DOI:10.15774/PPKE.ITK.2015.013

A MODEL OF COMPUTATIONAL MORPHOLOGY AND ITS APPLICATION TO URALIC LANGUAGES

Thesis for the Degree of Doctor of Philosophy

Attila Novák

Roska Tamás Doctoral School of Sciences and Technology

Pázmány Péter Catholic University, Faculty of Information Technology and Bionics

Academic advisor:
Dr. Gábor Prószéky

2015

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

A model of computational morphology and its application to Uralic languages

Can computers learn human languages? Can a computer program translate from one language to another adequately? Although we cannot answer with a definite ‘Yes’ to these questions, one thing is certain. The better the database of a linguistic program models the language, the better results it can produce. A key module in a linguistic model is the morphological component, which is responsible for the analysis and generation of words in the given language. In the past years, I have explored various ways of creating linguistically adequate computational morphologies for morphologically complex languages.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

Acknowledgements

You can skip this part. No names here. But if you want to check whether you are included, just go on.

First of all, I would like to thank those who managed to convince me that a PhD thesis need not to be a very sophisticated piece of art, and writing it is not a big deal after all. It is frustrating, but I accepted it as a fact in the end. This group of people includes, among others, my supervisor and reviewers of the PhD theses of some people I know. In addition, I would like to thank those who helped, hindered or bothered me, those who did not bother to interfere, those who sought and/or accepted my help, those who tolerated that I hindered or bothered them or did not bother to interfere, and those who needed me but waited for me patiently when I was absent.

And thanks to the hedgehogs. They are nice creatures.

DOI:10.15774/PPKE.ITK.2015.013

Contents

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

1

Introduction

The first chapter introduces the concepts of morphology, morphological analysis and the motivation of this thesis.

DOI:10.15774/PPKE.ITK.2015.013

Science primarily aims at describing and explaining various aspects of the world as we experience it. But since the second half of the 19$^{\text{th}}$ century, science has also grown to be a major contributor to technological knowledge. Nevertheless, language technology (i.e. computer technology applied to everyday language-related tasks) has coped in the past with a number of language-related problems without making much use of linguists' models. Doing morphology (i.e. handling word forms), for example, was not much of a technological problem for some of the commercially most interesting languages, especially for English, because it could be handled either by simple pattern matching techniques or by the enumeration of possible word forms.

Although there have been attempts to handle language technology tasks, such as spell checking, for languages that feature complex morphological structures and phonological alternations avoiding the use of a formal morphological description, these word-list-based attempts failed to produce acceptable results even recently, when corpora of sizes in an order of hundreds of millions of running words are available for languages such as Hungarian. However big the corpus is, even very common forms of not-extremely-frequent words are inevitably missing from it. Moreover, when I analyzed the word forms of the 150-million-word Hungarian National Corpus, I found that 60 percent of the theoretically possible Hungarian inflectional suffix morpheme sequences never occurs in the corpus. This figure does not include any of the numerous productive derivational suffixes. There is nothing odd about these suffix combinations. They are just rare. For a bigger 500-million-word Web corpus, I found the ratio to be 50 percent.

The creation of a formal morphological description is therefore unavoidable for this type of languages. The central theme of my thesis concerns **computational models of morphology that are applicable to such morphologically complex languages**. Some of these languages have also been commercially interesting to some extent: e.g. Turkish, Finnish or Hungarian, just to mention some from Europe. But as soon as there are tools which are readily applicable, what could stop us from applying these to languages that are spoken by less populous or rich speaker communities? So I also explored the task of creating computational morphologies for Uralic minority languages.

Beside the complexity of the morphology of these languages, another factor that makes a data-oriented approach unfeasible in some cases is the lack of electronically available linguistic resources.When working on Uralic minority languages, the corpora I had to do with did not exceed the size of a hundred thousand running words in the case of any of the languages involved, in some cases the size of the corpus did not even reach ten thousand words. In addition to a general lack of such resources concerning these languages, in the case of the most endangered ones, Nganasan and Mansi, there seems even to be a lack of really competent native speakers. The available linguistic data and their linguistic descriptions proved to be incomplete and contradictory for all of these languages, which also made numerous revisions to the computational models necessary.

The most successful and comprehensive analyzer for Hungarian (called Humor and developed by a Hungarian language technology firm, MorphoLogic) was based on an item-and-arrangement model analyzing words as sequences of allomorphs of morphemes and using allomorph adjacency constraints (Prószéky and Kis, 1999). Although the Humor analyzer itself proved to be an efficient tool, the format of the original database turned out to be

DOI:10.15774/PPKE.ITK.2015.013

problematic. A morphological database for Humor is difficult to create and maintain directly in the format used by the analyzer, because it contains redundant and hard-to-read low-level data structures. To avoid these problems, I created a higher-level morphological description formalism and a development environment that facilitate the creation and maintenance of the morphological databases.

I have created a number of complete computational morphologies using this morphological grammar development framework. The most important and most comprehensive of these is an implementation of Hungarian morphology. Its rule component was created relying mainly on my competence and on research I performed during the preparation of my theoretical linguistics Master's Thesis (Novák, 1999) and later refined during corpus testing and the actual use of the morphology in an English-to-Hungarian and a Hungarian-to-English machine translation system and various corpus annotation projects.

The first version of the analyzer's stem database was based on the original Humor database, from which all redundant (predictable) features were removed and unpredictable properties of words (e.g. category tag) were all manually checked and corrected, and missing properties were added (e.g. morpheme boundaries in morphologically complex entries). Currently, the size of the stem database is several times bigger than that of the original Humor analyzer and also contains specialized (e.g. medical, financial) vocabulary. The underlying grammatical model is much more accurate than that of the original morphology it replaced.

I have also created complete computational morphologies of Spanish and French using the morphological grammar development framework, and also adapted ones for Dutch, Italian and Romanian[i]. In addition, I co-authored morphologies for a number of endangered Finno-Ugric languages (Komi, Udmurt, Mari and Mansi)[ii]. In the same project, I created morphologies for two seriously endangered Northern Samoyedic languages: Nganasan and Tundra Nenets. The Uralic project was an attempt at using language technology to assist linguistic research, applying it to languages that can not otherwise be expected to be targets of the application of such technology due to the lack of commercial interest. The Uralic morphologies were further refined in a series follow-up projects, and two Khanty dialects (Synya and Kazym Khanty) were also described.

One aspect of morphological processing not covered by the original Humor implementation is that it does not support a suffix-based analysis of word forms whose stem is not in the stem database of the morphological analyzer, and the system cannot be easily modified to add this feature. Moreover, integration and appropriate usage of frequency information, as would be needed by data-driven statistical approaches to text normalization (e.g. automatic spelling error correction or speech recognition), is not possible within the original Humor system. A third factor that can be mentioned as a drawback of Humor is its closed-source licensing scheme that has been an obstacle to making resources built for morphological analyses widely available. The problems above could be solved by converting the morphological databases to a representation that can be compiled and used by finite-state morphological tools.

[i] These morphologies were used only in commercial products: in MorphoLogic's MoBiMouse/MorphoMouse pop-up dictionary program (see Section 8.1), in on-line dictionaries based on MorphoLogic technology and as on-line spell checkers.

[ii] This was done in a project called 'Complex Uralic Linguistic Database' (NKFP 5/135/2001), in which I worked together with László Fejes

DOI:10.15774/PPKE.ITK.2015.013

The Xerox tools implement a powerful formalism to describe complex types of morphological structures. This suggested that mapping of the morphologies implemented in the Humor formalism to a finite-state representation should have no impediment. However, the Xerox tools (called *xfst*), although made freely available for academic and research use in 2003 with the publication of Beesley and Karttunen (2003), do not differ from Humor in two significant respects: a) they are closed-source and b) cannot handle weighted models. Luckily, a few years later quite a few open-source alternatives to *xfst* were developed. One of these open-source tools, Foma (Huldén, 2009), can be used to compile and use morphologies written using the same formalism. Another tool, OpenFST (Allauzen et al., 2007), is capable of handling weighted transducers, and a third tool, HFST (Lindén et al., 2011), can convert transducers from one format to the other and act as a common interface above the Foma and OpenFST backends.

Creating high-quality computational morphologies is an undertaking that requires a considerable amount of effort, and requires threefold competence: familiarity with the formalism, knowledge of the morphology, phonology and orthography of the language, and extensive lexical knowledge. Thus another interesting subject is the learnability of (aspects of) morphology from corpora or existing lexical databases using automatic methods. Many morphological resources contain no explicit rule component. Such resources are created by converting the information included in some morphological dictionary to simple data structures representing the inflectional behavior of the lexical items included in the lexicon. The representation often contains only base forms and some sort of information (often just a paradigm ID) identifying the inflectional paradigm of the word, possibly augmented with a few other morphosyntactic features. With no rules, the extension of such resources with new lexical items is not such a straightforward task, as it is in the case of rule-based grammars. However, the application of machine learning methods may be able to make up for the lack of a rule component. Thus, I solved the problem of predicting the appropriate inflectional paradigm of out-of-vocabulary words, which are not included in the morphological lexicon. The method is based on a longest suffix matching model for paradigm identification, and it is showcased with and evaluated against an open-source Russian morphological lexicon.

Since the detailed presentation of each of the morphologies that I developed could itself be the subject of a separate research report, I cannot undertake to present them in full depth within the scope of this thesis. However, fragments of the grammars and lexicons are used in the corresponding chapters to illustrate features of the formalisms. Phenomena from the above-mentioned languages relevant to the subject of my work and the way they are handled in the models are presented and discussed. Emphasis is laid primarily on Hungarian, the language of which the most comprehensive description was created, but I also present some details about the morphologies created for other Uralic languages.

Another group of 'languages' for which I describe the method of adaptation of the general morphological analyzer, are some variants of Hungarian. First, the Humor morphological analyzer was extended to be capable of analyzing words containing morphological constructions, suffix allomorphs, suffix morphemes, paradigms or stems that were used in Old and Middle Hungarian but no longer exist in present-day Hungarian. A disambiguation system was also developed that can be used for automatic and manual disambiguation of the morphosyntactic annotation of texts and a corpus manager is described with the help

DOI:10.15774/PPKE.ITK.2015.013

of which the annotated corpora can be searched and maintained. Another ‘language’ for which the analyzer was adapted was the language of Hungarian clinical documents created in clinical settings. This language variant differs from general Hungarian in several respects. In order to process such texts written in a so called notational language, the morphological analyzer had to be adapted to the requirements of the domain by extending its lexicon applying a semi-automated algorithm.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

2

Background

Some background is needed in order to avoid reinventing the wheel. Some basic concepts and approaches from ancient Greeks to modern finite-state technology are introduced briefly in order to place my work on the map of computational morphologies.

Contents

DOI:10.15774/PPKE.ITK.2015.013

2.1 Computational morphology

Even though the formal representation of human language understanding as a whole might still be a fiction, there are some subtasks of natural language processing for which the achieved results are significant. Moreover, these solutions are already available for everyone using any kind of digital text processing tools and are part of many text processing algorithms. Such a field is that of computational morphology.

The morphology of agglutinating languages (such as Hungarian or other Uralic languages described in this Thesis) is rather complex. Words are often composed of long sequences of morphemes (atomic meaning or function bearing elements). Thus, agglutination and compounding yield a huge number of different word forms. For example while the number of different word tokens in a 20-million-word English corpus is generally below 100,000, in Finnish, it is well above 1,000,000, and the number is above 800,000 in the case of Hungarian. However, the 1:8 ratio does not correspond to the ratio of the number of possible word forms between the two languages: while there are about at most 4–5 different inflected forms for an English word, there are about a 1000 for Hungarian[i], which indicates that a corpus of the same size is much less representative for Hungarian than it is for English (Oravecz and Dienes, 2002). Figure 2.1 shows the number of different word forms in a 44-million-word corpus for English, Hungarian, Estonian, Finnish and Turkish (Creutz et al., 2007)[ii]. Finnish and Estonian both have a larger number of different word forms than Hungarian, since in these languages adjectives agree with the head of the noun phrase in case and number. Inflected adjectives are also correct forms in Hungarian, however they are used only if the noun is missing (i.e. in the case of ellipsis), and thus these forms are much rarer in Hungarian. This means that data sparseness is greater for Hungarian than it is for Finnish or Estonian: there are much more correct word forms not appearing in a corpus of any size.

The task of computational morphology is to handle the different word forms, generally applied to written language, while spoken or dialectal language raises special problems due to the lack of a standard orthography. This is the base of any further processing. The two most important tasks a computational morphology must be capable of are analysis and generation.

Morphological analysis is the task of recognizing words of the given language by finding its lemma and part-of-speech, the morphosyntactic features (i.e. coordinates that define the place of the actual word form in the paradigm of the lemma) and identifying derivations and compound structures. These are determined without considering the possible disambiguating effect of the lexical context the word occurs in. **Word form generation** is the task of producing the surface form corresponding to a given lemma and the morphological features.

The representation of the resulting analysis of a word depends on the morphological model used in the implementation. There are several such models describing the structure of words.

[i] Note that this number does not include theoretically possible but hardly ever occurring forms mentioned in the Introduction. Anyway, what I would like to emphasize here is the difference in the magnitudes. Note that the frequency of forms, on the other hand, is indeed relevant, and the lack of frequency information in "rule-based" systems may be an important source of problems when it comes e.g. to suggesting corrections for an erroneous word in a spell checker application.

[ii] The data in the chart for Hungarian is based on the Hungarian Webcorpus (Halácsy et al., 2004)

DOI:10.15774/PPKE.ITK.2015.013

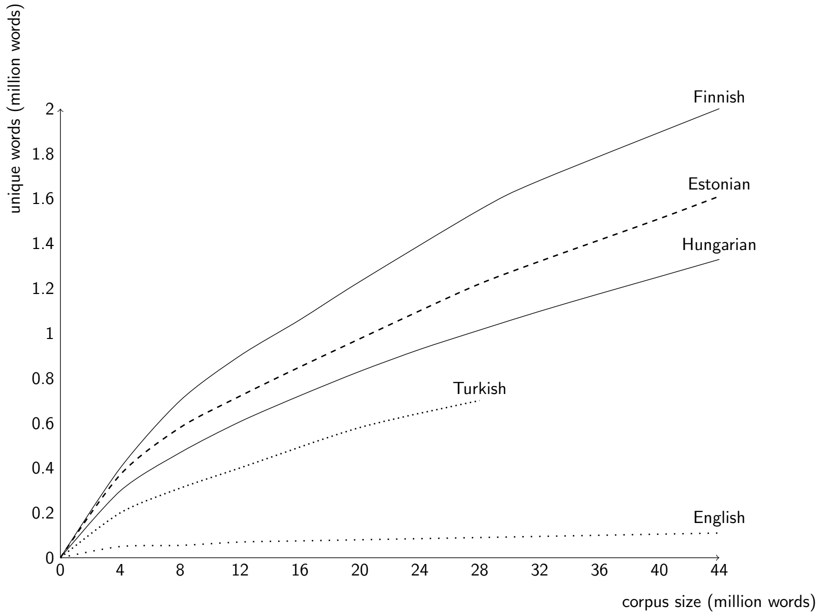


Figure 2.1: Different word forms in a corpus representative of the given language.

The classical morphological model following the traditions of Ancient Greek linguistics is the **word-and-paradigm** approach (Matthews, 1991). In this model, words are atomic elements of the language, lacking any internal structure (apart from letters/sounds). Thus, structures composed of smaller units are not considered as different inflected forms of the same lemma, but as different members of the inflectional paradigm of the word. At the beginning of the 20^{th} century, structuralists introduced *morphemes* as the smallest unit of a language to which a meaning or function may be assigned. The goal of linguistics according to this model is to find the morpheme set of a given language, the different realizations of each morpheme (allomorphs) and their distribution. This approach is called **item-and-arrangement** morphology (Hockett, 1954). Yet another theoretical approach to morphology is that of **item-and-process** models (Hockett, 1954), in which words are built through processes, not by a simple concatenation of allomorphs of morphemes. Thus, a word is viewed as the result of an operation (word formation rule) that applies to a root paired with a set of morphosyntactic features, and yields its final form.

Morphology, moreover, is closely related to **phonology**. The actual form of morphemes building up an actual word form depends on phonological processes or constraints both in a local (e.g. assimilation) or long-distance manner (e.g. vowel harmony). Generative phonology (Chomsky and Halle, 1968) introduced a system of a sequence of context-sensitive rewriting rules applied in a predefined order. In that model, there are several intermediate layers between the surface and the lexical layer. Even though this approach was suitable for phonological generation, it could not be applied to word form recognition or analysis, since due to the inherently non-deterministic manner in which rules are applied makes the search space explode in an exponential manner, which cannot be avoided by the prefiltering effect of rules applied earlier or by the use of a lexicon. However, the context-sensitive rules of phonology can be transformed to regular relations (Johnson, 1972; Kaplan and Kay, 1994). Their composition is also regular, which means that the whole system of rules can be represented by a single regular transformation, which can even be composed with the lexicon.

DOI:10.15774/PPKE.ITK.2015.013

Due to the limited memory available in contemporary computers, which was not enough for the implementation of such composite systems, Kimmo Koskenniemi introduced **two-level morhpology** (Koskenniemi, 1983), which solved the problem of intermediate levels, and could be implemented using little memory. In this approach, parallel transducers are applied, where each symbol pair of the lexical and surface layers must be accepted by each automaton. This approach led to the first viable finite-state representation of morphology.

By the beginning of the 1990's, computational morphologies were developed for morphologically complex languages using formal models that were adaptations of real linguistic models. The technology and the linguistic models that were used in the leading Finnish/Turkish and Hungarian morphological analyzers differed from each other. The Finnish model (also applied to Turkish) used **finite-state transducer** technology and was based on two-level phonological rules using the formalism defined by Koskenniemi (Koskenniemi, 1983). The most successful and comprehensive analyzer for Hungarian (called *Humor* and developed by a Hungarian language technology firm, MorphoLogic), on the other hand, was based on an **item-and-arrangement model** analyzing words as sequences of allomorphs of morphemes and using allomorph adjacency constraints (Prószéky and Kis, 1999). Although the two approaches differed from each other in the algorithms and data structures used, a common feature was that the linguistic database used by the morphological analyzer itself was optimized for computational efficiency and not for human readability and manual maintenance.

For other languages, different methods were applied to perform morphological analysis. An in-depth historical overview of all computational approaches to morphology and their applications to various languages would not fit into the scope of this Thesis, nevertheless, I will highlight a few points here. One branch of the approaches to computational morphology is based on the assumption that the language has a finite vocabulary, and all word forms can be enumerated. This finite list can then be compiled into an acyclic automaton, attaching information needed to return lemmas and morphosyntactic tags using pointers at certain (typically terminal) nodes. Another general assumption is that morphological paradigms are also simply enumerable, and each lexical item can be simply assigned a paradigm id from a finite set of such id's, and each of these paradigm id's corresponds to a simple operation that maps the lemma to a small finite set of word forms with analyses. Note, however, that the assumptions that all word forms and paradigms are simply enumerable does not hold for languages like Hungarian.

One system based on acyclic finite-state automata developed at the end of the 1990's, is Jan Daciuk's *fsa* package (Daciuk et al., 2000), and further similar implementations inspired by that tool (e.g. the *majka* system Šmerk (2009)) were used to create morphological analyzers for many European languages ranging from English, Dutch and Spanish to Bulgarian and Russian by compiling analyzed word lists created by various ad-hoc methods. Also for Czech, and some other Slavic languages, other finite-state tools were used (Hajič, 2001) with a finite vocabulary.

Another extensive stock of morphological resources were created with the *INTEX/UNITEX* tools using a similar enumerate-and-compile methodology (Silberztein, 1994; Paumier et al., 2009). Some morphologies (French, Arabic, etc.) were built using another linguistic development platform derived from INTEX, *NooJ* (Silberztein, 2005), which includes tools

DOI:10.15774/PPKE.ITK.2015.013

to construct, test and maintain wide-coverage finite-state lexical resources. An attempt at creating even a NooJ-based Hungarian morphology was made at the Research Institute for Linguistics if the Hungarian Academy of Sciences, converting a Hungarian inflectional dictionary (Elekfi, 1994). However, due to limitations of the approach, lack of coverage of derivation in the dictionary, and because the morphological description did not include a grammar that could be used to add new lexical items, the performance and coverage of this tool never approached that of either of the Hungarian computational morphologies mentioned in this Thesis (Gábor, 2010), and its further development was abandoned.

The *mmorph* tool (Petitpierre and Russell, 1994) developed at IISCO, Geneva represents another line of tools that do not make the finiteness assumption. This tool included a unification-based context-free word grammar, and orthographic alternations in allomorphs were handled by Kimmo-style two-level rules. Although the context-free word grammar rules implemented in this tool provide a simple solution to the problem of handling non-local dependencies between morphemes, finite-state automata can handle the same problem in a more efficient manner using an extended state space, as we will show in Sections 4.2.4 and 6.2. The English morphology implemented using mmorph was described in a detailed monograph (Ritchie et al., 1992), and further morphologies for German, French, Spanish and Italian were created using this formalism. These resources do not seem to be available any more, however.

A pair of tools that are often used for English morphological analysis and generation, *morpha* and *morphg* (Minnen et al., 2001) do not even contain an extensive stem lexicon, but instead comprise a set of morphological generalizations together with a list of exceptions for specific wordforms. The implementation of these tools is also based on finite-state techniques. The morpha analyzer depends on Penn-Treebank-style PoS-tags in its input and performs lemmatization only. It can also be used to analyze untagged input, but its performance is rather poor in that case due to lack of a lexical component.

The Xerox tools (Beesley and Karttunen, 2003), described in more details in the forthcoming sections, were used to implement two-level morphologies for Turkish (Oflazer, 1993) Finnish (http://www2.lingsoft.fi/doc/fintwol/) and many other languages including Hungarian. The state-of-the-art morphological systems for most languages are based on the Xerox finite-state formalism and its open-source alternatives, hfst (Lindén et al., 2011) and Foma (Huldén and Francom, 2012) that I will also cover in more detail.

2.2 Affix-stripping models

One branch of morphological grammar description tools is based on affix stripping. An earlier implementation is that of Packard (1973) from the 70s, parsing ancient Greek by iteratively stripping prefixes and suffixes of the word to be analyzed and then matching the remaining part against a lexicon (Jurafsky and Martin, 2000). The Porter stemmer (Porter, 1980), used for a long time in many English information retrieval applications, is also based on affix-stripping operations.

DOI:10.15774/PPKE.ITK.2015.013

Another implementation of affix-stripping methods are the descendants of the *Ispell* and *spell* tools (Peterson, 1980). These methods require a set of affix rules, where each rule contains a condition, a strip string and an append string. Lexical entries or base forms are also stored together with a set of features which identify the compatible affixes. When analyzing an input word, the algorithm strips off possible strip strings (i.e. possible affixes) according to the affix rules and appends the corresponding strings the rule prescribes. In each step, the resulting word is considered as a hypothesized lemma that is checked against the lexicon. The analysis is complete if a base form is found validating the actual affix as possible after the base form and all the other affixes fulfill the requirements of the conditions.

However, such a simple model of morphology was not applicable to complex languages. As an extension of the original *Ispell* line of models, *hunspell* and *hunmorph* were introduced (Trón et al., 2005, 2006), where the stripping and checking is performed iteratively, i.e. after stripping some affixes, the remaining part is checked again for possibilities of stripping and in each step the compatibility of affixes is also checked. Thus, it is also possible to handle circumfixes and this model is also capable of handling productive compounding. This set of generic tools, called *HunTools*, for morphological analysis and morphology development were created during a project called 'Szószablya' (Halácsy et al., 2003; Németh et al., 2004) and were designed to be able to handle complex morphologies like that of Hungarian. A resource manager tool, *HunLex*, was also developed for these tools, which is able to create optimized, language-specific resources for each module of HunTools. HunLex (Trón, 2004) uses a morphological database from which it is able to create the necessary output depending on its parameters. The morphological analyzer and spell tools use `.dic` and `.aff` files, the format of which is a modified (extended) version of the ones used by the *myspell* tools. HunLex implements a nice declarative morphological grammar resource description formalism, which was implemented a few years after the formalism and system described in Chapter 4 of this thesis. Unfortunately, the only working language resource I know about that was created using this formalism is the *morphdb.hu* morphology for Hungarian (Trón et al., 2006). Later, as the project financing the development of HunLex and morphdb.hu ended, further development and maintenance of these tools and resources was abandoned, but their revitalization and merging them with the Hungarian resources described in this Thesis is currently under way.

2.3 Finite-state models

As it was shown by Johnson (1972) and Kaplan and Kay (1994), rewrite rules are equivalent to finite-state transducers. As opposed to finite-state automata, transducers not only accept or reject an input string, but accept or reject two strings whose letters are pair-matched (Young and Chan, 2009), or, in practice, when a transducer accepts a string, it also generates all of the strings to which the regular relation implemented by the transducer maps the input string. However, handcrafting or manually checking a finite-state transducer for correctness even for a single phonological rule is a rather difficult task. Doing that for a single transducer representing a complete morphology with a lexicon and phonological/orthographic rules is more than difficult: it is impossible simply due to the sheer size of the model: the transducer for the Nganasan morphology described in Section 5.4 consists of 70,307 states and 209,346

DOI:10.15774/PPKE.ITK.2015.013

arcs, while the Hungarian morphology described in Chapter 6 consists of more than 1.3 million states and 3.1 million arcs. However, using a single transducer for the task instead of a set of simpler transducers is much more efficient in terms of speed. Thus, while finite-state transducers are simple and easy to implement, it were the algorithms implemented by Kaplan and Kay for compiling an ordered cascade of rewrite rules into a single transducer that made the finite-state implementation of morphology and phonology feasible and more efficient than the two-level implementation of Koskenniemi overcoming the limitations of the former approaches. This, however, could only be used in practice when 32-bit operating systems and computers with hundreds of megabytes of memory became available in the nineties.

The most elaborate toolkit developed for linguists to model morphology within this theoretical framework is the *xfst-lookup* combo of Xerox (Beesley and Karttunen, 2003), a program for compiling and executing rules. *Xfst* is an integrated tool that can be used to build computational morphologies implemented as finite-state transducers. The other tool, *lookup* consists of optimized run-time algorithms to implement morphological analysis and generation using the lexical transducers compiled by *xfst*.

The formalism for describing morphological lexicons in *xfst* is called *lexc*. It is used to describe morphemes, organize them into sublexicons and describe word grammar using continuation classes. A lexc sublexicon consists of morphemes having an abstract lexical representation that contains the morphological tags and lemmas and usually a phonologically abstract underlying representation of the morpheme, which is in turn mapped to genuine surface representations by a system of phonological/orthographic rules. The details of describing a morphology using this formalism is described in Sections 6.2 and 5.4, where it is shown how this approach can be used to describe morphologically complex languages.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

3

Humor

An introduction to Humor, which is the cornerstone of the research described in the following chapters. It is shown how Humor represents morphology and the constraints required to properly analyze complex word forms. Includes the binary representation of 'bush' and 'dog'. But not that of 'hedgehog'.

Contents

DOI:10.15774/PPKE.ITK.2015.013

Although morphological analysis is the basis for many natural language processing (NLP) applications, especially for languages with a complex morphology, descriptions of many morphological analyzers as separate NLP tools appeared with a significant delay in the literature and in a rather sketchy manner, since for a long time most of these tools were commercial products. The morphological analyzer called *Humor* ('High speed Unification MORphology'), which was used for tagging most publicly available annotated Hungarian corpora was also commercial product developed by a Hungarian language technology company, MorphoLogic (Prószéky and Kis, 1999). This commercial ownership prevented a detailed description of methods used in the Humor analyzer to be published for a long time.

The Humor analyzer performs a classical 'item-and-arrangement' (IA)-style analysis (Hockett, 1954), where the input word is analyzed as a sequence of morphs. Each morph is a specific realization (an allomorph) of a morpheme. Although the 'item-and-arrangement' approach to morphology has been criticized, mainly on theoretical grounds, by a number of authors (c.f. e.g. Hockett (1954); Hoeksema and Janda (1988); Matthews (1991)), the Humor formalism was in practice successfully applied to languages like Hungarian, Polish (Wołosz, 2005), German, Romanian, Spanish and Croatian (Aleksa, 2006).

The Humor analyzer segments the word into parts which have (i) a surface form (that appears as part of the input string, the morph), (ii) a lexical form (the 'quotation form' of the morpheme) and (iii) a (possibly structured) category label.

The analyzer produces flat morph lists as possible analyses, i.e. it does not assign any internal constituent structure to the words it analyzes, because it contains a regular word grammar, which is represented as a finite-state automaton. This is more efficient than having a context-free (CF) parser, and it also avoids most of the irrelevant ambiguities a CF parser would produce. In a Humor analysis, morphs are separated by + signs from each other. The representation of morphs is `lexical form[category label]=surface form`. The surface form is appended only if it differs from the lexical form. To facilitate lemmatization, a prefix in category labels identifies the morphological category of the morpheme (`S_`: stem, `D_`: derivational suffix, `I_`: inflectional suffix). In the case of derivational affixes, the syntactic category of the derived word is also given.

The following analyses of the Hungarian word form *Várnának* contain two morphs each, a stem and an inflectional suffix, delimited by a plus sign.

```
analyzer>Várnának
Várna[S_N]=Várná+nak[I_Dat]
vár[S_V]=Vár+nának[I_Cond.P3]
```

The lexical form of the stem differs from the surface form (following an equal sign) in both analyses: the final vowel of the noun stem (having a category label `[S_N]`) is lengthened from *a* to *á*, while the verbal stem (having a category label `[S_V]`) differs in capitalization. In this example, the labels of stem morphemes have the prefix `S_`, while inflectional suffixes have the prefix `I_`.

The category label of stems is their part of speech, while that of prefixes and suffixes is a mnemonic tag expressing their morphosyntactic function. In the case of homonymous lexemes where the category label alone is not sufficient for disambiguation, an easily identifiable

DOI:10.15774/PPKE.ITK.2015.013

```
bokor,       G,101..... .0.00010, ‘,........ ........,bokor,       FN
bokorbab,    B,10111111 11000011, ‘,........ ........,bokorbab,    FN
bokorrózsa,  C,100..... ...00011, ‘,........ ........,bokorrózsa,  FN
bokorrózsá,  D,10000100 11100011, ‘,........ ........,bokorrózsa,  FN
bokorugró,   A,10100100 11101011, ‘,........ ........,bokorugró,   MN
bokr,        H,10111010 01000010, ‘,........ ........,bokor,       FN
bokros,      B,10010010 10011010, ‘,........ ........,bokros,      MN
bokros,      B,10110010 10010010, ‘,........ ........,bokros,      FN
bokrosod,    A,00011010 10000000, ‘,........ ........,bokrosodik,  IGE
bokrosodás,  B,10110010 11000010, ‘,........ ........,bokrosodás,  FN
bokréta,     C,100..... ...00010, ‘,........ ........,bokréta,     FN
bokrétaünnep,B,11011010 11000011, ‘,........ ........,bokrétaünnep,FN
bokrétá,     D,10000100 11100010, ‘,........ ........,bokréta,     FN
bokrétás,    B,10010010 10001010, ‘,........ ........,bokrétás,    MN
...
kutya,       C,10...... ...00010, ‘,........ ........,kutya,       FN
kutyá,       D,10000100 01100010, ‘,........ ........,kutya,       FN
...
at,          A,00000000 00000000, 1,100.1... ........,at,          ACC
et,          A,00000000 00000000, 1,110.1... ........,et,          ACC
ot,          A,00000000 00000000, 1,101.1... ........,ot,          ACC
t,           A,00000000 00000000, 1,1...0... ........,t,           ACC
öt,          A,00000000 00000000, 1,111.1... ........,öt,          ACC
```

Figure 3.1: Humor representation of the allomorphs of the Hungarian stem morpheme *bokor* ‘bush’ (and some other stems starting with ‘bok’), *kutya* ‘dog’ and those of the accusative suffix. The fields separated by commas are the following: surface form, right-hand-side continuation class, right-hand-side binary properties vector, left-hand-side continuation class, left-hand-side binary requirements vector, lexical form, morphosyntactic tag

indexing tag is often added to the lexical form to distinguish the two morphemes in the Humor databases. The disambiguating tag is a synonymous word identifying the morpheme at hand. Using this disambiguating tag is important in the case of homonymous stems where there is also a difference in the paradigms of the distinct morphemes, especially when using the morphology to perform word form generation. E.g. in the Hungarian database, the word *daru* ‘crane’ is represented as two distinct morphemes: `daru_gép[N] ‘crane_machine[N]’` and `daru_madár[N] ‘crane_bird[N]’`, since a number of their inflected forms differ, e.g. plural of the machine is *daruk*, while that of the bird is *darvak*.

3.1 The lexical database

The lexical database of the Humor analyzer consists of an inventory of morpheme allomorphs, the word grammar automaton and two types of data structures used for the local compatibility check of adjacent morphs. One of these are continuation classes and binary continuation matrices describing the compatibility of those continuation classes (see Figure 3.2). The other are binary vectors of properties and requirements. Each morph has a continuation class identifier on both its left and right hand sides, in addition to a right-hand-side binary properties vector and a left-hand-side binary requirements vector. The latter may contain don’t care positions represented by dots. A sample of Humor representation of morphs can be seen in Figure 3.1.

DOI:10.15774/PPKE.ITK.2015.013

3.2 Morphological analysis

When doing morphological analysis, the program performs a depth-first search on the input word form for possible analyses. It looks up morphs in the lexicon the surface form of which matches the beginning of the yet unanalyzed part of the input word. The lexicon may contain morph sequences, i.e. ready-made analyses for irregular forms of stems or suffix sequences, which can thus be identified by the analyzer in a single step.

Two kinds of checks are performed at each morph lookup step: a **local compatibility check** of the next morph with the previous one and a **global word structure check** on each locally compatible candidate morph by traversing a deterministic extended finite-state automaton (EFSA) that describes possible word structures.

The data structures used and the steps taken by the analyzer during lookup are explained below. However, to better understand how the Humor lookup algorithm works, you might find it helpful also to take a look at the lookup trace of the morphological analyzer for the Hungarian input word *tör* 'breaks sg.' in Appendix A.2.

3.2.1 Local compatibility check

Local compatibility check is performed as follows: a morph (typically a suffix) may be attached to another morph (typically a stem) if the right-hand-side properties of the stem match the left-hand-side requirements of the suffix by checking both compatibility of the continuation matrix codes and matching of the corresponding binary vectors. Multiple binary continuation matrices can be defined; e.g. a different matrix for verbs and other stems. The gross size of matrices can thus be reduced by eliminating empty regions that would necessarily be there if a single matrix were used. The matrix to be used for compatibility check is selected using a subset of the binary properties of the left-hand-side (stem) morph e.g. a single binary feature bit differentiates verbal stems from other morphs in the Hungarian grammar, and the continuation matrix is selected using this single bit. Figure 3.2 shows a compatibility matrix for non-verbal categories in the original Hungarian Humor database. This data structure is indeed rather difficult to read. The corresponding matrix generated from the Hungarian description presented in Section 5.1, which is a more accurate representation of adjacency constraints of Hungarian nominal morphemes, has 1019 columns and 219 rows. That matrix is completely impossible to read for humans.

3.2.2 Word grammar automaton

The word grammar automaton used in the Humor analyzer to check overall word structure may have, in addition to its main state variable, extra binary state variables, which can be used to handle non-local constraints within the word without an explosion of the size of the automaton. An example of such a non-local constraint is related to the way superlatives are marked in Hungarian. Superlative is expressed by a combination of two morphemes: the superlative prefix *leg-* and the comparative suffix *-bb*. In general, a word form that contains a

DOI:10.15774/PPKE.ITK.2015.013

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	‘	a	b	c	d	e	f	g
‘	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
a	*	*	*	-	*	-	*	-	*	-	-	-	*	-	-	*	-	-	-	*	*	*	-	*	*	*	-	*	*	*	*	*	*	-	*	-	-	-	-
b	*	*	-	*	-	*	-	*	*	-	-	*	-	*	*	-	-	*	-	*	*	-	-	-	*	*	*	*	*	*	*	*	-	*	*	-	*	-	-
c	*	*	-	*	-	*	-	*	-	*	*	-	-	*	*	-	*	-	-	*	*	-	*	-	*	*	*	*	*	*	*	*	*	*	*	*	-	-	-
d	*	*	-	*	-	*	-	*	*	-	-	*	-	*	*	-	-	*	-	*	*	-	-	-	*	*	*	*	*	*	*	*	-	*	*	-	*	-	-
e	*	*	-	*	-	*	-	*	*	*	*	*	-	*	*	-	*	-	-	*	*	-	*	-	-	-	-	*	*	*	*	*	*	*	*	-	*	-	-
f	*	*	-	*	-	*	-	*	*	-	-	*	-	*	*	*	*	-	-	*	*	-	-	-	-	-	-	*	*	*	*	*	-	*	*	-	*	-	-
g	*	*	-	*	*	*	*	*	*	-	-	*	-	*	-	*	*	-	-	*	*	*	-	*	-	-	*	*	*	*	*	*	*	*	*	-	-	-	*
h	*	*	*	*	*	-	*	-	*	-	-	-	-	*	-	*	-	-	-	*	*	*	-	*	-	-	-	-	*	*	*	*	*	-	*	-	-	*	*
i	*	*	-	*	*	-	*	*	*	*	-	*	-	*	-	*	*	-	-	*	*	*	-	*	-	-	-	*	*	*	*	*	*	-	*	-	-	*	*
j	*	*	-	*	*	-	*	-	*	-	-	*	-	*	-	*	-	-	-	*	*	*	-	*	-	-	-	*	*	*	*	*	*	-	*	-	-	*	*
k	*	*	-	*	*	-	-	*	*	-	-	*	-	*	-	*	-	-	-	*	*	*	-	-	-	-	-	*	*	*	*	*	*	*	*	-	-	*	*
l	*	*	-	*	-	*	-	*	*	-	-	*	-	*	*	*	-	-	-	*	*	-	-	-	-	-	-	*	*	*	*	*	-	*	*	-	*	*	*
m	*	-	-	*	-	-	-	-	*	-	-	*	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	-	-	-
n	-	*	-	-	-	-	*	-	-	-	-	-	-	-	-	-	*	-	-	-	-	*	-	-	-	-	-	-	-	-	-	-	*	-	-	-	-	*	*
o	*	*	-	*	-	*	-	*	-	*	*	*	-	*	-	-	*	-	-	*	*	-	*	-	-	-	-	*	*	*	*	*	-	*	*	-	-	-	-
p	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	*	-	*	-	-	-	-	-	-	*	*	*	*	*	-	-	-	-	-	-	-
q	*	*	*	-	*	-	*	*	-	-	-	-	-	*	-	-	*	-	-	*	*	*	*	*	-	*	*	*	*	*	*	*	*	-	*	-	-	-	-

Figure 3.2: Compatibility matrix for non-verbal categories in the original Hungarian Humor database.

superlative prefix without the comparative suffix licensing it later within the word is ill-formed[i]. However, quite a few morphemes may intervene, e.g. *leg[Sup]+isten+tagad+ó+bb[Cmp]* SUP+God+deny+V>Adj+CMP, ‘the most atheist’. Similarly, verbal prefixes can either stand on their own or they must be followed by a verbal stem or a verb-forming suffix somewhere within the word, e.g. *be[VPfx]+zebra+csík+oz[N> V]* in+zebra+stripe+N>V ‘to make sg. zebra striped’.

An example of the word grammar automaton formalism is presented in Figure 3.3. This fragment of the Hungarian word grammar shows the non-final state N2 of the automaton reached when the final member of a nominal compound has been encountered. In this state, inflections (the arc labeled `inf`) may follow in addition to various derivational suffixes (represented by the rest of the outgoing arcs). When the inflection arc is traversed, the cleared state of two flags is checked: there must not have been either a dangling verbal prefix or a dangling superlative prefix in the word (both the `sup` ‘superlative prefix present’ and the `vpfx` ‘verb prefix present’ flag must be in the cleared state). When encountering a comparative `ndercmp`[ii] or a verbalizer suffix `vder`, the respective flag (`sup` or `vpfx`) is cleared[iii]. The `lcase` flag, which is also set in these cases, licenses lower-casing of proper nouns when certain derivational suffixes are attached to them.

As part of the word grammar implementation, the morphological database of the Humor analyzer contains a mapping (see Figure 3.4) from right-hand-side property vectors to sets of possible morphological category labels, which are used as arc labels in the word grammar automaton, such as `inf`, `ndercmp` or `vder` in Figure 3.3. The lookup and local compatibility check of each morph in the word form is followed by a move in the word grammar automaton. The move is possible if, at the current state of the automaton, there is an outgoing arc labeled by one of the morphological category labels to which the right-hand-side property

[i] There are a few stems (most of them containing the *-só/ső* suffix)that license the superlative prefix themselves, such as *utolsó* ‘last’: *leg+utolsó* ‘the very last’ is a well-formed Hungarian word.

[ii] A comparative suffix may also be attached to nouns in Hungarian.

[iii] Note that the binary vector of features is right-aligned in this representation.

DOI:10.15774/PPKE.ITK.2015.013

```
#right compound member encountered
N2:
inf ->      END ?{0.0}  #?{!sup !vpfx}
119sfx ->   ADJ ={1...} #={lcase}
nder-119 -> N2  ={1...} #={lcase}
ndercmp ->  ADJ ={10..} #={lcase !sup}
nder2_adj ->ADJ
nder2 ->    N2
vder ->     V   ={1..0} #={lcase !vpfx}
```

Figure 3.3: Fragment of the Hungarian word grammar automaton – non-final state N2.

vector of the morph currently looked up is mapped. At the end of the word, the automaton must be in final state for the current analysis to be acceptable.

```
#mapping used by the Humor morphological analyzer
inf:     R, 0x200, 0.1.....0..0........0..0........
ndercmp: L, 0x100, 101...............0....1........
vder:    L, 0x100, 111.................0...........

# converted from the following description fragment:
# inflectional suffix
'inf' =>     ['r','sfx&!inflable&!punct&!qtag&!sup&!119sfx'],
# the comparative sfx (a nominal case-lowering derivational suffix)
'ndercmp' => ['l','sfx&inflable&!cat_vrb&!cmp2&sup'],
# verbal derivational suffix
'vder' =>    ['l','sfx&inflable&cat_vrb&!HAT'],
```

Figure 3.4: Fragment of the mapping of right-hand-side properties to word grammar automaton arc label categories in the Hungarian morphological description.

DOI:10.15774/PPKE.ITK.2015.013

4

A morphological grammar development framework

A formalism is described here which facilitates writing Humor-based morphologies. It is a computational model of morphology that is able to handle morphologically complex agglutinating languages, is easy to extend, maintain and debug, and does not require much memory to compile and run.

More dogs and bushes. Quite an amount of technical details. But still no hedgehog.

Contents

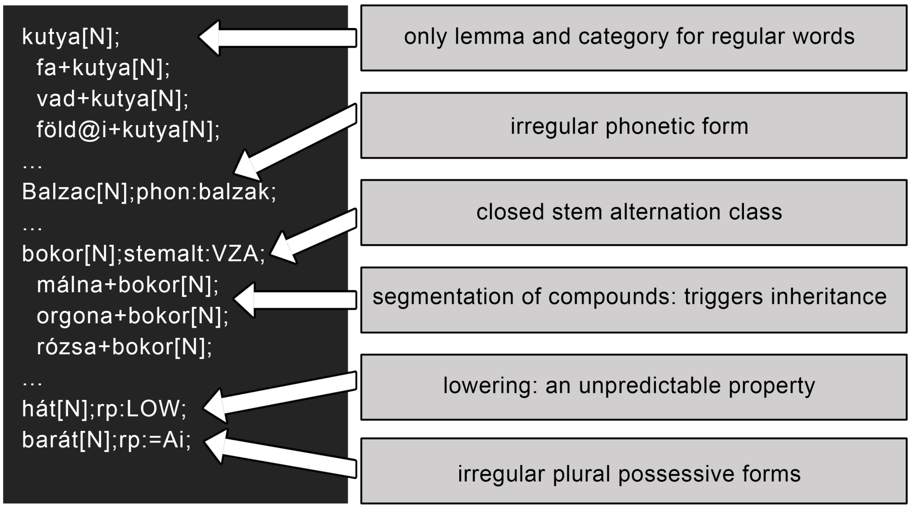
DOI:10.15774/PPKE.ITK.2015.013

Although the Humor analyzer itself proved to be an efficient tool, the format of the original database turned out to be problematic. The Humor system lacked a morphological grammar development component and the linguistic database format used in Humor was rather hard to read. The stem lexicon of the original Humor analyzer for Hungarian was converted in an ad-hoc manner from Papp (1969), a morphological dictionary of Hungarian, while suffix lexicons were manually created (Prószéky, 2001). Some words were later added to the stock of words imported from Papp (1969) by copying and editing allomorph entries manually, but since it was a very tedious and error-prone process, the coverage of the analyzer did not significantly exceed that of the source dictionary. The lexicons contained many errors and inconsistencies (both random and systematic ones) that were difficult to find because of the hard-to-read and very redundant lexicon formalism. In addition, the underlying linguistic model contained some design flaws (due to lack of distinctions in the source dictionary) that were never properly corrected. Moreover, the original database contained redundant and hard-to-read low-level data structures.

To avoid these problems, a higher-level morphological description formalism and a development environment were created that facilitate the creation and maintenance of the morphological databases (Novák, 2008). All Humor morphologies built after the creation of the development environment were developed using this higher-level formalism.

4.1 Creating grammar-based morphological models with minimal redundancy

A morphological description created using the higher-level formalism consists of morpheme-inventories that contain only unpredictable features of morphemes and rules that introduce all redundant features and generate allomorphs of each morpheme. The morpheme database may also contain irregular allomorphs. Figure 4.1 shows some entries from the high-level stem database.



```
kutya[N];
  fa+kutya[N];
  vad+kutya[N];
  föld@i+kutya[N];
...
Balzac[N];phon:balzak;
...
bokor[N];stemalt:VZA;
  málna+bokor[N];
  orgona+bokor[N];
  rózsa+bokor[N];
...
hát[N];rp:LOW;
barát[N];rp:=Ai;
```

only lemma and category for regular words
irregular phonetic form
closed stem alternation class
segmentation of compounds: triggers inheritance
lowering: an unpredictable property
irregular plural possessive forms

Figure 4.1: Entries in the high-level stem database.

DOI:10.15774/PPKE.ITK.2015.013

The high-level human-readable description is transformed by the system to a redundant but still human-readable allomorph database by applying the rules to the morpheme descriptions. This is then transformed to the low-level representations of the analyzer using an encoding definition description. This defines how each high-level feature should be encoded for the analyzer. Certain features are mapped to binary properties while the rest determine the continuation matrices, which are generated by the system dynamically. Figure 4.2 shows the architecture of the multilevel database.

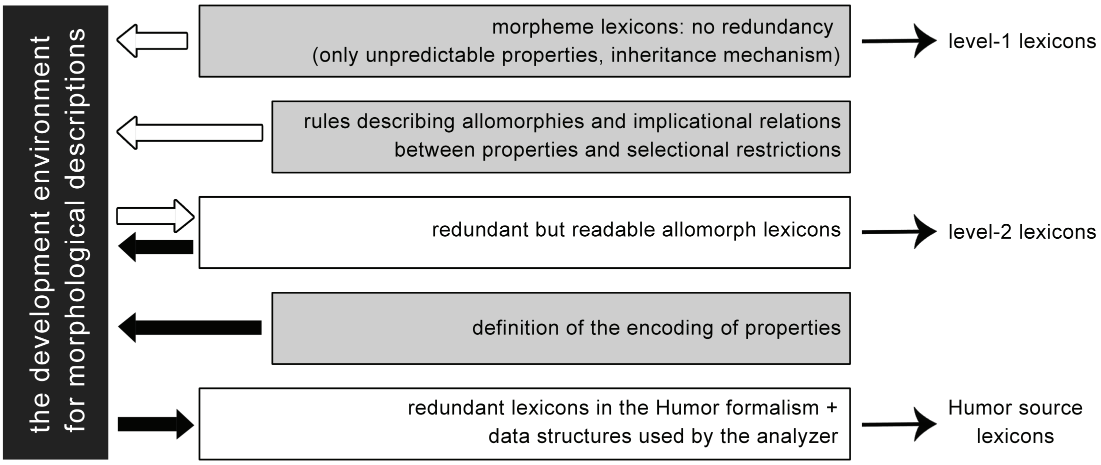

the development environment for morphological descriptions
morpheme lexicons: no redundancy (only unpredictable properties, inheritance mechanism)
level-1 lexicons
rules describing allomorphies and implicational relations between properties and selectional restrictions
redundant but readable allomorph lexicons
level-2 lexicons
definition of the encoding of properties
redundant lexicons in the Humor formalism + data structures used by the analyzer
Humor source lexicons

Figure 4.2: The multilevel database. Shaded blocks: input to the system. Unshaded blocks: generated by the system.

4.1.1 Creating a morphological description

When using the grammar development environment, the linguist has to create a high level human readable description which contains no redundant information and the system transforms it in a consistent way to the redundant representations which the analyzer uses. The work of the linguist consists of the following tasks:

Identification of the relevant morpheme categories in the language to be described (parts of speech, affix categories).

Description of stem and suffix alternations An operation must be described which produces each allomorph from the lexical form of the morpheme for each phonological allomorphy class. The morphs or phonological or phonotactic properties which condition the given alternation must be identified.

DOI:10.15774/PPKE.ITK.2015.013

Identification of features All features playing a role in the morphology of the language must be identified. These can be of various sorts: they can pertain to the category of the morpheme, to morphologically relevant properties of the shape of a given allomorph, to the idiosyncratic allomorphies triggered by the morpheme or to more than one of these at the same time.

Definition of selectional restrictions between adjacent morphs Selectional restrictions are described in terms of requirements that must be satisfied by the set of properties (features) of any morph adjacent to a morph. Each morph has two sets of properties: one can be seen by morphs adjacent to the left and the other by morphs adjacent to the right. Likewise, any morph can constrain its possible neighbors by defining a formula expressing its requirements on both sides.

Identification of implicational relations between properties of allomorphs and morphemes These implicational relations must be formulated as rules, which define how redundant properties and requirements of allomorphs can be inferred from their already known (lexically given or previously inferred) properties (including their shape). Rules may also define default properties. A relatively simple special-purpose procedural language, which I devised for this task, can be used to define the rules and the patterns producing stem and affix allomorphs.

Creation of stem and affix morpheme lexicons (level-1) In contrast to the lexicon used by the morphological analyzer, the lexicons created by the linguist contain the descriptions of morphemes instead of allomorphs. Morphemes are defined by listing their lexical form, category and all unpredictable features and requirements. Irregular affixed forms and suppletive allomorphs can also be listed in the lexicon instead of using very restricted rules to produce them. I implemented a simple inheritance mechanism to facilitate the consistent treatment of complex lexical entries (primarily compounds). Such items inherit the properties of their final element by default.

Creation of a word grammar Restrictions on the internal morphological structure of words (including selectional restrictions between nonadjacent morphemes) are described by the word grammar. The development environment facilitates the creation of the word grammar automaton by providing a powerful macroing facility. Another option is to use regular expressions.

Creation of a suffix grammar (optional) An optional suffix grammar can be defined as a directed graph, and the development environment can produce segmented suffix sequences using this description and the suffix lexicon. Using such preprocessed segmented sequences enhances the performance of the analyzer.

As for the methodology to follow by the linguist to draw a borderline between the word forms or morphological constructions to be considered part of the language and those to be considered as erroneous, it is difficult to give a recipe. One obviously needs to take different

DOI:10.15774/PPKE.ITK.2015.013

approaches depending both on the own level of proficiency in the language and the intended application or task to be solved.

When creating a morphological description for Hungarian (see Section 5.1), my initial approach was rather normative, listening primarily to my own intuition and taking authoritative dictionaries into account, excluding many forms that I considered substandard or dialectal. This approach was to a significant extent motivated by the fact that creating a spell checker was among the primary goals of the development. Nevertheless, as I and my colleagues began to apply the morphology to solve practical tasks such as analysis and word form generation in a machine translation system or annotation of corpora, the description had to be adapted to cover linguistic data that lie outside my own idiolect. The way I did it was an extensive usage of the notion of markedness. In addition to standard forms, I allowed the system to recognize marked (rare, substandard, dialectal) word forms. When the analysis database is converted into a word form generation database to be used for machine translation output, marked forms are eliminated so that the system would not generate them. Extension of Hungarian paradigms with marked forms gained momentum especially when I worked on adapting the morphology to analyze Middle Hungarian texts (see Section 5.2.1). These texts contained, in addition to archaic morphological constructions, many dialectal forms that still exist and are even frequent outside the standard. Modern corpora, on the other hand, contain a significant amount of slang that also needs to be handled with special care. Attempts at analyzing texts containing domain-spacific terminology resulted in adding many foreign (especially English and Latin) words to the Hungarian database. These may result in odd analyses unless one handles them with special care.

When there is a lack of linguistic competence on behalf of the linguist creating the description, there is always a struggle with the data, because one can never be sure whether word forms present in the corpora are noise or data. Written grammars one tries to use often tend to be rather coarse, vague or anecdotal. Statements that seem to be generalizations are sometimes valid for just a handful of lexical items. Exceptions are hardly mentioned, or if they are, the lists are hardly ever exhaustive.

4.1.2 Conversion of the morphological database

Using a description that consists of the information described above, the development environment can produce a lexical representation (consisting of the level-2 lexicons) which already explicitly contains all the allomorphs of each morpheme along with all the properties and requirements of each of them. This representation still contains the formulae expressing properties and selectional restrictions in a human-readable form, and it can thus be easily checked by a linguist.

The readable redundant representation is then transformed to the format used by the analyzer using an encoding definition description, which defines how each of the features should be encoded for the analyzer. In the next section, I describe the formalism used in the source files for a morphological database in detail.

DOI:10.15774/PPKE.ITK.2015.013

4.2 Components of the framework

In order to describe the morphology of a language and produce a Humor analyzer for it, the following high level source files must be produced:

- **Morpheme lexicon files** for stems and suffixes: these describe the lexicon of the language
- **Rule files** for generating allomorphs and compute their properties: these describe the morphology and phonology of the language (as far as orthography reflects it)
- An **encoding definition file** that defines the translation of properties to the bit vectors and matrices used in the Humor morphological analyzer

4.2.1 Lexicon files

There are three levels of source files for each entry in the morphological lexicon. Level-1 morpheme lexicon files basically contain only lemmas/lexemes along with some unpredictable information concerning the entry. These files are where lexical data enters into the system. Level-2 lexicon files, on the other hand, are generated by the system from the level-1 files using the rule files. They already explicitly contain all the allomorphs of each lemma along with all the relevant properties of each allomorph. The third level of source files are Humor lexicon source files, which are generated from level-2 files using the encoding definition file.

4.2.1.1 Level-1 lexicon files

The format of the level-1 stem files and suffix files differs from each other. The main reason for this is the fact that there is unavoidably a very large (and not bounded) number of entries in the stem lexicon, while the suffixes are not too numerous (and form a closed class). Thus there is a strong motivation for brevity of description in the stem lexicon, while the entries in the suffix lexicon may contain much more explicit information. For this reason, the format of the level-1 stem lexicon file and that of the level-1 suffix lexicon file is different (level-2 and level-3 stem and suffix lexicon files, on the other hand, have a uniform format, since they contain all relevant information explicitly for each allomorph).

The suffix lexicon file First-level lexicon files have a line-oriented format: each entry is on a line of its own. The data structure defining each entry is flat: it consists of fields that contain an attribute name and a value for that attribute. The format of a field is: `attribute:value;`. Each field is ended by a semicolon. The following is fragment of the level-1 suffix lexicon file for Hungarian defining the plural suffix. This line contains 10 fields and a comment.

```
#plural
type:infl;lr:cat_Nom;rp:mcat_infl;tag:PL;phon:LVk;sfxalt:VLCL;
        lp:FVL,VZA,SVS,vST,UDEL;lp:PL;mcat:PL;humor:PL;#-s
```

Attribute	Value	Interpretation
`type`	`infl`	The type of the suffix is inflection.
	`deriv`	The suffix is a derivational suffix.
`lr`		The left requirements of the morpheme.
`lp`		The left properties of the morpheme.
`rr`		The right requirements of the morpheme.
`rp`		The right properties of the morpheme.
`glr`		Requirements of the morpheme that must be satisfied by the global properties (`gp`) of some (not necessarily adjacent) morpheme to the left.
`gp`		Global properties of the morpheme.
`phon`		The phonological form of the morpheme.
`humor`		The traditional tag (used in previous versions of the Hungarian Humor analyzer) for this morpheme that can be output by the analyzer.
`tag`		Another set of (category) tags that can be output by the analyzer.
`sfxalt`		The suffix alternation the suffix participates in.
`mcat`		Morphological category.

Table 4.1: The fields used in the Hungarian suffix lexicon file

```
#Nominal inflections
# stem -- (number) -- Px -- Cx
#tag phon  rp  lp  lr    rr   mcat   #gloss
DU   qe9n      Du        !Px  Num    dual
PL   E6t       Pl        !Px  Num    plural

#tag phon  rp  lp  lr    rr   mcat   #gloss
NOM            Cx Nom         Cx     nominative case
LAT  a         Cx Lat         Cx     lative case
LOC  N         Cx Loc         Cx     locative case
LOC  na        Cx Loc na      Cx     locative case
```

Figure 4.3: Fragment of the tabular source of the Synya Khanty suffix lexicon

Table 4.1 explains the fields used in the Hungarian suffix lexicon file. However, most of this information is not specific to the Hungarian description with the exception of the usage of the `humor` feature. See also Sections 5.1 and 5.3 for further examples and explanation of the features and properties used.

Since the level-1 suffix lexicon format is difficult to read, suffix lexicons are defined in a tabular format, in which feature names are taken from table headers and values are taken from non-header rows. Comments may intervene. The example fragment in Figure 4.3 is from the Synya Khanty analyzer.

The stem lexicon file The main difference between the format of the suffix and stem lexicon files is that in the latter obligatory information (lemma and category of stem) must be given as the first field in the form `lemma[category];` without specifying the attribute names. When the level-1 stem lexicon file is interpreted by the program that translates it to a level-2 lexicon file, the lemma is assigned to the attribute `seg` (”segmented form”) and

```
Ithaca[FN];phon:itaka;
maca[FN];
  cica+maca[FN];
paca[FN];
  tinta+paca[FN];
Rábca[FN];
cca.[HA|ROV];equ:circa;phon:cirka;
...
veréb[FN]; zarte:e;stemalt:SVS;rp:=A;
  fi=a+veréb[FN]; zarte:e;
  nád@i+veréb[FN]; zarte:e;
Zala+cséb[FN];isa:település;
Özséb[FN];rp:=jA;isa:ffinév;
egyéb[MN|NM&FN|NM]; zarte:ë;stemalt:SVS;rp:=A;rp:ESS_no;
mi+egyéb[FN|NM]; zarte:ë;stemalt:SVS;rp:=A;
Gotlíb[FN];isa:ffinév;
lio#fób[MN];rp:=jA;rp:ESS_no;
hidro#fób[MN];rp:=jA;rp:ESS_no;
Jób[FN];isa:ffinév;
gardrób[FN];rp:=jA;
göb[FN];rp:=jA;
köb[FN];rp:=A;
küszöb[FN];rp:=A&=jA;
  inger+küszöb[FN]; zarte:ë;
  tudat+küszöb[FN];
  érz%et+küszöb[FN]; zarte:e;
...
ér[FN];stemalt:SVS;rp:=A;
...
facér[MN];rp:=jA;rp:VHB;rp:ESS_Vn LOW+;
kacér[MN];rp:=jA;rp:VHB;rp:ESS_Vn LOW+;
...
  arany+ér[FN];
```

Figure 4.4: A fragment of the Hungarian level-1 stem lexicon

the category is assigned the attribute `humor`. Optional fields, on the other hand, must be given as `attribute:value;` pairs, like in the suffix lexicon file. The same attribute name may appear more than once on a line in the stem lexicon file as well, and the values will be concatenated.

Figure 4.4 above shows a sample from the Hungarian level-1 stem lexicon file. Polymorphemic entries are segmented[i], and each lemma is followed by its category. The segmentation of lemmas plays an important role in determining the suffixation and allomorphy properties of the entries. Compound stems generally behave like the last compound member does (e.g. the compound *arany+ér* behaves like its last compound member *ér* (e.g. its plural is *aranyerek*), while *facér* (which is not a compound) behaves totally differently (regularly) both concerning allomorphy and vowel harmony (e.g. its plural is *facérok* or *facérak*)). Moreover, stems containing a particular final derivational suffix often behave differently from those ending in a similar sequence of phonemes but not containing that particular suffix. Examples include both productive and unproductive suffixes: verbs with a final (unproductive) -*Ad* suffix take the past -*t* suffix without a linking vowel in contrast to all other -*d* final verbs; nouns ending

[i] `+`: compound, `@`,`%`: derivational, `=`: inflectional, `#`: foreign morpheme boundary

DOI:10.15774/PPKE.ITK.2015.013

in the productive *-sÁg* suffix take the 3rd person and plural possessive markers in the *j*-less form (*-A*, *-Ai*). The rules in the stem allomorph generation rule file must thus rely on the segmentation given in the level-1 stem lexicon. Compounds inherit their suffixation properties from their last compound member. This is accomplished by a process that overwrites the properties of each compound by the properties of its last member before the level-2 stem lexicon is generated from the level-1 source file. This means that, for example, in spite of the fact that in the original Humor source *ingerküszöb* and *tudatküszöb* were suffixed differently (because they had a representation equivalent to the one below), they behave identically to *küszöb* in the current version.

```
küszöb[FN];rp:=jA&=A&=Ai&=jAi;
inger+küszöb[FN];rp:=jA&=jAi;
tudat+küszöb[FN];rp:=A&=Ai;
```

The order of listing of stems is important, because the inheritance mechanism relies on it: the stems in the level-1 stem lexicon file must be in reverse-alphabetic order (see the longer sample above). This ensures that the last member of a compound always precedes the compound itself, which makes the inheritance easy to implement. The algorithm also relies on the fact that no entries having a different ending may appear between the entry of the compound and that of its last member. The source files need not contain the stem morphemes in a sorted manner. The grammar development framework does the sorting automatically.

The category (i.e. the `humor` feature), and the following features may have list values: `phon`, `equ` and the property/requirement features `rp/lp/rr/lr/gp/glr`. The elements of the list must be separated by &'s. Spaces or commas may not be used here (except in the property/requirement features). In the case of the latter, this is really a conjunctive logical formula which is normally inherited by all the allomorphs of the lexeme. In the case of the `humor`, `phon` and `equ` features, however, the list notation is used differently: it is really a simple labor-saving and source file consistency enhancement device. If some of these features have a list value, the lexeme is split into as many independent lexemes as there are different values in each list, and all the possible combinations (i.e. the Cartesian product) are generated, so if e.g. humor is a list of length 2 (e.g. `FN&MN`) and `phon` is a list of length 2 (e.g. `süket&siket`), then 4 entries are generated: (`"süket"[FN], "süket"[MN], "siket"[FN], "siket"[MN]`). If this is not what one wants, one can list the combinations you want as independent entries in the lexicon file (with the rest of the features having identical values).

4.2.1.2 Level-2 lexicon files

Level-2 lexicons are generated by the system from the level-1 lexicons using the rule files. They already explicitly contain all the allomorphs of each lemma along with all the relevant properties of each allomorph. Each entry in the level-2 lexicon takes the form of a perl hash data structure declaration statement. Perl hash data structures straightforwardly implement the kind of attribute–value structures we need to properly describe all the linguistic information the morphological analyzer needs concerning each lemma and its allomorphs.

Each entry in the level-2 lexicon is a variable declaration statement of the form:

```
\$mrf = { 'attr1' => 'val1',
  'list_attr' => ['lval1', 'lval2'],
  'struc_attr' => {'attr2' => 'val2'}
};
```

The effect of this statement is that the variable called `$mrf` is assigned a pointer to the structure defined within the braces. The structure contains a comma separated list of attribute–value pairs. Attribute names appear between quotes and are separated from the value by an arrow =>.

The value of each attribute may be one of the following three kinds:

- Atomic string value: `'attribute_name' => 'string_value'`,
- List value: `'attribute_name' => [comma_separated_list_of_values]`,
- Embedded hash value: `'attribute_name' =>`
  `{comma_separated_list_of attribute_value_pairs}`,

In Figure 4.5 (which is the level-2 representation of the verb *fut* 'run' from the level-2 stem lexicon), the attribute `'cat'` has an atomic string value (`'V'`), the attribute `'allomfs'` has a list value, and each member of this list is in turn an embedded hash structure, which represents an allomorph of the verb stem. Entries are ended by a semicolon.

```
$mrf = {
 'phon' => 'fut',
 'humor' => 'IGE',
 'cat' => 'V',
 'seg' => 'fut',
 'root' => 'fut',
 'allomfs' => [
   {
     'allomf' => 'fut',
     'gp' => 'cat_V',
     'rr' => undef,
     'lr' => '!cat_vrb',
     'lp' => 'comp2 Cini',
     'rp' => '=Ott =OttAm =sz =nAk =tOk =lAk =jUk reg -O3 =tAt =OgAt cat_V mcat_stem
             inflable VHB A=1 B=1'
   },
   {
     'allomf' => 'fus',
     'gp' => 'cat_V',
     'rr' => undef,
     'lr' => '!cat_vrb',
     'lp' => 'comp2 Cini',
     'rp' => '=d =s cat_V mcat_stem inflable VHB A=1 B=1'
   }
 ],
};
```

Figure 4.5: The level-2 entry of the verb *fut* 'run'

Most of the top-level attributes in the level-2 lexical entries are directly inherited from the level-1 descriptions. The most important exception is the `allomfs` (allomorphs) attribute, the list value of which is generated using the rules given in the rule files that describe stem

DOI:10.15774/PPKE.ITK.2015.013

and suffix alternations and declare how the properties of each allomorph must be calculated. Table 4.2 shows the top-level attributes appearing in level-2 lexicon entries.

Attribute	Value	Interpretation
`seg`		Segmented base form of the stem (inherited from the 1st lemma field of the level-1 stem source)
`humor`		The traditional tag (used in previous versions of the Hungarian Humor analyzer) for this morpheme that can be output by the analyzer (inherited from the 2^{nd} category field of the level-1 stem lexicon file). This must contain as many `+`'s as the `phon/seg` features.
`tag`		Another set of (category) tags that can be output by the analyzer (inherited from the level-1 suffix lexicon file)
`cat`		Head category code of the morpheme for stems (including derivational suffixes). For stems, the value of this attribute is converted from the humor code, for derivational suffixes, it is calculated from the `mcat` value given in the level-1 lexicon. Possible values are:
	`N`	Noun
	`Adj`	Adjective
	`V`	Verb
	`Adv`	Adverb or postposition[ii]
	`Num`	Numeral
	`X`	Other (non-inflected)
	`Vpfx`	Verbal prefix
`stemalt`		Unproductive stem alternation the stem participates in (inherited form the level-1 stem source). See the list of stem alternation codes used in the Hungarian description in Table 5.2.
`phon`		Unpredictable pronunciation (usually of foreign words and names, inherited form the level-1 stem source, in standard orthography).
`equ`		What an abbreviation stands for (inherited form the level-1 stem source).
`allomfs`		The list value of this feature contains hash structures that represent allomorphs of the morpheme (or morpheme sequence) represented by the whole level-2 entry.

Table 4.2: Top-level attributes used in the level-2 lexicon files

Each hash structure in the `allomfs` list represents an allomorph of the morpheme (or morpheme sequence) represented by the whole level-2 entry. The values of the property and requirement attributes are similar to those of the respective fields in the level-1 suffix files (see the description of the fields `rp, lp, rr, lr, gp` and `glr` in Table 4.1 above). The main difference between the property and requirement attributes in level-1 and those in level-2 lexicons is that while the former contain mostly unpredictable properties and requirements only, the latter explicitly contain every relevant property and requirement of the morph that

[ii] Postpositions, even when they fuse with pronouns using special inflected forms, also have the `Adv` category in the morphology. The inflected forms, such as *(én+)mögött+em* 'behind me', *(te+)mögött+ed* 'behind you', *(ő+)mögött+e* 'behind him/her/(it)', etc., are generated off-line by the rule component.

is needed to describe its distribution. Some of these properties and requirements are directly inherited from the level-1 description, but most of them are generated using the rule files.

Although level-2 lexicons can also be saved in the format shown in Figure 4.5 for easy readability, they are generally stored in a much more compact format shown in Figure 4.6.

```
r,-O3&=OgAt&=Ott&=OttAm&=jUk&=lAk&=nAk&=sz&=tAt&=tOk&A=1&B=1&VHB&cat_V&inflable&mcat_stem&reg;
  l,Cini&comp2,!cat_vrb;fut;fut;fut;IGE;fut;fut;;S_IGE;fut[IGE];fut;fut;;;11‘
r,=d&=s&A=1&B=1&VHB&cat_V&inflable&mcat_stem;
  l,Cini&comp2,!cat_vrb;fus;fus;fut;IGE;fut;fus;;S_IGE;fut[IGE];fut;fut;;;11‘
```

Figure 4.6: The level-2 entry of the verb *fut* ‘run’ in a compact format

4.2.2 Rules

Rule files govern the translation of level-1 lexicon files to level-2 lexicon files. There are two such rule files: one of them describes stem allomorphies and contains rules for calculating stem properties, while the other one describes (stem conditioned) suffix allomorphies. Rule files are procedurally interpreted programs (they are in fact translated to perl code), and thus the order of rules is relevant.

Figure 4.7 shows an example of the rule formalism used to infer properties of morphemes and restrictions they impose on their neighbors and generate allomorphs and set their properties and restrictions. The rule in this example generates allomorphs of *a/e*-final Hungarian nouns, adjectives and numerals. The final vowel of such words is lengthened when one of a group of suffixes is attached to them. Suffixes that trigger lengthening have the property FVL (final vowel lengthening). Others do not have this property. The rule checks that the pronounciation of the word is also *a/e*-final (otherwise the rule does not apply). Then it generates an allomorph that is identical to the lemma and another one with the final vowel lengthened. The lemma-identical allomorph constrains morphs on their right not to have the FVL property. The lengthened allomorphs, on the other hand, require them to have that property.

```
#final vowel lengthening:           #o/ö-final lengthening
#kutya -> kutyá, eke ->eké          #Oslo -> Osló
  root:/[ae]\+*$/&&phon:/[ae]\+*$/    root:/[oö]$/
  +;!FVL;;                            +;(Omrf|comp2);;
  +/a(?=\+*$)/á/;FVL;;                +/o$/ó/;!Omrf !comp2;;
  +/e(?=\+*$)/é/;FVL;;                +/ö$/õ/;!Omrf !comp2;;
```

Figure 4.7: Fragment of the Hungarian rule grammar: a rule generating allomorphs of Hungarian final vowel lengthening stems and those of the orthographically similarly behaving *o/ö*-final stems.

4.2.2.1 The format of the rule files

The most important operations that can be done in the rule file (Appendix A.1 contains some examples for each operation):

DOI:10.15774/PPKE.ITK.2015.013

- Comments: Anything following a hash mark (#) is a comment, unless the # is part of a string (i.e. it is between single or double quotes) or a regular expression (i.e. it is between slashes //).
- Variable declaration and manipulation
  - Declaring named scalar variables: named scalar variables can be used as named constant character class shorthands in regular expressions (e.g. `$C` for consonant, `$V` for short vowel etc.)
  - Declaring named list variables: named list variables can be used for generating inflected forms of closed-class items like e.g. case-marked personal pronouns etc.
  - Declaring local attribute names: the attributes `rp` and `rr` are always handled by the program as local variables. Additionally, you can declare other attributes to be local.
- Attribute manipulation instructions:
  - The attribute manipulation instruction is most frequently used to manipulate properties and requirements of morphemes (if used outside of an allomorph manipulation block; this affects every allomorph) or individual allomorphs (if used within an allomorph manipulation block).
  - Manipulation of the values of other morpheme-level attributes.
- Blocks of statements: parts of the rule file can be enclosed within blocks. Each block has a modifier at the beginning. One type of block modifier is a condition that must be true in order for the block to be executed *(conditional blocks).* Another type of modifier indicates that the attribute manipulation instructions within the block are to be applied to individual allomorphs instead of the morpheme level attributes *(allomorph list manipulation blocks).* The third type of block is used to split allomorphs that have certain properties *(allomorph duplication blocks).* Allomorph generation blocks, allomorph list manipulation blocks and allomorph duplication blocks can all be referred to as allomorph manipulation blocks. In these contexts, the attribute manipulation instruction affects the properties and requirements of individual allomorphs instead of those of the whole morpheme.
  - Conditional blocks: conditional blocks are used to delimit parts of the rule file that should only apply if certain conditions are satisfied by the morpheme being processed.
  - Allomorph generation blocks: generating allomorphs and setting their properties. They are used to create the allomorphs belonging to the morpheme which is being processed and define their properties and requirements.
  - Allomorph list manipulation blocks: manipulating properties and requirements of individual allomorphs. If an attribute manipulation instruction appears in an allomorph list manipulation block, then the changes to requirements and properties affect the individual allomorph.
  - Allomorph duplication blocks: it is possible to handle certain properties as “disjunctive” or “underspecified” and split allomorphs containing them into independent allomorphs having the same form but different properties and requirements. This duplication of allomorphs is needed in the case of properties which are intended to be bit encoded in the level-3 (Humor) lexicon representation, since the bit vector

representation of properties is inherently conjunctive. This construction can be used to handle vacillating behavior: e.g. vacillating vowel harmony or the appearance of suffix-initial vowels (especially in the case of verbs). This construct can also be used to handle generic orthographic alternations such as the context-sensitive representation of palatal consonants in Russian-style Cyrillic-based orthographies, e.g. the ones used for Uralic languages spoken in Russia.

The format of the level-2 suffix rule file is the same as that of the level-2 stem rule file, so the description above applies to the suffix rule file as well. There is only one important difference between the stem and the suffix rule file. Since suffixes have their interface to the world on their left side, in the suffix rule file, the default attribute to check in conditionals is `lp` (in contrast to the case in the suffix rule file, where it is `rp`), the default attribute to set in field 2 is `lr` (not `rr`) and the default attribute to set in field 3 is `lp` (not `rp`).

4.2.2.2 Sequences of inflectional suffixes

Possible sequences of inflectional suffixes can be generated off-line during the generation of the level-2 suffix lexicon file in order to speed up the morphological analyzer. This also makes the word grammar automaton somewhat simpler.

In the Hungarian morphological description, harmonic variants of suffixes are generated by a Kimmo-style two-level finite-state transducer which is complemented by some regular-expression-based substitution expressions that make it able to handle additional regular suffix alternations such as suffix-initial vowel lowering. It also implements the most productive stem alternation: final low vowel lengthening, which all low-vowel-final suffixes also undergo if a lengthening suffix is attached to them. This mechanism for describing lengthening and lowering is needed to handle suffix sequences.

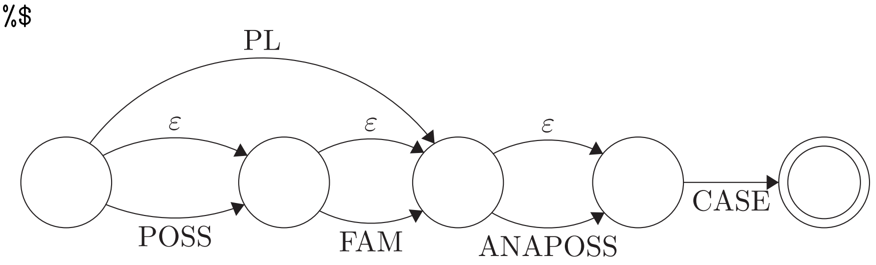
In addition to the mechanism that generates the surface form of suffix sequences, a description is also needed of what morpheme sequences are to be generated. The description I use is a finite-state automaton (in fact an arc-labeled graph) that describes the set of possible sequences.

The code in Figure 4.8 creates a representation of a finite-state automaton which describes possible Hungarian nominal inflectional suffix sequences. Although the most common representation of a finite state automaton (or a transducer) is a state table, here we use another representation: the (directed) graph of the machine is directly encoded in the data structure.

In the first statement, we create a tree subgraph of the whole graph. A tree is a connected graph that does not contain any nodes into which two different arcs run. In this form of declaration, the representaion of each subtree of the (arc labeled) tree is enclosed by a pair of braces { }. The braces enclose a list of the labels of all the arcs running from the root of the subtree.

The tree that the first statement declares is utterly simple: it contains 5 nodes and a single path from the start node to the only leaf, consisting of 4 arcs labeled: `'','',''` and

```
$sfxfsa->{’’}{’’}{’’}{’mcat:CASE’} = {};
$sfxfsa->{’’}{’’}{’mcat:ANAPOSS’} = $sfxfsa->{’’}{’’}{’’};
$sfxfsa->{’’}{’mcat:FAM’} = $sfxfsa->{’’}{’’};
$sfxfsa->{’mcat:PL’} = $sfxfsa->{’’}{’’};
$sfxfsa->{’mcat:POSS’} = $sfxfsa->{’’};
%$
```



PL	the plural suffix *-Vk* and the plural of possessives with plural possessors *-((j)A)i[nk/tOk/k]*
POSS	possessive suffixes *(-Vm, -Vd, -(j)A, -Unk, -VtOk, -(j)Uk)* including the plural of possessives with singular possessors *-((j)A)i(m/d)*
FAM	the familiar plural suffix *-ék*
ANAPOSS	the anaphoric possessive suffix *-é* and its plural form *-éi*
CASE	case suffixes (not including *-képp(en)*)

Figure 4.8: A sample suffix grammar describing Hungarian nominal inflectional suffix sequences

`’mcat:CASE’`. The whole graph is referred to by the variable `$sfxfsa`. Note that the arcs labeled `’’` are in fact ε transitions.

The next 4 statements add new paths to the graph that will thus not be a tree any longer. The right side of the assignment in the 2^{nd} statement, for example, `$sfxfsa->{’’}{’’}{’’}`, refers to the subtree that is reached on the path `’’,’’,’’` from the root of the tree referred to by the variable `$sfxfsa`. The effect of whole statement is that the graph will contain a new path `’’,’’,’mcat:ANAPOSS’` (this actually means that a single new outgoing arc labeled `’mcat:ANAPOSS’` is added to the (already existing) node ending the path `’’,’’`) and this path will lead to the same node as the (already existing) path `’’,’’,’’` does. The remaining 3 statements add three more arcs that run into already existing nodes.

Note that since the arcs running from a node are represented as hash keys, it is not possible to have two identically labeled arcs running from the same node, i.e. one can only represent deterministic finite-state automata using this notation.

Note also that the given representation does not explicitly differentiate final and non-final nodes. I simply assume here that leaves are final and all the rest are non-final.

The level-2 suffix lexicon file is produced from the level-1 suffix lexicon by the following procedure:

- The automaton representing possible sequences is read.
- The language of the automaton (the set of all sequences of category labels (`’mcat:CASE;’` etc.) accepted by the automaton) is generated (it is a finite set, because there are no loops).

DOI:10.15774/PPKE.ITK.2015.013

- The level-1 suffix lexicon is read and each entry (morpheme) in it is assigned the set of category labels that match it.
- For each sequence of category labels, every matching morpheme is substituted for each category label, and thus sequences of morphemes are generated, which inherit their properties from the morphemes in the sequence:
  - left-hand-side properties and requirements of the sequence are inherited from the leftmost morpheme in the sequence
  - right-hand-side properties and requirements of the sequence are inherited from the rightmost morpheme in the sequence and
  - global properties and requirements are inherited from all morphemes in the sequence (except for requirements satisfied within the sequence)
- The rules in the suffix rule file are applied to each sequence of suffixes (this generates allomorphs of the first suffix in the sequence), and the vowel harmony transducer is applied to the result (which again may multiply the allomorphs). The suffix allomorphs acquired this far are all independent entries.
- The rules in the stem rule file are applied to each sequence of suffixes (this generates allomorphs of the last suffix in the sequence). The suffix allomorphs generated in this step become members of the `allomfs` attribute within the same entry.

4.2.3 The encoding definition file

The encoding definition describes how the level-2 lexicon files should be translated to level-3 lexicon files and matrix definition files. The file also contains information that describes how the continuation matrix for each morph is selected (the $matrixsel= part) as well as definition of the category labels used in the word grammar (the $metacteg= part).

4.2.3.1 The representation of properties

Allomorph properties are defined as a set of propositional formulae. Each morpheme (allomorph) may have a formula encoding its *properties* visible from the left and right side along with the *requirements* that must be satisfied by any morpheme on its left/right. The formula expressing its right-hand-side properties (`rp`) must be defined for every morpheme because that formula contains the categorial information used in the word grammar, as well as the properties used for continuation matrix selection.

The formulae describing properties are composed of the conjunction of (optionally negated) properties. An example is the left-hand-side property formula of the Hungarian accusative suffix allomorph *–ot*, which is the following:

```
'lp' => 'FVL VZA SVS vST UDEL ACC Vini'
```

The formula states that this suffix triggers final low vowel lengthening (`FVL`), vowel–zero alternation (`VZA`), stem vowel shortening (`SVS`), v-insertion (`vST`), Ú-deletion (`UDEL`), is an accusative suffix (`ACC`) and is a vowel-initial morph (`Vini`). Ampersands (`&`) can also be used in place of the spaces to denote conjunction.

DOI:10.15774/PPKE.ITK.2015.013

Property-denoting formulae (**P-expressions**) have the following interpretation: the properties appearing non-negated in the right/left property formula are true for the morpheme when looked at from the right/left side. The properties not appearing in it are not true unless entailed by some of the properties that do appear. Explicitly negated properties are not true.

Formulae expressing requirements (**R-expressions**) may contain the conjunction and the disjunction of optionally parenthesized expressions containing possibly negated properties. An example is the left requirements formula of the Hungarian accusative suffix allomorph *–ot*, which is the following:

```
'lr' => '=Vt !LOW VHB cat_Nom'
```

The formula states that this suffix requires a non-lowering (`!LOW`), back harmonic (`VHB`) nominal stem (`cat_Nom`) that must select a vowel-initial form of the accusative suffix (`=Vt`). Ampersands (`&`) can also be used in place of the spaces to denote conjunction.

A morph may only be followed by another morph if its right-hand-side properties satisfy the left-hand-side requirements of the following morpheme and the left-hand-side properties of the latter satisfy the right-hand-side requirements of the former. If an atomic property appears in the requirements expression of a morph, it must be true for the other morph; if it appears negated, it must be false; and if it does not appear, then it is irrelevant for the match.

4.2.3.2 The translation of P- and R-expressions to the representation used by the analyzer

An atomic property has a different interpretation depending on whether it appears in a P-expression or an R-expression.

As follows from the interpretation of P- and R-expressions that we have seen above, the rules for setting bit-encoded properties are the following:

In property list:

property present → use the set operation (the property is true);
property missing → use the neg operation (the property is false);
property negated → use the neg operation (the property is false);

In requirements expression:

property present → use the set operation (the property is checked to be true);
property missing → use the ignore operation (the property is ignored);
property negated → use the neg operation (the property is checked to be false);

In the case of bit-encoded properties, the encoding also depends on whether the property is a right-hand-side property or a left-hand-side property. The reason for this is that in Humor, a mask can be defined for left-hand-side bitvectors (using the . character in the bit positions masked out) while no such mask can be defined for right-hand-side bitvectors (character . is equivalent to 0 on the right-hand-side). This means, in effect, that bit-encoded left-hand-side requirements are silently encoded using twice as many bits (data bits+mask) as right-hand-side requirements (data bits only). And from this, it follows that bit-encoded right-hand-side requirements (=left properties) must be encoded using twice as many data bit positions as needed for left-hand-side requirements (=right properties) so that the masking can also be provided for. We can thus use for example the following encoding for the operations:

for right-hand side (e.g. stem) properties (i.e. left-hand side requirements)

```
set=1
neg=0
ignore=.
```

for left-hand side (e.g. suffix) properties (i.e. right-hand side requirements)

```
set=.1
neg=1.
ignore=11
```

In practice, the encoding of properties using bit vectors depends on a number of factors, so the encoding above cannot always be used. These factors are the following:

- Binary vs. non-binary properties: although the simple propositional representation renders all properties binary in the sense that they are either true or false for every morpheme, the domain of description (the morphology of the language we are seeking to describe) is such that the truth of some of the properties implies that some other properties are not true. E.g. if `cat_v` is true for a morpheme, `cat_n`, `cat_adj` and `cat_adv` are not true. This is because `cat_v`, `cat_n`, `cat_adj` and `cat_adv` are in fact different possible values of the same feature: that of `cat`[iii]. Properties that are mutually exclusive possible values of the same feature (like the `cat_xxx` properties above) will be termed *non-binary properties*, and the rest (i.e. properties which would be the values of binary features) will be called *binary properties*.
- x-properties: binary properties can be encoded using the scheme in the previous section using the `set`, `neg` and `ignore` operations, which are implemented in a property-side dependent fashion, as we have seen above. In such a case, only the position of the relevant 1 or 2 (adjacent) bits must be specified in the bit vector when defining the property, and this is done by placing the symbol `x` in those positions. For this reason, I will use the term x-property for binary properties encoded in this fashion. The following examples show the definition of a right-hand-side and a left-hand-side x-property:

  ```
  ’take_clit’ => [’r’,’.....x’,’at_end’],
  ’stmalt_E’ => [’l’,’......xx’,1],
  ```

  There is a shorthand notation for the dots at the beginning: ’5>x’ can be written instead of `’.....x’`:

[iii] Homonyms are represented as distinct lexical entries.

```
'take_clit' => ['r','5>x','at_end'],
'stmalt_E' => ['l','6>xx',1],
```

The first parameter in the list defines whether it is a right-hand side or a left-hand side property. The second parameter is a bit mask that defines the position(s) used for the encoding of the property. (For left-hand-side properties, the positions marked by the `xx` must be adjacent.) The third parameter defines the domain of the property: it specifies a formula that selects the morphemes for which the property is defined. Whenever the properties and the requirements of a morpheme satisfy the formula that defines the domain of the property, the bit(s) for the property must be set using the appropriate operation (`set`, `neg` or `ignore`). In fact, for right-hand-side x-properties (having a single `x`), a correct encoding is always produced even if the domain of the property is not defined. However, this is not the case for left-hand side x-properties (having `xx`). For them, the domain must always be given. Note that properties with disjunct domains (e.g. a property pertaining only to verbs and another that is appropriate only to nominal stems) may be encoded using the same bit positions.

- Non-binary properties: the bit-encoding of non-binary properties is more complicated than that of the binary ones since the mutual exclusiveness that characterizes such properties must be provided for. There are also cases when morphemes having some of the mutually exclusive possible values of a feature have some properties in common or share some aspect of behavior. E.g. stems and derivational suffixes have the common property that they determine the syntactic category of the word; prefixes and stems have the common property that they may appear at the beginning of a word etc. A possible and the supported way of encoding non-binary properties is to decompose them into the conjunction of binary properties. Such decomposed non-binary properties are called complex properties. They entail their definition and thus the conjunctive formula that defines them is added to the formulae in which they appear unnegated. Note, however, that the negation of a complex non-binary property has a disjunctive (or a De Morgan-equivalent non-atom-negated) entailment that cannot be bit-encoded. For this reason, complex properties may not appear negated in formulae. Their encoding differs from that of x-properties also in that they are simply ignored without doing any bit operations if they do not appear in a formula. Figure 4.9 shows the definition of some complex properties using atomic ones.

```
'mcat_deriv' =>     ['r','','','sfx'], # derivational suffix
'mcat_stem' =>      ['r','','','!sfx'], # stem
'mcat_infl' =>      ['r','','','sfx&!inflable'], # inflectional suffix
'mcat_stem+infl' => ['r','','','!sfx&!inflable&!pfx'], # stem with inflectional suffix
'mcat_pfx' =>       ['r','','','!sfx&!inflable&pfx'], # prefix
'cat_Nom'  => ['r','','','inflable&!cat_vrb'],
'cat_N'    => ['r','','','inflable&!cat_vrb&!cat_num&cat_Nom'],
'cat_Adj'  => ['r','','','inflable&!cat_vrb&!cat_num&cat_Nom'],
'cat_Part' => ['r','','','inflable&!cat_vrb&!cat_num&cat_Nom'],
'cat_Num'  => ['r','','','inflable&!cat_vrb&cat_num&cat_Nom'],
'cat_V'    => ['r','','','inflable&cat_vrb'],
'cat_Adv'  => ['r','','','!inflable'],
'vpfx'     => ['r','4>x','','!sfx&!inflable&pfx'], # verbal prefix
'suppfx'   => ['r','5>x','','!sfx&!inflable&pfx'], # the superlative prefix
```

Figure 4.9: The definition of some complex properties using atomic ones

- Binary properties with entailments: properties may set their own bits in addition to having entailments (which may set other bits, see e.g. the `vpfx` property in Figure 4.9).

These properties may appear negated (in requirements or in entailments) and in such a case only the single bit (or, in the case of left-hand-side properties, only the single pair of bits) specified in the position field is negated while the entailments are ignored. Other examples include cases when a feature may have two complementary values both having a name.

- Manually encoded properties: it is also possible to manually define the binary encoding of binary and non-binary properties by using a bit mask that contains 1’s, 0’s and dots (or number and >) instead of using decomposition (entailments) and binary x-properties. As using this feature may be a potential source of errors and inconsistencies, the preferred way of handling complex properties is using decomposition. However, this notation can sometimes be useful, especially when the mutually exclusive properties cannot be decomposed in a meaningful or economical way.
- Automatic property range: the preferred method to handle cases when the mutually exclusive properties cannot be decomposed is to use a range of bits to represent the mutually exclusive possible values and have the system generate a unique pattern for each possible value, see Figure 4.10.

```
’wcat_Nom_stem’ =>    [’r’,’1>$5wcat’,’’],#nominal stem
’wcat_Nom_stem_infl’=>[’r’,’1>$5wcat’,’’],#nominal stem with inflection
’wcat_PP_stem’ =>     [’r’,’1>$5wcat’,’’],#locative postposition stem
’wcat_PP1_stem’ =>    [’r’,’1>$5wcat’,’’],#postposition stem
’wcat_V_stem’ =>      [’r’,’1>$5wcat’,’’],#verb stem
’wcat_V_stem_infl’ => [’r’,’1>$5wcat’,’’],#verb stem with inflection
’wcat_uninfl_stem’ => [’r’,’1>$5wcat’,’’],#not inflectable stem
’wcat_Nom_deriv’ =>   [’r’,’1>$5wcat’,’’],#nominal deriv suffix
’wcat_V_deriv’ =>     [’r’,’1>$5wcat’,’’],#verbal deriv suffix
’wcat_infl’ =>        [’r’,’1>$5wcat’,’’],#inflection
’wcat_Cx’ =>          [’r’,’1>$5wcat’,’’],#Cx suffix (may follow PP)
’wcat_Px’ =>          [’r’,’1>$5wcat’,’’],#Px suffix (may follow PP and PP1)
’wcat_clit’ =>        [’r’,’1>$5wcat’,’’],#clitic
’wcat_dot’ =>         [’r’,’1>$5wcat’,’’],#is a dot
’wcat_hyph’ =>        [’r’,’1>$5wcat’,’’],#hyphen
’wcat_VMod’ =>        [’r’,’1>$5wcat’,’’],#verbal prefix
’wcat_Adv_deriv’ =>   [’r’,’1>$5wcat’,’’],#adverbial deriv suffix
’wcat_ImpVx’ =>       [’r’,’1>$5wcat’,’’],#imperative verbal suffix
’wcat_Inf’ =>         [’r’,’1>$5wcat’,’’],#infinitive suffix
’wcat_Passive’ =>     [’r’,’1>$5wcat’,’’],#passive verbal suffix
’wcat_Tense’ =>       [’r’,’1>$5wcat’,’’],#tense suffix
’wcat_Vx’ =>          [’r’,’1>$5wcat’,’’],#verbal agreement suffix
’wcat_TenseVx’ =>     [’r’,’1>$5wcat’,’’],#verbal Tense+Vx(sg3) suffix
```

Figure 4.10: The definition of mutually exclusive properties using a 5-bit automatic range from the encoding definition of the Synya Khanty analyzer

4.2.3.3 The encoding of matrix properties

The properties not defined as bit-encoded properties and the requirements concerning them will be represented by the continuation matrices. The matrices directly encode the already stated matching mechanism according to which a morpheme may only be followed by another morpheme if its right-hand-side (matrix-encoded) properties satisfy the left-hand-side (matrix-encoded) requirements of the following morpheme and the left-hand-side (matrix-encoded) properties of the latter satisfy the right-hand-side (matrix-encoded) requirements of the former.

DOI:10.15774/PPKE.ITK.2015.013

The selection of the continuation matrix for a morpheme is determined by a subset of its right-hand-side bit-encoded properties (i.e. requirements may not be used). The expressions may not contain disjunction (like any bit-encoded expressions) but they must be disjunct (neither of them may entail any other) so that a unique matrix can be selected for every morpheme. The formulae determining matrix selection are defined as given in the following example from the Spanish Humor morphology:

```
\$matrixsel={
'cat_v&thm_a'=>'va',
'cat_v&thm_e'=>'ve',
'cat_v&thm_i'=>'vi',
'have_cat&cat_nom'=>'nom',
'!have_cat'=>'rest',
'cat_adv'=>'rest',
};
```

The matrix selection definition above states that each class of verbal roots with either of the theme vowels *a, e* and *i* have their own matrix, nominal (noun and adjective) stems have another, and all the rest are poured into the same matrix called `rest`.

Note that although disjunction may not be used in the expressions, the same matrix (e.g. the one called `rest` in the example above) may be selected by more than one expression: this is the standard way of resolving the ban on disjunction for bit-encoded properties.

The Humor 'meta-matrix' file and the part of the layout file that describes the matrices is generated using the matrix-selection definition above. The number of bits used for representing matrix codes (8 or 16) in the analyzer is also automatically determined by the program.

In order to be able to generate the matrices, it is necessary that for each ⟨left/right Side, Properties, Requirements⟩ triple (SPR-triple) that occurs in the allomorph-database, the set of matrices which are affected by the given SPR-triple be identified. This is done by a procedure that returns the list of matrices from `$matrixsel` the property-list of which (a) is satisfied by the right-hand-side properties in the SPR if Side is '`right`'; (b) satisfies the left-hand-side requirements in the SPR if Side is '`left`'. In case (a), the list must contain exactly one matrix (unless the morpheme appears only in final position, so that a unique continuation matrix can be selected); in case (b), the list must contain at least one (unless the morpheme appears only at the beginning of words, so that the morpheme be reachable).

Note that the matrix-encoded part of a left-PR may participate in more than one matrix. For example, the Spanish verbal inflectional suffix *–o* of indicative present tense first person singular may follow verbs of either theme vowel, i.e. the left-PR of this morpheme appears in three different matrices. Such left-PR's must have the same continuation class identifier (i.e. they appear in the same row) in all of the matrices in which they participate. This may in some cases necessitate the insertion of empty rows in some of the matrices.

DOI:10.15774/PPKE.ITK.2015.013

In order to minimize the number of such empty rows, the SPR's are ranked with regard to the number and size of matrices they affect (calculating $\Sigma(1 - mxtdim/10000)$[iv] summing over the matrices the expression participates in) and they appear in the matrices in the order determined by the ranking (higher-ranked score first). Further optimization removes all identical rows and columns which do not appear in any other matrices including empty rows and columns.

4.2.3.4 The format of the encoding definition file

In the encoding definition file, the following data must be declared:

- The length of bit vectors: the length of the bit vectors containing the bit-encoded properties of lexical items can be declared by assigning a value to the variable `$bitlength`. The value must be one of 8, 16, 24 or 32.
- Matrix selection: when the morphological analyzer identifies two possible morphs next to each other, it is checked whether they are compatible (i.e. whether they satisfy each other's requirements). One part of the compatibility test involves checking whether the value indexed by the right-hand-side matrix code of the left-hand-side morph (usually a stem) and the left-hand-side matrix code of the right-hand-side morph (usually a suffix) in the continuation matrix selected by the left-hand-side morph (the stem) indicates compatibility or incompatibility. Different types of stems may select different continuation matrices for the matrix check, i.e. nominal types of stems may use a different matrix from verbal stems and other morphs (e.g. non-inflectible stems) may specify still another matrix. The matrix is always selected by the left-hand-side morph for each pair of morphs. The selection is made by a subset of the morph's right-hand-side bits and it must be unambiguous. Matrix selection is defined by assigning a structure to the variable `$mtxsel`. The structure defines a matrix name to use for every relevant combination of right-hand-side properties. Only bit-encoded right properties may be used (requirements not). The expressions may not contain disjunction but they must be disjunct (i.e. more than one of them may not be true at the same time; this is needed for the unambiguous choice of a continuation matrix).
- The definition of categories: when a possible analysis for a word is being created by the morphological analyzer, a finite state automaton (the word grammar) is used by the analyzer to check whether the analysis being generated conforms to the morphosyntax of the language. The atomic symbols used by the automaton are morpheme category labels. The category assigned to a morph is determined by a subset of its *right bit-encoded properties* (similarly to the case of matrix selection, as described above). In addition to the formula which must be true for a morph to have a certain category, it must be declared whether the morphs having that category should be searched for from the left or the right end of the word (i.e. whether the lexical lookup direction is left to right or right to left). For stems, the lexical lookup direction is left to right. For inflections, it is normally right to left.
- The declaration of the encoding of properties: the encoding of properties is defined by assigning a structure to the variable `$Gprops`. The structure that defines each property may contain four fields:

[iv] *mtxdim* is the size of the matrix, SPR's are sorted for both dimensions.

- field 0: right/left-side property, indicated by 'r' or 'l'
- field 1: bits or empty ('') for matrix-encoding, '*' if to be ignored, prefix dots or num> to show bit position ('...1' = '3>1') – the representation of bit vectors is left-aligned here
- field 2: the domain of the property: bits must be set if this expression is true (this is really only needed for left-hand-side x properties)
- field 3: entailments: use this to define complex properties

Field 0 (side) is mandatory, the other fields do not have to be present. The default values are then: matrix encoding, no domain and no entailments.

4.2.4 The word grammar

In the Humor analyzer, non-local constraints on word structure are handled by a finite-state automaton that has an extended set of binary state variables in addition to the main state variable. In the grammar development framework, two methods are provided to support the generation of a word grammar automaton.

```
$flags=
{
  'vpfx',   'x',    #verbal prefixes encountered
  'no_vpfx','0',    #no verbal prefixes encountered
  '1_vpfx', '1',    #one verbal prefix encountered
  'sup',    'x<2',  #superlative prefix encountered
  'lcase',  'x<3',  #case lowering suffix encountered
  'ezer',   'x<4',  #"ezer" (1000) encountered
  'hyph',   'x<5',  #hyphen encountered
  ...
};

...

NUM+N:
#sok+emelet+es
119sfx ->    N2
NKENT_sfx -> END    ?{!sup no_vpfx}
nder2 ->     ADJ+N
hyphen ->    HYPHEN ={hyph}

#Is expanded to
NUM+N:
119sfx ->    N2
NKENT_sfx -> END    ?{0.0}
nder2 ->     ADJ+N
hyphen ->    HYPHEN ={1.....}
```

Figure 4.11: Definition, usage and expansion of extended word grammar category macros

DOI:10.15774/PPKE.ITK.2015.013

4.2.4.1 A macro expansion facility

The first method is based on a macro expansion facility that makes it possible for the creator of the grammar to use mnemonic feature names instead of binary vectors for the specification of constraints on extended state variables and the way these extended variables are to be updated when passing an arc in the automaton. An example from the Hungarian description is shown in Figure 4.11.

In addition, list macros can be used to specify sets of arcs in a single step after defining a set of arc labels and the corresponding feature checking and feature setting operations, as shown in the example in Figure 4.12 which is expanded by the system to the Humor word grammar fragment in Figure 4.13.

```
#Definition of word grammar list macros
#inf=inflection
#Advdercmp=adverbial derivational suffix which licenses the superlative prefix
#inf requires sup to be false (no unlicensed superlative prefix is allowed)
@inf=([’inf’,’?{!sup no_vpfx}’],[’NKENT_sfx’,’?{!sup no_vpfx}’],
[’Advdercmp’,’?{no_vpfx}       ={lcase lcase2 !sup}’]);

#some numerals and adjectives license the superlative prefix:
#these delete the sup flag, others do not
@sup=([’_sup’,’!sup’],[’_!sup’,’’]);

#compound word members increment the cmpdm flag, others do not
@cmp=([’_cmpd’,’cmpdm’],[’_!cmpd’,’’]);

@nstem2=(’nstem2’,’nstem12’);
@hyphlc=([’!hyph’,’hyph’],[’lcase’,’lcase2’]);

#Usage of word grammar list macros
#possible left compound member found
N1:
@inf[0] ->               END    @inf[1]
119sfx  ->               ADJ    ?{@hyphlc[0]}     ={@hyphlc[1]}
nder-119 ->              N1     ?{@hyphlc[0]}     ={@hyphlc[1]}
ndercmp ->               ADJ    ?{@hyphlc[0]}     ={@hyphlc[1] !sup}
nder2_adj ->             ADJ
nder2 ->                 N1
vder ->                  V      ?{@hyphlc[0]}     ={@hyphlc[1] !vpfx}
pfx_V ->                 N+P    ={vpfx cmpd}
vstem@cmp[0] ->          N+V    ={@cmp[1] cmpd}
@nstem2@sup[0]@cmp[0] -> N2     LCA{lcase}        ={@sup[1] @cmp[1] cmpd}
nstem2+inf@cmp[0] ->     END    ?{!sup no_vpfx}   ={@cmp[1] cmpd}
nstem_uninfl_cmp2+dot -> DOTREQ ?{!sup no_vpfx}   ={cmpd}
astem@sup[0]@cmp[0] ->   N+ADJ  LCA{lcase}        ={@sup[1] @cmp[1] cmpd}
```

Figure 4.12: Definition and usage of word grammar list macros

4.2.4.2 Regular-expression-based word grammar definition

The other method relies on a regular-expression-based definition of the word grammar automaton. This method was used in most of the more recent morphologies created in the framework. I used the xfst language to define the word grammar categories and the finite-state description of the word grammar itself. The automaton generated by xfst from these

DOI:10.15774/PPKE.ITK.2015.013

```
N1:
inf ->        END   ?{0.0}
NKENT_sfx ->  END   ?{0.0}
Advdercmp ->  END   ?{0}      ={1..........10..}
119sfx ->     ADJ   ?{0.....} ={1...}
119sfx ->     ADJ   ?{1.....} ={1..............}
nder-119 ->   N1    ?{0.....} ={1...}
nder-119 ->   N1    ?{1.....} ={1..............}
ndercmp ->    ADJ   ?{0.....} ={10..}
ndercmp ->    ADJ   ?{1.....} ={1...........0..}
nder2_adj ->  ADJ
nder2 ->      N1
vder ->       V     ?{0.....} ={1..0}
vder ->       V     ?{1.....} ={1.............0}
pfx_V ->      N+P             ={1.......1}
vstem_cmpd -> N+V             ={11.......}
vstem_!cmpd ->N+V             ={1........}
nstem2_sup_cmpd ->   N2       ={11....0..}  LCA{1...}
nstem12_sup_cmpd ->  N2       ={11....0..}  LCA{1...}
nstem2_!sup_cmpd ->  N2       ={11.......}  LCA{1...}
nstem12_!sup_cmpd -> N2       ={11.......}  LCA{1...}
nstem2_sup_!cmpd ->  N2       ={1.....0..}  LCA{1...}
nstem12_sup_!cmpd -> N2       ={1.....0..}  LCA{1...}
nstem2_!sup_!cmpd -> N2       ={1........}  LCA{1...}
nstem12_!sup_!cmpd ->N2       ={1........}  LCA{1...}
nstem2+inf_cmpd ->   END        ?{0.0}   ={11.......}
nstem2+inf_!cmpd ->  END        ?{0.0}   ={1........}
nstem_uninfl_cmp2+dot -> DOTREQ ?{0.0}   ={1........}
astem_sup_cmpd ->    N+ADJ    ={11....0..}  LCA{1...}
astem_!sup_cmpd ->   N+ADJ    ={11.......}  LCA{1...}
astem_sup_!cmpd ->   N+ADJ    ={1.....0..}  LCA{1...}
astem_!sup_!cmpd ->  N+ADJ    ={1........}  LCA{1...}
```

Figure 4.13: Expansion of a word grammar fragment containing list macros in Figure 4.12

regular expressions is exported using the `net` xfst command and converted to the Humor word grammar automaton format. The orthographies of the Uralic minority languages described using the framework did not require the usage of extended state variables. Although there is a limitation in the xfst version released with the book (Beesley and Karttunen, 2003) on the size of the automaton that can be exported using the `net` command, the word grammar automata for the languages I described were all far below this complexity limit. In addition to the ease and flexibility provided by the fact that automata need not to be defined and edited manually, another advantage of this solution is that the regular expressions can be automatically filtered to remove edges from the automata that make it cyclic or are labeled with clitics. The morphology generated using these limited acyclic word grammars can be used to easily generate specific inflectional paradigms that can be compared to model paradigms or manually checked by linguists, which is a nice feature facilitating checking and debugging of the morphological grammar. Figures 4.14 and 4.15 show the xfst source of the word grammar for the Udmurt morphological description and the Humor word grammar automaton generated from it.

4.3 Lemmatization and word-form generation

There is an important difference between the way stem allomorphy (conditioned by an attached suffix) and suffix allomorphy (conditioned by the stem to which the suffix is

DOI:10.15774/PPKE.ITK.2015.013

```
define Digits [
(1dig [0dig|1dig]*) [1dig|10dig|fdig]
];

define Roman [
[[(roman_1000) (roman_100) (roman_10) (roman_1)] & $[?]]
];

define N_stem [
Nom_stem|
ltr|
[[[ltr abbrdot]* (Surname|[Surname | Nom_stem] SurnameSfx)] & $[?]]
];

define infl_stem [
N_stem|V_stem
];

define deriv [
Nom_deriv|V_deriv
];

define Word [
uninfl_stem|
Nom_stem_infl|
V_stem_infl|
[infl_stem|Digits|Roman] [deriv]* infl|
N_stem CxN infl|
NomPron infl
];

define clWord [
Word (clit)
];

regex [
clWord [hyph clWord]* (dot)
];
```

Figure 4.14: The xfst regex source of the Udmurt word grammar

attached, e.g. vowel harmony) must be handled by the morphological analyzer. The reason for this is that the output of the analyzer must be usable for lemmatization purposes. Lemmatization is the identification of the quotation (“uninflected”) form of a word form. This means that all inflectional affixes must be removed from the word and the quotation form of the rest (the lemma) must be produced. In contrast to inflectional affixes, most of the derivational suffixes are normally considered to be part of the lemma. An example is shown in Table 4.3.

surface form	**butá**	**cská**	**bb**	**já**	**tól**	**nadrág**	**ocská**	**tól**
lexical form (lemma)	*buta*	*cska*	*bb*	*ja*	*tól*	*nadrág*	*ocska*	*tól*
abstract lexical form	buta	LVcskA	LAObb	LjA	LtŐl	nadrág	LVcskA	LtŐl
tag	A	DIM	CMP	PSS3	ABL	N	DIM	ABL
lemma 1	**butá**	**cská**	*bb*					
lemma 2	**butá**	*cska*				**nadrág**	*ocska*	
lemma 3	*buta*					*nadrág*		

Table 4.3: Examples of lemmatizing derived and inflected words

In order to produce the lemma correctly (whether or not we consider the comparative and the diminutive suffixes part of the lemma) the *lexical form* of the final morph in the

```
s0:%
NomPron -> s1 FAM
Nom_stem -> s2 FAM
Nom_stem_infl -> fs3 FAM
Surname -> s2 FAM
V_stem -> s4 FAM
V_stem_infl -> fs3 FAM
fdig -> s4
ltr -> s5 FAM
roman_1 -> s4
roman_10 -> s6
roman_100 -> s7
roman_1000 -> s8
uninfl_stem -> fs3 FAM
1dig -> s9
10dig -> s4

s1:
infl -> fs3

s2:
CxN -> s1
Nom_deriv -> s4
SurnameSfx -> s10
V_deriv -> s4
infl -> fs3

fs3:$
clit -> fs11
dot -> fs12
hyph -> s0

s4:
Nom_deriv -> s4
V_deriv -> s4
infl -> fs3

s5:
CxN -> s1
Nom_deriv -> s4
V_deriv -> s4
abbrdot -> s13
infl -> fs3

s6:
Nom_deriv -> s4
V_deriv -> s4
infl -> fs3
roman_1 -> s4

s7:
Nom_deriv -> s4
V_deriv -> s4
infl -> fs3
roman_1 -> s4
roman_10 -> s6

s8:
Nom_deriv -> s4
V_deriv -> s4
infl -> fs3
roman_1 -> s4
roman_10 -> s6
roman_100 -> s7

s9:
Nom_deriv -> s4
V_deriv -> s4
fdig -> s4
infl -> fs3
0dig -> s14
1dig -> s9
10dig -> s4

s10:
CxN -> s1
Nom_deriv -> s4
V_deriv -> s4
infl -> fs3

fs11:$
dot -> fs12
hyph -> s0

fs12:$

s13:
CxN -> s1
Nom_deriv -> s4
Nom_stem -> s15 FAM
Surname -> s2 FAM
V_deriv -> s4
infl -> fs3
ltr -> s16 FAM

s14:
fdig -> s4
0dig -> s14
1dig -> s9
10dig -> s4

s15:
SurnameSfx -> s10

s16:
abbrdot -> s13
```

Figure 4.15: The Humor word grammar automaton for Udmurt

lemma must be concatenated to the **surface form** of the non-final morphs. Note that the `abstract lexical form` does not look at all like the quotation form and thus it cannot be used for lemmatization purposes. Note also that the lexical form used for lemmatization must reflect suffix allomorphies triggered by the stem to which a derivational suffix is attached: the diminutive suffix is consonant-initial *(cska)* after a vowel final stem *(buta)*, but it assumes a vowel-initial form *(ocska)* after a consonant-final stem *(nadrág)*.

For correct lemmatization, the various surface allomorphs of a stem (where allomorphy is triggered by the presence of a suffix) must refer to the same lexical form, while the surface allomorphs of a suffix (where allomorphy is triggered by the stem on the left)[v] must have different lexical forms. Note that applying the same principle to inflectional suffixes may cause some aesthetic problems in the output. E.g. one of the returned segmented analyses of the word form *finnek* 'Finnish+dative', `finn[ADJ|nat]+ek[DAT]`, does not indicate that the lexical form of the dative suffix is *n*-initial. Some users might find this annoying. An analysis of the form `finn[ADJ|nat]+nek[DAT]=ek` could be preferable.

[v] including cases where the stem-triggered allomorphy manifests itself in vacillating behavior, e.g. *mágnes+nak/nek* 'magnet+dative', *olcsó+k/ak* 'cheap ones'

This contrast between the two kinds of allomorphy is also reflected in the representation of morphemes in the level-2 lexicons. While stem allomorphs are represented as members of the `allomfs` list within a single lexical entry (all of them referring to the same lexical form given in the common `seg` feature of the entry), allomorphs of a suffix (in the case of stem triggered allomorphy) are represented as independent entries, each of them having a different lexical form stored as the value of its `seg` attribute. The example in Figure 4.16 illustrates this: it shows the level-2 representation of the dative case marker suffix as four independent lexical entries. (The *-ak/-ek* entries are used for stems ending in geminate *n*, e.g. *finn* 'Finnish'.)

```
\$mrf = {
  'humor' => 'DAT',
  'tag' => 'DAT',
  'allomfs' => [
    {
      'rp' => 'mcat_infl',
      'allomf' => 'nak',
      'lp' => 'FVL n_ini Cini',
      'lr' => ' VHB cat_Nom'
    }
  ],
  'seg' => '=nak'
};
```

```
\$mrf = {
  'humor' => 'DAT',
  'tag' => 'DAT',
  'allomfs' => [
    {
      'rp' => 'mcat_infl',
      'allomf' => 'ak',
      'lp' => 'FVL Vini',
      'lr' => 'n_fin =Al VHB cat_Nom'
    }
  ],
  'seg' => '=ak'
};
```

```
\$mrf = {
  'humor' => 'DAT',
  'tag' => 'DAT',
  'allomfs' => [
    {
      'rp' => 'mcat_infl',
      'allomf' => 'nek',
      'lp' => 'FVL n_ini Cini',
      'lr' => ' VHF cat_Nom'
    }
  ],
  'seg' => '=nek'
};
```

```
\$mrf = {
  'humor' => 'DAT',
  'tag' => 'DAT',
  'allomfs' => [
    {
      'rp' => 'mcat_infl',
      'allomf' => 'ek',
      'lp' => 'FVL Vini',
      'lr' => 'n_fin =Al VHF cat_Nom'
    }
  ],
  'seg' => '=ek'
};
```

Figure 4.16: The level-2 entry of the Hungarian dative case marker suffix

4.3.1 The lemmatizer

The Humor 'lemmatizer' tool, built around the analyzer core, does more than just identifying lemmas of word forms: it also identifies the exposed morphosyntactic features. In contrast to the more verbose analyses produced by the core analyzer, compound members and derivational suffixes do not appear as independent items in the output of the lemmatizer, so, in contrast to analyses returned by the morphological analyzer, the internal structure of words is not revealed, as shown in the example below. The analyses produced by the lemmatizer are well suited for tasks like corpus tagging, indexing and parsing.

DOI:10.15774/PPKE.ITK.2015.013

```
analyzer>fejetlenséget
 fej%etlen%ség[S_N]=fejetlenség+et[I_ACC]
 fej@etlen[S_A]=fejetlen+ség[D=N_PROP]+et[I_ACC]
 fej[S_N]+etlen[D=A_NPRIV]+ség[D=N_PROP]+et[I_ACC]
 fej[S_V]+etlen[D=A_VPRIV]+ség[D=N_PROP]+et[I_ACC]

lemmatizer>fejetlenséget
 fejetlenség[N][ACC]
```

There are two implementations of the lemmatizer. One of them, called *HumorLem*, relies on the tag format presented in Chapter 3 containing a prefix for morpheme category: `S_` for stems, `P_` for prefixes, `D=PoS_` for derivational suffixes resulting in a stem of category `PoS` and `I_` for inflectional suffixes. The other implementation, *stem2005*, uses a separate configuration file for morpheme categories and other features. Both implementations merge derivational affixes into the stem dynamically tagging the resulting stem with the resulting part of speech category. They can properly handle the task of correctly lemmatizing and filtering special Hungarian word constructions, such as words containing more than one suffixed stem where the stem categories and the affixes must match each other e.g. *jövök-megyek* 'I come and go'[vi], or words like *ablak+mos@ó* 'window+clean@er' even in cases where the programs are parametrized not to consider the -*Ó* suffix to be part of the lemma. They are also capable of filtering out analyses matching some regular expression, marking or filtering out productive compounds, and returning full analysis in addition to the lemmatized analysis.

Furthermore, *HumorLem* has the following additional features:

- adding segmentation marks to the lemma (`+`: compound, `@`: derivational);
- optional warnings about stems being already included in the lexicon, equivalent and different analyses, non-lexical forms, category mismatches w.r.t. the PoS tag given in the input, no analysis etc.;
- filtering out a.) non-lexical forms if there are lexical forms, b.) any analyses in addition to lexically given ones;
- treat a set of derivational affixes as if they were inflection;
- doing segmentation only, returning surface form for all segments.

These make *HumorLem* easily applicable to the task of checking and filtering word lists and automatically adding segmentation marks to compounds and derived stems to be added to the lexicon.

The following features of the *stem2005* lemmatizer, on the other hand, make it especially suitable for integration in other applications:

- it is multithread-safe;
- Optional caching of lemmatized analyses makes it fast;
- it can use a text-file-based inflecting dictionary to handle domain-specific words not included in the Humor lexicon. The inflecting dictionary consists of lines of the following format: [lemma] [similarly inflected lemma known to the MA] [optional PoS label].

[vi] This construction is called *ikerszó* 'twin word' in traditional Hungarian grammars.

DOI:10.15774/PPKE.ITK.2015.013

4.3.2 Word form generation

The original Humor system lacked the capability of word form generation and the existing lemmatizer was unable to correctly lemmatize certain non-trivial word constructions. These shortcomings of the system had to be overcome so that the morphology could be integrated in the English–Hungarian machine translation system, MetaMorpho (Novák et al., 2008) and in several other applications. I solved these problems by extending the system so that it can also be used as a morphological generator and implementing better lemmatization algorithms, as described in 4.3.1.

The Humor generator is not a simple inverse of the corresponding morphological analyzer. One requirement was that in the case of free alternation it should generate only the least marked form, consequently the Hungarian morphological description was extended with information expressing markedness. Marked forms are automatically removed during compilation from the version of the database that is intended for word form generation in the machine translation system, where the generation of a single preferred word form is required.

Another unique feature of the word form generator is that it can generate the inflected and derived forms of any multiply derived and/or compound stem without explicitly referring to compound boundaries or derivational suffixes in the input, even if the whole complex stem is not in the lexicon of the analyzer. This is a useful feature in the case of languages where morphologically very complex stems are commonplace. When generating inflected (or derived) forms of a morphologically complex stem, one does not have to be concerned whether the stem is included in the stem database. If the corresponding analyzer can analyze it in any way, the generator will be able to generate its inflected forms correctly.

The generator produces all word forms that could be realizations of a given morpheme sequence. The input for the generator is a lemma followed by a sequence of category labels that express the morphosyntactic features the word form should expose.

The following examples show how the generator produces an inflected form of the derived nominal stem *félkarúság*, which is not part of the stem lexicon, and the explicit application of the derivational suffix (and the same inflectional suffix) to the absolute verbal root of the word.

```
generator>félkarúság[N][ACC]
 félkarúságot
generator>félkarú[A][_PROP][ACC]
 félkarúságot
```

In some languages, there is considerable variation in suffix ordering (see e.g. Section 5.3), thus I also created a version of the generator that has another useful feature: it does not assume that the morphosyntactic features are properly ordered in the input, rather it considers them a set.

DOI:10.15774/PPKE.ITK.2015.013

4.3.2.1 Creating a morphological generator database

The fact that the morphological database contains both surface and lexical forms at the same level of representation makes it easy to transform the analysis database to a flexible generation database. When doing morphological analysis, the analyzer maps input surface forms to lexical forms and category labels, which are three separate pieces of data for each allomorph entry in the lexicon. When creating the word form generation database, each entry of the analysis database is in general mapped to more than one generation entry. The following mappings are performed:

1. The tag fields are all empty in the generator lexicon.
2. Prefix entries are treated as stem entries.
3. All entries are transferred with a `surface form` → `surface form` mapping, unless the entry is restricted to the generator. These entries make sure that the generator maps each surface form that the corresponding analyzer accepts to itself.
4. The mappings from this point on (5.–8.) are only applied to entries not restricted to the analyzer. This restriction makes sure that the generator does not generate marked (dispreferred) forms.
5. Inflectional suffixes are transferred with an `[inflection tag]` → `surface form` mapping.
6. Stem entries are transferred with a `lexical form[stem PoS tag]` → `surface form` mapping.
7. derivational entries are transferred with a `lexical form[derivational tag]` → `surface form` mapping and a `lexical form[stem PoS tag]` → `surface form` mapping, where `[stem PoS tag]` is the resulting PoS category the application of the derivational suffix results in.
8. For segmented polymorphemic entries, the mappings above are done for each morph in the sequence with the restriction that partial surface forms are only produced for entries not restricted to the generator.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

5

Applications of the model to various languages

After diving deep into the abyss of Hungarian morphology, join us on a time-travel adventure including witch trials and a quick medical training. Finally, a linguistic excursion in the Nordic area of the World, with a special tour of reindeer hunting.
¿Erizos?
Quoi? ≪Erizos≫? ... Ah, ≪erizos≫! Non.

Contents

5.1 The Hungarian analyzer

Of the computational morphologies I implemented using the formalism described in this thesis, the most elaborate and extensive one is the morphological database created for contemporary Standard Hungarian. I this subchapter, I present the main characteristics of this morphological description, also illustrating specific constructs of the formalism.

The present state of the morphological database for Standard Hungarian consists of the parts shown in Table 5.1.

stem lexicons		lemmas/lexemes	allomorphs
generic vocabulary		**95811**	**141718**
	original lexicon extended	75132	105473
	closed-class stems (pronouns, numerals, etc.)	744	3675
	from dictionaries and corpora	19935	32570
terminological lexicons		**110129**	**178324**
	Geographical and human names	40262	
	Nuclear technology	911	
	Financial/Administration	4736	
	English	1920	
	Medical	40813	
	Defense	21487	
all		**205940**	**320042**
	of that compounds	89415	126728
	polymorphemic/suffixed		7720

suffix lexicon		lexemes	allomorphs
	all	**283**	**12041**
	polymorphemic		10959

rule files	operations	rules	lines
stem rules file			
	45 declarations	520 rules	2074 lines
	596 allmorph generating operations	220 stem allomorphy rules	
suffix rules file		50 rules	233 lines
	86 allmorph generating operations	34 allomorphy rules	

word grm	states	transitions	flags
word grammar automaton			
	47 states	602 transitions	20 flags

	categories	properties	
encoding definition of features and word grammar categories			
	102 word grammar categories	102 vector-encoded properties 187 matrix-encoded properties	

Table 5.1: Components of the Hungarian morphological description

DOI:10.15774/PPKE.ITK.2015.013

5.1.1 Stem lexicon

The core of the stem lexicon is a resource that was the lexicon of the original Humor morphological analyzer. It contained about 69500 lemmas (with 93345 allomorphs, 110 inflectional + 41 derivational suffix morphemes and 4887+793 suffix sequence allomorphs). The original lexicon contained morphological information in the format that the Humor analyzer uses: binary vectors and continuation matrices. The information contained in the binary vectors was decoded and converted into a human-readable feature representation. The redundant features were deleted and inconsistencies and errors in the database were manually corrected including category tags and erroneous features. When doing this task, and during the whole process of rewriting the morphology as detailed below, I mainly relied on my own competence and on authoritative dictionaries. Later, however, when the morphology was used for different tasks by myself and other people, such as machine translation and corpus annotation, I needed to fine-tune the grammar and revise my initial decisions to match data encountered in the corpora. This has been an iterative process that spanned over a decade.

Allomorphs belonging to the same lemma were clustered and, based on the allomorphy patterns, a stem alternation feature was assigned to irregular alternating stems. Regular stem alternations, like stem final vowel lengthening (`VZA`) or hyphen insertion (-), are introduced by the stem alternation rules and need not to be included in the lexicon. Table 5.2 shows the lexically determined stem alternation classes that were introduced. (The `+` at the end of some codes (`VZA, SVS, vshrt`) indicates that the base allomorph of the stem has a full paradigm: i.e. it may also take the suffixes that normally trigger the alternation, e.g. *sárt, sáros* or *sarat, saras:* `SVS+`.)

In addition to these, stem alternation classes for inflected pronouns and words containing possessive suffixes in the middle of the word (`stemalt:poss1`, `L` indicates lowering here) or possessive suffixes in agreement at multiple positions (`stemalt:poss2`) were introduced:

```
szavL=a+járás=a[FN];stemalt:poss1;
hír=e-+hamvL=a[FN]; zarte:e;stemalt:poss2;
kény=e-+kedv=e[FN]; zarte:eee;stemalt:poss2;
```

E.g. `stemalt:poss1` generates the forms *szavamjárása* ‘my favorite words’, *szavadjárása* ‘your favorite words’, *szavajárása* ‘his/her favorite words’ etc.; `stemalt:poss2` generates *hírem-hamvam* ‘no news of me’, *híred-hamvad* ‘no news of you’, *híre-hamva* ‘no news of him/her/it’ etc.

Moreover, polymorphemic lexical entries were segmented. The following segmentation marks were introduced:

- `+`: compound boundary,
- `@`: productive derivational suffix boundary,
- `%`: unproductive derivational suffix boundary,
- `=`: inflectional suffix boundary,
- `#`: morpheme boundary in foreign stems.

DOI:10.15774/PPKE.ITK.2015.013

`stemalt`	**interpretation**	**example**
`-`	orthographic variation in abbreviations and foreign words (a hyphen is added when the stem is suffixed)	ABC → ABC-*hez* Bretagne → Bretagne-*ban*
`VZA(+)`	vowel-zero alternation (the last (mid) short vowel of the stem disappears in certain suffixed forms)	lélek → lelk*em*, bokor → bokr*ot*, fészek → fészk*en*,
`SVS(+)`	stem vowel shortening (the last long vowel of the stem is shortened in certain suffixed forms)	nyár → nyar*at*
`vins`	*v* insertion (in some long-vowel-final stems: a stem-final *v* appears in certain suffixed forms)	mű → műv*e*
`vshrt(+)`	*v* insertion and vowel shortening (in some long-vowel-final stems: a stem-final *v* appears and the stem vowel is shortened in certain suffixed forms; `vins` and `vshrt` is triggered by the same suffixes)	tó → tav*on*, ló → lov*on*
`vVST(+)`	vowel-*v* alternation (in some short *u/ü*-final stems: the stem-final vowel alternates with *v* in some suffixed forms: the words in this group do not exhibit a uniform pattern concerning the suffixes that trigger the alternation)	falu → falv*ak*
`UDEL`	final *ú* deletion (in certain suffixed forms, the *mag* → *magv* alternation is triggered by the same suffixes)	borjú → borj*ak*, mag → magv*ak*
`PVS`	”possessive vowel shortening” (the stem-final *ó/ő* is shortened to *a/e* in some suffixed forms: the words in this group do not exhibit a uniform pattern concerning the suffixes that trigger the alternation)	ajtó → ajta*ja*
`KST`	kinship terms (the stem-final short low vowel disappears before the 3sg possessive marker *-ja*)	apa → ap*ja*
`SPEC`	other special alternations	e.g. száj → szá*m*
`lex`	all allomorphs are explicitly given in the stem lexicon	e.g. kend → kend*tek*

Table 5.2: Stem alternation codes used in the Hungarian description

Since the morphological formalism includes an inheritance mechanism that copies features of compound stems from the features of their final member by default, redundant features of compounds were also deleted. The inheritance mechanism is blocked by the presence of explicit features. This makes it possible for exceptional compounds to have a morphological behavior that deviates from that of the standalone variant of the final compound member. An additional mechanism makes it possible to have all compounds ending in a specific word to behave in an identical manner, which, however, is different from the behavior of the standalone non-compound stem. In some cases, this has been achieved as part of the rule system (e.g. for some members of some closed stem alternation classes, e.g. *szó* ‘word’). This can also be achieved by defining a stem which only provides features for the inheritance mechanism, but is not included in the compiled morphology as a standalone stem. This mechanism has primarily been used to describe compound adjectives or geographical names, where the final compound member is often not a standalone word, and the compounding construction is not fully productive.

Entries of this type are marked by `**...` as shown in the example below. The presence of certain features does not block inheritance, e.g. the semantic `isa` feature in the examples, or the phonemic `zarte` 'mid-e' feature used to indicate whether the *e*'s in the stem are mid or low in the dialects distinguishing the two.

```
**...forma[MN|NM];rp:ESS_Vn;
  mag=a+forma[MN|NM];
  olyan+forma[MN|NM];
  ilyen+forma[MN|NM]; zarte:e;
**...falv=a[FN];rr:!PL !POSS;++!falv+i[IKEP];cat:Adj;rp:LOW;
  Kerka+falv=a[FN];isa:település; zarte:ë;
  Duna+falv=a[FN];isa:település;
  Klára+falv=a[FN];isa:település;
**...túl[FN];rp:=jA;
  Duná=n+túl[FN];
  Dél-+(Duná=n+túl[FN];
  Tiszá=n+túl[FN];
 **...szárny@ú[MN];rp:%ESS_Vn;
  fedel@es+szárny@ú[MN&FN]; zarte:ëee;
  egyen@es+szárny@ú[MN&FN]; zarte:ëeë;
  pikkely@es+szárny@ú[MN&FN]; zarte:ëë;
  hártyá@s+szárny@ú[MN&FN];
  recé@s+szárny@ú[MN&FN]; zarte:e;
```

Certain parts or the whole of the lexical representation of a stem may be explicitly excluded from inheritance or inheritance can be blocked by using the `no_inh` feature. E.g. in the example below, *családapa* 'family man' does not inherit features of *apa* 'father', and the comparative form *nyugatabbi* 'more to the west' of *nyugati* 'western' is not inherited by *nyugati*-final compound adjectives.

```
apa[FN];stemalt:KST;
  bérma+apa[FN];
  család+apa[FN];no_inh:;
  déd+apa[FN];
  öreg+apa[FN]; zarte:e;
  kis+apa[FN];no_inh:;
nyugat@i[MN];rp:ESS_Vn;no_inh:;++!nyugat+abb@i[FOK];cat:Adj;rp:LOW;gp:sup;
  vad+nyugat)@i[MN];
  nap+nyugat)@i[MN];
  közép+nyugat)@i[MN];
```

5.1.1.1 Types of features

The lexicon includes only unpredictable features of stems. The two properties every stem has are the lemma and the category tag. In addition, irregular pronunciation can be defined using the `phon` feature. Stems belonging to closed alternation classes have the `stemalt` feature. E.g. the stems *apa* 'father' and *anya* 'mother' belong to `stemalt:KST` (kinship term) . Some semantic properties of words may also be included in the lexicon using the `isa` feature, such as the type of entity denoted by a proper name, e.g. `isa:település` (settlement). Some orthographical rules and certain suffixes are sensitive to this information, e.g. geminate-final given names are suffixed using a hyphen when a suffix starting with the same letter is attached. This is not so for other geminate-final words. In addition, the suffix *–né* can only be attached to surnames, male given names and names of certain professions. In addition to the features mentioned above, exceptional lexical items may also contain binary properties and restrictions. These are defined using the `rp, lp, rr, lr` and `gp` features:

```
 *java takes only possessive endings, is a lowering stem,
 *and licenses the superlative prefix:
jav=a[FN];no_inh:;rr:POSS;rp:LOW;gp:sup;
 *térd has j-less possessive forms, is a lowering stem:
térd[FN];rp:=A;rp:LOW;
 *3 sg possessive: j-initial, plural of possessives: j-less:
barát[FN];rp:=jA&=Ai;
```

Certain properties indicate that a certain suffix morpheme or an identically behaving group of suffix morphemes can be attached to the given stem allomorph. The name of properties of this type begins with a hyphen (e.g. `-i, -kor, -Om`), as shown in the following examples:

```
vacsora[FN];rp:-kor; *takes the temporal -kor suffix
lejj@ebb[HA];rp:-i; zarte:ëe; *takes the -i adjectivizer suffix
 *este does not take the -i adjectivizer suffix
 *the corresponding form is esti, which is a lowering adjective.
este[FN];rp:!-i; zarte:ee;++!est+i[IKEP];cat:Adj;rp:LOW; zarte:e;
 *1sg subject may be marked by the suffix -Om for many -ik-final verbs:
öreg@ed=ik[IGE]; rp:-Om; zarte:eë;stemalt:szd;
```

Another group of properties indicates in what form a suffix or a group of suffixes can be attached to the given stem (or stem allomorph). The name of properties of this type begins with an `=`. See the examples below.

```
 *possessive is either j-initial or j-less, optionally lowering,
 *accusative is zordon+t, essive modalis is -Ul
zordon[MN];rp:=A&=jA&LOW+ =t;rp:ESS_Ul;
él[IGE];rp:=tAt; *causative is él+tet
```

The presence of a property of the latter type implies the presence of a corresponding hyphen-initial property. These implicatures are defined in the encoding definition file, as follows:

```
’-jA’ =>[’r’,’’,’’],             #the stem takes the 3rd person possessive suffix
’=jA’ =>[’r’,’10>1’,’’,’-jA’],      #the form of the 3rd person possessive suffix
’=A’ => [’r’,’’,’’,’-jA&!=jA’],
’-jAi’ =>[’r’,’’,’’],              #the stem takes the plural possession suffixes
’=Ai’ =>[’r’,’11>0’,’’,’-jAi&Cfin’],  #the form of the plural possession suffixes
’=jAi’ =>[’r’,’’,’’,’-jAi&!=Ai’],
’=i’ =>[’r’,’’,’’,’-jAi&!Cfin&!ifin&!=jAi’],
 #possible combinations:
 #jA i, jA jAi, A Ai, jA Ai, jA jAi A Ai
```

5.1.1.2 Vacillating behavior

Certain stems can be combined with more than one allomorph of a certain suffix morpheme. One of the most frequent example of this in Hungarian is vacillating vowel harmony. This harmony type may only apply to front harmonic stems containing at least one back vowel which is followed by a sequence of neutral *(e,é,i,í)* vowels. Vacillating vowel harmony is marked in the lexicon using the property `VHV` (vowel harmony: vacillating). In the case of a subset of vacillating stems, one of the harmonic variants is preferred. Such items are marked by the features `VHVB` (back harmony preferred) and `VHVF` (front harmony preferred). Vacillating stems having a preferred harmony variant take vowel-initial suffix allomorphs only in the preferred harmonic form, while consonant-initial suffixes may alternate (*klarinét: -os/-n[ae]k, mágnes: -es/-n[ae]k*). For items marked by `VHVB+` or `VHVF+`, all suffixed forms can be either front or back, but one of them is marked as preferred.

```
gálic[FN];rp:VHB;*back harmony
október[FN]; zarte:ë;rp:=A;rp:VHFU;*unrounded front harmony
kupec[FN]; zarte:ë;rp:VHV;*vacillating harmony
Marcell[FN];rp:=jA;rp:VHVF+; zarte:ë;*front harmony preferred, all forms allowed
```

Another example are stems taking the possessive suffix both in its j-initial and j-less forms. These can be marked by either including both the property `=A` and `=jA` in the `rp` feature or by entering the property `=*j0A`. For entries like that, a preferred stem allomorph having only the property `=A` and a dispreferred allomorph having only the property `=jA` is generated. The allomorphs are generated by the following generic allomorph duplication block in the stem rule file based on the `=*0` notation.

```
#split allomorphs having specifications like =A0lAk
#to one having =lAk and another having =AlAk
dup(rp:/=[^ &]+0/)
{
 #the preferred version
 rp:s/=\*[^ &]+0|=([^ &]+)0[^ *&]*\*/=$1/g;;;
 #the dispreferred version
 !restr:/^g/&&rp:s/=\*([^ &]+)0|=[^ &]+0([^ *&]*)\*/=$1$2/g;;restr:a;
}
#remove =* and -* properties in generator, remove * in analyzer
dup(rp:/[=-][^0\s|()]*\*[^0\s|()]+(?:\s+|$)/)
{
 #remove =* and -* properties in generator
 !restr:/^a/&&rp:s/[=-][^0\s|()]*\*[^0\s|()]+(?:\s+|$)//g;;restr:g;
 #remove * in analyzer
 !restr:/^g/&&rp:s/([=-][^0\s|()]*)\*([^0\s|()]+(?:\s+|$))/$1$2/g;;restr:a;
}
```

The dispreferred allomorph is filtered out from the generator lexicon. The same applies to entries with vacillating harmony, allomorphs of which are generated by the following allomorph duplication code:

DOI:10.15774/PPKE.ITK.2015.013

```
###split underspecified allomfs
map(@allomfs)
{
 #split vacillating stems
 #VHV:balett -[oe]t, n[ae]k
 #VHVF:mágnes -es, n[ae]k
 #VHVB:klarinét -ot, n[ae]k
 #restrict dispreferred form to analyzer

 dup(VHV[FB]?\+?)
 {
  #back preferred
  rp:s/VHVB\+?/VHB/;;;
  #front unrounded preferred
  rp:s/VHVF\+?/VHFU/;;;
  #for VHV: the value calculated in $vhrm used as preferred form
  $vhrm ne 'VHV'&&rp:s/VHV(?= |$)/$vhrm/;;;
  #VHFU if $vhrm also contains VHV
  $vhrm eq 'VHV'&&rp:s/VHV(?= |$)/VHFU/;;;
  #back harmony is dispreferred unless $vhrm is VHB
  $vhrm ne 'VHB'&&rp:s/VHV(?= |$)/VHB/&&!restr:/^g/;;restr:a;
  #front harmony is dispreferred if $vhrm is VHB
  $vhrm eq 'VHB'&&rp:s/VHV(?= |$)/VHFU/&&!restr:/^g/;;restr:a;
  #front unrounded only for consonant-initial suffixes
  rp:s/VHVB(?!\+)/VHFU/&&!restr:/^g/;(Cini|comp2);restr:a;
  #back only for consonant-initial suffixes
  rp:s/VHVF(?!\+)/VHB/&&!restr:/^g/;(Cini|comp2);restr:a;
  #the dispreferred variants only in the analyzer
  rp:s/VHVB\+/VHFU/&&!restr:/^g/;;restr:a;
  rp:s/VHVF\+/VHB/&&!restr:/^g/;;restr:a;
 }
...
}
```

5.1.1.3 Irregular forms

In addition to entries marked as belonging to some closed stem alternation class, some entries have irregular suffixed forms declared in the stem lexicon. Irregular forms which are to be suffixed further, e.g. ones containing derivational suffixes are introduced following a `++!` mark, while irregular case suffixed forms generally follow a `++`. A form introduced using the `++!` mark undergoes all stem allomorphy rules, while a form introduced using the `++` mark is simply included among the allomorphs of the given stem. See the examples below.

```
este[FN];rp:!-i; zarte:ee;++!est+i[IKEP];cat:Adj;rp:LOW; zarte:e;
kicsi[MN];++!kis+ebb[FOK];cat:Adj;gp:sup; zarte:e;
idén[FN];cat:N;rr:CASE !0mrf;++!idén+[TMP_INL];cat:Adv;
jó[MN];rp:!grad ESS_no -kor;++!jo+bb[FOK];cat:Adj;
        rp:-kor ESS_Vn VHB LOW;++!jó+l[ESSMOD];rp:cat_Adv;

Vác[FN];isa:település;++Vác+ott[INL];rp:mcat_stem+infl;
szabad[IGE];cat:X;rp:mcat_stem+infl;++szabad+[e3];++szabad+jon[Pe3];
        ++szabad+na[Fe3];++szabad+ott[Me3];
szomj[FN];rp:LOW;no_inh:;++szomj+ot[ACC];rp:mcat_stem+infl;
gát@ol[IGE];++gátl;rp:-Ó -Ás;
```

There are 15 irregular verbs which are marked by the property `irreg`. The allomorphy patterns of these verbs are defined in the stem allomorphy rule file similarly to those of entries marked as belonging to a named stem alternation class (see the example paradigms below). Even some stems of the latter type exhibit idiosyncratic behavior characterizing only a single lexical entry, e.g. *szó* and *tó* in the stem alternation class `vshrt` (v-insertion and vowel shortening).

```
##verbs marked as irregular
if(irreg)
{
 #nincs
 root:/(^|[${cmpsep}])[ns]incs$/
 +L;; -O3 =AnAk
 +L/$/en/;; -O3
 #megy
 root:/(^|[${cmpsep}])m[ëe]gy$/
 +L;; -O3 -Ok -Unk =OgAt
 #më
 +L/gy$//;; -hAt
 #mën
 +L/gy$/n/;; -vA =j -Ás -Ó =nA =nAk =tOk =tAm =t -tAbAn
 +L//$1mé/;; =sz
 +L//$1mé+gy[e2]/;restr:a; mcat_stem+infl;#unless($generator)
```

-*ik*-final verbs belonging to the *sz-d* or *sz-d-v* alternation class have two possible lemmas, an *sz* or a *d/z*-final one. The following fragment of the rule file describes their behavior:

```
elsif(stemalt:szdv?)
{
 #defectives: -Agsz,-Aksz,-Alsz,gyarapsz...
 #dsz(v) verbs
 ;;irreg;
 ;;gseg:$seg;
 seg:s/[aeou]?[dz](?==)/sz/;;;
 if(stemalt:szdv)
 {
  #esküszik,alkuszik
  root:/($C$C[uü])d$/
  +L;;=Ott =*Ol =nA =nAk =tOk =tAm =lAk =d =j =jUk -hAt -vA -*Ás -*Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
  +L//$1v/;;-Ás -Ó;
  #
  root:/($C$C[uü])z$/
  +L;;=Ott =*Ol =nA =nAk =tOk =tAm =lAk =d =j =jUk -hAt -vA -*Ás -*Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
  +L//$1v/;;-Ás -Ó;
  #igyekezik...
  root:/${V}z$/
  +L;;=Ott =*Ol =nA =nAk =tOk =tAm =lAk =d =j =jUk -hAt -vA -*Ás -*Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
  +L//v/;;-Ás -Ó;
  #növekedik...
  root:/${V}d$/&&stemalt:szdv
  +L;;=Ott =*Ol =nA =nAk =tOk =tAm =lAk =d =j =jUk -hAt -vA -*Ás -*Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
  +L//v/;;-Ás -Ó;
 }
 else
 {
  #emlékezik...
  root:/${V}z$/
  +L;;=Ott =*Ol =nA =nAk =tOk =tAm =lAk =d =z =zUk -hAt -vA -Ás -Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
  #melegedik...
  root:/${V}d$/
  +L;;=Ott =*sz =nA =nAk =tOk =tAm =lAk =d =j =jUk -hAt -vA -Ás -Ó -*ik -*Ok -*Unk -AndÓ
      -AtlAn =tAt
 }
 {
  #the -sz stem of dsz verbs: esküszik,alkuszik
  root:/($C$C[uü])[dz]$/
  +L//$1sz/;;-ik -Ok -Om -Unk =Ol =*nAk =*tOk;
  #the -sz stem of all other dsz verbs: verekszik, kisebbszik...
  #root:/(${V_}$C)$V[dz]$/
  root:/$V[dz]$/
  +L//sz/;;-ik -Ok -Om -Unk =Ol =*AnAk =*OtOk;
 }
}
```

These lexical items only have a single *dik/zik*-final entry in the stem lexicon, however, they are

Symbol	Interpretaion	Symbol	Interpretaion
`A`	a/e (`bAn`)	`V`	a/e/(ë)/o/ö/0 (suffix-initial lowerable linking vowel; `Vt, Vk, Vs` etc.)
`Á`	á/é (`nÁl`)	`Q`	o/ö (used only for the implementation of the behavior of V)

`O`	o/e(ë)/ö (`hOz`)	`OO, AO,`
		`UO`
			unstable suffix-initial vowel (`OOn, UOnk, AOcskA` etc.)

`Ő`	ó/ő (`bŐl`)	`B`	back harmony (`hídB, hidBL`)
`U`	u/ü (`jUk`)	`vO`	underspecified consonant slot (v-assimilation): `vOAl, vOÁ`
`Ű`	ú/ű (`jŰ`)	`L+`	(`L` before boundary) suffix vowel lowering stem (`házL`)
		`+L`	(`L` after boundary) stem final vowel lengthening suffix (`+LbAn`)

Table 5.3: The interpretation of special characters in the value of the `phon` feature in the Hungarian description:

marked by either the `stemalt:dsz / stemalt:dszv` or the `stemalt:szd/stemalt:szdv` feature depending on whether the *d/z*-final or the *sz*-final form is preferred. For entries having the feature `stemalt:szd/stemalt:szdv` a *szik*-final lemma is generated, while those marked by `stemalt:dsz/stemalt:dszv` retain their *dik/zik*-final lemma.

5.1.2 Suffix lexicon

The Hungarian suffix lexicon has a tabular format. Each item has a tag, a phonemic form, a suffix alternation class, a morphological category, and additional properties and requirements. Allomorphies belonging to each suffix alternation class are defined in the suffix alternation rule file.

The phonological form of suffixes may contain archi-phonemes that define harmonic behavior. The harmonic variants of suffixes and inflectional suffix sequences which appear in the Hungarian level-2 suffix lexicon file are generated using a two-level Kimmo-style transducer augmented with some regular-expression-based substitution expressions. Table 5.3 explains the special characters that may appear in the value of the `phon` feature in the Hungarian description. They are used to represent harmonic vowels and some special morphophonological properties of morphs that may affect the form of the following suffix or preceding stem.

Nominal inflectional suffixes belong to one of the morphological categories (the possible values of the `mcat` feature) described in Table 5.4. The `mcat` feature of derivational suffixes marks the category of stems to which the suffix can be attached and the category of the suffixed form. E.g. *-beli* is an `N>Adj` suffix.

5.1.3 Rule files

The stem allomorphy rule file consists of 2074 lines of code. The declaration of string variables mainly used in regular expressions is followed by the declaration of list variables, which are used for the generation of inflected forms of pronouns and postpositions. The file contains about 520 rules, 220 of which are allomorphy rules consisting of 596 individual allomorph generation operations. The

DOI:10.15774/PPKE.ITK.2015.013

Value	Interpretation
POSS	Possessive marker that may be followed by the familiar plural marker *-ék*
PL	Plural or plural possessive marker which may not be followed by *–ék*
FAM	Familiar plural *–ék*
ANP	Anaphoric possessive marker *–é* and *–éi*
CASE	Case ending
KEPP	the suffixes -képp and -képpen
INFL	Verbal inflection
Pos1>PoS2	Derivation from PoS1 to PoS2

Table 5.4: Possible values of the `mtag` feature in the Hungarian suffix lexicon file

rest of the rules add morpheme-level or allomorph-level properties and requirements to morpheme representations coming from the lexicon or to already generated allomorphs.

The suffix allomorphy rule file is less complex with just 233 lines of code and 50 rules, of which 34 are allomorphy rules (conditioned mainly on the value of the `sfxalt` feature in the suffix lexicon) containing 86 allomorph generation operations.

The syntax and semantics of rules are described in detail in the Appendix with examples for each construction. Some more examples are included in Section 5.1.1.

5.2 Adaptation of the Hungarian morphology to special domains

Language use in special domains and language variants may deviate in a significant manner from what one encounters in the standard written dialect of the language. The morphological model needs to be adapted when texts from such a special language variant are to be analyzed. In this subchapter, two such examples are described, demonstrating the adaptibility of the analyzer built using the framework described earlier. First, an adapted model of Hungarian morphology is introduced that is applicable to the annotation of Old and Middle Hungarian texts. The adapted analyzer can handle extinct morphological constructions as well as dialectal variants missing from Modern Standard Hungarian. The other example is an adaptation of the Hungarian morphology to the clinical domain, where the domain-specific terminology, which includes a vast amount of word forms of foreign origin, had to be treated in a robust manner.

Since in general, a disambiguated morphological analysis is needed in practical natural language processing tasks, the analyzer is seldom applied on its own. Rather, it is used as a component of a morphological tagging tool, generally built around a part-of-speech tagger. The performance of the computational morphology can thus be evaluated by measuring the quality of a morphological tagger tool solving this higher-level task, and specifically to what extent the integrated morphology is responsible for the quality of the combined morphological tagging tool. So, in order to demonstrate the utility of the adapted morphologies, we evaluate the tagging performance of a combined morphological tagging tool with a stand-alone data-driven tagger in these domains.

DOI:10.15774/PPKE.ITK.2015.013

5.2.1 Morphological annotation of Old and Middle Hungarian corpora

In order to be able to automatically analyze texts in Old and Middle Hungarian, the Humor morphological analyzer was extended to be capable of analyzing words containing morphological constructions, suffix allomorphs, suffix morphemes, paradigms or stems that existed in Old and Middle Hungarian but are no longer used in present-day Hungarian. A disambiguation system was also developed that can be used for automatic and manual disambiguation of the morphosyntactic annotation of texts. In addition, I created a corpus manager with the help of which the annotated corpora can be searched and maintained. The adapted morphology and the automatic and manual annotation tools were used in two parallel OTKA projects[i] of the Research Institute for Linguistics of the Hungarian Academy of Sciences since one of the major aims of these projects was is to create morphologically analyzed and searchable corpora of texts from the Old Hungarian and Middle Hungarian period.

5.2.1.1 Preprocessing

The overwhelming majority of extant texts from the Old Hungarian period are codices, mainly containing texts translated from Latin. The texts selected for the Corpus of Informal Language Use, however, are much closer to spoken language: minutes taken at court trials, such as witch trials, and letters sent by noblemen and serfs. In the case of the latter corpus, metadata belonging to the texts are also of primary importance, as these make the corpus fit for historical-sociolinguistic research.

All the texts selected for our corpora were originally hand-written. However, the basis for the digitized version was always a printed edition of the texts published earlier. The printed texts were scanned and converted to a character stream using OCR. This was not a trivial task, owing to the extensive use of unusual characters and diacritics. In the lack of an orthographic norm, each text applied a different set of characters; moreover, the printed publications used different fonts. Thus the only way to get acceptable results was to retrain the OCR program[ii] for each text from scratch since the out-of-the-box Hungarian language and glyph models of the software did not fit any of the texts. Subsequently, all the automatically recognized documents had to be manually checked and corrected, but even so, this workflow proved to be much faster than attempting to type in the texts.

The next step of preprocessing was normalization, i.e. making the texts uniform regarding their orthography and phonology. Normalization, which was done manually, meant modernization to present-day orthography. Note that this also implies differences in tokenization into individual words between the original and the normalized version. During this process, which also included segmentation of the texts into clauses, certain phonological dialectal variations were neutralized.

Morphological variation, however, was left untouched: no extinct morphemes were replaced by their present-day counterparts. Extinct allomorphs were also retained unless the variation was purely phonological. In the case of potential irresolvable ambiguity, the ambiguity was preserved as well, even if it was due to the vagueness of the orthography of the era.

An example of this is the non-consistent marking of vowel length. The definite and indefinite 3^{rd} person singular imperfect of the frequently used word *mond* 'say' was *mondá* $\sim$ *monda* respectively, but accents are often missing from the texts. Furthermore, in many texts in the corpus, these two forms were used with a clearly different distribution from their present-day (3^{rd} person singular past) counterparts *mondta* $\sim$ *mondott*. Therefore, in many cases, neither the orthography, nor the

[i] *Hungarian historical generative syntax* [OTKA NK78074], and *Morphologically analysed corpus of Old and Middle Hungarian texts representative of informal language use* [OTKA 81189]

[ii] I used FineReader, which makes full customization of glyph models possible, including the total exclusion of out-of-the-box models.

DOI:10.15774/PPKE.ITK.2015.013

usage was consistent enough to decide unambiguously how a certain appearance of *monda* should be annotated concerning definiteness.

Another example of inherent ambiguity is a dialectal variant of possessive marking, which is very frequent in these corpora and often neutralizes singular and plural possessed forms. For example, *cselekedetinek* could both mean 'of his/her deed' or 'of his/her deeds', which in many cases cannot be disambiguated based on the context even for human annotators. Such ambiguous cases were annotated as inherently ambiguous regarding number/definiteness etc.

Some of the Old Hungarian codices (Jókai (Jakab, 2002), Guary (Jakab and Kiss, 1994), Apor (Jakab and Kiss, 1997), and Festetics (Jakab and Kiss, 2001)) were not digitized using the OCR technique described above, as these were available in the form of historical linguistic databases, created by Jakab László and his colleagues between 1978 and 2002. However, the re-creation of the original texts out of these lexical databases was a difficult task. The first problem was that the locus of word token occurrences only identified codex page, column and line number in the databases, but there was no information concerning the order of words within a line. The databases also contain morphological analyses, but they were encoded in a hard-to-read numerical format, which occasionally was incorrect and often incomplete. Furthermore, the categorization was in many respects incompatible with my system. However, finally I managed to re-create the original texts. First the order of words was manually restored, and incomplete and erroneous analyses were fixed. Missing lemmas were added to the lexicon of the adapted computational morphology, and the normalized version of the texts was generated using the morphology as a word form generator. Finally, the normalized texts were reanalyzed to get analyses compatible with the annotation scheme applied to other texts in the corpora.

5.2.1.2 The morphological analyzer

The lexicon of lemmas and the affix inventory of the program were augmented with items that have disappeared from the language but are present in the historical corpora. Just the affix inventory had to be supplemented with 50 new affixes (not counting their allomorphs).

Certain affixes have not disappeared, but their productivity has diminished compared to the Old Hungarian era. Although words containing these morphemes are still present in the language, they are generally lexicalized items, often with a changed meaning. An example of such a suffix is *–At*, which used to be a fully productive nomen actionis suffix. Today, this function belongs to the suffix *–Ás*. The (now lexicalized) words, however, that end in *–At* mark the (tangible) result of an action (i.e. nomen acti) in present-day standard Hungarian, as in *falazat* 'wall' vs. *falazás* 'building a wall'.

One factor that made adaptation of the morphological model difficult was that there are no reliable accounts on the changes of paradigms. Data concerning which affix allomorphs could be attached to which stem allomorphs had to be extracted from the texts themselves. Certain morphological constructions that had already disappeared by the end of the Old Hungarian era were rather rare (such as some participle forms) and often some items in these rare subparadigms have alternative analyses. This made the formal description of these paradigms rather difficult.

However, the most time-consuming task was the enlargement of the stem inventory. Beside the addition of a number of new lemmas, the entries of several items already listed in the lexicon of the present-day analyzer had to be modified. The causes were various: some roots now belong to another part of speech, or in some constructions they had to be analyzed differently from their present analysis.

Furthermore, the number of pronouns was considerably higher in the examined period than today. The description of their extensive and rather irregular paradigms was really challenging as some forms were underrepresented in the corpora.

DOI:10.15774/PPKE.ITK.2015.013

Some enhancements of the morphological analyzer made during the corpus annotation projects were also applicable to the morphological description of standard modern Hungarian. One such modification was a new annotation scheme applied to time adverbials that are lexicalized suffixed (or unsuffixed) forms of nouns, like *reggel* 'morning/in the morning' or *nappal* 'daytime/in daytime', quite a few of which can be modified by adjectives when used adverbially, such as *fényes nappal* 'in broad daylight'. This latter fact sheds light on a double nature of these words that could be captured in an annotation of these forms as specially suffixed forms of nouns instead of atomic adverbs, an analysis that is compatible with X-bar theory (Jackendoff, 1977).

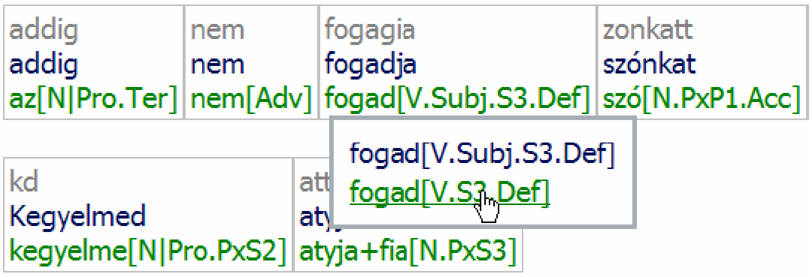
5.2.1.3 Disambiguation

With the exception of already analyzed sources (i.e. the ones recovered from the Jakab databases), the morphological annotation had to be disambiguated. The ambiguity rate of the output of the extended morphological analyzer on historical texts is higher than that for the standard Humor analyzer for present-day corpora (2.21 vs. 1.92[iii] analyses/word with an identical (high) granularity of analyses). This is due to several factors: (i) the historical analyzer is less strict, (ii) there are several formally identical members of the enlarged verbal paradigms including massively ambiguous subparadigms like that of the passive and the factitive,[iv] (iii) a lot of inherent ambiguities described above.

The workflow for disambiguation of morphosyntactic annotation was a semi-automatic process: an automatically pre-disambiguated version of each text was checked and corrected manually. For a very short time, I considered using the Jakab databases as a training corpus, but recovering them required so much development and manual labor and the analyses in them lacked so much distinction I wanted to make that I opted for creating the training data completely from scratch instead.

5.2.1.4 The manual disambiguation interface

To support the process of manual checking and the initial manual disambiguation of the training corpus, I created a web-based interface using JavaScript and Ajax where disambiguation and normalization errors can be corrected very effectively. The system presents the document to the user using an interlinear annotation format that is easy and natural to read. An alternative analysis can be chosen from a pop-up menu containing a list of analyses applicable to the word that appears when the mouse cursor is placed over the problematic word. Note that the list only contains grammatically relevant tags and lemmas for the word returned by the morphological analyzer. This is very important, since, due to the agglutinating nature of Hungarian, there are thousands of possible tags (see Figure 5.1).



addig	nem	fogagia	zonkatt
addig	nem	fogadja	szónkat
az[N|Pro.Ter]	nem[Adv]	fogad[V.Subj.S3.Def]	szó[N.PxP1.Acc]

fogad[V.Subj.S3.Def]
fogad[V.S3.Def]

kd	att
Kegyelmed	aty
kegyelme[N|Pro.PxS2]	atyja+fia[N.PxS3]

Figure 5.1: The web-based disambiguation interface

[iii] measured on newswire text

[iv] This ambiguity is absent from modern standard Hungarian because the passive is not used any more.

DOI:10.15774/PPKE.ITK.2015.013

The original and the normalized word forms as well as the analyses can also be edited by clicking on them. An immediate reanalysis by the morphological analyzer running on the web server can be initiated by double clicking the word.

As there is an inherent difference between the original and normalized tokenization, and because, there may remain tokenization errors in the normalized texts, it is important that tokens and clauses can also be split and joined using the disambiguation interface.

I designed the automatic annotation system bearing in mind the requirement that it should make it possible that details of the annotation scheme be modified in the course of work. One such modification was e.g. the change to the annotation of time adverbs mentioned in Section 5.2.1.2 above. The modified annotation can be applied to texts analyzed and disambiguated prior to the modification relatively easily, because the program chooses the analysis most similar to the previously selected analysis (based on a letter trigram similarity measure). Nevertheless, the system highlights all tokens the reanalysis of which resulted in a change of annotation, so that these spots can be easily checked manually. For changes in the annotation scheme where the simple similarity-based heuristic could not be expected to yield an appropriate result (e.g. when I decided to use a more detailed analysis of derived verb forms as before), a more sophisticated method was devised to update the annotations: old analyses were replaced using automatically generated regular expressions. These were created using a manually checked output of the morphological generator.

5.2.1.5 Automatic disambiguation

While the first few documents were disambiguated completely manually using the web-based tool, I soon started to train and use a tagger for pre-disambiguation applying the tagger incrementally, trained on an increasing number of disambiguated and checked text. First the HMM-based trigram tagger HunPos (Halácsy et al., 2007) was used. HunPos is not capable of lemmatization, but I used a straightforward method to get a full analysis: I applied reanalysis to the text annotated only by the tags assigned by HunPos using the automatic similarity-based ranking of the analyses. This approach yielded quite good results, but one problem with it was that the similarity-based ranking always prefers shorter lemmas. This was not appropriate for handling the case of a frequent lemma ambiguity for verbs with one of the lemma candidates ending in an *–ik* suffix and the other lacking a suffix (such as *dolgozik* ‘work’ vs. *(fel)dolgoz* ‘process’). Always selecting the *–ik*-less variant is not a good bet in the case of many frequent words in this ambiguity class.

Later, HunPos was replaced with another HMM-based trigram tagger, PurePos (Orosz and Novák, 2013), that has many nice extra features. It can process morphologically analyzed ambiguous input and/or use an integrated analyzer constraining possible analyses to those proposed by the analyzer or read from the input. This boosts the precision of the tagger dramatically in the case of languages like Hungarian and small training corpora. The fact that PurePos can be fed analyzed input makes it easy to combine with constraint-based tools that can further improve the accuracy of the tagging by handling long distance agreement phenomena not covered by the trigram model, or by simply removing impossible tag sequences from the search space of the tool.

PurePos can perform lemmatization, even for words unknown to the morphological analyzer (and not annotated on the input) learning a suffix-based lemmatization model from the training corpus along with a similar suffix-based tag guessing model, thus it assigns a full morphological analysis to each token. It is also capable of generating an n-best list of annotations for the input sentence when using beam search instead of the default Viterbi decoding algorithm.

DOI:10.15774/PPKE.ITK.2015.013

5.2.1.6 Disambiguation performance

I performed an evaluation of the accuracy of PurePos on an 84000-word manually checked part of the historical corpus using five-fold cross-validation with a training corpus of about 67000 words and a test corpus of about 17000 words in each round. The ratio of words unknown to the morphological analyzer in this corpus is rather low: 0.32%.

The average accuracy of tagging, lemmatization and full annotation for different versions of the tagger are shown in Table 5.5. In addition to token accuracy, I also present clause accuracy values in the table. Note that, in contrast to the usual way of evaluating taggers, these values were calculated excluding the always unambiguous punctuation tokens from the evaluation. The baseline tagger uses no morphological information at all. Its current lemmatization implementation uses suffix guessing in all cases (even for words seen in the training corpus) and selects the most frequent lemma, which is obviously not an ideal solution.

The disambiguator using morphology performs significantly better. Its clause-level accuracy is 81.50%, which means that only every fifth clause contains a tagging error. The tag set I used in the corpus differentiates constructions which are not generally differentiated at the tag level in Hungarian corpora, e.g. deictic pronouns (*ebben* 'in this') vs. deictic pre-determiners (*ebben a házban* 'in this house'). Many of these can only be disambiguated using long-distance dependencies, i.e. information often not available to the trigram tagger. Combination of the tagger with a constraint-based tool (see e.g. Huldén and Francom (2012)) would presumably improve accuracy significantly.

The rightmost column contains a theoretical upper limit of the performance of the current trigram tagger implementation using 5-best output and an ideal oracle that can select the best annotation.

		baseline	morph	5-best+oracle
token	Tag	90.17%	96.44%	98.97%
	Lem.	91.52%	98.19%	99.11%
	Full	87.29%	95.90%	98.53%
clause	Tag	62.48%	83.81%	93.99%
	Full	54.68%	81.50%	91.47%

Table 5.5: Disambiguation performance of the tagger

5.2.1.7 Searching the corpus

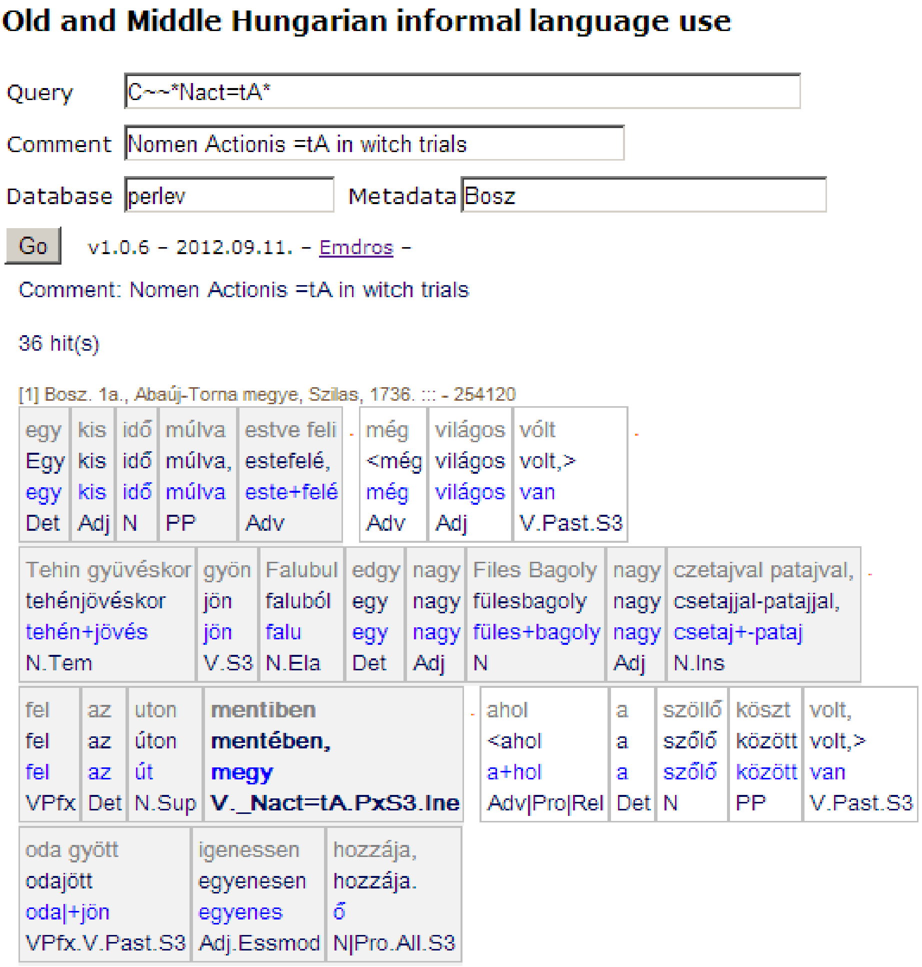
The web-based tool I created as a corpus query interface does not only make it possible to search for different grammatical constructions in the texts, but it is also an effective correction tool. Errors discovered in the annotation or the text appearing in the "results" box can immediately be corrected, and the corrected text and annotation is recorded in the database. Naturally, this latter functionality of the corpus manager is only available to expert users having the necessary privileges.

A fast and effective way of correcting errors in the annotation is to search for presumably incorrect structures and to correct the truly problematic ones at once. The corrected corpus can be exported after this procedure and the tagger can be retrained on it.

The database used for the corpus manager is based on the Emdros corpus manager (Petersen, 2004). In addition to queries formulated using MQL, the query language of Emdros, either typed in at the query box or assembled using controls of the query interface, advanced users can use a custom-made corpus-specific query language (MEQL), which makes a much more compact formulation of queries possible than MQL. It is e.g. extremely simple to locate a specific locus in the corpus: one simply needs

DOI:10.15774/PPKE.ITK.2015.013

to type in the sequence of words one is looking for. Queries formulated in MEQL are automatically converted to MQL queries by the query processor.



Old and Middle Hungarian informal language use

Query: C~~*Nact=tA*

Comment: Nomen Actionis =tA in witch trials

Database: perlev Metadata: Bosz

Go v1.0.6 - 2012.09.11. - Emdros -

Comment: Nomen Actionis =tA in witch trials

36 hit(s)

[1] Bosz. 1a., Abaúj-Torna megye, Szilas, 1736. ::: - 254120

egy	kis	idő	múlva	estve feli	még	világos	vólt
Egy	kis	idő	múlva,	estefelé,	<még	világos	volt,>
egy	kis	idő	múlva	este+felé	még	világos	van
Det	Adj	N	PP	Adv	Adv	Adj	V.Past.S3

Tehin gyüvéskor	gyön	Falubul	edgy	nagy	Files Bagoly	nagy	czetajval patajval,
tehénjövéskor	jön	faluból	egy	nagy	fülesbagoly	nagy	csetajjal-patajjal,
tehén+jövés	jön	falu	egy	nagy	füles+bagoly	nagy	csetaj+-pataj
N.Tem	V.S3	N.Ela	Det	Adj	N	Adj	N.Ins

fel	az	uton	**mentiben**	ahol	a	szöllő	köszt	volt,
fel	az	úton	**mentében,**	<ahol	a	szőlő	között	volt,>
fel	az	út	**megy**	a+hol	a	szőlő	között	van
VPfx	Det	N.Sup	**V._Nact=tA.PxS3.Ine**	Adv|Pro|Rel	Det	N	PP	V.Past.S3

oda gyött	igenessen	hozzája,
odajött	egyenesen	hozzája.
oda|+jön	egyenes	ő
VPfx.V.Past.S3	Adj.Essmod	N|Pro.All.S3

Figure 5.2: The query interface

The search engine makes it possible to search inside sentences, clauses, or texts containing grammatical constructions and/or tagged with metadata matching the criteria specified in the query. Units longer than a sentence can also be searched for. The context displayed by default for each hit is the enclosing sentence with focus words highlighted. Clauses may be non-continuous. This is often the case for embedded subordinate clauses. But the corpus also contains many injected parenthetical coordinate clauses and many examples where the topic of a subordinate clause precedes its main clause with the net effect of the subordinate clause being interrupted by the main clause. The query example in Figure 5.2 shows a sentence containing several clauses with gaps: the clauses enclosed in angle brackets are wedged between the topic and comment part of the clauses which they interrupt. Emdros is capable of representing these interrupted clauses as single linguistic objects with the interrupting clause not being considered part of the interrupted one.

5.2.1.8 Availability of the resulting corpus

The extended morphological analyzer is used for the annotation of the constantly growing Old and Middle Hungarian corpora. Part of these corpora are already searchable by the public. The Old Hungarian Corpus is available at `http://omagyarkorpusz.nytud.hu`, while the analyzed part of the Historical Corpus of Informal Language Use can be searched at `http://tmk.nytud.hu`.

DOI:10.15774/PPKE.ITK.2015.013

5.2.2 Extending the lexicon of the morphological analyzer with clinical terminology

In this section, I describe the methods by which the database of the contemporary Hungarian morphological analyzer was extended for better coverage of the medical domain. Methods similar to the ones described here can also be applied when the coverage of the specific terminology of other domains need to be improved.

Processing clinical texts is an emerging area of natural language processing. Even though there are some algorithms used for parsing English clinical notes, these cannot be applied to Hungarian medical records. Moreover, NLP tools for general Hungarian perform poorly when applied as they are. This is due to the special characteristics of clinical texts. Such records are created in a special environment, i.e. in the clinical settings, thus they differ from general Hungarian in several respects. These attributes are the following (cf. Orosz et al. (2013); Siklósi and Novák (2013); Siklósi et al. (2012)):

- notes contain a lot of erroneously spelled words,
- sentences generally lack punctuation marks and sentence-initial capitalization,
- punctuation is often erroneous when present,
- measurements are frequent and have plenty of different (erroneous) forms,
- a lot of (non-standard) abbreviations occur in such texts,
- and numerous Latinate medical terms are used.

In order to process such texts, the morphological analyzer had to be adapted to the requirements of the domain. In order to achieve a performance comparable to that obtained in the case of general Hungarian texts, the lexicon of the analyzer had to be extended.

The primary source for the extension process was a spelling dictionary of medical terms (Fábián and Magasi, 1992), which contains about 90,000 entries. This dictionary does not contain any information about the part-of-speech, language or the pronunciation of these words. However, when adding them to the morphological database, this information was necessary. In addition, I had to determine the compound boundaries for compound words. Since the number of words to be manually annotated was several thousands, the process of categorization and definition of additional information had to be aided by automated methods.

Assignment of part-of-speech was based first on surface form features (e.g. names and abbreviations in the dictionary could be separated from other words based on such characteristics). Second, after having a portion of the words manually categorized, I trained the guesser algorithm used in the TnT (Brants, 2000) and PurePos (Orosz and Novák, 2012) taggers on this set. Then, this model was applied iteratively to the rest of the words, which were manually checked in each step.

In the case of Latinate words with certain endings, it was quite difficult to decide whether it was a noun or an adjective, or could be used in both ways. In order to be able to efficiently categorize these words, another aspect was considered: in the case of multiword Latinate terms, the last element is usually an adjective (unless it is a possessive phrase), while the first one is usually a noun. The order of the elements is thus systematically different from that of Hungarian noun phrases. The difficulties of annotating Latin adjectives arise from this phenomenon, and such words are quite frequent in Hungarian clinical texts. The corresponding Hungarian form of these words (which are the phonetic transliteration of the masculine nominative form in Latin) is definitely an adjective, thus being in the usual adjective–noun order. In real Latin multiword phrases, the order is noun–adjective and the two elements are in agreement. Latin adjectives being members of non-masculine or non-nominative phrases are to be considered nouns from the aspect of Hungarian categorization. In theory, this would be the case for masculine nominative phrases as well, if the corpus was not full of instances of phrases

DOI:10.15774/PPKE.ITK.2015.013

which follow the ordering of Hungarian nominal phrases, but are constructed from words written according to Latin orthography (at least partially), as shown in Table 5.6.

Latin NP word order	**Hungarian NP word order**
Degeneratio *marginalis* pellucida corneae	*marginalis* degeneratio
ulcus *marginalis*	alsó szemhéj *marginalis* részéhez közel
Cataracta *progrediens*	*progrediens* maghomályok
Membrana *epiretinalis*	*epiretinalis* membrán, Bal szem *epiretinalis* membranja
keratitis *superficialis* punctata	egy *superficialis* basalsejtes carcinoma

Table 5.6: Latinate adjectives used in Hungarian NP's using Latin orthography – examples from the ophthalmology corpus

Thus, I decided to assign distinctive tags in the lexicon to nouns and adjectives written according to Latin orthography, tagging masculine nominative adjectives as adjectives and the rest as nouns. These lexical items got a special tag in addition, marking their Latinate spelling.

Besides assigning part-of-speech information, elements written according to foreign or Hungarian orthography had to be differentiated. This was also necessary in order to determine the pronunciation of foreign words so that they would be affixed properly. The dictionary itself helped this process by containing several word pairs being spelling variants of the same word. In most cases, one of these variants is the Hungarian form, the other is the foreign one.

In most cases, the Hungarian variant was marked in the dictionary as the preferred form, however, there were many exceptions as well. After performing a partial categorization manually, an adapted version of the TextCat algorithm (Cavnar and Trenkle, 1994) was applied, which is able to decide about short strings whether these are in Hungarian or not. The situation was quite clear when this system qualified one member of the word pair as rather Hungarian, while the other as rather foreign. However, many of the word pairs are such that both members are foreign spelling variants. The TextCat algorithm was also able to filter these entries. Thus, this language identification method was integrated into the iterative lexicon extension procedure. The dictionary contained some other words of foreign origin (mainly Latinate terms, but there were many terms of English and French origin as well), which did not have their transliterated Hungarian equivalent in the dictionary. These had to be identified, but in these cases, I could not rely on any implicit extra information, which was available in the case of word pairs.

After having decided whether an entry is in Hungarian or not, the actual pronunciation had to be assigned to foreign words. In the case of word pairs, it was partially given, but in most cases, beside the Hungarian version listed in the dictionary, another transliteration of the Latin term was also necessary, especially for words ending in *s*. The reason for this is that for multiword phrases, it is always the Latin pronunciation that determines affixation (the same often applies to standalone words as well). The assignment of pronunciation was also done algorithmically and was corrected manually. This task was not solved by a traditional G2P (grapheme-to-phoneme) machine learning algorithm, but by a simple regular-expression-based heuristic method. This can be invoked directly from the editor used for creating the lexicon and the input can be blocks of highlighted words (if, for example, some are found to be foreign words not categorized as such by the language detection module).

Another task was the identification of compound boundaries with a special emphasis on identifying elements frequently being members of compounds. These were prioritized when processing the dictionary, thus compounds including such words could already be analyzed by the morphology, decreasing the amount of entries to be processed and the chance of inconsistency when entering data manually. The procedure for finding compound members was the following. Words from the general

DOI:10.15774/PPKE.ITK.2015.013

Hungarian spelling dictionary and the medical spelling dictionary that were at least two character long and contained at least one vowel were stored in a trie data structure. Then, the implemented algorithm searched for these as postfixes in the words of the dictionary and built statistics from the elements of these words. The prefixes found were marked by several features: whether having a length less than 4 characters, being included as a word in the dictionary, containing a hyphen, and being postfixes of other words. Using the result of this classification and the manually checked suspicious compounds, the most frequent pre- and postfixes were added to the lexicon first. Then the real compounds produced by these elements were also added, thus adding all compounds with a representation indicating compound boundaries.

Surprisingly, the dictionary contained a great amount of words derived from verbs (mainly participles and nomen actionis), of which the base verb (mainly derived from Latinate stems) was not included. Instead of including these derived words, the base form was added to the lexicon. Thus, the analyzer can generate a proper analysis for the derived forms. Moreover, there were many adjectives with the derivational suffix *-s*, of which the base form was also in the dictionary. These were also skipped, because adding the base form to the dictionary resulted in adding the derived word automatically.

Another aspect of processing the dictionary was that it included a number of typographical errors, thus the data found in this resource could not be considered as totally reliable.

Beside the spelling dictionary, another important set of words was taken from the database of medications and active ingredients downloaded from the webpage of OGYI [v]. Here, the part-of-speech categorization of words was not a problem. However, assigning the proper pronunciation was rather important. The algorithm producing these had to be adapted, because even though the names of the active ingredients are composed of Latinate elements, but their written form corresponds to the English spelling of Latinate terms of the original Latin spelling. These often end in an unpronounced *-e*, e.g. *hydroxycarbamide, ifosfamide, cyclophosphamide.*

The third resource was of course the corpus itself. Words occurring in the corpus were prioritized when processing the spelling dictionary. But, after including the two external resources into the lexicon of the analyzer, there were still some frequent words in the corpus that could not be analyzed, thus these had to be added, too. Most of these words were abbreviations. The resolution of the abbreviations, that was necessary for defining the correct part-of-speech, was done based on corpus concordances.

From the medical dictionary and the corpus 36,000 entries were added to the stem lexicon of the analyzer (another 25,000 have not been processed yet). From the database of names of medicines and active ingredients, 4860 entries were added.

The morphological analyzer was then integrated into the part-of-speech tagger. The precision of the latter system with the extended version of the morphology was 93.25%. The difference between using the original and the extended morphological analyzer was significant, i.e. 6.4%.

Investigating the errors still present, I found that the most frequent ones are not relevant from the aspect of further processing these texts syntactically or semantically, e.g. cases when the morphology makes a distinction between nouns and adjectives of Latin or Hungarian origin, or the distinction of participles and the corresponding lexicalized adjectives. Not considering these errors, the precision of the part-of-speech tagger was 93.77% with the extended morphology.

For the application of the morphology to clinical texts, see also Section 8.5.

[v] `http://www.ogyi.hu/listak/`

DOI:10.15774/PPKE.ITK.2015.013

5.3 Examples from other Uralic languages

Beside the national languages spoken by several million speakers: Hungarian, Finnish and Estonian, the Uralic language family includes a number of minority languages with significantly smaller speaker communities, the majority of which are spoken on the territory of the Russian Federation. The goal of various projects I participated in was to create computational morphologies and annotated corpora for several of these languages: Udmurt, Komi, Eastern Mari, Northern Mansi, Synya and Kazym Khanty, Tundra Nenets and Nganasan. Table 5.7 presents information concerning the alternative names of the languages, their geographical distribution, the estimated number of speakers[vi] and the branch to which they belong within the language family.

Language	A.k.a.	Geographical distribution	Speakers	Language branch
Komi	Zyrian	Komi Republic, west of the Urals	156,000	Permic (Finnic)
Udmurt	Votyak	Udmurtia, west of the Urals	460,000	Permic (Finnic)
Eastern (Low) Mari	Cheremis	Mari El Republic, by the Volga	480,000	Volgaic (Finnic)
Northern Mansi	Vogul	west of the Urals, between the Urals and the Ob River	<1,000	Ugric
Synya Khanty	Ostyak	at the Synya tributary of the Ob River	<9,600	Ugric
Kazym Khanty	Ostyak	at the Kazym tributary of the Ob River	<9,600	Ugric
Nganasan	Tavgi	Taymyr Peninsula, North Siberia	200	Northern Samoyedic
Tundra Nenets	Yurak	Northwest Siberia	22,000	Northern Samoyedic

Table 5.7: The languages and dialects covered by the Uralic projects

As is evident even from the number of speakers, Mansi, Khanty and Nganasan are on the verge of extinction. In their case, the documentation of the language and any remnants of the oral cultural heritage of these peoples is an urgent scientific task.

One aim of this research was to make linguistic data concerning these languages available for research to a broader community of linguists, not only the Uralist specialists, and to make corpus-based investigation of these languages possible. Many of these languages exhibit phenomena that would be exciting to explore for a variety of linguists, such as theoreticians specializing in any module of grammar or those interested in language typology. Annotated corpora make it possible to carry out research on various aspects of the language without a long preliminary study of the language itself. Since many details of the description which often remain vague in written grammars must unavoidably be made explicit in a computationally implemented grammar, the process of creating the implementations as well as the resulting programs themselves shed light on inconsistencies and gaps in the available descriptions of the phonology and morphology of the language, and often help correcting them. Moreover, while examining linguistic models with regard to exactness and completeness by hand is an impossible task, the computational implementation makes an exhaustive testing of the adequacy of our grammatical models possible against a great amount of real linguistic data. Systematic comparison of word forms generated against model paradigms has pinpointed errors not only in the computational implementation (which were then eliminated) but also in the model paradigms or the grammars the computational implementation was based on.

[vi] The number of all Khanty speakers is about 9,600 according to 2010 census data.

DOI:10.15774/PPKE.ITK.2015.013

Another fact makes a more thorough documentation of these languages urgent. Due to the nature of Russian minority policy, the school system, the great degree of dispersion, the low esteem of the ethnic language and culture and the general lack of an urban culture of their own, all these languages are endangered. On the other hand, there are significant differences among these languages concerning the number of speakers and the exact sociolinguistic situation they are in.

Some of the languages can be categorized as moribund, with virtually no chance of the language still being spoken in another 50 years. This is not only due to the low number of speakers (some of these languages have existed and developed as the communication medium of small nomadic communities of about a thousand people for thousands of years without an immediate risk of disappearance), but because one generation of speakers has already failed to pass on the language to the next and thus hardly any children speak it. In the case of these languages, the most we can do is trying to document as much of the language as possible. Documenting these languages is not a trivial task though, not only because of the extreme complexity of some of them (e.g. in terms of their morpho-phonology), but also because the speaker communities are disintegrating into a small assembly of individuals with more and more uncertain language skills. A heavy influence from their parallel knowledge of the majority language, Russian, seems to impact not only the syntactic structures they use, but even the morpho-phonology.

But these languages are not only very difficult to learn for anybody but babies, but they are not considered very useful to know, either. They have lost much of their function when these nomadic peoples were forced to settle as a minority in settlements inhabited by people speaking another language and to give up their traditional way of life, their rituals and practices. Their tame reindeer herds were collectivized (which subsequently fell victim to epidemics), and they were practically prohibited from reindeer hunting. But the fatal blow on these languages was the schooling of minority children in boarding schools hundreds of kilometers away from their home where the language of education was exclusively Russian. The children had no contact at all with their parents and their home community during the school year, and both their knowledge and their esteem of their mother tongue deteriorated significantly. This was the generation that growing up failed to pass on the language to their children.

There is another factor that makes the documentation of some of these languages difficult. During the Soviet era, making field trips to areas where many of these small minority languages are spoken was only possible for linguists from within the Soviet Union. In the nineties, during the Yeltsin era, an unprecedented freedom of movement made it possible also for foreign linguists to travel freely to the areas previously inaccessible to them and to do research there. Fortunately, this is still true for many areas (such as the region of the River Ob, where the Mansi and Khanty live). Certain areas of the northern Arctic regions where some of these minority languages are spoken, however, (the Taymyr Peninsula in particular, where the Nganasans live) have unfortunately been declared divisions of restricted access again. Foreign linguists intending to do field work in the region must apply for an entrance permit at the local security authorities, which they may fail to issue. This might make it necessary to find alternatives to field trips such as carrying native speakers to places accessible for the researchers as well.

Another group of the languages mentioned do not seem to be threatened by an immediate language death, but even within this group there are significant differences. Although Udmurt and Mari have a similar number of speakers according to the census data, Mari seems to have a different sociolinguistic status than Udmurt due to the native speakers' different attitude toward their mother tongue. While the Mari are proud of their language and their cultural heritage, Udmurts have a rather low esteem of their mother tongue, which they consider inferior to Russian. On the other hand, Maris tend to have more conflicts with the Russian majority than Udmurts for the same reason.

In the case of these languages, the computational tools I created can also be adapted for practical purposes, such as providing the speaker communities with spell checkers and electronic dictionaries in

DOI:10.15774/PPKE.ITK.2015.013

their native language in the hope that the existence of such applications can help to raise the prestige of these languages.

These languages, being members of the Uralic language family, are of the agglutinating type, thus their **morphology** is characterized by the relatively high frequency of words containing long suffix sequences.

The following example is from Udmurt.

jaratonoosynyz ‘with the sweethearts (ones in love)’

jarat	on	o	os	yny	z
to love	nomen acti = love	having love	plural	instr.	def.

The high number of productive suffixes and possible suffix positions results in a combinatorial explosion of the number of possible word forms (yielding several thousands) for each stem in the open word classes. In some of the languages (e.g. Mari and Nganasan) certain suffixes (clitics) can assume a wide variety of positions within the suffix sequence.

The following corpus examples are from Mari. Both the form (*wlak* vs. šaməč) and the position of the plural suffix relative to other suffixes (whether it precedes or follows other inflectional endings) exhibit variation:

jeŋ[N]+že[Def]+[NOM]+-wlak[Pl]	‘the people’
artist[N]+kə[COM]+že[Def]+-wlak[Pl]	‘with the artists’
šydər[N]+-wlak[Pl]+še[Def]+[NOM]	‘the stars’
jeŋ[N]+-wlak[Pl]+lan[DAT]	‘for people’
pašajeŋ[N]+-šaməč[Pl]+ən[GEN]	‘of workers’

Table 5.8 summarizes properties of the morphologies created in this research. The size of the affix lexicons is indicated as a number of stem morphemes and lexicalized morpheme sequences in the source lexicon. Some lexicons also contain glosses, in that case, the number of different senses in the lexicon is also given in parentheses. There are three versions of the Northern Mansi analyzer, using three different transcriptions and based on three different lexical resources: Mansi (Chr. Vog.) is based on Kálmán (1963), Mansi (WT) on Kálmán (1976), and Mansi (VNGY) on Munkácsi (1892) and Munkácsi (1986). See further details in Section 8.4.

Language	**Stem lexicon**		**Affix lexicon**
	lemmas	**(senses)**	(UR entries)
Komi$_1$	2,100		156
Komi$_2$	37,000		193
Udmurt	14,100	(18,500)	286
Mari	2,200		189
Mansi (Chr. Vog.)	1,400	(1,600)	271
Mansi (WT)	3,778	(4,230)	376
Mansi (VNGY)	11,240	(16,600)	300
Kazym Khanty	1,800	(2,100)	150
Synya Khanty	3,100	(3,540)	150
Nganasan	4,150		334
Tundra Nenets	19,500		254

Table 5.8: Properties of the morphologies

The rather complex morphological makeup of words is a manifestation of the agglutinating nature of all languages belonging to the Uralic language family. If it were just simple concatenation that happens to morphemes making up a word, creating a formal grammar describing the morphology of these languages would not be a difficult task even in spite of all the variation of suffix ordering that occurs e.g. in the Permic languages or Mari. The following examples from Udmurt show that the order of possessive and case suffixes is different depending on the case.

kyšno[N]+je[PSS1]+ly[DAT]	‘to my wife’
ares[N]+a[INE]+m[PSS1]	‘in my age’

In Komi, there are even cases where there is free variation, or the order depends on both the case and the possessive suffix:

along my man (transitive case)	
mortöjti	mort[N]+öj[PSS1]+ti[TRA]
morttiym	mort[N]+ti[TRA]+ym[PSS1]
without my/your man (caritive case)	
mortöjtög	mort[N]+öj[PSS1]+tög[CAR]
morttögyd	mort[N]+tög[CAR]+yd[PSS2]
towards my/your/etc. man (approximative case)	
mortöjlań	mort[N]+öj[PSS1]+lań[APP]
mortlańyd	mort[N]+lań[APP]+yd[PSS2]
mortlańys	mort[N]+lań[APP]+ys[PSS3]
mortlańnym	mort[N]+lań[APP]+nym[PSP1]
mortnydlań	mort[N]+nyd[PSP2]+lań[APP]
mortlańnys	mort[N]+lań[APP]+nys[PSP3]

Linguists dealing with Finno-Ugric and in general with Uralic languages outside Russia tend to use Latin based **phonological transcriptions** instead of the eventual Cyrillic orthographies of the languages. Since the tools we created were intended for linguists, we decided to use a Latin-based phonological notation in the morphologies instead of the standard Cyrillic orthographies of the languages. As a result, the tools cannot be applied to orthographic input directly, only with an intermediate converter, which makes the operation of the analyzer less efficient in terms of speed. With the exception of the Tundra Nenets analyzer, the mapping between the orthographic forms and our representation is rather straightforward. In the case of Tundra Nenets, I used Tapani Salminen’s phonological notation (Salminen, 1997), which is less phonetic than the standard orthography.

Although it might not have been anticipated by the initiators of the project, an immediate result was that the process of the creation of these inevitably completely formalized language descriptions itself shed light on many gaps, uncertainties and errors in the textbook grammars on which the computational grammars were based on. Furthermore, the fact that the implemented morphologies could be tested against real language data in the form of corpora made it possible that we could improve the linguistic descriptions of these endangered languages. There is also hope that the uncertainties discovered during the development and validation process of these computational grammars will induce further field research. In addition to that, the morphological analyzers can be utilized in the process of semiautomatic annotation of corpora that can be effectively used in the research of other aspects of these languages, among others their syntax.

In the following sections two morphologies are described in detail. The Komi analyzer was implemented using the Humor models, while that for the Samoyedic language, Nganasan, was implemented using finite-state models.

5.3.1 The Komi alanyzer

Komi (or Zyryan, Komi-Zyryan) is a Finno-Ugric language spoken in the northeastern part of Europe, West of the Ural Mountains. The number of speakers is about 156,000. Komi has a very closely related language, Komi-Permyak (or Permyak, about 63,000 speakers), which is often called a dialect, but with a standard of its own. As a minority language spoken in Russia, Komi is an endangered language. Although it has an official status in the Komi Republic (Komi Respublika), this means hardly anything in practice. The education is in Russian, children attend only a few classes in their mother tongue. A hundred years ago, 93% of the inhabitants of the region were of Komi nationality. Thanks to the artificially generated immigration (industrialization, deportation) their proportion is under 25% today.

Komi is a relatively well documented language. The first texts are from the 14th century, and there is a great collection of dialect texts from the 19th and 20th centuries. There are linguistic descriptions of Komi from the 19th century, but hardly anything is described in any of the modern linguistic frameworks.

5.3.1.1 Creating a Komi Morphological Description

The first piece of description created was a lexicon of suffix morphemes along with a suffix grammar, which describes possible nominal inflectional suffix sequences. One of the most complicated aspect of Komi morphology is the very intricate interaction between nominal case and possessive suffixes.

Another problem was that none of the linguistic descriptions we had access to describes in detail the distribution of certain morphemes or allomorphs. In some of these cases I managed to get some information by producing the forms in question (along with their intended meaning) with the generator and having a native speaker judge them. In other cases I tried to find out the relevant generalizations from the corpus.

Then, the stem lexicon was created along with the formal description of stem alternations triggered by an attached suffix. Fortunately, all of the stem alternations are triggered by a simple phonological feature of the following suffix: that it is vowel initial. The alternations themselves are also very simple (there is an l∼v alternation class and a number of epenthetic classes).

```
töv[N];stemalt:LV;  *töl+ös[ACC]
kyv[N];stemalt:Jep; *kyvj+ön[INS]
oš[N];stemalt:Kep;  *ošk+ös[ACC]
un[N];stemalt:Mep;  *unm+ön[INS]
göp[N];stemalt:Tep; *göpt+yn[INE]
kov[V];stemalt:LV;  *kol+ö[PrsSg3]
lok[V];stemalt:Tep; *lokt+as[FutSg3]
jul[V];stemalt:Yep; *july+ny[Inf]
```

Figure 5.3: A list of all nominal and verbal alternation classes in Komi

On the other hand, it does not seem to be predictable from the (quotation) form of a stem whether it belongs to any of the alternation classes. This information must therefore be entered into the stem lexicon. Figure 5.3 contains a list of all nominal and verbal alternation classes with an example for each of them from the stem lexicon with a comment containing an example of a suffixed form where the stem undergoes the alternation. These are the actual entries representing these stems in the stem lexicon. The quotation form is followed by a label indicating its syntactic category and

DOI:10.15774/PPKE.ITK.2015.013

its unpredictable idiosyncratic properties (in this case the stem alternation class it belongs to). For regular stems only the lexical form and the category label has to be entered.

Irregular suffixed forms and suppletive or unusual allomorphs can be entered into the lexicon by listing them within the entry for the lemma to which they belong. The following example shows the entry representing a noun which has an irregular plural form.

```
pi[N];rr:!Pl; ++!pi+jan[PL];rr:(Cx|Px);
```

The entry defines the noun *pi* 'boy, son', which requires that the morph following it should not be the regular plural suffix (which is *-jas*) and introduces the irregular plural form *pijan*, which in turn must be followed by either a case marker or a possessive suffix.

In Komi, personal pronouns are inflected for case, while reflexive pronouns are inflected for case, number and person. Locative case suffixes can be attached to postpositions and adverbs. Certain parts of these paradigms are identical to that of regular nominal stems, but there are also idiosyncrasies. Especially among the forms of reflexive pronouns there are very many idiosyncratic ones. We handled regular subparadigms by introducing lexical features and having the analyzer process the corresponding word forms like any regular suffixed word. Idiosyncratic forms, on the other hand, were listed in the lexicon along with their analysis.

The first version of the Komi morphology contained only the vocabulary of the small corpus we managed to acquire, but later we got the dictionary of Beznosikova (2000) from the author in a digital form, which I parsed and converted into a stem database. The second version of the analyzer, which can directly analyze Cyrillic input, is primarily based on the vocabulary of this dictionary.

5.4 Finite-state implementation of Samoyedic morphologies

The morphology of some languages turned out to be particularly hard to model using the formalism presented in earliear chapters. The two Samoyedic languages: Tundra Nenets and Nganasan have a particularly complex phonology with a great abundance of very productive and quite complex phonological and surface phonetical processes. This makes not only the implementation of a computational model of the morphology of these languages very difficult, but poses a serious problem even for the linguists trying to do field work and gather linguistic data concerning these languages in a consistent notation, and for the linguists trying to create any acceptable grammar of them. Morphemes in Nganasan and Nenets tend to have numerous surface forms (allomorphs), which hardly resemble each other. Thus what is common in all of the surface forms of a morpheme (its 'underlying form') is necessarily something very abstract, which in turn hardly resembles any of the allomorphs.

Due to their high complexity, it is very difficult to construct an adequate description of the phonology and morphology of the Samoyedic languages, and the first such descriptions appeared only quite recently (Salminen, 1997; Helimski, 1998; Wagner-Nagy, 2002).

In order to be able to model the morphology of these languages, I used the finite-state formalism of the Xerox programs described in Chapter 6, after a failed attempt to implement it using the adjacency-based formalism presented in Chapters 3 and 4.

DOI:10.15774/PPKE.ITK.2015.013

5.4.1 Nganasan

A formal description of Nganasan was written by fellow linguists, Beáta Wagner-Nagy, Zsuzsa Várnai and Sándor Szeverényi (Wagner-Nagy, 2002). In this study, Nganasan phonology is described as a set of context dependent re-write rules of the form widely used by generative phonologists (Várnai, 2002). They also digitized a Russian-Nganasan dictionary (Kost'erkina et al., 2001) and converted it to the phonemic transcription based on Latin script used by their team. The dictionary contains approximately 3650 non-derived roots. The Nganasan team also provided category labels for each item, which was missing from the original source. This dictionary was to serve as a basis for the stem database of the analyzer. The description (Wagner-Nagy, 2002) also contains some short texts which could be used as a corpus along with a collection of text from other sources. Later another 500 roots were added encountered when testing the analyzer on this corpus.

The dictionary was converted to the stem database format of the morphological grammar development environment. I transformed the text which was originally printed using special accented and phonetic characters from various fonts to the transcription used by the analyzer where digraphs are used for the palatal segments (`t1, d1, s1, n1, l1`) and the velar nasal (`ng`). In addition, the schwa phoneme is denoted by `e`, the full *e* vowel, which only occurs in initial syllables of words, by `@`, the glottal stop by `'`, and the voiced dental fricative by `q`.

During the preparation of this stem dictionary, we also started to describe the suffixes of Nganasan in a formal manner. The first step of this was the creation of a list of the suffixes that contained the underlying phonological form of each suffix together with its category label, plus a feature that indicates which morphological root form the suffix can attach to. We used the following model to describe Nganasan morphology: following Helimski (1998), we hypothesized that each root morpheme has three morphological stem variants (out of which two or all three might have the same form), and suffixes are sorted into three groups depending on which root allomorph they attach to. Some suffixes (such as the Lative case marker) exhibit vacillating behavior: they can attach to two of the root allomorphs.

The underlying phonological representation contains a number of abstract archiphonemes: harmonic vowels (in the case of suffixes the quality of these vowels depends on the harmonic features of the root they are attached to) and 'quasi-consonants' which never appear on the surface but condition gradation, one of the key processes in Nganasan phonology. Of these abstract segments, harmonic vowels appear only in suffixes, the other abstract segments turn up both in the suffix and the root lexicon, most notably in stems which are irregular and in inflectional (word final) suffixes. The first suffix list compiled contained additional information for derivational suffixes: the category label was given for the root it attaches to and for the derived form as well.

The next step was to convert the suffix list into a format that was compatible with the development environment: I refined the description and added data required by the formalism.

The selectional restrictions between the root and the suffix (i.e. 1^{st}, 2^{nd} or 3^{rd} root) were described as left-hand side requirements of the suffix. Other left-hand side constraints on roots were used primarily in the case of verbal suffixes: suffixes attaching to perfective or imperfective verbal roots, suffixes of verbs requiring an Agent, suffixes attaching to transitive verbs etc.

I also described the morphotactic restrictions governing the linear order of suffixes by defining a suffix grammar. I set up morphotactic classes for suffixes: e.g. Possessive suffixes in Nominative/Accusative (`NomPx`), Possessive suffixes used in other cases (`Px`), Oblique case endings (`OblCx`), Nominal Predicative suffixes (`NVx`) etc. I defined a finite-state automaton whose edges are labeled by the names of the morphotactic classes and which describes the possible linear order of the morphemes belonging to these classes. In the present model, the possible order of derivational morphemes is only constrained by the syntactic category of the base morpheme and that of the derived form.

DOI:10.15774/PPKE.ITK.2015.013

5.4.1.1 The complexity of Nganasan morpho-phonology

Having written the suffix inventory, the rules governing root allomorphy were to be defined. The Humor development environment prepares the allomorph database using these rules and the morpheme inventories.

In Nganasan, nominal and verbal roots follow different alternation patterns. Additionally, vowel-final and consonant-final roots also exhibit different behavior. Some root-final changes are restricted to lexically marked root classes. Each of these roots must have a relevant lexical mark in the root inventory. Other root-final changes occur in each root that satisfies the formal requirements of the rule.

It was relatively easy to describe root-final sound alternations in the Humor formalism. Productive phonological processes that are sensitive to local contexts (such as degemination) could be formalized as separate rules. However, the phenomenon of gradation (i.e. the systematic alternation of obstruents in syllable onsets) proved to be so complex that I could not describe it satisfactorily. The root of the problem is that the Humor analyzer sees each word as a sequence of allomorphs and during analysis it checks whether the adjacent morphs are locally compatible with each other.

Nganasan gradation, however, does not depend on the morphological make-up of the word: the only factor at play is syllable structure. Syllable boundaries and morph boundaries do not usually coincide. In the case of short suffixes (made-up of a single segment), it is possible that even non-adjacent morphs belong to the same syllable. Moreover, the rules governing gradation in Nganasan are quite intricate. An obstruent in the onset position is in strong grade (i) in even-numbered open syllables (if not preceded by a long vowel) and (ii) if it is preceded by a non-nasal coda consonant. Otherwise, it is in rhythmical weak grade (i) if preceded by a long vowel (or vowel sequence) or (ii) if it is in odd-numbered syllable. Otherwise, it is in syllabic weak grade in even-numbered closed syllables. Gradation combines with other alternations in the language: vowel harmony, degemination, root alternations and various morpho-phonological suffix alternations (as a result of which a monosyllabic suffix can have as many as 32 different allomorphs). Moreover, there are also apparent lexical exceptions to the general gradation patterns. These words exhibit gradation patterns which look as if there were a consonant at a point in the word where there is in fact none. Assuming underlying abstract consonants in these words is motivated by the fact that the exceptions include a number of words in which historical language data indicate that these consonants were actually present at an earlier stage of the history of the language, but they later disappeared on the surface due to a diachronic process. The exceptions also include loan words (including names) which did not fully assimilate to the native stock of words. Assuming underlying abstract segments in these words is only motivated by analogy.

As an illustration, the examples in Table 5.9 demonstrate the purely phonological allomorphy of a single verbal mood suffix (of narrative mood used in the subjective and the non-plural objective conjugations) in Nganasan. Each of the rows of Table 5.9 shows a Nganasan word form segmented into a stem, followed by the narrative mood suffix and a subject agreement ending with a gloss added at the end. Each of the word forms contains a different allomorph of the mood suffix. The superscript letters indicate the lexical vowel harmony class of the stems (I: unrounded, U: rounded).

The underlying representation of the morpheme is `hA2nhV`, and its 12 allomorphs are: *banghu, bjanghy, bambu, bjamby, bahu, bjahy, hwanghu, hjanghy, hwambu, hjamby, hwahu, hjahy.* These allomorphs are produced from the underlying representation by the general phonological processes of the language, undergoing vowel harmony, *a*-diphthongization and gradation, as follows.

The harmonic vowel `A2` surfaces as *a* or the diphthong *ja* as a result of root dependent roundness harmony. *a* diphthongizes to *wa* when it follows a *h*. (Roots are sorted into lexical classes depending on their harmonic features. This feature must be marked in the lexicon as it is totally arbitrary. Stem

DOI:10.15774/PPKE.ITK.2015.013

stem	narr. sfx (`hA2nhV`)	subj. agr.	'the rumor is that...'
i^U	bahu	[Sg3]	he is
aukum^U	hwahu	[Sg3]	he tames sg.
ngungkegimtü^U	banghu	[Sg3]	he increases sg.
ngungkegimtü^U	bambu	ng[Sg2]	you increase sg.
nguem^U	hwanghu	[Sg3]	he attaches to sg.
nguem^U	hwambu	ng[Sg2]	you attach to sg.
ngumsyqe^I	bjahy	[Sg3]	he answers
ngus1iir^I	hjahy	[Sg3]	he moves
ngya'kebty^I	bjanghy	[Sg3]	he annoys sy.
ngya'kebty^I	bjamby	ng[Sg2]	you annoy sy.
ini'jaim^I	hjanghy	[Sg3]	he becomes old
ini'jaim^I	hjamby	ng[Sg2]	you become old

Table 5.9: Purely phonological allomorphy of a single verbal mood suffix (of narrative mood used in the subjective and the non-plural objective conjugations) in Nganasan

harmony is indicated above as upper indexes: ^{I} for unrounded harmony, ^{U} for rounded harmony. Some roots may belong to more than one class, as they exhibit vacillating behavior.) The harmonic vowel `V` can surface as *u*, *y*, *ü* or *i*, its behavior being regulated by roundness and frontness harmonies (in the suffix being discussed it can only surface as *u* or *y*, however, as there is a back vowel (*a, ja, wa*) in the previous syllable in every case). The consonant `h` appears as *h* in strong grade and as *b* in weak grades. The consonant cluster `nh` surfaces as (i) *ngh* in strong grade or if it undergoes the so-called nunnation effect (see below), as (ii) *h* in rhythmical weak grade, and as (iii) *mb* in syllabic weak grade. A nasal consonant assimilates in place of articulation to the following consonant, and it disappears in rhythmical weak grade unless there is an immediately preceding nasal on the consonantal tier: this latter phenomenon is called nunnation. Moreover, the same morpheme has another 12 allomorphs used in the reflexive and the plural objective conjugations, six of which coincide with six of the forms cited above. The underlying form of these is `hA2nhA1`[vii].

While gradation is extremely difficult to formalize as a set of allomorph adjacency restrictions, it is such a productive process in Nganasan that it must be included in a proper morphological analyzer. It seemed, however, that though the formalism of the Humor analyzer proved to be adequate for the description of most phenomena in the language, the rule-formalism of the development environment could not easily cover all of the essential processes.

The complexity of Nganasan phonology was not the only factor that prevented the application of the feature-based formalism. Although the description of Nganasan phonology and morphology upon which I tried to base my implementation (Várnai, 2002) aimed to present a formalized account of the language, it proved to be incomplete. It describes phonological processes using context-dependent rewrite rules of the Generative Phonology tradition, but the ordering of rules is not made explicit. In addition, The formulation of a number of rules was too vague.

These vague details had to be made explicit, and it was clear that my first guess at the setting of these parameters is not likely to be correct. Rather I would have to experiment with various parametrizations and test them on the available linguistic data to find a model which adequately describes the morphology of the language. It was obvious that this experimentation would require

[vii] `A1` is another harmonic vowel that surfaces as *a* following a rounded stem, as *i* following an unrounded stem and *ü/i* in previous syllable or if a palatal consonant precedes, and as *y* following an unrounded stem and any other vowel in the previous syllable.

DOI:10.15774/PPKE.ITK.2015.013

much less human effort if the computational model which I apply were closer to the formalism used in the original account.

5.4.1.2 The application of the finite-state formalism

In the end, I managed to create a full description of Nganasan using the *xfst* formalism. The calculus implemented by the program makes it possible to ignore irrelevant symbols (such as morpheme boundaries in the case of gradation) in the environment description of rewrite rules, therefore environments encompassing non-adjacent morphemes can be easily defined. As during composition the program automatically eliminates intermediate levels of representation created by individual rules producing a single finite-state transducer, generation and analysis can be performed efficiently.

Nganasan gradation was described in *xfst* as a cascade of rules performing syllabification, the identification of syllable grades, changing the quality of the obstruents in syllable onsets and removing auxiliary symbols. The rule system covers the irregularities of Nganasan syllabification: the glottal stop closes the syllable even if it is not followed by another consonant (i.e. `V'V` is syllabified as `V'.V`), and the *b* in a *bt* cluster does not normally close the preceding syllable (i.e. `V.btV`). The rules describing gradation are reproduced in Figure 5.4. The whole of the rule system naturally contains several other rules. It describes all productive, automatic phonological rules (e.g. the assimilation of nasals to the immediately following obstruent, degemination, vowel harmony, nunnation, palatalization etc.) and morphologically or lexically constrained root and suffix alternations.

5.4.1.3 The conversion of morpheme inventories

Naturally, the new formalism not only affected the rules, but morpheme inventories and morphotactic rules had to be converted as well. I had to create a converter that would translate the morpheme inventories into the new formalism. The feature-based description of Humor proved to be very efficient concerning morphological constraints (e.g. the root selection of suffixes), though it could not handle the very complex morpho-phonological processes. Fortunately, the Xerox tools also contain a method for the description of feature-value constraints (Flag Diacritics), therefore these rules could be automatically translated.

In the formalism of *lexc*, the lexicon is composed of sublexicons that contain the description of morphemes. A continuation class must be defined for each morpheme. The continuation class is either the name of a sublexicon each member of which may follow the given morpheme, or the word-boundary symbol. The example in Figure 5.5 shows a part of the suffix list written for the Humor development environment (above), and the representation transformed into *lexc* items (below). The example shows the Narrative Mood suffix, the first form of which is used in Subjective, Objective singular and dual, while the second form is used in Objective plural and Reflexive conjugations.

The symbol `@U.S.1@` here means that the suffix attaches to the root belonging to the first root class (`@U.S.1@` in *xfst* means 'unify the current value of feature S with the value 1'). The symbol `@C.S@` clears the value of feature S. The regular root alternation rules create the three root class forms of derivational suffixes that function as roots.

I converted the graph describing the morphotactics of suffix sequences into a *lexc* lexicon fragment which defines the sequencing of suffixes and the general categorial constraints on stem+suffix combinations using the continuation class notation of *lexc*. This, together with the converted suffix representations defines the underlying representation of all possible suffix sequences in Nganasan, and the whole suffix grammar combined with the stem lexicon defines the underlying representation of the set of all possible Nganasan word forms, which our model licenses.

```
########## RULES and PROCESSES ##############

#syllabification: syllable boundaries are marked by a dot or a comma in an
 alternating fashion:
#.S: even syllable
#,S: odd syllable (except the first one, which is unmarked)
#/NSeg makes sure that non-segmental material is ignored

define Syllab [
#a dot after every syllable that is followed by a syllable which has an onset
[[C* V C*]/NSeg @-> ... "." || _ [C V]/NSeg ]
#a dot before syllables without an onset
.o.
[ V @-> "." ... || V/NSeg _ ]
#resyllabify ’ from onset to coda
#insert syllable boundary after ’
.o.
[’ -> ... "."  || "."/NSeg _ ]
#delete syllable boundary before ’
.o.
["." -> 0 || _ [’ "."]/NSeg ]
#resyllabify b from coda to onset if followed by t
#insert syllable boundary before b
.o.
[b -> "." ... || _ ["." t]/NSeg]
#delete syllable boundary after b
.o.
["." -> 0 || ["." b]/NSeg _ t/NSeg]
#strong grade after non-nasal codas and m codas not followed by b
.o.
["." -> ... "^S" || [[C-[n|m|n1|ng|N|M|N1|NG|Ng]] (Nas)]/NSeg _ ,
      [m|M]/NSeg _ [Seg-[b|B]]/NSeg]
#rhythmical weak grade after long vowels
.o.
["." -> ... "^W1" || [V V (Nas)]/NSeg _ ]
#change every second dot to a comma
#. = even syllable
#, = odd syllable
.o.
["." -> "," \/ "." ~\$["."|","] _ ]
#rhythmical weak grade in odd syllables not yet marked as strong
(NGrd=not a grade mark)
.o.
["," -> ... "^W1" || _ NGrd]
#syllabic weak grade in even syllables with a coda not yet marked as weak
.o.
["." -> ... "^W2" || _ [NGrd ?* & [C* V [C-"^X"]]/\\[Seg|"."|","]]]
#strong grade in other even syllables (codaless ones)
.o.
["." -> ... "^S" || _ NGrd]
];
```

Figure 5.4: The rules describing gradation.

```
#mode suffixes
#tag    phon     lp    mcat    comment
(...)
Narr    HA2NHU   S1    VTM     narrative subj/obj
Narr    HA2NHA1  S1    VTMR    narrative refl.
```

(a)

```
LEXICON infl_V
(...)

@U.S.1@@C.S@h^A2nh^V[Narr]      infl_VTM_r;
@U.S.1@@C.S@h^A2nh^A1[Narr]     infl_VTMR_r;
```

(b)

Figure 5.5: A part of the suffix list written for the Humor development environment(a) and the same suffixes converted to the *lexc* formalism (b).

In the case of the stem lexicon, a number of extra steps had to be taken to make it compatible with the rest of the grammar. The stem lexicon which was converted from the original dictionary data contained the surface representation of the singular nominative absolute form of nominal stems and the infinitive of verbs. This is not exactly what was needed as the input of the phonological rules. First of all, the vowel harmony class of the stem had to appear in the lexicon. This information was represented in the form of an abstract symbol (`^U` for rounded harmony stems and `^I` for unrounded harmony stems). This feature was entered into the lexicon manually in the case of nominal stems where the harmony class could be inferred from inflected forms listed in the dictionary. For verbs, it was deduced by a script which analyzed the infinitive form, cutting the ending at the same time. For stems of unknown harmony class two versions were generated, one with rounded harmony and one with unrounded harmony, so that we can analyze whatever we find in the corpus. Later the missing information could be entered into the lexicon by analyzing the corpus data concerning these words to properly constrain the analyzer. There are also a number of words which are known to take both rounded and unrounded harmonic suffixes. The entries representing these words explicitly contain both flags in the lexicon source.

Another conversion which was applied to the stem lexicon was the inverse of gradation (so that the original surface form of the stem be restored when gradation is applied to the whole word form) and a special repair rule, which introduced the assumed underlying abstract consonant segments for words which are irregular stem-internally concerning gradation.

The sample below shows a couple of entries from the original stem lexicon and the way they are represented in the *lexc* format. (Comments in *lexc* are introduced by the `!` character, while in the lexicon format of the Humor development environment, the comment character is `*`, and, to make things worse, in *xfst*, it is `#`.)

DOI:10.15774/PPKE.ITK.2015.013

```
(original lexicon format)
Ugaarne[N|GN:Ugarnaja];
ün1s1üqeU[N:1)utas, szánon utazó, 2)elso szán a fogatban];
ukud1aryI[N:búvármadár (Colymbus)];
zakazU[N:rendelés];
iked1a[Vi:általában van];

(lexc lexicon format -- converted by a script)
!a stem of vacillating (or unknown) harmony (a geographical name)
Ugaarne0:Ugaarne^U        [N][GN];
Ugaarne0:Ugaarne^I        [N][GN];
!regular entries with inverse gradation applied
ün1s1üqe0:ün1s1üte^U        [N];
ukud1ary0:ukusary^I        [N];
!irregular entry (a loan word): abstract consonant ^C inserted
!to preserve strong grade k in the closed 2nd syllable
za0kaz0:za^Ckaz^U        [N];
!verb stem: harmony class (^U) is deduced from the infinitive
iked1a:ike^U0        [Vi];
```

Beside irregularities of gradation there is a number of stem alternations which are peculiar to *j*-final stems and a subclass of vowel-final stems. Nouns and verbs follow different alternation patterns, and words having the same ending often behave differently. By analyzing dictionary data, I considered some of these alternations regular (if the pattern was unambiguous or one pattern was significantly more frequent than the others for a certain stem ending). These cases were handled by defining rules in the *xfst* rule component which create the allomorphs and introduce the necessary flag diacritics codes. The following fragment from the rule component defines regular stem alternations of *j*-final nouns.

```
#Nouns --- alternations of j
define Stemalt [
#monosyllabic regular case (no syllable boundary precedes,
#no S code in lexicon)
o j -> [o j "#P.S.1#" | u e "#N.S.1#"] ,
u j -> [u j "#P.S.1#" | u u "#N.S.1#"] ,
ü j -> [ü j "#P.S.1#" | ü ü "#N.S.1#"]
     || .#. ~\$[V] _ [Harm* TAG* Noun]
#polysyllabic regular case (syllable boundary precedes)
.o.
u j -> [u j "#P.S.1#" | u == "#N.S.1#"] ,
ü j -> [ü j "#P.S.1#" | ü == "#N.S.1#"] ,
a j -> [a j "#P.S.1#" | a a "#P.S.2#" | a u "#P.S.3#"]
     || .#. \$[V] _ [Harm* TAG* Noun]
```

Words which do not follow the regular patterns had to be marked in the lexicon source by entering the irregular allomorphs (∗ is a comment character here).

```
*irregular third stem. S1=S2
beuremuU[N:átkelohely];S3:beurema;
*S3 is irregular because it coincides with S1 and S2
bieI[N:szél];S3:bie;
*S2 and S3 irregular
tugy’I[N:anyag, rongy];S2:tukyd1e;S3:tukyd1i;
*S2 and S3 irregular but they coincide
tujU[N:tuz];!S1:tuu;
*S3 irregular with two alternative forms, S1=S2
tyraaU[A:sekély];S3:tyraa/tyrau;
```

These entries converted to the *lexc* lexicon format look like the following. The regular stem alternation rules do not apply to these entries because they have explicit stem flag marking in the lexicon.

```
beuremu@P.S.3@0:beurema#P.S.3#^U        [N];
beuremu@N.S.3@0:beuremu#N.S.3#^U        [N];
bie@P.S.3@0:bie#P.S.3#^I        [N];
bie@N.S.3@0:bie#N.S.3#^I        [N];
tugy'@P.S.1@0:tuky'#P.S.1#^I        [N];
tugy'0@P.S.2@0:tukyse#P.S.2#^I        [N];
tugy'0@P.S.3@0:tukysi#P.S.3#^I        [N];
tuj@P.S.1@0:tuj#P.S.1#^U        [N];
tuj@N.S.1@0:tuu#N.S.1#^U        [N];
tyraa@N.S.3@0:tyraa#N.S.3#^U        [A];
tyraa@P.S.3@0:tyraa#P.S.3#^U        [A];
tyraa@P.S.3@0:tyrau#P.S.3#^U        [A];
```

5.4.1.4 The analyzer

Due to the complexity of Nganasan phonology, the standard procedure of building an analyzer with the Xerox tools could not be applied, because the compiler used up all available memory and crashed. Thus, instead of first compiling the phonology transducer by composing the rules with each other and then composing the phonology network with the lexicon, I composed the rules one by one with the lexicon. Thus the lexicon constrained the space of possible underlying representations from the very beginning, and the size of the network remained manageable throughout the whole compilation process.

I compiled two variants of the Nganasan analyzer which differ only in the verbosity of the analyses they produce. The output of the less verbose variant consists of the citation form and the category tag of the absolute stem and the morpho-syntactic features the suffixes expose. The more verbose variant displays the underlying phonological representation of the suffixes as well. This latter is in fact the basic variant, the other one is obtained by composing an appropriate filter. The following is the output of the two variants for the word form *meunte* 'of your land' or 'to (the) land':

```
(terse variant of analyzer)
meunte  meu[N][Lat][Sg]
meunte  meu[N][Gen][Sg][Px][2][Sg]

(verbose variant of analyzer)
meunte  meu[N]+[Gen][Sg]+nte[Px][2][Sg]
meunte  meu[N]+nte^C[Lat][Sg]
```

I created a Nganasan word form generator by inverting the terse version of the analyzer. It generates the possible word forms exposing a stem-morpho-syntactic feature set combination. The following is an example of the application of the generator to the morpho-syntactic feature set 'meu[N][Lat][Sg]':

```
(generator)
meu[N][Lat][Sg] meute
meu[N][Lat][Sg] meunte
```

The word *meu* has two possible alternative forms because although normally the underlying `nt` of the suffix is in rhythmical weak grade due to the preceding vowel sequence *eu* (the rhythmical weak grade of `nt` is *t*), nunnation may optionally apply here (since the preceding consonant is a nasal *m*), preserving the *n* in the coda.

DOI:10.15774/PPKE.ITK.2015.013

5.4.1.5 Testing the morphological analyzer

Computational morphologies can not only be used for text markup but also for the validation of the adequacy of the grammar implemented, provided that there is a body of linguistic data they can be tested on. Therefore a set of Nganasan data was prepared to perform the tests upon.

First I checked whether the word forms listed in the original dictionary do in fact arise if I apply the generator to their lexical representation. For verb stems I checked the infinitive form. The other internal test performed was generating a random example word form for each allomorph of each suffix morpheme and checking whether these are in fact valid word forms.

I found, and, with the help of the above mentioned group of linguists, I corrected a number of errors with the help of these tests. The order of rules had to be changed and we found that the rule describing the simplification of consonant clusters overapplied to morpheme-internal clusters. We found a number of verbs in the stem database where the form of the infinitive ending in the lexicon contradicted the model (the form listed was not a possible infinitive according to the grammar). Some of these errors were introduced when the dictionary data was converted. A common source of errors was that the notation used in the dictionary does not distinguish rising diphthongs, e.g. *ja* and *wa*, from vowel sequences, e.g. *ia* and *ua*. Two words which only differ in that one of them contains a diphthong and the other contains a vowel sequence, behave quite differently, because what is an odd syllable in one is an even syllable in the other and vice versa, so gradation renders the subsequent syllables differently. These errors were not hard to fix, however some of these words appear printed in the dictionary in a form which is obviously impossible according to our model.

The model was also tested on a corpus. We used two kinds of material for testing our morphology: a corpus and model inflectional paradigms. As a corpus we used the texts which appeared in Wagner-Nagy (2002) and a collection of folklore texts from Labanauskas (2001).

I compiled a word form frequency list from the corpus and tested the analyzer on it. I first checked words for which the analyzer produced no analysis checking the list in decreasing order of word form frequency. The most frequent reason for the lack of analysis was that the stem of the word was missing from the dictionary our stem database was derived from. These words were added to the stem database.

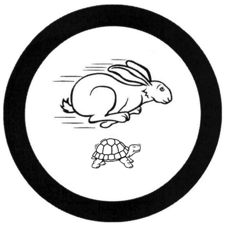
DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

6

Generating a finite-state implementation of morph-adjacency-constraint-based models

Humor vs. Xerox: How to turn a small but slow turtle into a fast but big hare...
The dog and the bush also reappear.
OK, but how about hedgehogs?
Well, you need to be patient...



Contents

DOI:10.15774/PPKE.ITK.2015.013

6.1 Difficulties for the morph-adjacency-constraint-based model

One aspect of morphological processing is not covered by the original Humor implementation. It does not support a suffix-based analysis of word forms whose stem is not in the stem database of the morphological analyzer. The system cannot be easily modified to add this feature. Such a morphological guesser would be a very useful tool, because every corpus of natural language text contains a significant amount of words with novel stems.

Moreover, integration and appropriate usage of frequency information, as would be needed by data-driven statistical approaches to text normalization (e.g. automatic spelling error correction or speech recognition), is not possible within the original Humor system. Being able to create a statistical model could also be useful when building an unknown-word guesser, since this could provide a natural way of ranking weakly constrained guessed analyses.

A third factor that can be mentioned here is that Humor's closed-source licensing scheme has been an obstacle to making these resources widely available.

The problems above could be solved by converting the morphological databases to a representation that can be compiled and used by finite-state morphological tools.

6.2 The Xerox tools

The most influential implementation of finite-state tools for morphological processing is the *xfst-lookup* combo of Xerox (Beesley and Karttunen, 2003). *xfst* is an integrated tool that can be used to build computational morphologies implemented as finite-state transducers. The other tool, *lookup* consists of optimized run-time algorithms to implement morphological analysis and generation using the lexical transducers compiled by *xfst*.

The formalism for describing morphological lexicons in *xfst* is called *lexc*. It is used to describe morphemes, organize them into sublexicons and describe word grammar using continuation classes. A lexc sublexicon consists of morphemes having an abstract lexical representation that contains the morphological tags and lemmas and usually a phonologically abstract underlying representation of the morpheme, which is in turn mapped to genuine surface representations by a system of phonological rules.

The phonological rules can either be formulated as a sequential or a parallel rule system. *xfst* can be used to compile and compose sequential rule systems with a lexc lexicon, yielding a single transducer mapping surface word forms to lexical representations directly. A similar compiler, *twolc* is available for implementing parallel two-level constraints.

The Xerox finite-state transducer implementation makes a factorization of the state space of the transducers possible in a manner similar to the extended word grammar automaton of the Humor analyzer. The construct is called *flag diacritics*. Flag diacritics are implemented as special epsilon arcs, traversed by the lookup algorithm without consuming input. At the same time, traversal of the arc affects the extended state of the transducer: the state of the variable denoted by the flag can be checked, set or cleared by the operation specified on the flag diacritics arc. If a flag checking operation fails, the lookup algorithm must stop exploring the given path in the transducer and backtrack.

Although handling of flag diacritics during lookup incurs some speed penalty, this feature is very useful. Using flag diacritics can help prevent the size explosion of the transducer due to long distance dependencies in the morphology. Furthermore, it can also be used to describe constraints between

DOI:10.15774/PPKE.ITK.2015.013

adjacent morphemes in a linguistically expressive and easy-to-understand manner. Using an *xfst*-operation, flag diacritics expressing such local constraints can often be eliminated from the transducer gaining lookup speed benefits without a significant transducer size penalty.

The Xerox tools implement a powerful formalism to describe complex types of morphological structures. This suggested that mapping of the morphologies implemented in the Humor formalism to a finite-state representation should have no impediment.

6.3 Transforming Humor descriptions to a finite-state representation

The standard procedure of building an analyzer with the Xerox tools is first to compile a lexicon transducer using *lexc*, which describes possible underlying word forms. Then, using *xfst*, one compiles a transducer from each rule in the rule component and composes them into a single transducer which defines the whole morpho-phonology of the language. Finally, one composes these two transducers into a single transducer (called a lexical transducer) which can be used as a morphological analyzer.[i]

However, as the morphological models created with the Humor formalism contain a full description of the morphology including morphophonology, neither the sequential (xfst) nor the parallel (twolc) rule component of the finite-state formalism is needed for the conversion of the Humor grammars to a finite-state representation.

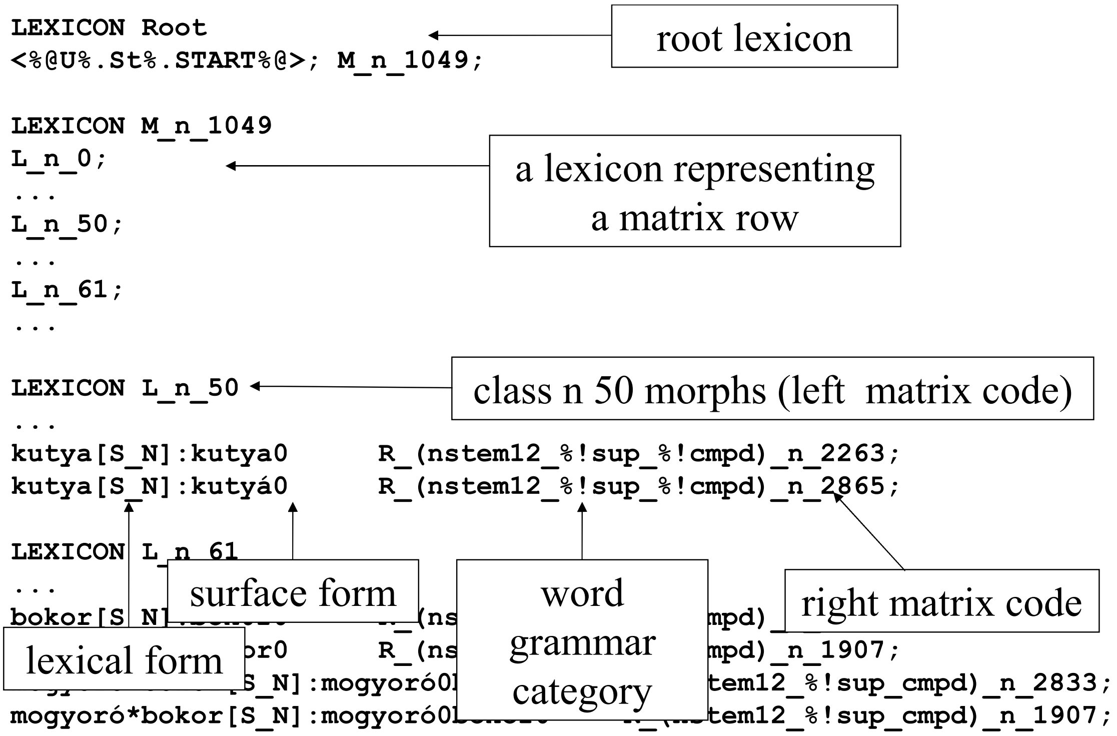
The lexical form and category label of each morph is mapped to the lexical side of the lexc representation of the morpheme, while its surface form is mapped to the surface side. The latter one is real surface form instead of the abstract underlying phonological representation that is common in usual lexc lexicon sources. Appropriate alignment of corresponding symbols in the lexical and surface representations is provided by the implementation of the lexicon converter. Tags are represented as single multicharacter symbols.

Local morph adjacency constraints represented as matrix codes, continuation matrices, binary properties and requirements vectors can be represented directly as lexc continuation classes. To simplify the mapping, a switch was added to the piece of code in the development environment that generates the Humor encoding. When the switch is present, the program creates matrices that alone completely describe all morph adjacency constraints, thus binary vectors can be ignored. When generating the lexc representation of each morph, the sublexicon it is to be included in is determined by its left matrix name and code. Its continuation class is determined by its right-hand-side matrix name and code along with its word grammar category. Figure 6.1 below shows some entries from the stem database converted to their lexc representation as well as a fragment of a sublexicon representing a row of a Humor continuation matrix. Right matrix name and code hook back to the morpheme lexicons indexed by their left matrix name and code through these sublexicons, directly encoding the compatibility relations encoded in the Humor matrices. Also note that in the representation in Figure 6.1, the word *kutya* 'dog' has a different left and right continuation class code than in the Humor lexicon fragment shown in Figure 3.1. The reason for this is that the matrices alone represent all adjacency constraints here and are thus more complex than those in the original Humor lexicon.

The easiest way to map the Humor word grammar to a finite-state representation is using flag diacritics. The main state variable of the automaton is mapped to one flag (called *St*), while the extended binary state variables to one additional generated flag each. The exact set of flag diacritics arcs attached to the representation of each morph is determined by the word grammar category of

[i] The motivation for doing it this way is that when all errors in the rule component have been fixed, it never has to be recompiled again when one adds new stems to the lexicon, which generally speeds up the compilation of the whole network.

DOI:10.15774/PPKE.ITK.2015.013



```
LEXICON Root
<%@U%.St%.START%@>; M_n_1049;

LEXICON M_n_1049
L_n_0;
...
L_n_50;
...
L_n_61;
...

LEXICON L_n_50
...
kutya[S_N]:kutya0      R_(nstem12_%!sup_%!cmpd)_n_2263;
kutya[S_N]:kutyá0      R_(nstem12_%!sup_%!cmpd)_n_2865;

LEXICON L_n_61
...
bokor[S N]              (ns                mpd)_
          or0          R_(ns                mpd)_n_1907;
           S_N]:mogyoró0l                   tem12_%!sup_cmpd)_n_2833;
mogyoró*bokor[S_N]:mogyoró0l                tem12_%!sup_cmpd)_n_1907;
```

Figure 6.1: Fragment of the lexc representation of converted Humor data structures: a row of a continuation matrix and stem allomorphs

the morph. The sublexicon fragment at the bottom of Figure 6.2 illustrates how this is implemented. Figure 6.2 also shows the converted representation of the allomorphs of the Hungarian accusative suffix. Elimination of the word grammar state flag `St` is possible to increase the speed of lookup on the transducer. However, it may result in a considerable growth of the state space.

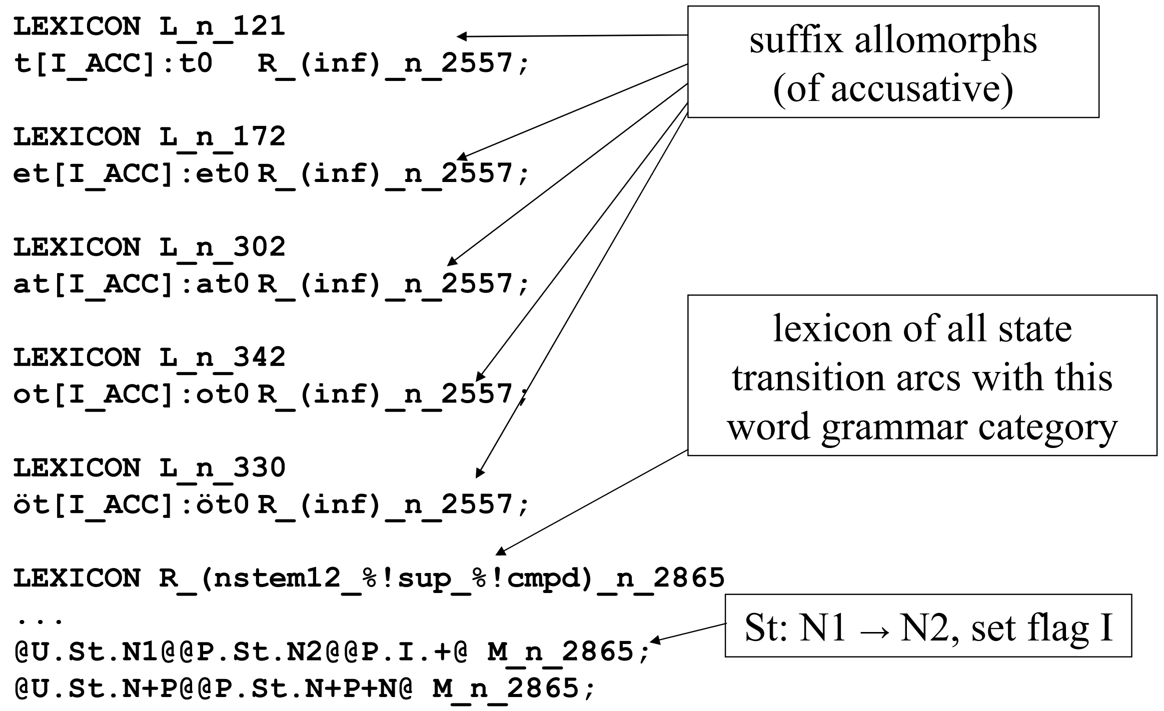
6.4 Comparison of Humor and xfst

Advantages and drawbacks of the two toolsets follow from the properties of the internal representation of the morphological database of the analyzer/generator and that of the linguistic formalisms used to create the databases.

6.4.1 Speed and memory requirement

In the Xerox tools, morphology is represented by a simple and homogeneous data structure, a set of finite-state transducers. An analysis of a word form is simply the traversal of a path in this homogeneous stream of states and transitions. On the other hand, since transducers are indeterministic with regard to their input side, the traversal of the net usually involves a lot of backtracking. Moreover, in real life situations, normally a synchronized traversal of a chain of transducers is needed, and lookup also has to handle epsilon arcs containing flag diacritics.

DOI:10.15774/PPKE.ITK.2015.013



```
LEXICON L_n_121
t[I_ACC]:t0  R_(inf)_n_2557;

LEXICON L_n_172
et[I_ACC]:et0 R_(inf)_n_2557;

LEXICON L_n_302
at[I_ACC]:at0 R_(inf)_n_2557;

LEXICON L_n_342
ot[I_ACC]:ot0 R_(inf)_n_2557;

LEXICON L_n_330
öt[I_ACC]:öt0 R_(inf)_n_2557;

LEXICON R_(nstem12_%!sup_%!cmpd)_n_2865
...
@U.St.N1@@P.St.N2@@P.I.+@ M_n_2865;
@U.St.N+P@@P.St.N+P+N@ M_n_2865;
```

suffix allomorphs (of accusative)
lexicon of all state transition arcs with this word grammar category
St: N1 → N2, set flag I

Figure 6.2: Fragment of the lexc representation of converted Humor data structures: allomorphs of the Hungarian accusative suffix and a sublexicon of state transitions labeled by the word grammar category `nstem12_!sup_!cmpd`

The database of the Humor analyzer is less homogeneous and the search for analyses involves more different operations such as lexical lookup, checking of morph adjacency constraints, word grammar automaton traversal and case conversion checks.

Due to the simpler data structure and lookup algorithm, the Xerox analyzer is faster. In fact, there is a trade-off between speed and memory requirement: the Humor analyzer, on the other hand, requires much less memory.

Table 6.1 presents a brief comparison of a version of our Hungarian morphology containing about 144000 morphs in the original Humor-compiled lexicon format and the converted version compiled by the *Xerox xfst* tool, with and without the elimination of the St flag and used for analysis using the Xerox lookup tool.

	Humor lex	lexc with St	lexc no St
run-time mem	3.3 MB	20.6 MB	38.5 MB
lookup speed	4700 words/s	12500 words/s	33333 words/s

Table 6.1: Comparison of the original Humor and *xfst*-compiled equivalents of a 144000-morph Hungarian lexicon

Finite-state conversion results in a significant increase of the memory footprint (>11 times) of the morphological analyzer. However, it also yields a significant analysis speed benefit (>7 times). Elimination of further flags roughly doubles the size of the compiled lexicon for each eliminated flag.

DOI:10.15774/PPKE.ITK.2015.013

It also leads to an extremely long compilation time, but it does not result in any significant analysis speed benefit.

Especially in the case of morphologies with a separate rule component, the ratio of compile-time memory requirement seems to be at least another order of magnitude higher than the difference between the run-time memory loads.

When working on the Nganasan morphology described in Section 5.4, the standard procedure of compiling the rule component separately by compiling and composing all the rules using *xfst* and then composing it with the lexicon compiled by *lexc* completely failed in an at that time standard 512 MB machine for lack of memory.

Finally, I tackled this problem by changing the procedure of creating the final transducer: I composed the rules one by one with the lexicon. The lexicon constrained the space of possible underlying representations from the very beginning and thus the size of the network remained manageable throughout the whole compilation process.

25 years ago, when the Humor analyzer was conceived, the compile-time and even the run-time memory requirements of the finite-state tools would have been unfeasibly high. With today's RAM sizes, even a 40 MB analyzer lexicon does not seem to be a serious problem anymore. Nevertheless, Humor might be a feasible alternative for handheld devices with limited memory capabilities.

6.4.2 The grammar formalisms

Both grammar formalisms are powerful enough to handle the complex morphology of agglutinating languages without difficulties or compromises. However, the *xfst* formalism seems to have an advantage when one has to deal with very complex phonology. Context dependent re-write rules of the Generative Phonology tradition have been a popular formalism to describe phonological processes. Such a grammar is obviously easier to translate to the *xfst* formalism than to a Humor grammar. Moreover, many details of the description often remain vague in written grammars (such as the ordering and exact formulation of rewrite rules). These must unavoidably be made explicit in a computationally implemented grammar. It is also clear that one's first guess at the setting of these parameters is not likely to be totally correct, especially if the model is very abstract, as it was in the case of the Samoyedic languages (see Section 5.4). Rather, one has to experiment with various parametrizations and test them on the available linguistic data to find a model that adequately describes the morpho-phonology of the language. It is obvious that this experimentation requires much less human effort if the computational model which one applies is closer to the formalism used in the original account.

6.4.3 Lemmatization and generation

There is a point where the lexicon format geared to the slicing-up approach of the Humor analyzer seems to have a clear advantage over the transducer-based lexicon implementation. The fact that the Humor analyzer returns both the lexical and surface form of each morph allows for a high level of parameterizability when doing lemmatization or word form generation. The key difference between a usual Xerox lexicon created using *xfst* and a Humor lexicon is that while the 'lexical form' of suffixes is normally an abstract deep phonological representation in the former, in the latter it is the form that the suffix would assume if no further suffixes were attached.

Whether various derivational affixes should be considered to be part of the lemma, and in what constructions, often depends on the actual application. In the Humor lemmatizer, these parameters can be set at run time without a recompilation of the lexicon. The rich output of the analyzer

DOI:10.15774/PPKE.ITK.2015.013

and the non-abstract lexical forms returned make merging the morphs constituting the lemma very straightforward. The flexibility of the word form generator described in Section 4.3.2 (i.e. the fact that, if necessary, the generator can handle non-atomic stems as if they were atomic) is also made possible by the fact that the corresponding analyzer database can be easily converted into a generator lexicon in which stems, derivational affixes and inflections have exactly the kind of representation needed for versatile word form generation.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

7

Extending morphological dictionary databases without developing a morphological grammar

In which we take time to rest, and instead of writing grammars for more languages, a method is proposed for the extension of dictionary-based computational morphologies without writing a grammar. And finally, they are here: живые и неживые ежи.

Contents

DOI:10.15774/PPKE.ITK.2015.013

The morphological grammar development framework described in Chapter 4 allows an easy extension of the morphology with new lexical items. This approach also gives the creator of the morphology complete control over the quality of the resource. Building rule-based morphological grammars, however, requires threefold competence: familiarity with the formalism, knowledge of the morphology, phonology and orthography of the language, and extensive lexical knowledge. Many morphological resources, on the other hand, contain no explicit rule component. Such resources are created by converting the information included in a morphological dictionary to some simple data structures representing the inflectional behavior of the lexical items included in the lexicon. The representation often only contains base forms and some information (often just a paradigm ID) identifying the inflectional paradigm of the word, possibly augmented with some other morphosyntactic features.

With no rules, the extension of such resources with new lexical items is not such a straightforward task, as it is in the case of rule-based grammars. However, the application of machine learning methods may be able to make up for the lack of a rule component. The context in which I explored the possibilities of automatic paradigm identification, was the following task. I needed to make a pop-up dictionary capable of handling and correctly lemmatizing all inflected word forms of the vocabulary of a specific Russian–Hungarian dictionary (see also Section 8.1). The method by which I solved the problem of predicting the appropriate inflectional paradigm of out-of-vocabulary words is based on a longest suffix matching model for paradigm identification, and it is showcased with and evaluated against an open-source Russian morphological lexicon.

Morphological paradigm prediction has been a field of interest, especially for researchers dealing with inflectional, or at least compounding languages. Some studies aim at solving this problem by learning inflectional paradigms from raw text corpora by clustering word-forms in the corpus and analyzing the resulting clusters (Nakov et al., 2005; Monson et al., 2008; Dreyer and Eisner, 2011). Other unsupervised methods applied to morphology induction are that of Wicentowski (2002); Hammarström and Borin (2011) and Goldsmith (2001), the latter using morphemes to encode a corpus by grouping morphemes into structures, called signatures, representing inflectional paradigms. These models, however, mainly aim at only segmenting word forms into stems and affixes: stem alternations cause paradigms to be scattered into unrelated subparadigms. However, the performance of unsupervised methods is far behind those using existing resources either as an inventory of inflectional pattern rules, or as annotated data for supervised machine learning algorithms. For example, when creating a Humor-based French morphology, I experimented with the *Linguistica* system described in Goldsmith (2001), trying to get the system generate an optimal stem-suffix segmentation for French verbal inflectional paradigms. The system failed to come up with anything usable. Another attempt at using *Morfessor* (Creutz and Lagus, 2007) to segment the Russian lexical database used in the research presented in this chapter resulted in a similar fiasco.

Raw text corpora are also used in approaches where word form statistics are used to validate inflectional forms generated by a predicted paradigm candidate for a given word. If the resulting word forms are not represented in a corpus, then the paradigm is not valid. Some examples for such methods are that of Forsberg et al. (2006) and Oliver and Tadić (2004). The research done in Lindén (2009) exploits both lexical features and corpus-based information to determine inflectional behavior by analogy. Šnajder (2013) also defines string-based and corpus-based features used for a support vector machine classifier to decide if a predicted paradigm is valid or not.

My approach differs from most of the previous ones in that I use a morphological lexicon as annotated data and the frequency distribution of raw text corpora. I address the problem of predicting inflectional paradigms based on the lemma and some given lexical features which are usually available even in some less-sophisticated dictionaries. Based on the information coming from the dictionary, the morphological lexicon can be extended in a more robust manner than in cases when only raw word form corpus frequency data is available, and lemma, categorial features and the paradigm all need to be estimated from that data.

ёж[num:Sg.cas:Nom]	ёж[num:Sg.cas:Nom]
ежа[num:Sg.cas:Gen]	ежа[num:Sg.cas:Gen]
ежу[num:Sg.cas:Dat]	ежу[num:Sg.cas:Dat]
ежа[num:Sg.cas:Acc]	ёж[num:Sg.cas:Acc]
ежом[num:Sg.cas:Ins]	ежом[num:Sg.cas:Ins]
еже[num:Sg.cas:Prp]	еже[num:Sg.cas:Prp]
ежи[num:Pl.cas:Nom]	ежи[num:Pl.cas:Nom]
ежей[num:Pl.cas:Gen]	ежей[num:Pl.cas:Gen]
ежам[num:Pl.cas:Dat]	ежам[num:Pl.cas:Dat]
ежей[num:Pl.cas:Acc]	ежи[num:Pl.cas:Acc]
ежами[num:Pl.cas:Ins]	ежами[num:Pl.cas:Ins]
ежах[num:Pl.cas:Prp]	ежах[num:Pl.cas:Prp]
(a) ёж[N.gnd:Mas.ani:Ani][:8];	(b) ёж[N.gnd:Mas.ani:Ina][:9];

Figure 7.1: Differences in case syncretism of the lemma (*ёж* 'hedgehog') depending on whether it is animate (a) or inanimate (b).

7.1 Features affecting the paradigmatic behavior of Russian words

When attempting to predict the inflectional paradigm for Russian words, certain grammatical features of the lexical item need to be known in order to have a good chance of guessing right. Lemma and part of speech are obviously necessary features, although part of speech can be guessed from the lemma for adjectives and verbs with high confidence. Nevertheless, I assumed these to be known, as these properties of words are present in any dictionary. In the experiments, I used the LGPL-licensed open-source Russian morphology available from `www.aot.ru` (Sokirko, 2004). The core vocabulary of this morphology is based on Zaliznyak's morphological dictionary (Zaliznyak, 1980). It contains 174, 785 lexical entries, each of which are classified into one of 2, 767 paradigms.

For nouns, a number of additional features (gender, countability and animacy) play a role in determining the morphosyntatctic feature combination slots which make up the paradigm of the given lemma. There are also nouns, which are undeclinable. Of these features, gender is indicated for each headword in any dictionary, and undeclinable nouns are also usually marked as such. Certain abstract, collective and mass nouns (and, in the `aot` resource, also many proper names) lack plural forms, while there are also pluralia tantum, which have no singular. Some of the latter, however, are easier to recognize, due to their lemma exhibiting typical plural morphology.

Animacy affects the nominal paradigm in a manner that does not influence the actual set of possible word forms. However, there is a case syncretism in Russian, which depends on animacy. For animate nouns, plural accusative coincides with genitive (for masculine nouns, the same applies also to singular). For inanimate nouns, on the other hand, the form of accusative matches that of the nominative. This difference is still present in the case of homonyms, where one of the senses of the word is animate, and another form is inanimate. This phenomenon is illustrated in Figure 7.1 with the word *ёж* 'hedgehog: animal', and 'Czech hedgehog: a static anti-tank obstacle'. Note, however, that the animacy feature, although it is present in the `aot` lexicon, is not generally made explicit in other dictionaries, because a human user can infer this information from the meaning of the word. We thus have not used this information.

Similarly, the set of valid morphosyntactic feature combinations for verbs depends on verbal aspect and transitivity/reflexivity. Thus, these properties need to be known for verbs, and, indeed, they are listed in dictionaries. E.g. non-transitive verbs lack passive participles; verbs of perfective aspect lack

DOI:10.15774/PPKE.ITK.2015.013

present participle forms; and many verbs of imperfect aspect lack past participial (especially passive) forms. The adverbial participial forms a verb may assume also depend on aspect (and also on other idiosyncratic lexical features).

Defectivities of the adjectival paradigm, e.g. the lack of short predicative forms and synthetic comparative and superlative forms depends on semantic and other, seemingly idiosyncratic, features of the lexeme. E.g. relational adjectives usually lack these forms. Such properties, however, were not made explicit in the `aot` lexicon, neither are they present in normal dictionaries, so I did not use any lexical features for adjectives beside part of speech.

Thus, when defining the feature set for predicting inflectional paradigms of words, I assumed that the lemma and the lexical properties mentioned above: part of speech, gender, verb type, etc., are known. However, some morphological characteristics relevant from the aspect of inflection cannot be derived neither from a simple dictionary, nor from the surface form of a word. Such features are whether the meaning of a noun is an animate or inanimate entity; an adjective lacks certain grammatical forms; there is stress variation, idiosyncratic orthographic variations, or other irregularities. Thus, my model is not necessarily able to predict paradigmatic behavior depending on such features, since the necessary information is not available to it.

The other set of features I used are n-character-long suffixes of the lemma for various lengths n. The maximum suffix length is a parameter of the algorithm. It was set to 10 in all the experiments. In order to exploit this information, a suffix model is created based on the lexicon. An illustration of how this model including both the endings and the lexical features is generated is shown in Figure 7.2.

7.2 Creation of the suffix model

A suffix trie is built of words input to the training algorithm in the form shown in the right column of Figure 7.2. The lemma is decorated with the following features (from right to left):

- The tag in brackets consists of two parts: part of speech (and, in the example: gender) is followed by the appropriate paradigm ID from the `aot` database; the two are separated by a hyphen. This is the information to be predicted by the algorithm for unknown words. After processing the training data, terminal nodes of the suffix trie link to a data structure representing the distribution (relative frequency) of tags for the given suffix.
- A suffix following a vertical bar is attached to the end of the lemma. This represents the available lexical knowledge about the lexical item in an encoded form.[i]
- Some paradigms are restricted to lemmas ending in a specific suffix. There is a hash mark at the beginning of the suffix of the lemma that is required by the given paradigm ID to be valid. The given paradigm ID is not applicable to words not having that ending. E.g. all lemmas in paradigm 1433 must end in *ьё*.

7.3 Ranking

The suffix-trie-based ranking algorithm that I used was inspired by the suffix guesser algorithm used in Brants' TnT tagger to estimate the lexical probability of out-of-vocabulary words (Brants, 2000). However, that model did not prove to perform well enough in this task. So I modified the model step-by-step until I arrived at a model that turned out to be simpler, yet to perform much better.

[i] n: neuter noun, *: undeclinable, s: singular only

DOI:10.15774/PPKE.ITK.2015.013

<table>
<tr><td>мумиё[N.n.*.-];prd:25</td><td>мумиё|n*[N.n-25]</td></tr>
<tr><td>остриё[N.n.-];sfx:ё;prd:1709</td><td>остри#ё|n[N.n-1709]</td></tr>
<tr><td>бабьё[N.n-];sfx:ё;prd:210</td><td>бабь#ё|ns[N.n-210]</td></tr>
<tr><td>дубьё[N.n-];sfx:ё;prd:210</td><td>дубь#ё|ns[N.n-210]</td></tr>
<tr><td>свежевьё[N.n-];sfx:ё;prd:210</td><td>свежевь#ё|ns[N.n-210]</td></tr>
<tr><td>цевьё[N.n.-];sfx:ьё;prd:1433</td><td>цев#ьё|n[N.n-1433]</td></tr>
<tr><td>жнивьё[N.n];sfx:ё;prd:1103</td><td>жнивь#ё|n[N.n-1103]</td></tr>
<tr><td>суровьё[N.n];sfx:ё;prd:210</td><td>суровь#ё|ns[N.n-210]</td></tr>
<tr><td>мостовьё[N.n];sfx:ё;prd:210</td><td>мостовь#ё|ns[N.n-210]</td></tr>
</table>

Figure 7.2: A portion of the suffix model. The format of the right column is: `lem#ma|lex-features[PosTag-paradigmID]`, where `ma` is a required ending of the lemma for all items in the paradigm identified by `paradigmID`.

<table>
<tr><td>гурба|f [N.f]</td><td>[N.f:50]#2.857270 [N.f:175]#0.756756 [N.f:48]#0.293840
[N.f:105]#0.175658 [N.f:88]#0.098045 [N.f:103]#0.051742
[N.f:396]#0.03995 [N.f:611]#0.039730 [N.f:69]#0.029693
[N.f:121]#0.021167</td></tr>
<tr><td>дурака|f [N.f]</td><td>[N.f:88]#4.466005 [N.f:15]#1.341181 [N.f:273]#0.904291
[N.f:36]#0.738748 [N.f:50]#0.467147 [N.f:16]#0.443249
[N.f:39]#0.300179 [N.f:105]#0.175658 [N.f:96]#0.155983
[N.f:103]#0.051742</td></tr>
</table>

Figure 7.3: The ten highest ranked paradigm candidates for the input words гурба—f and дурака—f. The candidates are listed sorted by their rank, with the calculated score separated by the # mark for each tag.

The paradigms are predicted by assigning a score to each paradigm for each word. Then, the higher this score is for a paradigm tag for a certain word, the more probable it is that the word belongs to that paradigm. I select the top-ranked paradigm to be the predicted inflectional class.

The score for each paradigm in the case of a word is calculated for all suffixes of the word, including the lexical properties, from shortest to longest. More formally, for all tags, the rank is calculated iteratively according to Formula 7.1.

$$rank^{i+1}[tag] = sign \times len\_sfx \times rel\_freq + rank^{i}[tag] \tag{7.1}$$

where

$sign$	is negative if the suffix is shorter than the minimal suffix required by the given paradigm
len_sfx	is the length of suffix not including lexical properties
rel_freq	is the relative frequency of tag for the suffix
$rank^{i}[tag]$	is divided by len_sfx if $len\_sfx > 1$ is negated if $sign > 0$ and $rank^{i}[tag] < 0$ before calculating $rank^{i+1}[tag]$

The applied ranking score clearly prefers the most frequent paradigm for the longest matching suffix. Some examples for the ranked candidates are shown in Figure 7.3.

DOI:10.15774/PPKE.ITK.2015.013

7.4 Evaluation

For the evaluation of the performance of the paradigm assignment algorithm, I used the frequency distribution of Russian lemmas, taken from Serge Sharoff's Russian internet frequency list.[ii]

Evaluation of the ranking algorithm was performed for the four different test sets. These are rare words (LT10), average words (LT100), and frequent words (MT1000). I used standard evaluation metrics for measuring the performance of my method.

Evaluation of the ranking algorithm was performed on different training and test set combinations. In each case, I applied five-fold crossvalidation. In order to see how the performance of the algorithms is affected by the frequency of the lemmas in the training and test sets, I split the `aot` lexicon into parts that contained rare words (LT10; not more than 10 occurrences in the Internet corpus; 91,770 words), average words (LT100; between 11 and 100 occurrences; 33,990 words), and frequent words (MT1000; more than 1000 occurrences; 9,650 words). Moreover, I also evaluated performance on a random 20% sample of the lemmas disregarding frequency (RAND; 159,935 words).

I used standard evaluation metrics for measuring performance. *First-best accuracy* measures the ratio of having the correct paradigm ranked at the first place. This reflects the ability of the system to automatically classify new words to paradigms. In addition, the accuracy values for 2^{nd} to 9^{th} ranks were also calculated. *Recall* is the ratio of having the correct paradigm in the set of the first ten highest ranked candidates. Following the metrics used by Lindén (2009), precision was calculated as *average precision at maximum recall*, i.e. $1/(1+n)$ for each word, where n is the rank of the correct paradigm. This measures the performance of the ranking algorithm. As it might be the case that paradigm prediction is used to aid human classification, this metric reflects the ratio of noise a human must face with when verifying the results. Finally, *f-measure* is the harmonic mean of precision and recall.

I evaluated our algorithm comparing it to two baseline methods. The first one uses Brants' suffix guesser model (Brants, 2000) instead of the longest suffix matching method. This model uses a θ factor to combine tag probability estimates for endings of different length in order to get a smoothed estimate. θ is set as the standard deviation of the probabilities of tags. First, the probability distribution for all suffixes is generated from the training set, then it is smoothed by successive abstraction according to Formula 7.2.

$$P(t|l_{n-i+1}, ...l_n) = \frac{\hat{P}(t|l_{n-i+1}, ...l_n) + \theta_i P(t|l_{n-i}, ...l_n)}{1+\theta_i} \tag{7.2}$$

for $i = n...0$, with the initial setting $P(t) = \hat{P}$, where

$\hat{P}$	are maximum likelihood estimates of the tag t from the frequencies in the lexicon of suffixes consisting of letters l
θ_i	weights are the standard deviation of the unconditioned maximum likelihood probabilities of the tags in the training set for all i

The other baseline assigns the most frequent paradigm identifier to each word based on its part of speech and the additional features available (e.g. gender, aspect, etc.). The results of these baselines compared to our system are shown in Table 7.1. As expected, the second baseline, choosing the most frequent tag, has a rather low accuracy. However, our longest suffix method outperforms the first baseline as well. A key difference between the two models is that Brants' model assigns more weight

[ii] http://corpus.leeds.ac.uk/frqc/internet-ru.num

Table 7.1: First-best accuracy of paradigm identifiers achieved by the longest suffix match algorithm, Brants' model, and by assigning the most frequent paradigm tag

	Longest suffix	Brants' model	Most frequent tag
MT1000	0.768	0.587	0.410
LT100	0.876	0.593	0.473
LT10	0.887	0.698	0.480
RAND	0.862	0.632	0.466

Table 7.2: Results on full tag agreement (FULL), paradigm identifiers (ID) and equivalent paradigm classes (EQUIP). The results are measured by first-best accuracy, precision, recall and f-measure.

	MT1000	LT100	LT10	ALL/MT1000	ALL/LT100	ALL/LT10	RAND
FULL	0.752	0.849	**0.879**	0.759	0.855	0.872	0.848
	0.819	0.903	0.926	0.823	0.910	0.923	0.903
	0.903	0.979	0.991	0.923	0.989	0.994	0.982
	0.859	0.940	**0.958**	0.870	0.948	0.957	0.941
ID	0.768	0.876	**0.887**	0.771	0.872	0.885	0.862
	0.830	0.920	0.934	0.834	0.924	0.933	0.915
	0.905	0.980	0.992	0.926	0.990	0.994	0.983
	0.866	0.949	**0.962**	0.878	0.956	0.962	0.948
EQUIP	0.819	0.889	**0.892**	0.813	0.884	0.890	0.875
	0.869	0.929	0.937	0.866	0.932	0.936	0.924
	0.929	0.984	0.993	0.951	0.993	0.995	0.988
	0.898	0.956	**0.964**	0.907	0.961	0.965	0.955

to unconditioned tag distributions and ones conditioned on shorter suffixes than those conditioned on longer ones. This is just the other way round in the longest suffix algorithm.

The tags containing paradigm ID's as well as detailed PoS and subcategorial features define a very sophisticated classification of words. However, some of the features that distinguish two different paradigms are not relevant from the aspect of their inflectional behavior, such as the subtype of a non-inflecting adverb. Moreover, some of these features cannot even be predicted. In many cases, there is stress variation yielding a different paradigm ID, which, however, does not affect the set of orthographic forms in the paradigm. Moreover, some paradigm differences are irrelevant from the point of view of our dictionary lookup task, because they do not affect the set of word forms in the paradigm. The case syncretism differences between animate and inanimate nouns are examples of such differences. To see how the algorithms perform in our original lemmatization task, equivalence classes of paradigms were generated, and a prediction was considered correct if the set of inflected forms generated by the predicted paradigm was identical to the set of word forms generated by the correct paradigm. Of the 2,767 different paradigms, 921 non-unique paradigms could be collapsed into 283 equivalence classes. Table 7.2 shows the results for each setup, where rows FULL, ID and EQUIP correspond to full tag, paradigm ID, and equivalence class evaluations, respectively. In the rows marked by ID, instead of full tag agreement, which might include hard-to-predict information like that the word is the name of an organization, only the paradigm identifiers were considered. Thus `[N.n._nam:Org.--49]`, and `[N.n.--49]` were considered as equivalent.

The three columns on the left show results where the models were trained only on words in the same frequency class they were tested on. The test set was always 20% of the lemmas in the given frequency

Table 7.3: First-best accuracy of paradigm ID prediction in the case of all types of words, nouns, verbs and adjectives

	ALL	NOUNS	VERBS	ADJECTIVES
MT1000	0.768	0.814	0.702	0.683
LT100	0.876	0.935	0.802	0.772
LT10	0.887	0.968	0.869	0.732
RAND	0.862	0.947	0.848	0.682

range. Results in the next four columns were obtained by training the models on the complement of the test set w.r.t. the whole lexicon.

As the numbers show, our system performs best on rare words, while it achieved the worst results on very frequent words. This is not very surprising, as irregular words tend to be frequent words, while rare words have regular inflectional behavior. Correctly predicting the exact paradigm of an unknown personal pronoun or an irregular verb is indeed a rather difficult task. Since our aim was to extend existing morphological lexicons, and such resources already contain the most frequent words of the language, the results obtained for rare words are the ones which are relevant for our task.

Also note that beside similar recall values, precision and first-best accuracy are higher when equivalent paradigms are collapsed. The prediction algorithm works reasonably well for extending resources for tasks that do not require full morphological analysis such as indexing for information retrieval or dictionary lookup.

Table 7.3 shows the first-best paradigm ID accuracy results for all words, nouns, verbs and adjectives separately. The exact paradigm of verbs and adjectives turned out to be more difficult to guess than that of nouns. The results achieved for adjectives seem to be especially contradictory to the overall performance, which can be explained by the unpredictable behavior of adjectives. Semantic factors and hard-to-predict stress variation affecting paradigmatic classification are explained in the next section of this paper.

7.5 Error analysis

The most frequent confusions of the longest suffix algorithm for infrequent words are due to failure to correctly predict

- whether an adjective has synthetic comparative, superlative and/or short predicative forms,
- whether a *-ние*-final abstract noun has an alternative *-нье* spelling,
- whether a noun has a second genitive (used in partitive constructions) or locative form,
- stress in past passive participles of certain verb classes and in short and comparative, forms of certain adjectives, or other optional stress variation across the paradigm (this results in an *е*∼*ё* contrast not normally reflected in orthography),
- whether a non-inflecting noun can be interpreted as plural,
- whether an imperfective verb has past passive participle forms.

Except for stress-related issues and semantically motivated or idiosyncratic defectivity, incorrect forms are very rarely predicted by the algorithm. Humans would probably make similar mistakes for words they do not know, especially if they do not know the meaning of the word either. The system sometimes highlights inconsistencies in the original `aot` data that even I, not being a native or even advanced speaker of Russian, can identify as errors, e.g. that while the name of the energy company

DOI:10.15774/PPKE.ITK.2015.013

Кубаньэнерго is categorized as lexically non-plural, the similarly formed *Сахалинэнерго* does not have this property.

When looking at errors the algorithm makes when applied to frequent words, we find that the types of errors are similar to the ones listed above. Nevertheless, failure to predict superlatives, comparatives, second genitives or special locative forms is more prevalent for this data, as a much higher proportion of very frequent words have these “irregular” forms.

The most frequent errors of Brants’ original suffix guesser algorithm, on the other hand, include absurd errors that would not be made even by beginning learners of Russian. This is due to overemphasizing distributions conditioned on shorter suffixes over those on longer ones. The top-ranked candidate paradigm is often totally inapplicable to words having the ending the given lexical item has, such as the paradigm of -*кий*-final adjectives to -*ный*-final ones (the most frequent error of that algorithm for infrequent words).

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

8

Applications

"All the proofe of a pudding, is in the eating." (Camden, 1605)

Contents

DOI:10.15774/PPKE.ITK.2015.013

The Humor morphological analyzer and the computational morphologies described in this thesis have been used in various commercial tools and research projects. These include spell checker applications, machine translation systems, corpus annotation projects and tools, search tools, information retrieval systems etc.

8.1 Integration into commercial products

Spell checker in Microsoft Office The Hungarian morphology described here has been used to implement a spell checker for Microsoft Word products. The spell checker has been extended with add-on lexicons covering special, such as financial, medical, nuclear and military, terminology.

Application of the word form generator in the Hungarian stemmer/word breaker used in Microsoft's indexing service Microsoft's indexing service was integrated in various versions of the Windows operating system (Windows search) and the stemmer/word breaker is also used in Microsoft database, web service and information retrieval solutions (e.g. SharePoint Server). This solution works by indexing word forms in documents as they are, and performing query extension at query time by generating the full paradigm of query terms. Since generating the full paradigm for languages like Hungarian is not feasible (especially taking into account that derived forms, e.g. participles, and their further suffixed forms can also be of interest), the specific solution delivered to Microsoft generates only the at most 200 most frequent forms for each term using the word form generator described in Section 4.3.2.

Lemmatization in the MoBiMouse/MorphoMouse pop-up dictionary program MorphoLogic has had a line of pop-up dictionary tools which are capable of coming up with dictionary entries for words and multiword expressions pointed at or clicked on by the mouse. Obviously, the tool must also work for suffixed word forms, thus both the dictionary indexes (created for headwords, expressions and examples) and the query words are lemmatized using the *stem2005* lemmatizer described in Section 4.3.1. The same indexing system is used in on-line dictionary services run by MorphoLogic (e.g. `www.webforditas.hu/szotar.php`) and the Grimm Publishing House (`www.grimmonlineszotar.hu/`). In order to cover the languages of the dictionaries, I created computational morphologies using the formalism described in Section 4 for French and Spanish, and adapted ones for Italian, Dutch and Russian. The methods by which the Russian morphology was extended to cover the full vocabulary of the Russian–Hungarian dictionary published by the Grimm Publishing House is described in Section 7. The creation/adaptation and extension of the morphologies got me in many cases also deeply involved in the proofreading process of the dictionaries.

8.2 Machine translation systems

The need for the automated translation of texts has made the development of machine translation systems a popular and important field of research. Beside rule-based applications, statistical methods have also been applied to solve the task of translation. For both methods, a high-accuracy morphological analyzer and generator is of crucial importance, especially for languages with a complex morphology like Hungarian.

DOI:10.15774/PPKE.ITK.2015.013

Morphological analysis and generation in the MetaMorpho machine translation system of MorphoLogic MorphoLogic's MetaMorpho (Novák et al., 2008; Prószéky and Tihanyi, 2002) is a rule based machine translation system, which is also at the core of the online English–Hungarian and Hungarian–English translation services at the sites `webforditas.hu` and `itranslate4.eu`. MetaMorpho is a rule-based system the architecture of which differs from that of most well-known rule-based systems: it does not contain a separate transfer component. Its grammar operates with pairs of patterns (synchronous context-free rules enriched with features) that consist of one source pattern used during bottom-up parsing and one or more target patterns that are applied during top-down generation of the translation. The architecture of the grammar is completely homogeneous: the same formalism is used to represent general rules of grammar, more-or-less idiomatic phrases and fully lexicalized items, these differ only in the degree of underspecification.

In this system, morphological analysis is performed by the Humor analyzer. Moreover, Humor is used as a word form generator to synthesize output word forms generated by the machine translation system.

Utilizing morphology in hybrid statistical machine translation systems Statistical machine translation (SMT) systems do not rely on rules or grammatical pattern matching, but are trained on parallel corpora of the corresponding languages. Phrase-based SMT methods build statistical models of the possible translations of words and multi-word phrases from automatically sentence-and-word-aligned parallel corpora. Moreover, a target-side language model is also built containing the distribution of n-grams in the training corpus, and it used enable the system to produce sentences which are more-or-less fluent in the target language.

However, for languages with a rich morphology, pure word-based statistics do not provide a satisfactory model for translations. Many English function words, such as prepositions, possessive and other pronouns etc. correspond to bound morphemes in Hungarian. It is difficult to capture these generalizations using a word-form-based representation. Moreover, the word alignment phase of training an SMT system makes many alignment errors due to these differences between grammatically distant languages. Thus, an English-Hungarian SMT system can only achieve satisfactory results if augmented with a morphological analyzer capable of handling language-specific phenomena.

An example of a state-of-the-art system of this type is that presented in Laki et al. (2013b,a). In this system, training is performed on a morphologically segmented version of the Hungarian side of the parallel corpus, while manually crafted syntactic transformation rules are applied to the English side to reduce word order differences. Thus, a morpheme-based translation system was created in order to diminish the differences in the number of words and word order between the two languages. In this system, the Humor analyzer was used to create the morphologically segmented Hungarian side of the training set. Then, the Humor word form generator is used to generate words from the morphemes created by the translation process. The results described in Laki et al. (2013a) proved that morphological analysis has a positive effect on translation quality, resulting in an about 6% improvement in the mm-BLEU evaluation framework, compared to the word-based system. Moreover, in human evaluation, the morpheme-based system also ranked better than any of the tested word-based systems. The quality improvement is due to the capability of the system to translate word forms that were not present in the training corpus in cases when the individual morphemes were.

8.3 Part-of-speech tagging

Part-of-speech tagging is one of the first steps in a natural language processing pipeline, thus its accuracy is of crucial importance. Although part-of-speech tagging is generally understood as a task

DOI:10.15774/PPKE.ITK.2015.013

where only a single part-of-speech label is to be assigned to each token in the text, for morphologically rich languages, like Hungarian, this is hardly enough for any upstream task. A full morphological disambiguation including the assignment of a lemma and a rich morphosyntactic annotation is needed. The Humor analyzer has been used in two implementations of automatic morphological annotation systems built around a part-of-speech tagger core. Both systems were first trained to handle Hungarian texts.

PurePos PurePos is an open-source HMM-based automatic morphological annotation tool (Orosz and Novák, 2013). In order to achieve high accuracy and fast training time, PurePos uses methods introduced in TnT (Brants, 2000) and HunPos (Halácsy et al., 2007). The tagging model is a linearly interpolated ngram-based contextual model, and it uses unigram or bigram lexical models.

In addition to statistical modeling, the tagger can incorporate knowledge provided by a morphological analyzer. The integrated morphological analyzer has a twofold role. On the one hand, it helps to determine the correct tag for words unseen in the training corpus by eliminating false tag candidates generated by the suffix guesser. Thus, the tagger achieves better accuracy, since the suffix guesser may have many false guesses that can be filtered out by using a morphological analyses. On the other hand, since the identification of lemmata is a subtask of the morphological annotation task, the most probable lemma that corresponds to each selected tag and token needs to be selected. For tokens recognized by the morphological analyzer, it supplies lemmata, while in the case of out-of-vocabulary words, a lemma guesser is used to generate possible lemmata. Each guess contains a morphosyntactic tag with which it is compatible. Having the guesses for each token and the best tag sequence for the sentence, the most probable compatible lemma (having the same morphosyntactic tag) is selected for each token.

HuLaPos2 HuLaPos2 (Laki et al., 2013c) is another automatic tagger/morphological annotation tool, based on a statistical machine translation system. It considers the disambiguation process as a translation task, in which the original text is the source side and the lemmatized and tagged version is the target side of the translation. The drawback of this purely statistical method is its inability to handle word forms not seen in the training corpus. Thus, beside using a suffix guesser, the quality of the tagger was improved by integrating the Humor analyzer into the original system. The morphological analyzer is used to filter out the incorrect part-of-speech and lemma candidates of the suffix guesser.

8.4 Corpus annotation

Morphological annotation of the Hungarian Gigaword Corpus The second version of the Hungarian National Corpus (MNSZ2, also called the Hungarian Gigaword Corpus at `http://clara.nytud.hu/mnsz2-dev/` (Oravecz et al., 2014)) was annotated using the Hungarian morphology described in Section 5.1 and the *stem2005* lemmatizer described in Section 4.3.1. The annotation includes full segmented morphological analysis in addition to lemmatized analysis.

Old and Middle Hungarian annotated corpora The adapted Old and Middle Hungarian morphology and the automatic and manual annotation tools described in Section 5.2.1 were used in two parallel OTKA projects of the Research Institute for Linguistics of the Hungarian Academy of Sciences (*Hungarian historical generative syntax* [OTKA NK78074], and *Morphologically analysed corpus of Old and Middle Hungarian texts representative of informal language use* [OTKA 81189]). One of the major aims of these projects was is to create morphologically analyzed and searchable

DOI:10.15774/PPKE.ITK.2015.013

corpora of texts from the Old Hungarian and Middle Hungarian period. The corpora are accessible at `omagyarkorpusz.nytud.hu` and `tmk.nytud.hu`.

Uralic resources The computational morphologies available at the site `www.morphologic.hu/urali` have been created in a series of research projects by me and Researchers of the Department of Finno-Ugric and Historical Linguistics of the Research Institute for Linguistics of the Hungarian Academy of Sciences (see Section 5.3).

The projects, funded by *Hungarian Scientific Research Fund (OTKA)* and the *National Research and Development Programme (NKFP)*, laying the foundations of the tools presented here were the following:

OTKA 71707	*Ob Ugric morphological analyzers and corpora*
OTKA K 60807	*Development of a morphological analyzer for Nganasan*
OTKA T 048309	*Linguistic databases for Permic languages*
NKFP-5/135/01	*A Complex Uralic Linguistic Database*

For the Finno-Ugric languages, Komi (see Section 5.3.1), Udmurt, Northern Mansi, and the Khanty dialects, the Humor morphological analyzer engine was used (see Chapters 3 and 4). These analyzers were developed by László Fejes and myself. The Nganasan morphology was implemented using the Xerox xfst toolset by me (see Section 5.4). Some analyzers also present glosses.

The web interface was created by *István Endrédy* and myself. The site includes corpora for all the languages we created morphologies for, see below. For the following three languages/corpora: Mansi (Chr. Vog.), Kazym Khanty and Synya Khanty, the corpus was automatically annotated and disambiguated using a combination of the respective morphological analyzer and the Hunpos tagger (Halácsy et al., 2007), and the annotation was manually checked and corrected using the web interface described in 5.2.1.4. Since the texts are glossed and contain translations, these corpora are not only morphosyntactically tagged, but the glosses also provide sense tagging. Word sense disambiguation was already performed in the automatic disambiguation phase by comparing the glosses of the candidate analyses to the translations we had using a letter-trigram-based similarity measure, which performed rather well, and could even be used to help bootstrap the iterative training cycle of the tagger.

The analyzers

Komi-Zyryan The lexicon for the analyzer for standard Komi-Zyryan with Cyrillic orthography was derived from Beznosikova (2000).

Mansi (WT) In all analyzers for Northern Mansi, glosses for the stems are presented in English, German and Hungarian. This version uses the transcription of the text collection Kálmán (1976). The lexicon is based on the vocabulary of the same book.

Mansi (Chr. Vog.) This version uses the transcription of the text collection Kálmán (1963). The lexicon is based on the vocabulary of the same book.

Mansi (VNGY) This version uses the transcription of the text collection Munkácsi (1892). The lexicon is based on the vocabulary Munkácsi (1986), digitized by myself, corrected by László Fejes.

Kazym Khanty The stem lexicon was created and the morphological annotation of the corpus was disambiguated by Mária Sipos. Glosses for the stems are presented in English and Hungarian.

Synya Khanty The corpus presented on the website was collected and transcribed, morphological annotation of the corpus was disambiguated and the stem lexicon was created by Eszter Ruttkay-Miklián. Glosses for the stems are presented in English and Hungarian.

DOI:10.15774/PPKE.ITK.2015.013

Udmurt The lexicon is derived from the dictionary Kozmács (2002). Glosses are presented in Hungarian.

Nganasan The Nganasan analyzer uses a Latin-based phonemic transcription (various transcription versions are available on the web interface of the tools, which are variants of the transciption used in the text collection Wagner-Nagy (2002). The lexicon is based on the vocabulary of the same book and the dictionary Kost'erkina et al. (2001).

8.5 Information retrieval systems

In search tools, both at indexing and at query time, it is a common practice to use stemming in order to be able to find all the different forms corresponding to the words in the query.

Search engine for Hungarian ophthalmology notes In Section 5.2.2, I have described how the Humor analyzer was extended to be able to handle the special vocabulary of clinical documents. This version of the morphological analyzer was not only used to process clinical notes, but was also integrated into a search engine built to retrieve information from a database of ophthalmology notes. In this system, the Humor analyzer is used both at indexing and query time to be able to find all inflected forms of words including this special medical vocabulary.

Solr integration of Humor As the search engine for ophthalmology notes uses the Lucene/Solr document retrieval system, the integration of Humor in the system required the implementation of a Lucene/Solr API to Humor. Thus, Humor can be used as an alternative analyzer and stemmer in other Solr-based search engine solutions.

Precognox search engine The search engine of Precognox is a domain-independent intelligent indexing and search solution built on enterprise search platforms enhanced by natural language processing applications. The Humor analyzer is used in this system to process query terms and both the stored and retrieved documents. The concept of the search process is a simple user interface, but the results to the entered query are presented in an organized manner.

News management system of the MTI Hungarian News Agency The stemmer engine built around the Hungarian Humor morphological analyzer has been integrated into the document retrieval system used by news editors of the MTI Hungarian News Agency.

Archives of the Hungarian Atomic Energy Authority The Hungarian Atomic Energy Authority, a government office, has a sizable archive of technical documentation concerning nuclear facilities, transport containers, as well as with the security of nuclear and other radioactive materials and associated facilities in Hungary. Documentation in the archive in various languages had to be made digitally accessible. A Humor-based stemmer was used to index the documents in a custom-made information retrieval system in Hungarian, English and Russian. For Hungarian, the morphology was extended with nuclear terminology to improve the coverage of the stemmer. Erroneous words were indexed with an additional automatically corrected stem, hyphenated words were properly dehyphenated.

DOI:10.15774/PPKE.ITK.2015.013

8.6 Other tools

Accent restoration Due to clumsy mobile device interfaces and reluctance of users to spend too much time entering their message, a great amount of text is generated in a format that lacks the diacritic marks normally used in the orthography of the language the text is written in. An automatic system that can restore the fully accented equivalent of such accent-stripped texts can be used not only to make the text more readable for humans but is also a prerequisite for the applicability of higher-level natural language processing tools. Thus, I created a method based on statistical machine translation that is able to restore accents in Hungarian texts with high accuracy. However, due to the agglutinating characteristic of Hungarian, there are always wordforms unknown to any system trained on a fixed vocabulary. In order to be able to handle such words, I integrated the Humor analyzer into the system that can suggest accented word candidates for unknown words as well.

I considered the problem of accent restoration as a translation task, where the source language is the unaccented version, and the target language is the accented Hungarian. Since it is easy to come up with a parallel training corpus for this task, methods of statistical machine translation (SMT) can be applied.

In my implementation, I used Moses (Koehn et al., 2007), a widely used SMT toolkit for building the translation models and performing decoding, and SRILM (Stolcke et al., 2011) to build the necessary language models.

In the baseline setup, only the translation and language models built from the training corpus were used. The input for the decoder was Hungarian raw texts with all the accents removed. The translation model was responsible for predicting the distribution of accented forms and the language model exploited contextual information. The combination of these two models could determine the accented forms but only for words already seen in the training corpus.

In order to be able to restore accents in unseen words as well, Humor was integrated. For unknown words, all possible correct accented candidates were generated by the morphological analyzer. These candidates were then fed to the Moses decoder with a probability assigned to each candidate. First, I assumed uniform distribution among the candidates. However, this approach assigned the same probability to the most common and the most nonsensical (although grammatical) candidates as well. Thus, in some cases these forms showed up in the results. In order to avoid the system to make such errors, a more sophisticated distribution was estimated for the candidate set. For this, I applied a linear regression model based on corpus frequency data determined for the lemma and other features of the candidate word (since the actual wordform was not present in the corpus). Thus, for each candidate,

- its lemma frequency *(LEM)*,
- the number of productively applied compounding *(CMP)*,
- the number of productively applied derivational affixes *(DER)*, and
- the frequency of the inflectional suffix sequence returned by the analysis *(INF)*

were determined. Based on these components, a score was assigned to each candidate based on Formula (8.1).

$$score = -\lambda_c CMP - \lambda_d DER + log_{10} LEM + \lambda_i log_{10} INF - MS \tag{8.1}$$

, where

$$MS = \begin{cases} minscore - 1 & if \quad minscore \leq 0 \\ 0 & otherwise \end{cases} \tag{8.2}$$

DOI:10.15774/PPKE.ITK.2015.013

The MS component was used to scale up the scores by adding $|minscore| + 1$, i.e. the lowest score received for any candidate in the actual candidate set. After this score was counted for each candidate for a certain word, the individual scores were normalized in order to have values in the range of a probability distribution.

Grapheme-to-phoneme conversion A crucial component of text-to-speech systems is the one responsible for the transcription of the written text to its phonemic representation. Though the complexity of the relation between the written and spoken form of languages varies, most languages have their regular and irregular phonological set of rules. In the case of phonetic languages, such as Finnish, Estonian or Hungarian, the transcription of a written wordform is almost always straightforward. However, there are two types of phenomena that make the transcription nontrivial: changes in pronunciation due to the interference of certain sounds, and traditional or foreign words. Another problem is the normalization of semiotic systems (e.g numbers written in digits, abbreviations resolved when pronounced, etc.).

I have developed a tool for the phonemic transcription of Hungarian that performs well even on texts containing a high number of (foreign or other) names. My method is based on three components: the morphological analyzer, Humor, a lexicon for irregular stems and the implementation of phonological rules using *xfst*.

First, the morphological structure of each word is identified. This is necessary in order to find morpheme boundaries to which certain morpho-phonological rules refer. Compounds, which are also quite frequent in Hungarian, might contain components that have an irregular pronunciation. These should also be recognized by the analyzer to avoid their transcription by the regular phonological rules.

Due to the incorporation of Humor, the system is more than a look-up tool for individual words, but is able to transcribe whole phrases or sentences, taking into account sound assimilations appearing at word boundaries as well. Moreover, as the system is not limited to the vocabulary of a prebuilt dictionary, it is capable of transcribing any wordforms, which is of crucial importance in languages like Hungarian, where agglutination and compounding can produce an unlimited number of words. I compared the performance of the system to that of the most popular freely-available grapheme-to-phoneme tool, eSpeak. The word-error-rate of my system was 0.35%, while that of eSpeak was 0.98%.

In 2007, the database of Geographical names of the Institute of Geodesy, Cartography and Remote Sensing (FÖMI) was converted to a phonemic transcription using this tool.

Helyesírás.hu The `helyesiras.mta.hu` website is an interactive automatic online consultation service concerning Hungarian orthography. It is able to tell the user what the orthographically correct form of a query word is. Moreover, it also gives an explanation and presents the corresponding rules of orthography. The website uses different natural language processing tools to analyze the query and to find the answer to specific questions. The Hungarian Humor analyzer, which is capable of returning orthographically relevant information (such as number of syllables and compound members in the stem) explicitly in addition to morphological analyses, is a key component of the system.

A special accent restoration tool: marking the mid-*ë* phoneme The official Hungarian orthography does not make a difference between the short low *e* and the short mid *ë* sounds, as in the standard dialect, there is only a single short *e* phoneme. Though most native speakers of Hungarian are not able to recognize or to produce the latter one, some of those who use it, feel the need to differentiate these two phonemes in written texts as well. I participated in a project, in which a

DOI:10.15774/PPKE.ITK.2015.013

semi-automated tool was created to mark closed *ë* phonemes in texts. In this project, the database of Humor was extended to handle suffixes correctly, moreover these phonemes appearing in stems were marked by a number of competent speakers. The system is able to analyze mid-*ë* marked text, it is able to transform standard texts semi-automatically (the tool has an easy-to-use user interface to correct cases where the system failed to select the contextually correct variant), and it can also be used as a spell checker for this language variant.

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

9

Conclusion
New scientific results

The most important part of this thesis, summarizing new scientific results described in the previous chapters. A must-read.

Contents

DOI:10.15774/PPKE.ITK.2015.013

The better the database of a linguistic program models the language, the better results it can produce. A key module in a linguistic model is the morphological component, which is responsible for the analysis and generation of words in the given language. In this research, I have explored various ways of creating linguistically adequate computational morphologies for morphologically complex languages.

I have created a model for morphological description that has been successfully applied to a number of different languages. The language descriptions and the tools created using the model have been used in various commercial products, such as spell checkers, stemmers, pop-up dictionaries, a rule-based machine translation system, and in various scientific projects.

9.1 A morphological grammar development framework

Creating a morphological analyzer for an agglutinative language is quite a challenge, as the number of morpheme combinations is practically infinite. Thus, the standard methods applied for isolating languages, such as English, do not give satisfactory results for morphologically complex ones. The standard tool used for Hungarian has long been MorphoLogic's Humor ('High speed Unification MORphology') morphological analyzer engine (Prószéky and Kis, 1999). The model this analyzer uses is based on constraints on adjacent morphs. It performs a classical 'item-and-arrangement' (IA)-style analysis. I used this program as the starting point for my research.

Humor analyzes the input word as a sequence of morphs. Each morph is a specific realization (an allomorph) of a morpheme. The word is segmented into parts which have a surface form (that appears as part of the input string, the morph); a lexical form (the 'quotation form' of the morpheme) and a (possibly structured) category label.

The program performs a depth-first search on the input word form for possible analyses. Two kinds of checks are performed at every step: a local compatibility check of the next morph with the previous one and a global word structure check on each locally compatible candidate morph by traversing a deterministic extended finite-state automaton (EFSA) that describes possible word structures. The lexical database of the Humor analyzer consists of an inventory of morpheme allomorphs, the word grammar automaton and two types of data structures used for the local compatibility check of adjacent morphs. One of these are continuation classes and binary continuation matrices describing the compatibility of those continuation classes. The other are binary properties and requirements vectors. Each morph has a continuation class identifier on both its left and right hand sides, in addition to a right-hand-side binary properties vector and a left-hand-side binary requirements vector.

The database is difficult to create and maintain directly in the format used by the analyzer, because it contains redundant and low-level data structures. To avoid these problems, I designed and implemented a morphological grammar development environment that automatically creates the lexical resources used by the Humor analyzer from a high-level human readable redundancy-free morpheme-based grammatical and lexical representation that only contains idiosyncratic features of morphemes and is easy to maintain and extend. Polymorphemic entries, such as compounds, can also be added to the lexicon, and an inheritance mechanism ensures that these entries inherit idiosyncratic properties from their final element by default, thus minimizing redundancy in the description and enhancing consistency. The system also creates a redundant but still easy-to-read intermediate representation that facilitates the checking of the correctness of allomorph creation rules. The system ensures the consistency of the created lexical resources and automatically checks for possible syntactic errors and contradictions in the source descriptions.

DOI:10.15774/PPKE.ITK.2015.013

The high-level human-readable description is transformed by the system to the redundant representations of the analyzer by performing the operations described by the rules and converting the features and constraint expressions using an encoding definition description. This defines how each high-level feature should be encoded for the analyzer. Certain features are mapped to binary properties while the rest determine the continuation matrices, which are generated by the system dynamically. The system is not geared to a particular language; it can be effectively used to describe the morphologies of various languages without any modification of the programs.

All Humor morphologies built after the creation of the development environment were developed using this higher-level formalism.

THESIS 1:

> *I designed and implemented a morphological grammar development environment that automatically creates the lexical resources used by the Humor analyzer from a high-level human readable redundancy-free morpheme-based grammatical and lexical representation that only contains idiosyncratic features of morphemes and is easy to maintain and extend.*

Related publications: 13, 14, 54, 55, 56, 61, 63

9.2 Application of the model to various languages

The development environment was used to create computational morphologies for a number of languages, among them agglutinating ones. I created state-of-the-art morphologies for *Hungarian*, *Spanish* and *French.* In addition, I co-authored Humor morphologies for the following Uralic languages: *Komi*, *Udmurt*, *Mari*, *Northern Mansi* and various *Khanty* dialects. Moreover, based on various morphological descriptions, I also created Humor-compatible morphologies for *Dutch*, *Italian*, *Romanian* and *Russian*, extending the coverage of the original descriptions as required. I also created computational morphologies for two *Samoyedic* languages. Although these agglutinating languages are also members of the Uralic language family, describing their very intricate morpho-phonology using the constraint-based Humor formalism turned out to be too difficult. Thus, these morphologies were implemented using the finite-state formalism of Xerox using *lexc* and *xfst.*

THESIS 2

> *I created and co-authored computational morphologies for several languages.*

THESIS 2a:

> *I created, co-authored or adapted computational morphologies for the following languages in the formalism I introduced demonstrating the capabilities of the framework: Hungarian, Spanish, French, Dutch, Italian, Romanian, Russian, Komi, Udmurt, Mari, Northern Mansi and the Synya and Kazym Khanty dialects.*

THESIS 2b:

> *I created morphologies for two seriously endangered Northern Samoyedic languages, Nganasan and Tundra Nenets, using the Xerox finite-state formalism.*

Related publications: 22, 63, 59, 61, 14, 58, 55, 56, 54, 48, 49, 50

DOI:10.15774/PPKE.ITK.2015.013

9.3 Adaptation of the Hungarian morphology to special domains

Language use in special domains and language variants may deviate in a significant manner from what one encounters in the standard written dialect of the language. The morphological model needs to be adapted when texts from such a special language variant are to be analyzed. Two examples of such phenomena were described here, demonstrating the adaptibility of the analyzer built using the development framework. The first task for which the analyzer was adapted, was the annotation of Old and Middle Hungarian texts. The adapted analyzer can handle extinct morphological constructions as well as dialectal variants missing from Modern Standard Hungarian. The other example is an adaptation of the Hungarian morphology to the clinical domain, where the domain-specific terminology, which includes a vast amount of word forms of foreign origin, had to be treated in a robust manner.

THESIS 3:

I demonstrated the adaptability of morphologies created using the formalism I introduced by extending the Hungarian morphology I created.

THESIS 3a:

I adapted the Humor morphological analyzer for Hungarian to be capable of analyzing words containing morphological constructions, suffix allomorphs, suffix morphemes, paradigms and stems that were used in Old and Middle Hungarian but no longer exist in present-day Hungarian.

Related publications: 38, 42

THESIS 3b:

I created a method for semi-automatically extending the Humor morphological analyzer to be capable of analyzing words used in the clinical language that contains non-standard word constructions, phrases of foreign origin and a high ratio of abbreviations. I created methods to distinguish words of foreign origin, to predict their pronunciation and to predict the part of speech of words to be added to the lexicon. The resulting analyzer is able to analyze medical language in an appropriate and robust manner.

Related publications: 2, 37, 65, 36, 7, 1

9.4 Finite-state implementation of constraint-based morphologies

Humor's closed-source licensing scheme has been a limitation to making resources made for it widely available. Moreover, there are a few limitations of the rule-based Humor engine: lack of support for morphological guessing and the use of frequency information or other weighting of the models. These problems were solved by converting the databases to a finite-state representation that allows morphological guessing and the addition of weights and has open-source implementations.

The Xerox tools (Beesley and Karttunen, 2003) implement a powerful formalism to describe complex types of morphological structures. This suggested that mapping of the morphologies implemented in the Humor formalism to a finite-state representation should have no impediment.

THESIS 4:

I created a method to convert the Humor databases to a finite-state representation that allows morphological guessing and the addition of weights and has open source implementations.

Related publications: 30, 31

DOI:10.15774/PPKE.ITK.2015.013

9.5 Extending morphological dictionary-based models without writing a grammar

Most freely available morphological resources contain no rule component. They are usually based on just a morphological lexicon, containing base forms and some information (often just a paradigm ID) identifying the inflectional paradigm of the word, possibly augmented with some other morphosyntactic features. Resources of this type are much more difficult to extend with new words than rule-based morphologies. However, the application of machine learning methods may be able to make up for the lack of a rule component. I prepared an algorithm that makes the integration of new words into such resources as easy as a rule-based morphology can be extended. This is achieved by predicting the correct paradigm for words, which are not present in the lexicon. The suffix-trie-based supervised learning algorithm is based on longest matching suffixes and lexical frequency data, and it was used to extend a Russian morphology, on which its performance was evaluated. With minimal adaptation, the tool can be used for any language provided there is a morphological resource available. I assumed that a dictionary with some lexical features is also available, thus such features could be used for disambiguating paradigm candidates. The results showed that the method performs with an accuracy of about 90% in all different setups, achieving the best performance on relatively rare words, which are good candidates of being absent in the original lexicon.

I found that assigning more weight to distributions conditioned on longer suffixes than on shorter ones yields much better prediction performance, not only in terms of the number of exact predicted paradigm matches, but especially when taking into account what sorts of errors the system makes. While the baseline suffix guesser algorithm often proposes paradigms inapplicable to the given lexical item, my algorithm makes errors that arise due to the lack of lexical semantic information. Humans would make similar errors in similar situations.

THESIS 5:

> *I prepared an algorithm that makes the integration of new words into lexicon-based morphological resources easy by automatically predicting the correct paradigm for words which are not present in the lexicon.*

Related publications: 27, 5

DOI:10.15774/PPKE.ITK.2015.013

9.6 A flexible model of word form generation and lemmatization

The original Humor system lacked the capability of word form generation and the existing lemmatizer was unable to correctly lemmatize certain non-trivial word constructions. I solved these problems by extending the system so that it can also be used as a morphological generator and implementing better lemmatization algorithms.

The generator produces all word forms that could be realizations of a given morpheme sequence. The input for the generator is a lemma followed by a sequence of category labels that express the morphosyntactic features the word form should expose.

The Humor generator is not a simple inverse of the corresponding analyzer: it can generate the inflected and derived forms of any multiply derived and/or compound stem without explicitly referring to compound boundaries and derivational suffixes in the input even if the whole complex stem is not in the lexicon of the analyzer. This is a useful feature in the case of languages where morphologically very complex stems are commonplace. When generating inflected (or derived) forms of a morphologically complex stem, one does not have to be concerned whether the stem is included in the stem database. If the corresponding analyzer can analyze it in any way, the generator will be able to correctly generate its inflected forms.

It is possible to describe preferences for the cases when a certain set of morphosyntactic features may have more than one possible realization. This can be useful for such applications of the generator as text generation in a machine translation system, where the generation of a single preferred word form is required. The Hungarian morphological description was extended with information expressing markedness. Marked forms are automatically removed during compilation from the version of the database that is intended for word form generation in the machine translation system.

Since there is considerable variation in suffix ordering in some of the languages for which I created morphologies (e.g. Komi), I also created a version of the generator that has another useful feature: it does not assume that the morphosyntactic features are properly ordered in the input, rather it considers them a set.

The Humor 'lemmatizer' tool, built around the analyzer core, does more than just identifying lemmas of word forms: it also identifies the exposed morphosyntactic features. In contrast to the more verbose analyses produced by the core analyzer, compound members and derivational suffixes do not appear as independent items in the output of the lemmatizer, so the internal structure of words is not revealed. The analyses produced by the lemmatizer are well suited for such tasks as corpus tagging, indexing and parsing.

There are two implementations of the lemmatizer. Both can properly handle the task of correctly lemmatizing and filtering special word constructions, e.g. ones that are not (only) suffixed at the end of the word thus improving the accuracy of lemmatization.

THESIS 6:

> *I created new word form generator and lemmatizer tools for the Humor system.*

THESIS 6a:

> *I implemented a word form generator as a Humor module, which can generate the inflected and derived forms of any multiply derived and/or compound stem without explicitly referring to compound boundaries and derivational suffixes in the input even if the whole complex stem is not in the lexicon of the analyzer. The generator was used in various commercial applications and research prototype systems in the domain of information retrieval and machine translation.*

THESIS 6b:

> *I created a lemmatizer tool for Humor, which can properly handle the task of correctly lemmatizing and filtering special word constructions, e.g. ones that are not (only) suffixed at the end of the word. The lemmatizer was used in various corpus annotation projects and it was integrated into information retrieval and machine translation systems.*

Related publications (on applications of the tools): 3, 35, 41, 53, 51, 52, 11, 49

9.7 A tool for annotating and searching text corpora

To support the process of manual checking and the initial manual disambiguation of an annotated corpus, I created a web-based interface where disambiguation and normalization errors can be corrected very effectively. The system presents the document to the user using an interlinear annotation format that is easy and natural to read and it supports handling glosses, normalization and translations.

I also created a web-based corpus query tool, which does not only make it possible to search for different grammatical constructions in the texts, but it is also an effective correction tool. Errors discovered in the annotation or the text appearing in the “results” box can immediately be corrected and the corrected text and annotation is recorded in the database. Naturally, this latter functionality of the corpus manager is only available to expert users having the necessary privileges.

A fast and effective way of correcting errors in the annotation is to search for presumably incorrect structures and to correct the truly problematic ones at once. The corrected corpus can be exported after this procedure and the tagger can be retrained on it.

THESIS 7:

> *I developed a disambiguation system that can be used for automatic and manual disambiguation of the morphosyntactic annotation and glossing of texts and I created a corpus manager appropriate for searching and correcting annotated corpora.*

Related publications: 42, 38, 49, 50

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

10

List of Papers

Journal publications

1 Borbála Siklósi, **Attila Novák**, Gábor Prószéky (2016): Context-aware correction of spelling errors in Hungarian medical documents, In: *Computer Speech & Language*, Vol. 35, pp. 219-233, ISSN 0885-2308, http://dx.doi.org/10.1016/j.csl.2014.09.001.

2 György Orosz, **Attila Novák**, Gábor Prószéky (2014): Lessons learned from tagging clinical Hungarian. In: *International Journal of Computational Linguistics and Applications*, Vol. 5 no. 2. ISSN 0976-0962

3 László János Laki, **Attila Novák**, Borbála Siklósi, György Orosz (2013): Syntax-based reordering in phrase-based English-Hungarian statistical machine translation. In: *International Journal of Computational Linguistics and Applications*, Vol. 4 no. 2. pp. 63–78. ISSN 0976-0962

4 István Endrédy, **Attila Novák** (2013): More effective boilerplate removal – the GoldMiner algorithm. In: *Polibits 48.* pp. 79–83. ISSN 1870-9044

Book chapters

5 **Attila Novák** (2015): Making morphologies the 'easy' way, In: A. Gelbukh (ed.) *Lecture Notes in Computer Science Volume 9041: Computational Linguistics and Intelligent Text Processing* Springer International Publishing, Berlin–Heidelberg. Part I pp. 127–138. ISBN 978-3-319-18110-3

6 Borbála Siklósi, **Attila Novák** (2014): Identifying and Clustering Relevant Terms in Clinical Records Using Unsupervised Methods. In: Besacier, L.; Dediu, A.-H. and Martín-Vide, C. (eds.) *Lecture Notes in Computer Science Volume 8791: Statistical Language and Speech Processing* Springer International Publishing, Berlin–Heidelberg. pp. 233–243 ISBN 978-3-319-11396-8

DOI:10.15774/PPKE.ITK.2015.013

7 Borbála Siklósi, **Attila Novák**, Gábor Prószéky (2013): Context-Aware Correction of Spelling Errors in Hungarian Medical Documents. In: Adrian-Horia Dediu, Carlos Martín-Vide, Ruslan Mitkov, Bianca Truthe (eds.) *Lecture Notes in Computer Science Volume 7978: Statistical Language and Speech Processing, First International Conference, SLSP 2013.* Springer, Berlin Heidelberg. pp. 248–259 ISBN 978-3-642-39592-5

8 György Orosz, László János Laki, **Attila Novák**, Borbála Siklósi (2013): Improved Hungarian Morphological Disambiguation with Tagger Combination. In: Habernal, Ivan; Matousek, Vaclav (eds.) *Lecture Notes in Computer Science, Vol. 8082: Text, Speech, and Dialogue, 16th International Conference, TSD 2013.* Pilsen, Czech Republic. Springer, Berlin–Heidelberg. pp. 280–287. ISBN: 978-3-642-40584-6

9 Nóra Wenszky, **Attila Novák** (2013): The hypercorrect key witness. In: Péter Szigetvári (ed.) *VLlxx: Papers presented to Varga László on his 70th birthday.* Department of English Linguistics, Eötvös Loránd University. ISBN 978-963-284-315-5

10 Borbála Siklósi, **Attila Novák** (2013): Detection and Expansion of Abbreviations in Hungarian Clinical Notes. In: F. Castro, A. Gelbukh, M.G. Mendoza (eds.) *Lecture Notes in Computer Science, Vol. 8265: Advances in Artificial Intelligence and Its Applications.* Springer, Berlin Heidelberg. pp. 318–328. ISBN 978-3-642-45114-0

11 László János Laki, György Orosz, **Attila Novák** (2013): HuLaPos 2.0 – Decoding morphology. In: F. Castro, A. Gelbukh, M.G. Mendoza (eds.) *Lecture Notes in Computer Science, Vol. 8265: Advances in Artificial Intelligence and Its Applications.* Springer, Berlin–Heidelberg. pp. 294–305. ISBN 978-3-642-45114-0

12 György Orosz, **Attila Novák**, Gábor Prószéky (2013): Hybrid text segmentation for Hungarian clinical records. In: F. Castro, A. Gelbukh, M.G. Mendoza (eds.) *Lecture Notes in Computer Science, Vol. 8265: Advances in Artificial Intelligence and Its Applications.* Springer, Berlin–Heidelberg. pp. 306–317. ISBN 978-3-642-45114-0

13 **Novák Attila**, Wenszky Nóra (2007): Mire jó és hogyan készül egy számítógépes morfológia. In: Alberti Gábor, Fóris Ágota (eds.) *A mai magyar formális nyelvtudomány műhelyei.* Nemzeti Tankönyvkiadó, Budapest. 157–169.

14 Gábor Prószéky, **Attila Novák** (2005): Computational Morphologies for Small Uralic Languages. In: A. Arppe, L. Carlson, K. Lindén, J. Piitulainen, M. Suominen, M. Vainio, H. Westerlund, A. Yli-Jyrä (eds.) *Inquiries into Words, Constraints and Contexts. Festschrift in the Honour of Kimmo Koskenniemi on his 60th Birthday.* Gummerus Printing, Saarijärvi/CSLI Publications, Stanford. pp. 116–125.

15 **Novák Attila** (2002): Többértelmű vagy homályos? In: Kálmán László, Trón Viktor, Varasdi Károly (eds.) *Lexikalista elméletek a nyelvészetben.* Tinta Könyvkiadó, Budapest. (Segédkönyvek a nyelvészet tanulmányozásához 13.) pp. 277–287.

16 **Novák Attila** (2002): HPSG fonológia. In: Kálmán László, Trón Viktor, Varasdi Károly (eds.) *Lexikalista elméletek a nyelvészetben.* Tinta Könyvkiadó, Budapest. (Segédkönyvek a nyelvészet tanulmányozásához 13.) pp. 99–128.

DOI:10.15774/PPKE.ITK.2015.013

17 Kálmán László, **Novák Attila** (2001): A magyar egyszerű mondat fajtái. In: Kálmán László (ed.): *Magyar leíró nyelvtan*, Mondattan I. Tinta Könyvkiadó, Budapest, 2001. pp. 10–23.

18 Gyuris Bea, **Novák Attila** (2001): A topik és a kontrasztív topik. In: Kálmán László (ed.): *Magyar leíró nyelvtan, Mondattan I.* Tinta Könyvkiadó, Budapest, 2001. pp. 24–53.

19 **Novák Attila**, Dudás Kálmán, Kálmán László (2001): Igevivők. In: Kálmán László (ed.): *Magyar leíró nyelvtan, Mondattan I.* Tinta Könyvkiadó, Budapest, 2001. pp. 54–75.

20 **Novák Attila** (2001): A kommentelőzmények. In: Kálmán László (ed.): *Magyar leíró nyelvtan, Mondattan I.* Tinta Könyvkiadó, Budapest, 2001. pp. 76–91.

21 **Novák Attila** (2001): A hatókör felszíni egyértelműsítése. In: Kálmán László (eds.) *Magyar leíró nyelvtan, Mondattan I.* Tinta Könyvkiadó, Budapest, 2001. pp. 92–97.

22 **Novák Attila** (1999): Inflectional paradigms in Hungarian – The conditioning of suffix- and stem-alternations (Ragozási paradigmák a magyarban – A toldalék- és tőalternációkat kiváltó tényezők), Szakdolgozat, ELTE Elméleti Nyelvészet Szak, Budapest.

23 **Attila Novák** (1998): HPSG Phonology. In: *Lexicon Matters.* ELTE Theoretical Linguistics Programme, Budapest, 1998. pp. 33–48

24 **Attila Novák** (1998): Ambiguity and Vagueness. In: *Lexicon Matters.* ELTE Theoretical Linguistics Programme, Budapest, 1998. 115–120

Conference proceedings

25 **Novák Attila**, Siklósi Borbála (2015): Automatic Diacritics Restoration for Hungarian. In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics. pp. 2286–91.

26 Siklósi Borbála, **Novák Attila** (2015): Restoring the intended structure of Hungarian ophthalmology documents. In: *Proceedings of the BioNLP 2015 Workshop at the* 53^{rd} *Annual Meeting of the Association for Computational Linguistics, ACL 2015.* Beijing, China. pp. 152–157

27 **Novák Attila** (2015): "Olcsó" morfológia In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *XI. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 145–157

28 Siklósi Borbála, **Novák Attila** (2015): Nem felügyelt módszerek alkalmazása releváns kifejezések azonosítására és csoportosítására klinikai dokumentumokban. In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *XI. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 237–248

29 Borbála Siklósi, **Attila Novák**, Gábor Prószéky (2014): Resolving Abbreviations in Clinical Texts Without Pre-existing Structured Resources. In: *Proceedings of the Fourth Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing (BioTxtM 2014).* Reykjavík. pp. 69–75

30 **Attila Novák** (2014): A New Form of Humor – Mapping Constraint-Based Computational Morphologies to a Finite-State Representation. In: *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC-2014).* Reykjavík. pp. 1068–1073

31 **Novák Attila** (2014): A Humor új Fo(r)mája. In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *X. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 303–308. ISBN 978-963-306-246-3

32 Siklósi Borbála, **Novák Attila** (2014): Rec. et exp. aut. Abbr. mnyelv. KLIN. szöv-ben – rövidítések automatikus felismerése és feloldása magyar nyelvű klinikai szövegekben. In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *X. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 167–176. ISBN 978-963-306-246-3

33 Siklósi Borbála, **Novák Attila** (2014): A magyar beteg. In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *X. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 188–198. ISBN 978-963-306-246-3

34 Orosz György, **Novák Attila** (2014): PurePos 2.0: egy hibrid morfológiai egyértelműsítő rendszer. In: Tanács Attila, Varga Viktor, Vincze Veronika (eds.) *X. Magyar Számítógépes Nyelvészeti Konferencia.* Szegedi Tudományegyetem, Informatikai Tanszékcsoport, Szeged. pp. 373–377. ISBN 978-963-306-246-3

35 Laki László, **Novák Attila**, Siklósi Borbála (2013): Hunglish mondattan – átrendezésalapú angol-magyar statisztikai gépifordító-rendszer. In: Tanács Attila; Vincze Veronika (eds.) *A IX. Magyar Számítógépes Nyelvészeti Konferencia előadásai.* SZTE, Szeged. pp. 71–82 ISBN 978-963-306-189-3

36 Siklósi Borbála, **Novák Attila**, Prószéky Gábor (2013): Helyesírási hibák automatikus javítása orvosi szövegekben a szövegkörnyezet figyelembevételével. In: Tanács Attila; Vincze Veronika (eds.) *A IX. Magyar Számítógépes Nyelvészeti Konferencia előadásai.* SZTE, Szeged. pp. 148–158 ISBN 978-963-306-189-3

37 Orosz György, **Novák Attila**, Prószéky Gábor (2013): Magyar nyelvű klinikai rekordok morfológiai egyértelműsítése. In: Tanács Attila; Vincze Veronika (eds.) *A IX. Magyar Számítógépes Nyelvészeti Konferencia előadásai.* SZTE, Szeged. pp. 159–169 ISBN 978-963-306-189-3

38 **Novák Attila**, Wenszky Nóra (2013): O & kọzepmaǵar zoalactany elemzọ. In: Tanács Attila; Vincze Veronika (eds.) *A IX. Magyar Számítógépes Nyelvészeti Konferencia előadásai.* SZTE, Szeged. pp. 170–181 ISBN 978-963-306-189-3

39 Endrédy István, **Novák Attila** (2013): Egy hatékonyabb webes sablonszűrő algoritmus – avagy miként lehet a cumisüveg potenciális veszélyforrás Obamára nézve. In: Tanács Attila; Vincze Veronika (eds.) *A IX. Magyar Számítógépes Nyelvészeti Konferencia előadásai.* SZTE, Szeged. pp. 297–301 ISBN 978-963-306-189-3

40 György Orosz, László János Laki, **Attila Novák**, Borbála Siklósi (2013): Combining Language-Independent Part-of-Speech Tagging Tools. In: J. P. Leal, R. Rocha, and A. Simoes (eds.) *2nd Symposium on Languages, Applications and Technologies.* Porto: Schloss Dagstuhl–Leibniz-Zentrum für Informatik. pp. 249–257 ISBN 978-3-939897-52-1

41 László János Laki, **Attila Novák**, Borbála Siklósi (2013): English-to-Hungarian Morpheme-based Statistical Machine Translation System with Reordering Rules. In: Marta R. Costa-jussa, Reinhard Rapp, Patrik Lambert, Kurt Eberle, Rafael E. Banchs, Bogdan Babych (eds.) *Proceedings of the Second Workshop on Hybrid Approaches to Machine Translation (HyTra).* Association for Computational Linguistics. pp. 42–50

42 **Attila Novák**, György Orosz, Nóra Wenszky (2013): Morphological annotation of Old and Middle Hungarian corpora. In: Piroska Lendvai, Kalliopi Zervanou (eds.) *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities.* Association for Computational Linguistics. pp. 43–48

43 György Orosz, **Attila Novák** (2013): Purepos 2.0: a hybrid tool for morphological disambiguation. In: Galia Angelova, Kalina Bontcheva, Ruslan Mitkov (eds.) *Proceedings of the international conference Recent Advances In Natural Language Processing RANLP 2013.* Hissar, Bulgaria. pp. 539–545 ISSN 1313-8502

44 György Orosz, **Attila Novák** (2012): PurePos – an open source morphological disambiguator. In: Bernadette Sharp, Michael Zock (eds.) *Proceedings of the 9th International Workshop on Natural Language Processing and Cognitive Science.* Wrocław, Poland. pp. 53–63

45 Borbála Siklósi, György Orosz, **Attila Novák**, Gábor Prószéky (2012): Automatic structuring and correction suggestion system for Hungarian clinical records. In: *LREC-2012: SALTMIL-AfLaT Workshop on "Language technology for normalisation of less-resourced languages".* Istanbul, Turkey, 2012. pp. 29–34

46 Siklósi Borbála, Orosz György, **Novák Attila** (2011): Magyar nyelvű klinikai dokumentumok előfeldolgozása. In: *VIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2011).* Szegedi Tudományegyetem, pp. 143–340

47 **Novák Attila**, Orosz György, Indig Balázs (2011): Javában taggelünk. In: *VIII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2011).* Szegedi Tudományegyetem, pp. 336–340.

48 Fejes László, **Novák Attila** (2010): Obi-ugor morfológiai elemzők és korpuszok. In: *VII. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2010).* Szegedi Tudományegyetem, pp. 284–291

49 Bakró-Nagy Marianne, Endrédy István, Fejes László, **Novák Attila**, Oszkó Beatrix, Prószéky Gábor, Szeverényi Sándor, Várnai Zsuzsa, Wagner-Nagy Beáta (2010): Online morfológiai elemzők és szóalakgenerátorok kisebb uráli nyelvekhez. In: *VII. Magyar*

DOI:10.15774/PPKE.ITK.2015.013

Számítógépes Nyelvészeti Konferencia (MSZNY 2010). Szegedi Tudományegyetem, pp. 345–348

50 István Endrédy, László Fejes, **Attila Novák** , Beatrix Oszkó, Gábor Prószéky, Sándor Szeverényi, Zsuzsa Várnai, Beáta Wágner-Nagy (2010): Nganasan – Computational Resources of a Language on the Verge of Extinction. In: *Creation and Use of Basic Lexical Resources for Less-Resourced Languages: 7th SaLTMiL Workshop (LREC-2010).* La Valletta, Malta, pp. 41–44

51 **Novák Attila**, Prószéky Gábor (2009): Kísérletek statisztikai és hibrid magyar–angol és angol– magyar fordítórendszerek megvalósítására. In: **VI. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2009)**. Szegedi Tudományegyetem, pp. 25–34

52 **Attila Novák** (2009): MorphoLogic's submission for the WMT 2009 Shared Task. In: *Proceedings of the Fourth Workshop on Statistical Machine Translation at EACL 2009.* Athens, Greece. pp. 155–159

53 **Attila Novák**, László Tihanyi, Gábor Prószéky (2008): The MetaMorpho translation system. In: *Proceedings of the Third Workshop on Statistical Machine Translation at ACL 2008.* Columbus, Ohio. pp. 111–114

54 **Attila Novák** (2008): Language resources for Uralic minority languages. In: *Proceedings of the SALTMIL Workshop at LREC-2008: Collaboration: interoperability between people in the creation of language resources for less-resourced languages.* Marrakech, pp. 27–32

55 **Novák Attila**, M. Pintér Tibor (2006): Milyen a még jobb Humor. In: *IV. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2006).* Szegedi Tudományegyetem, pp. 60–69

56 **Attila Novák** (2006): Morphological Tools for Six Small Uralic Languages. In: *Proceedings of The Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, pp. 925–930

57 **Novák Attila**, Endrédy István (2005): Automatikus ë-jelölő program. In: *III. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2005).* Szegedi Tudományegyetem, pp. 453–454

58 **Novák Attila**, Wenszky Nóra (2005): Tundrai nyenyec morfológiai elemző és generátor. In: *III. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2005).* Szegedi Tudományegyetem, pp. 200–208

59 **Novák Attila** (2004): Az első nganaszan szóalaktani elemző. In: *II. Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2004).* Szegedi Tudományegyetem, pp. 195–202

60 **Attila Novák**, Viktor Nagy, Csaba Oravecz (2004): Combining symbolic and statistical methods in morphological analysis and unknown word guessing. In: *Proceedings of The Fourth International Conference on Language Resources and Evaluation (LREC-2004).* Lisbon, pp. 1255–1258

DOI:10.15774/PPKE.ITK.2015.013

61 **Attila Novák** (2004): Creating a Morphological Analyzer and Generator for the Komi language. In: *Proceedings of the SALTMIL Workshop at LREC-2004: First Steps in Language Documentation for Minority Languages.* Lisbon, pp. 64–67.

62 **Novák Attila**, Nagy Viktor, Oravecz Csaba (2003): Magyar ismeretlenszó-elemző program fejlesztése. In: *Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2003).* Szegedi Tudományegyetem, 45–57

63 **Novák Attila** (2003): Milyen a jó Humor? In: *Magyar Számítógépes Nyelvészeti Konferencia (MSZNY 2003).* Szegedi Tudományegyetem, pp. 138–145

64 **Attila Novák**, Viktor Nagy, Csaba Oravecz (2003): Corpus assisted development of a Hungarian morphological analyser and guesser. In: Dawn Archer, Paul Rayson, Andrew Wilson and Tony McEnery (eds.) *Proceedings of the Corpus Linguistics 2003 conference.* UCREL technical paper number 16. UCREL, Lancaster University, pp. 583–590

Research reports

65 Borbála Siklósi, **Attila Novák**, György Orosz, Gábor Prószéky (2014): Processing noisy texts in Hungarian: a showcase from the clinical domain, In: Péter Szolgay (ed.), *Jedlik Laboratories Reports*, Vol. II, no. 3, pp. 5–62 ISSN 2064-3942

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

List of Figures

DOI:10.15774/PPKE.ITK.2015.013

DOI:10.15774/PPKE.ITK.2015.013

List of Tables

DOI:10.15774/PPKE.ITK.2015.013

Bibliography

Aleksa, M. (2006). Automatic morphological analysis of the croatian language: The verbal, adjectival and nominal inflections within the morphological parser humor. In *CESCL1, Proceedings of the First Central European Student Conference in Linguistics*, Budapest. Institute for Lingistics, Hungarian Acadey of Sciences.

Allauzen, C., Riley, M., Schalkwyk, J., Skut, W., and Mohri, M. (2007). OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In Holub, J. and Zdárek, J., editors, *Proceedings of the Ninth International Conference on Implementation and Application of Automata, (CIAA 2007)*, volume 4783 of *Lecture Notes in Computer Science*, pages 11–23. Springer.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology.* CSLI Publications, Ventura Hall.

Beznosikova, L., editor (2000). *Komi-Roč Kyvčukör.* Komi Nebör Ledzanin, Syktyvkar.

Brants, T. (2000). TnT – A Statistical Part-of-Speech Tagger. In *Proceedings of the Sixth Conference on Applied Natural Language Processing*, pages 224–231. Association for Computational Linguistics.

Camden, W. (1605). *Remaines of a greater worke, concerning Britaine.*

Cavnar, W. B. and Trenkle, J. M. (1994). N-Gram-Based Text Categorization. In *Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, Las Vegas, US.

Chomsky, N. and Halle, M. (1968). *The Sound Pattern of English.* Harper & Row, New York, NY.

Creutz, M., Hirsimäki, T., Kurimo, M., Puurula, A., Pylkkönen, J., Siivola, V., Varjokallio, M., Arisoy, E., Saraçlar, M., and Stolcke, A. (2007). Morph-based Speech Recognition and Modeling of Out-of-vocabulary Words Across Languages. *ACM Trans. Speech Lang. Process.*, 5(1):3:1–3:29.

Creutz, M. and Lagus, K. (2007). Unsupervised models for morpheme segmentation and morphology learning. *ACM Trans. Speech Lang. Process.*, 4(1):3:1–3:34.

Daciuk, J., Watson, B. W., Mihov, S., and Watson, R. E. (2000). Incremental construction of minimal acyclic finite-state automata. *Comput. Linguist.*, 26(1):3–16.

Dreyer, M. and Eisner, J. (2011). Discovering Morphological Paradigms from Plain Text Using a Dirichlet Process Mixture Model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 616–627, Stroudsburg, PA, USA. Association for Computational Linguistics.

Elekfi, L. (1994). *Magyar ragozási szótár [Dictionary of Hungarian inflections].* MTA Nyelvtudományi Intézet, Budapest.

Fábián, P. and Magasi, P. (1992). *Orvosi helyesírási szótár.* Akadémiai Kiadó, Budapest.

Forsberg, M., Hammarström, H., and Ranta, A. (2006). Morphological Lexicon Extraction from Raw Text Data. In Salakoski, T., Ginter, F., Pyysalo, S., and Pahikkala, T., editors, *Advances in Natural Language Processing*, volume 4139 of *Lecture Notes in Computer Science*, pages 488–499. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gábor, K. (2010). Creating a shallow-parsed Hungarian corpus with NooJ. In Váradi, T., Kuti, J., and Silberztein, M., editors, *Applications of Finite-State Language Processing: Selected Papers from the 2008 International NooJ Conference*, pages 318–328. Cambridge Scholars Publishing, Newcastle upon Tyne.

Goldsmith, J. (2001). Unsupervised Learning of the Morphology of a Natural Language. *Comput. Linguist.*, 27(2):153–198.

Hajič, J. (2001). *Disambiguation of Rich Inflection - Computational Morphology of Czech*, volume I. Prague Karolinum, Charles University Press. 334 pp.

DOI:10.15774/PPKE.ITK.2015.013

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., and Trón, V. (2003). A Szószablya projekt. In *I. Magyar Számítógépes Nyelvészeti Konferencia*, Szeged. Szegedi Tudományegyetem.

Halácsy, P., Kornai, A., Németh, L., Rung, A., Szakadát, I., and Trón, V. (2004). Creating open language resources for hungarian. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal.*

Halácsy, P., Kornai, A., and Oravecz, C. (2007). HunPos: an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 209–212, Stroudsburg, PA, USA. Association for Computational Linguistics.

Hammarström, H. and Borin, L. (2011). Unsupervised Learning of Morphology. *Comput. Linguist.*, 37(2):309–350.

Helimski, E. (1998). Nganasan. In Abondolo, D., editor, *The Uralic Languages*, pages 480–515. Routledge, London.

Hockett, C. J. (1954). Two Models of Grammatical Description. *Word*, 10:210–231. Reprinted in M. Joos, ed. (1957), ıt Readings in Linguistics I, 386–399.

Hoeksema, J. and Janda, R. (1988). Implications of Process-Morphology for Categorial Grammar. In Oehrle, R., Bach, E., and Wheeler, D., editors, *Categorial Grammars and Natural Language Structures*, volume 32 of *Studies in Linguistics and Philosophy*, pages 199–247. Springer Netherlands.

Huldén, M. (2009). Foma: a Finite-State Compiler and Library. In Lascarides, A., Gardent, C., and Nivre, J., editors, *Proceedings of EACL 2009*, pages 29–32, Athens, Greece. The Association for Computer Linguistics.

Huldén, M. and Francom, J. (2012). Boosting statistical tagger accuracy with simple rule-based grammars. In Chair), N. C. C., Choukri, K., Declerck, T., Doğan, M. U., Maegaard, B., Mariani, J., Odijk, J., and Piperidis, S., editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey. European Language Resources Association (ELRA).

Jackendoff, R. (1977). *X-bar-Syntax: A Study of Phrase Structure.* Linguistic Inquiry Monograph 2. MIT Press, Cambridge, MA.

Jakab, L. (2002). *A Jókai-kódex mint nyelvi emlék szótárszerű feldolgozásban.* Számítógépes Nyelvtörténeti Adattár. Debreceni Egyetem, Debrecen.

Jakab, L. and Kiss, A. (1994). *A Guarÿ-kódex ábécérendes adattára.* Számítógépes nyelvtörténeti adattár. Debreceni Egyetem, Debrecen.

Jakab, L. and Kiss, A. (1997). *Az Apor-kódex ábécérendes adattára.* Számítógépes nyelvtörténeti adattár. Debreceni Egyetem, Debrecen.

Jakab, L. and Kiss, A. (2001). *A Festetics-kódex ábécérendes adattára.* Számítógépes nyelvtörténeti adattár. Debreceni Egyetem, Debrecen.

Johnson, C. D. (1972). *Formal Aspects of Phonological Description.* Mouton, The Hague.

Jurafsky, D. and Martin, J. H. (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition.

Kaplan, R. M. and Kay, M. (1994). Regular Models of Phonological Rule Systems. *Computational Linguistics*, 20(3):331–378.

Kálmán, B. (1963). *Chrestomathia Vogulica.* Tankönyvkiadó, Budapest.

Kálmán, B. (1976). *Wogulische Texte mit einem Glossar.* Budapest.

DOI:10.15774/PPKE.ITK.2015.013

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the ACL 2007 Demo and Poster Sessions*, pages 177–180, Prague. Association for Computational Linguistics.

Koskenniemi, K. (1983). Two-level Morphology: a General Computational Model for Word-form Recognition and Production. Technical Report 11, Department of General Linguistics, University of Helsinki.

Kost'erkina, N. T., Momd'e, A. ., and Ždanova, T. J. (2001). *Slovar' nganasansko-russkij i russko-nganasanskij.* Prosvesčen'ije, Sankt-Pet'erburg.

Kozmács, I. (2002). *Udmurt-Magyar Szótár.* Savaria University Press, Szombathely.

Labanauskas, K., editor (2001). *Nganasanskaja Fol'klornaja Xrestomat'ija.* Fol'klor Narodov Tajmyra 6. Tajmyrskij Okružnyj Centr Narodnogo Tvorčestva, Dud'inka.

Laki, L. J., Novák, A., and Siklósi, B. (2013a). English to Hungarian Morpheme-based Statistical Machine Translation System with Reordering Rules. pages 42–50, Sofia, Bulgaria. Association for Computational Linguistics.

Laki, L. J., Novák, A., and Siklósi, B. (2013b). Syntax-based reordering in phrase-based English-Hungarian statistical machine translation. *International Journal of Computational Linguistics and Applications.*

Laki, L. J., Orosz, G., and Novák, A. (2013c). HuLaPos 2.0 – Decoding Morphology. In Castro, F., Gelbukh, A., and González, M., editors, *Advances in Artificial Intelligence and Its Applications*, volume 8265 of *Lecture Notes in Computer Science*, pages 294–305. Springer Berlin Heidelberg.

Lindén, K. (2009). Entry Generation by Analogy – Encoding New Words for Morphological Lexicons. *Northern European Journal of Language Technology*, 1(1):1–25.

Lindén, K., Silfverberg, M., Axelson, E., Hardwick, S., and Pirinen, T. (2011). HFST—Framework for Compiling and Applying Morphologies. In Mahlow, C. and Pietrowski, M., editors, *Systems and Frameworks for Computational Morphology*, volume Vol. 100 of *Communications in Computer and Information Science*, pages 67–85.

Matthews, P. H. (1991). *Morphology 2nd Edition.* Cambridge Textbooks in Linguistics. Cambridge University Press, Cambridge.

Šmerk, P. (2009). Fast morphological analysis of czech. In *Proceedings of the Raslan Workshop 2009*, Brno. Masarykova univerzita.

Minnen, G., Carroll, J., and Pearce, D. (2001). Applied morphological processing of English. *Natural Language Engineering*, 7:207–223.

Monson, C., Carbonell, J. G., Lavie, A., and Levin, L. S. (2008). ParaMor: Finding Paradigms across Morphology. In Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D. W., Peñas, A., Petras, V., and Santos, D., editors, *Advances in Multilingual and Multimodal Information Retrieval*, volume 5152 of *Lecture Notes in Computer Science*, pages 900–907. Springer Berlin Heidelberg.

Munkácsi, B. (1892). *Vogul Népköltési Gyűjtemény.* Magyar Tudományos Akadémia, Budapest.

Munkácsi, B. (1986). *Wogulisches Wörterbuch.* Akadémiai Kiadó, Budapest.

Šnajder, J. (2013). Models for predicting the inflectional paradigm of Croatian words. In *Slovenšcina 2.0*, pages 1–34.

Nakov, P., Bonev, Y., Angelova, G., Gius, E., and von Hahn, W. (2005). Guessing morphological classes of unknown German nouns. In Nicolov, N., Bontcheva, K., Angelova, G., and Mitkov, R., editors, *RANLP*, volume 260 of *Current Issues in Linguistic Theory (CILT)*, pages 347–356. John Benjamins, Amsterdam/Philadelphia.

DOI:10.15774/PPKE.ITK.2015.013

Németh, L., Trón, V., Halácsy, P., Kornai, A., Rung, A., and Szakadát, I. (2004). Leveraging the open-source ispell codebase for minority language analysis. In *Proceedings of SALTMIL 2004*. European Language Resources Association.

Novák, A. (1999). Inflectional paradigms in hungarian – the conditioning of suffix and stem alternations. Master's thesis, ELTE Theoretical Linguistics Programme, Budapest.

Novák, A. (2008). Language resources for Uralic minority languages. In *Proceedings of the SALTMIL Workshop at LREC-2008: Collaboration: Interoperability between People in the Creation of Language Resources for Less-resourced Languages*, pages 27–32.

Novák, A., Tihanyi, L., and Prószéky, G. (2008). The MetaMorpho Translation System. In *Proceedings of the Third Workshop on Statistical Machine Translation*, StatMT '08, pages 111–114, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oflazer, K. (1993). Two-level description of turkish morphology. In *Proceedings of the Sixth Conference on European Chapter of the Association for Computational Linguistics*, EACL '93, pages 472–472, Stroudsburg, PA, USA. Association for Computational Linguistics.

Oliver, A. and Tadić, M. (2004). Enlarging the Croatian Morphological Lexicon by Automatic Lexical Acquisition from Raw Corpora. In *LREC*. European Language Resources Association.

Oravecz, C. and Dienes, P. (2002). Efficient Stochastic Part-of-Speech Tagging for Hungarian. In *Proceedings of the 3rd Language Resources and Evaluation Conference*, pages 710–717, Las Palmas, Espanha.

Oravecz, C., Váradi, T., and Sass, B. (2014). The hungarian gigaword corpus. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Orosz, G. and Novák, A. (2012). PurePos – an open source morphological disambiguator. In Sharp, B. and Zock, M., editors, *Proceedings of the 9th International Workshop on Natural Language Processing and Cognitive Science*, pages 53–63, Wroclaw.

Orosz, G. and Novák, A. (2013). PurePos 2.0: a hybrid tool for morphological disambiguation. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing*, pages 539–545, Hissar, Bulgaria.

Orosz, G., Novák, A., and Prószéky, G. (2013). Hybrid text segmentation for Hungarian clinical records. In Castro, F., Gelbukh, A., and González, M., editors, *Advances in Artificial Intelligence and Its Applications*, volume 8265 of *Lecture Notes in Computer Science*, pages 306–317. Springer Berlin Heidelberg, Heidelberg.

Packard, D. W. (1973). Computer-Assisted Morphological Analysis of Ancient Greek. In *COLING*, pages 343–356.

Papp, F. e. (1969). *A magyar nyelv szóvégmutató szótára (Reverse-alphabetized Dictionary of the Hungarian Language)*. Akadémiai Kiadó, Budapest.

Paumier, S., Nakamura, T., and Voyatzi, S. (2009). UNITEX, a Corpus Processing System with Multi-Lingual Linguistic Resources. In *eLexicography in the 21st century: new challenges, new applications (eLEX'09)*, pages 173–175.

Petersen, U. (2004). Emdros — a text database engine for analyzed or annotated text. In *Proceedings of COLING 2004.*, pages 1190–1193.

Peterson, J. L. (1980). *Computer programs for spelling correction : an experiment in program design*. Lecture notes in computer science. Springer-Verlag, Berlin, New York. Includes index.

Petitpierre, D. and Russell, G. (1994). MMORPH - The Multext morphology program.

Porter, M. (1980). An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

DOI:10.15774/PPKE.ITK.2015.013

Prószéky, G. (2001). Az ‘újrafelhasznált’ szóvégmutató szótár. *Modern filológiai közlemények*, 3(2):121–123.

Prószéky, G. and Kis, B. (1999). A unification-based approach to morpho-syntactic parsing of agglutinative and other (highly) inflectional languages. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL ’99, pages 261–268, College Park, Maryland. Association for Computational Linguistics.

Prószéky, G. and Tihanyi, L. (2002). MetaMorpho: A Pattern-Based Machine Translation System. In *Proceedings of the 24th ’Translating and the Computer’ Conference*, ASLIB, pages 19–24, London, United Kingdom.

Ritchie, G. D., Black, A. W., Russell, G. J., and Pulman, S. G. (1992). *Computational Morphology: Practical Mechanisms for the English Lexicon.* MIT Press, Cambridge Mass.

Salminen, T. (1997). *Tundra Nenets inflection.* Mémoires de la Société Finno-Ougrienne 227, Helsinki.

Siklósi, B. and Novák, A. (2013). Detection and Expansion of Abbreviations in Hungarian Clinical Notes. In Castro, F., Gelbukh, A., and González, M., editors, *Advances in Artificial Intelligence and Its Applications*, volume 8265 of *Lecture Notes in Computer Science*, pages 318–328. Springer Berlin Heidelberg, Heidelberg.

Siklósi, B., Orosz, G., Novák, A., and Prószéky, G. (2012). Automatic structuring and correction suggestion system for Hungarian clinical records. In De Pauw, G., De Schryver, G.-M., Forcada, M., M. Tyers, F., and Waiganjo Wagacha, P., editors, *8th SaLTMiL Workshop on Creation and use of basic lexical resources for less-resourced languages*, pages 29–34, Istanbul, Turkey.

Silberztein, M. (1994). Intex: A corpus processing system. In *COLING*, pages 579–583.

Silberztein, M. (2005). Nooj: a linguistic annotation system for corpus processing. In *HLT-Demo ’05 Proceedings of HLT/EMNLP on Interactive Demonstrations*, pages 10–11. The Association for Computational Linguistics.

Sokirko, A. V. (2004). Morphological modules at the site www.aot.ru. In *Dialog’2004*.

Stolcke, A., Zheng, J., Wang, W., and Abrash, V. (2011). SRILM at sixteen: Update and outlook. In *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, Waikoloa, Hawaii.

Trón, V. (2004). Hunlex – morfológiai szótárkezelő rendszer. In *II. Magyar Számítógépes Nyelvészeti Konferencia*, Szeged. Szegedi Tudományegyetem.

Trón, V., Halácsy, P., Rebrus, P., Rung, A., Vajda, P., and Simon, E. (2006). Morphdb.hu: Hungarian lexical database and morphological grammar. In *Proceedings of the Fifth conference on International Language Resources and Evaluation*, pages 1670–1673, Genoa.

Trón, V., Kornai, A., Gyepesi, G., Németh, L., Halácsy, P., and Varga, D. (2005). Hunmorph: Open Source Word Analysis. In *Proceedings of the Workshop on Software*, Software ’05, pages 77–85, Stroudsburg, PA, USA. Association for Computational Linguistics.

Várnai, Z. (2002). Hangtan. In Wagner-Nagy, B., editor, *Chrestomathia Nganasanica*, Studia Uralo-Altaica Supplementum 10. SZTE Finnugor Tanszék – MTA Nyelvtudományi Intézet, Szeged – Budapest.

Wagner-Nagy, B., editor (2002). *Chrestomathia Nganasanica.* Studia Uralo-Altaica Supplementum 10. SZTE Finnugor Tanszék – MTA Nyelvtudományi Intézet, Szeged – Budapest.

Wicentowski, R. (2002). Modeling and Learning Multilingual Inflectional Morphology in a Minimally Supervised Framework. Technical report.

Wołosz, R. (2005). *Efektywna metoda analizy i syntezy morfologicznej w jezyku polskim.* Problemy współczesnej nauki, Teoria i zastosowania. Akademicka Oficyna Wydawnicza Exit.

Young, C. and Chan, E. (2009). Review of Finite State Morphology. *Word Structure*, pages 245–254.

Zaliznyak, A. A. (1980). *Russian grammatical dictionary – Inflection.* Russkij Jazyk, Moskva.

DOI:10.15774/PPKE.ITK.2015.013

A

Appendix

A.1 Format of the rule files

A.1.1 Variable declaration and manipulation

A.1.1.1 Declaring named scalar variables

Declaration example	1. `$V="[aeiouüöAEIOUÜÖ]";` #short vowels (this is a comment) 2. `$C="(?:dzs|[ds]z|[cz]s|[ltgn]y|[rtpsdfghjklcvbnmz])";` #consonants
Description of the declaration	The name of scalar variables begins with a `$` sign and contains one or more alphanumeric characters `[a-zA-Z0-9_]` the first of which is not a digit. When assigning a value to the variable, the name of the variable is followed by an equal sign and the value assigned to the variable between single or double quotes. Double quoted strings are "interpolated" in the sense that variable names appearing in the string are substituted with the value of the variable.

Usage example (in regular expressions – i.e. between //)	`#jósol/kotor`
	1. `root:/($C)$V([lr])$/&&!-ik #if the root ends in a consonant+`
	`#short vowel+[l or r] sequence and is not an -ik verb ...`
	`...`
	`#sokall, rühell...`
	2. `root:/$Vll$/&&!1syl #if the root ends in a short vowel+ll`
	`#sequence and is not monosyllabic...`

Using the variable	The same kind of variable interpolation happens in regular expressions (between //). If the name of the variable is followed by alphanumeric characters in the string that could be mistakenly be regarded to be part of the name, you can put the alphanumeric part of the variable name between braces: in `$Vll` only `$V` is the variable (followed by the string `ll`, see example 2.).

DOI:10.15774/PPKE.ITK.2015.013

A.1.1.2 Declaring named list variables

Declaration example	1. `@ppron=qw/én te õ mi ti õ/;#personal pronouns` 2. `@ppron=('én','te','õ','mi','ti','õ');# this is equivalent` 3. `@psfxneki=qw/em[e1] ed[e2] i[e3] ünk[t1] tek[t2] ik[t3]/;`
Description of the declaration	The name of list variables begins with a @ sign and contains one or more alphanumeric characters `[a-zA-Z0-9_]` the first of which is not a digit. When assigning a value to the variable, the name of the variable is followed by an equal sign and the value assigned to the variable. The value is a list that can be given as a comma separated list of values between parentheses (example 2.), or, in the case of the list of words, as a whitespace separated list of words preceded by `qw/` and followed by `/` (examples 1. and 3.).

Usage example (generating the dative forms of personal pronouns)	`root:/neki/&&humor:DAT`
	`+//@ppron+nek+@psfxneki/;;`

Using the list variable	List variables can be used in an allomorph generation block to generate a list of polymorphemic entries. Note that when using list variables, it is obligatory to include the braces around the name of the variable (e.g. `@ppron`) even if no alphanumeric character follows it. If more than one list appears in the allomorph generation expression, they have to have the same length, and the resulting list will contain strings in which the nth element of the one list is concatenated with the nth element of the others. E.g. the usage example above produces the following polymorphemic entries (assuming the declarations of `@ppron` and `@psfxneki` as given in examples 1. and 3. above):
	`én+nek+em[e1], te+nek+ed[e2], õ+nek+i[e3],`
	`mi+nek+ünk[t1], ti+nek+tek[t2], õ+nek+ik[t3]`

A.1.1.3 Declaring local attribute names

Declaration example	`my root;`
	`my seg;`
	`my equ;`
	`my phon;`
	`my* humor;`
	`my stemalt;`

Description of the declaration	Local attributes are declared using the my keyword, which is followed by the name of the attribute to be handles locally. The declaration is ended by a semicolon. The keyword `my` may be followed by an asterisk (`my*`).

Declaring an attribute local has the following effects:

- You can refer to the value of the attributes as `$rp`, `$rr` etc. in attribute manipulation instructions, i.e. you can refer to them the same way as to scalar variables. Attributes not declared local cannot be referred to as `$attr`, they must be referred to as `$mrf >'attr'`.
- Locally handled attributes are manipulated more efficiently than non-localized ones.
- Local attributes are not written to the level 2 lexicon file (i.e. for local attributes, the level 2 file contains exactly the same attributes and values as given in the level 1 lexicon file) *unless* the `my` keyword is followed by an asterisk (e.g. `my* humor;`). If the `phon` attribute is declared to be local (using `my`), for example, the `phon` attribute will only appear in entries which already had it in the level 1 lexicon (i.e. only irregular pronunciation will appear in the level 2 lexicon), even if the pronunciation is calculated for every inflectable lexical item in the rule file, as in the following example 2.

DOI:10.15774/PPKE.ITK.2015.013

Usage example

1. `;;root:$seg; #the root is the same as seg`
2. `!phon:/./;;phon:$root; #phon defaults to be the same as the root # (unless otherwise specified)`

A.1.2 Attribute manipulation instructions

Examples

1. `;;Cfin; #add the property Cfin to rp`
2. `phon:/$C2$/;;DIG; #mark it digraph-final if it is`
3. `Vfin&&!ifin&&!rr:vST;;=jA =i;`
4. `else rr:(SVS|vST|VZA);;!=jA !=jAi;`
5. `humor:/MNa/||seg:/.?bb$/;;non_grad&&no_sÁg;`
6. `rp:s/VHVB/VHFU/;Cini;;`

Description

The instruction consists of the following fields (delimited by semicolons): `<conditionals>; <requirement setting>; <property setting>;` In examples 1–5., there is no requirement setting (note the two semicolons `;;` in the middle). In example 1., the conditional part is also missing (note the two semicolons `;;` at the beginning of the line). In example 6., the property setting is missing (note the two semicolons `;;` at the end of line).

The conditional

Example 2. shows that in the conditional part you may check a property of an attribute: here a regular expression (/$C2$/) is matched against the value of the attribute `phon`. The attribute to use for checking is `rp` (='right properties') by default (in example 3., `rp` is checked for the presence of the `Vfin` feature and the absence of the `ifin` feature (expressed by the negation operator `!`, see below)). If you want to check a different attribute, you have to prefix the name of it with a colon to the checking expression. In example 3., the absence of the `vST` feature is checked in `rr` (expressed as `!rr:vST`). If the checking expression contains a regular expression operator (`//`) or a substitution operator (`s///`, see below), using the attribute name prefix is always mandatory (even if the attribute involved is `rp`, see example 6.).

Using Boolean operators

You can check more than one condition and use the conjunction ("and", `&&`) or the disjunction ("or", `||`) operators between the checking expressions (examples 3. and 5.). You can also use negation (marked by a `!` before the expression to be negated).

You can not at present use parentheses in the Boolean conditional expressions. (And thus negation may only appear before atomic expressions.) The reason for this is the following:

Not only expressions delimited by slashes `//`, but also the other checking expressions not containing slashes are implemented as regular expression matching. Example 4. shows that this is the case: `(SVS|vST|VZA)` is really a regular expression containing the regular expression disjunction operator `|` (this is a single `|` in contrast to the double `||` of Boolean disjunction) and the grouping operator `()` (i.e. parentheses). Since parentheses are part of the regular expression syntax, they are never considered to partition the Boolean conditional expression. (Note that the parentheses are in fact superfluous in the case of `(SVS|vST|VZA)`.)

	In the Boolean expression, negation (`!`) has the highest precedence and disjunction (`||`) has the lowest. Thus `A&&!B||C` is interpreted as `(A&&(!B))||C`. Note that you can always use regexp disjunction (`|`) instead of Boolean disjunction (`||`) if the two conditions to be checked refer to the same attribute. (In example 4. the all refer to the attribute `rr` (='right requirements').) The regexp disjunction (`|`) has higher precedence than either negation (`!`) or (Boolean) conjunction (`&&`).
The evaluation of Boolean expressions	Only as much is evaluated of the Boolean expression as is needed for the determination of its truth value: if a left conjunct is false, the right one is not evaluated (and the whole expression is false); similarly, if the left member of a disjunction is true, the right one is not evaluated (and the whole expression is true).
Using `else`	Example 4. also shows that the keyword else may appear at the beginning of the conditional. In that case, the whole instruction is executed only if the conditional of the previous instruction was false. This means that example 4. is never executed if example 3. is.
Using substitution in the conditional	Example 6. also shows that the conditional may even contain a sort of expression that actually changes the value of the attribute to which it refers to: `rp:s/VHVB/VHFU/` changes the first occurrence of the string `VHVB` to `VHFU` within the value of the attribute `rp`. `s/regex/subst/` is a regular expression based substitution expression, which changes the substring matched by the regular expression `regex` to the string given as `subst`. If you add a switch `g` to the end: `s/regex/subst/g`, then not only the first matched substring is replaced, but all such substrings are (`g` stands for 'global' matching).

A.1.2.1 Manipulating properties and requirements of morphemes

1. Add left or right properties if certain conditions are met.

Examples	1. `phon:/$V_$/;;Vfin;` #add the property Vfin to rp if vowel final 2. `allomf:/^$/;;lp:0mrf;` #mark zero morphs 3. `Vfin&&!ifin&&!rr:vST;;=jA =i;`
Description	The third field is normally used to set properties. The instruction affects the attribute `rp` by default. Use a prefix to affect another attribute (`lp` in example 2.) You can add more than one property by giving a space separated list of them (example 3.).

2. Delete left or right properties if certain conditions are met

Examples	1. `rr:(SVS|vST|VZA);;!=jA !=jAi;`
Description	You can also delete properties from `rp`, `lp` or `gp` by preceding them by an `!` in field 3. If the condition in example 1. is met, the properties `=jA` and `=jAi` are deleted from `rp`. (In fact, an `!` in field 3 can be used to remove a word from the value of any attribute, including the removal of requirements.)

DOI:10.15774/PPKE.ITK.2015.013

3. Add left or right requirements if certain conditions are met

Examples	1. `rp:s/VHVB/VHFU/;Cini;;` 2. `lp:comp2;lr:!cat_vrb;;` #right compound members must follow a nominal (non-verbal) stem
Description	You can also add requirements. Field 2 is normally used for this purpose. The attribute affected by the operation in field 2 is `rr` (='right requirements') by default. If you want to add a requirement to another attribute (e.g. to `lr`), you must use a prefix (example 2.). When used in field 2, the `!` does not mark that the requirement should be removed but it marks the addition of a negative requirement (i.e. in contrast to field 3, the `!` is not treated specially; example 2.).

A.1.2.2 Manipulation of the values of other morpheme level attributes

1. Simple value assignment

Examples	1. `;;root:$seg;` #the root is the same as seg 2. `!phon:/./;;phon:$root;` #phon defaults to be the same as the root (unless otherwise specified)
Description	You can use field 3 for the purpose of simple value assignment. If you change the value of a non-list valued attribute (i.e. something other than the property or requirement list attributes: `rp, rr, lp, lr, gp, glr, grr`), then the effect is not the addition of the value to the value list, but simply the assignment of the given value to the attribute. The result of `;;root:$seg;` is not the concatenation of the value of the `seg` attribute with that of the root, but the root attribute is simply assigned the same value as the `seg` attribute. Value assignment can of course be combined with condition checking (if given in field 1, see example 2.). You can refer to the values of attributes as `$attr` if you declared them to be local using `my attr;`. Otherwise you must refer to them as `$mrf->'attr'`.

2. Regular expression based substitution

<table>
<tr><td>Examples</td><td>1. <code>root:s/["?!#=%@(̂){}]|[<[].*?[]>]|\.\.\.//g;;;</code> #remove special segmentation characters from root
2. <code>seg:/=ik$/&&root:s/ik$//;;-ik;</code></td></tr>
<tr><td>Description</td><td>Regular expression based substitution has the following syntax: <code>attr:s/regexp/subst/;;;</code> (see example 1.)
You can use field 1 (i.e. the condition field) for this purpose. If substitution has a precondition, you can add that before the substitution expression using conjunction (see example 2.), and, since the right conjunct is not evaluated if the left conjunct is false, no substitution occurs if the precondition fails. The substitution expression itself is true if actual substitution occurs. Thus if the attribute manipulation instruction also specifies e.g. the addition of a property (in field 3, <code>-ik;</code> in example 2.), this addition only occurs if both the precondition is satisfied and actual substitution occurs.</td></tr>
</table>

DOI:10.15774/PPKE.ITK.2015.013

3. Character translation

Examples	1. `phon:tr/A-ZÁÉÍÓÚÖÜÕÛ/a-záéíóúöüõû/;;;` #decapitalize phon
Description	The format of character translation is: `attr:tr/fromchars/tochars/;;;` You can use field 1 (i.e. the condition field) for this purpose. Every occurrence of the nth character from between the 1st pair of slashes is replaced by the nth character from between the 2nd pair of slashes. You can use character ranges (as example 1. shows). A range is actually an ASCII code range, thus accented characters must be listed explicitly. All the remarks about the preconditions etc. described at the regular expression based substitution apply here as well. One important difference between character translation and regular expression based substitution is that in the case of the former the match and the replacement character lists are taken verbatim: variables are not interpolated into either of them.

A.1.3 Blocks of statements

A.1.3.1 Conditional blocks

Examples

```
1. if(cat_(N|Adj|Num|Part))
    {
        ... #all rules specific to nominal stems go within this block
    }
2. unless(phon:/(?:[aeoö]|$C2)$/||GEM||stemalt:/./)
    {
        ... #the most common case: no alternation
    }
    elsif(!stemalt:/./)
    {
    ... #productive alternations
    }
    else
    {
    ... #improductive stem alternations
    }
```

The `if` block	The most common conditional block is the `if` block (example 1.). The modifier of the `if` block is of the form `if(<Boolean expression>)`, i.e. the keyword `if` is followed by a Boolean expression in parentheses. The block following the `if` modifier is only evaluated if the condition expressed by the Boolean expression is true. All the properties of the conditional field of the attribute manipulation instruction apply to the Boolean expression in the conditional block modifier: you can use Boolean operators (`&&`, `||` and `!`, but not parentheses), regular expressions, attribute name prefixes (the default attribute to check is `rp`), substitution etc.

The unless block

The only difference between the if block and the unless block (example 2.) is that in the case of the latter the block is only executed if the Boolean expression in the modifier is false.

The elsif block

An elsif block (example 2.) is executed if neither the preceding if or unless block or any of the preceding elsif blocks were executed and the Boolean expression in the modifier is evaluated to be true.

The else block

An else block (example 2.) is executed if neither the preceding if or unless block or any of the preceding elsif blocks were executed. An else or elsif block can also be used right after a single attribute manipulation instruction if that instruction has a conditional.

```
cat_N&&!no_-Vs;;-Vs;
else
{
...
}
```

A.1.3.2 Generating allomorphs and setting their properties (allomorph generation blocks)

Examples

```
1. root:/dzs$/&&!GEM&&DIG #if the root is non-geminate dzs final
     +;;DIG; #bridzs (base allomorph)
     +//ddzs/;INS;=Al; #briddzs(el), +// is a shorthand for +/dzs$/
2. root:/([ds]z|[cz]s|[ltgn]y)$/&&!GEM&&DIG #non-gem. digraph final
     +;;DIG; #fagy (base allomorph)
     +/(.)(.)$/$1$1$2/;INS;=Al; #faggy(al)
```

Description

Allomorph generation rules consist of a conditional row (bold) that checks whether the conditions for the stem allomorphy described by the rule are met and a sequence of allomorph generation instructions that actually generate the allomorphs (these begin with a +).

The conditional row

The conditional row first names the attribute from the value of which the form of allomorphs is generated (this is normally the root attribute) and checks some property (usually the ending) of it using a regular expression. Additional properties may be checked using Boolean operators (&&, || and !). Note that this conditional row does not contain semicolons (;) in contrast to attribute manipulation rows.

The allomorph generation rows

Allomorph generation rows conform to the general format of attribute manipulation instructions. In this case, however, the conditional field begins with a + (this indicates the addition of an allomorph to the list of allomorphs.) The other fields can be used as usual to set requirements and properties. In this context, however, not global (i.e. morpheme level) properties and requirements are set, but local ones: i.e. the requirements and properties set in the allomorph generation rows pertain only to the allomorph generated by the statement. Negative properties can also be given and thus morpheme-level properties can be deleted in individual allomorphs (see the example below).

Example: negative properties

```
root:/hamu$/
+;;;
+/u/v/;vST (POSS|PL|-jÚ);LOW !Vfin Cfin;
```

DOI:10.15774/PPKE.ITK.2015.013

Adding the base allomorph	If the `+` is not followed by slashes (i.e. field 1 consists only of a `+` or the `+` is followed by a Boolean operator), like in the first allomorph generation row of both examples 1. and 2. above, then the form of the allomorph to be added is identical to the value of the attribute named in the conditional row (i.e. that of the root attribute). This is the base ('lexical') allomorph of the morpheme.

Adding other allomorphs	Otherwise the expression that follows the `+` sign must in fact be a regular expression based substitution expression (but no s before the first slash). The form of the allomorph to be added is produced from the value of the attribute named in the conditional row (i.e. that of the root attribute) by applying the changes specified by the substitution expression to it. If the regular expression in the row does not match the value of the specified attribute (i.e. `root`), then no allomorph is added by the row. E.g., in the case of the example below, while the first allomorph generation row applies to both o-final and ö-final stems (generating a base allomorph), the second row applies only to o-final ones, while the last one applies only to ö-final ones.
	`#o-final lengthening Oslo -> Osló`
	`root:/[oö]$/`
	`+;0mrf;;`
	`+/o$/ó/;!0mrf;;`
	`+/ö$/ő/;!0mrf;;`

If the regexp is empty	If the regular expression part of the substitution expression is empty (i.e. two slashes follow the `+`, as in the second allomorph generation row of example 1.), then the regular expression given in the conditional row (`/dzs$/` in the case of example 1.) is used.

Inserting a harmonic vowel	In the Hungarian description, an `!` at the end of the replacement part of a substitution expression in an allomorph generation row marks that the automaton computing vowel harmony must be applied to the form of the allomorph generated by the statement. In the case of the following example, the second allomorph generation row first generates the form *`fürOd!`* from the `root` *`fürd`* by inserting the mid harmonic vowel *`O`*, to which vowel harmony is applied (because of the final `!`), and thus we obtain the correct allomorph form *`füröd`*.
	`#fürd.ik`
	`root:/($C)d$/&&-ik&&1syl`
	`+;; -ik -Ok -Unk -Ás -Ó -AndÓ -AtlAn =At =Asz =AlAk =AnA =AnAk =OtOk`
	`+//$1Od!/;; -hAt -vA =jUk =j =d =gAt =sz =lAk =nA =nAk =tOk =tAm`
	This is not a built-in feature, however, but it is achieved by the following row in an allomorph list manipulation block (`for(@allomfs)`, see below) that affects all allomorphs (`dovhrm()` applies a Kimmo vowel harmony automaton to the string given as its argument):
	`#! at the end of allomorph marks that vowel harmony must be done:`
	`allomf:s/(.*)!$/dovhrm($1)/e;;;`
	(The `/e` switch at the end of the substitution expression indicates that the replacement part of the expression should be evaluated as a perl expression (and call the subroutine `dovhrm`) instead of as a string.)

Flow of control in allomorph generation rules	A single allomorph generation rule behaves like a conditional block concerning control flow, i.e. if the initial conditional is satisfied, then all the allomorph generation rows in the block are evaluated in sequence. A sequence of allomorph generation rules behave like a sequence of `if-elsif` blocks: the first rule the initial conditional of which is satisfied is executed, the rest in the sequence are skipped.

Setting variables in the conditional row

You can add special variable assignments to the end of the conditional row of allomorph generation rules in the form of a comment. This construction can be used when some material from an attribute different from the attribute normally used for generating the form of allomorphs (i.e. from something other than `root`) must be used for generating the form of an allomorph. The material from the other attribute (in the example below: from `phon`) can be extracted using the regular expression grouping operator `()`. The material matched by the first, second etc. group in the last regular expression is always stored in the temporary variables `$1, $2` etc. (in the example below: the final digraph of the value of `phon` feature is grouped and stored: `($C2)`), and it can be permanently stored in a variable by adding a comment containing the variable assignment statement (here: in `$finC2`) and later used in an allomorph generation row. Note that the comment that contains the variable assignment must begin with two `!`'s. If you want to have more than one variable assignments, you must separate them with commas (,) (not semicolons (;)). Normal comments are not allowed in the initial conditional row of allomorph generation blocks.

```
#foreign digraph-final Milosevic -> Miloseviccs(el)
#if the phon is digraph final but the root is not:
!root:/$C2$/&&phon:/($C2)$/#!!$finC2=$1
+;;DIG; # Milosevic
+/$/$finC2/;INS;=Al; # Miloseviccs
```

Preconditions in allomorph generation rows

You can add preconditions to the end of allomorph generation rows (after the third `;` closing the property setting field). The preconditions follow an `if` or an `unless` in parentheses and have the same format as all other conditionals. These are evaluated first, and the rest of the line is only evaluated if the preconditions are all satisfied.

An example of preconditions

```
root:/($V_$C)[eoöau]($C)$/
+;!VZA;;if(stemalt:VZA)
+;;!stmalt;if(stemalt:VZA\+)
```

A.1.3.3 Manipulating properties and requirements of individual allomorphs (allomorph list manipulation blocks)

Allomorph list manipulation blocks

Allomorph list manipulation blocks begin with either `for(@allomfs)`, to manipulate all allomorphs, or `for($allomfs[0])`, to manipulate only the first (normally the base) allomorph and are enclosed in braces { ... }. The opening and closing braces must appear on a line of their own.

Attribute manipulation instructions

Within the block, you can use the attribute manipulation instructions to
a) add left or right properties if certain conditions are met,
b) delete left or right properties if certain conditions are met,
c) add left or right requirements if certain conditions are met.
Like in the case where morpheme-level properties are set (i.e. when you use the attribute manipulation instructions outside of an allomorph manipulation block), the default attribute to check in the condition field (field 1) is `rp`, the default attribute to set in the requirements field (field 2) is `rr`, and the default attribute to set in the properties field (field 3) is `rp`.

Allomorph-local attributes

In contrast to the morpheme-level case, however, the following attributes are assumed to be allomorph-local in this context: `gp, glr, grr, lr, rr, lp, rp` and `allomf`. All the other attributes are global (i.e. they belong to the morpheme as a whole) in this context as well. If, for any reason, you want to refer to the morpheme level variant of the `rr, rp` etc. attributes, or a property within them, precede them with a caret (ˆ).The caret must precede the negation mark if one is present, e.g. `!̂1syl`. See the examples below. You can also use conditional blocks within allomorph list manipulation blocks. In these, the same attributes are assumed to be allomorph-local.

Examples: reference to global properties

```
for($allomfs[0])
{
##mark the first allomorph of all verbs with -ik/-03 and regular
    unless marked otherwise
#add -ik or -03 property marking sg3 ind pres ending
^-ik&&!-ik;;-ik;
if(!^irreg&&!irreg)
{
;;reg; #mark it regular
!^-ik&&!-03;;-03;
...
}}
```

Morpheme-level properties and requirements

The properties and requirements of individual allomorphs specific to the stem allomorphy are set by the allomorph generation rules already. The morpheme-level properties and requirements coming from the lexicon and specified by morpheme level attribute manipulation rules (`gp, glr, grr, lr, rr, lp` and `rp`) can be added to these by using the `&addprops;` statement at the beginning of the for(`@allomfs`) block (see the example below). Another variant of this statement, `&addlexprops;`, adds only properties and requirements specified in the lexicon, but not the ones set by the morpheme-level attribute manipulation rules.

An example

```
#add properties of nominal allomorphs
for(@allomfs)
{
#add morpheme-level properties to each allomorph
&addprops;

#fix the v-final allomf of V->v stems
Cfin;;!Vfin;

#the 0 allomf of -i[IKEP]
humor:IKEP&&allomf:/^$/;;=_t =Vs =AtlAn;

#accusative sfx -t/Vt
if(!=Vt&&!=_t)
{
phon:/(ss|ssz|n$Sb|$V_$Sb|$Sn$Sn|$V_$Sn|$V_)$/&&!LOW;;=_t;
else seg:/($V__$Sb|$V__$Sn)$/&&!LOW;;=_t;
else phon:/$V_$/&&!rr:vST;;=_t;
else ;;=Vt;
}
...
}
```

A.1.3.4 Allomorph duplication blocks

An example

```
###split underspecified allomfs
map(@allomfs)
{
#split vacillating harmonic stems
#VHV:balett -[oe]t, n[ae]k
#VHVF:mágnes -es, n[ae]k
#VHVB:klarinét -ot, n[ae]k
dup(VHV[FB]?)
{
rp:s/VHVB?/VHB/;;;
rp:s/VHVF?/VHFU/;;;
rp:s/VHVB/VHFU/;Cini;; #only consonant-initial front suffixes
rp:s/VHVF/VHB/;Cini;; #only consonant-initial back suffixes
}
#split allomorphs having specifications like =A0lAk
#to one having =lAk and another having =AlAk
dup(rp:/=[^ &]+0/)
{
rp:s/=[^ &]+0/=/g;;;
rp:s/=([^ &]+)0/=$1/g;;;
}
}
```

Description

Allomorph duplication can be done within a `map(@allomfs)` block. Within the block, only comments and `dup` (=allomorph duplication) blocks may appear. The `dup` keyword is followed by a Boolean expression in parentheses. The dup block is executed if this conditional expression is true.

Within the `dup` block, there is a sequence of attribute manipulation instructions, the conditional part of which normally contains a substitution expression which changes the disjunctive property into one of its possible alternatives. Each of the attribute manipulation instructions the conditional of which matches, produces an allomorph with the property changed as specified by the substitution expression, and possibly with the addition or deletion of other properties or requirements.

A.2 A sample analysis trace

The following is the trace of the analysis of the Hungarian word form *tör* ‘breaks sg.’ by the Humor morphological analyzer engine. At each lookup position (marked by `|->|` in the input), the actual state of the word grammar automaton is shown in the lines marked by `State Name:`, while the extended state variables in the lines marked by `Flags:`. Morphs looked up successfully are marked by `OK:`. The output contains their lexical and surface form and tag to be output (`Morpheme:`), their left-hand-side requirement vector and matrix code (`Required:`), their right-hand-side feature vector and matrix code (`Features:`), and their word grammar category (`Categ. Name:`). Morphs looked up that do not match the previous morph have exclamation marks (`!`) in the positions where there is a vector or matrix clash. In addition, this situation is marked by `Error: not matching to previous morpheme or dialect condition`. Another error message, `Error: matching morphemes not found` marks situations where no matching morphemes with an appropriate outgoing category are found in the given state. A successful analysis is marked by `******** Result:`. Backing up in the word grammar automaton is marked by a right brace (}).

```
>tör
-------------------------------------------------------
Processing: "tör"
-------------------------------------------------------
Looking up:   "|->|tör"
State Name:   START
Flags:        00000000 00000000 00000000 00000000
  {
  Morpheme:     "tör=ik[S_IGE]=tör"
  Required:     .0...... ........ ....... ........ 8
  OK:           (first)
  Features:     11011001 11000010 00000000 00000000 #24
  Categ. Name:  vstem_!cmpd
  -------------------------------------------------------
  Looking up:   "tör|->|"
  State Name:   V
  Flags:        00000000 00000000 00000000 00000000
    {
    Morpheme:     "[I_NOM]"
    Required:     10...... ........ ....... ........ #44
    Error:        =!====== ======== ======== ======== = (#24)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[I_e2]"
    Required:     11...11. ........ ....... ........ #28
    Error:        =====!!= ======== ======== ======== = (#24)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[I_e3]"
    Required:     11...... ........ ....... ........ #26
    Error:        ======== ======== ======== ======== ! (#24)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[I_TPe2]"
    Required:     11...... ........ ....... ........ M
    Error:        ======== ======== ======== ======== ! (#24)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[D=HA_ESSMOD=0]"
    Required:     ........ ........ ....... ........ #20
    Error:        ======== ======== ======== ======== ! (#24)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    }
  Morpheme:     "tör=ik[S_IGE]=tör"
  Required:     .0...... ........ ....... ........ 8
  OK:           (first)
  Features:     11011000 00000000 00000000 00000000 V
  Categ. Name:  vstem_!cmpd
  -------------------------------------------------------
  Looking up:   "tör|->|"
  State Name:   V
  Flags:        00000000 00000000 00000000 00000000
    {
    Morpheme:     "[I_NOM]"
    Required:     10...... ........ ........ ........ #44
    Error:        =!====== ======== ======== ======== ! (V)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[I_e2]"
    Required:     11...11. ........ ........ ........ #28
    Error:        =====!!= ======== ======== ======== ! (V)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[I_e3]"
    Required:     11...... ........ ........ ........ #26
    Error:        ======== ======== ======== ======== ! (V)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[I_TPe2]"
    Required:     11...... ........ ........ ........ M
    Error:        ======== ======== ======== ======== ! (V)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[D=HA_ESSMOD=0]"
    Required:     ........ ........ ........ ........ #20
    Error:        ======== ======== ======== ======== ! (V)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    }
  Morpheme:     "tör[S_IGE]"
  Required:     .0...... ........ ........ ........ 8
  OK:           (first)
  Features:     11011001 11000010 00000000 00000000 #5
  Categ. Name:  vstem_!cmpd
  -------------------------------------------------------
  Looking up:   "tör|->|"
  State Name:   V
  Flags:        00000000 00000000 00000000 00000000
    {
    Morpheme:     "[I_NOM]"
    Required:     10...... ........ ........ ........ #44
    Error:        =!====== ======== ======== ======== = (#5)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[I_e2]"
    Required:     11...11. ........ ........ ........ #28
    Error:        =====!!= ======== ======== ======== = (#5)
    Error:        not matching to previous morpheme or dialect condition
    ------------------------------------------------------
    Morpheme:     "[I_e3]"
    Required:     11...... ........ ........ ........ #26
    OK:           ======== ======== ======== ======== = (#5)
    Features:     00100000 00000000 00000000 00000100 #666
    Categ. Name:  inf
******** Result:  "tör[S_IGE]+[I_e3]"
    ------------------------------------------------------
    Looking up:   "tör|->|"
    State Name:   END
    Flags:        00000000 00000000 00000000 00000000
      {
      Morpheme:     "[I_NOM]"
      Required:     10...... ........ ........ ........ #44
      Error:        !======= ======== ======== ======== = (#666)
      Error:        not matching to previous morpheme or dialect condition
      ------------------------------------------------------
      Morpheme:     "[I_e2]"
      Required:     11...11. ........ ........ ........ #28
      Error:        !!===!!= ======== ======== ======== = (#666)
      Error:        not matching to previous morpheme or dialect condition
      ------------------------------------------------------
    Morpheme:     "[I_e3]"
    Required:     11...... ........ ........ ........ #26
    Error:        !!====== ======== ======== ======== = (#666)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[I_TPe2]"
    Required:     11...... ........ ........ ........ M
    Error:        !!====== ======== ======== ======== = (#666)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "[D=HA_ESSMOD=0]"
    Required:     ........ ........ ........ ........ #20
    Error:        ======== ======== ======== ======== ! (#666)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    }
  Morpheme:     "[I_TPe2]"
  Required:     11...... ........ ........ ........ M
  Error:        ======== ======== ======== ======== ! (#5)
  Error:        not matching to previous morpheme or dialect condition
  -------------------------------------------------------
  Morpheme:     "[D=HA_ESSMOD=0]"
  Required:     ........ ........ ........ ........ #20
  Error:        ======== ======== ======== ======== ! (#5)
  Error:        not matching to previous morpheme or dialect condition
  -------------------------------------------------------
  }
Morpheme:     "t._tisztelt[S_MN|ROV]=t+[I_NOM]"
Required:     ........ ........ ........ ........ #0
OK:           (first)
Features:     00000000 00000100 00000000 00000100 #883
Categ. Name:  uninfl+dot
-------------------------------------------------------
Looking up:   "t|->|ör"
State Name:   DOTREQ
Flags:        00000000 00000000 00000000 00000000
  {
  Error:        matching morphemes not found
  -------------------------------------------------------
  }
Morpheme:     "t[S_FN|BETU]"
Required:     ........ ........ ........ ........ #0
OK:           (first)
Features:     10010000 00100100 00010000 00001100 #999
Categ. Name:  qstem_!sup_!cmpd_nosug
-------------------------------------------------------
Looking up:   "t|->|ör"
State Name:   NUM
Flags:        10000000 00000000 00000000 00000000
  {
  Morpheme:     "Ör[S_FN]=ör"
  Required:     ........ ........ ........ ........ #0
  Error:        ======== ======== ======== ======== ! (#999)
  Error:        not matching to previous morpheme or dialect condition
  -------------------------------------------------------
  Morpheme:     "ö[S_FN|BETU]"
  Required:     ........ ........ ........ ........ #0
  Error:        ======== ======== ======== ======== ! (#999)
  Error:        not matching to previous morpheme or dialect condition
  -------------------------------------------------------
  }
Morpheme:     "t_tonna[S_FN|ME|ROV]=t"
Required:     ........ ........ ........ ........ #23
OK:           (first)
Features:     10000001 00100100 00011000 00000100 #999
Categ. Name:  acron_!cmpd
-------------------------------------------------------
```

```
  Looking up:   "t|->|ör"
  State Name:   N2u
  Flags:        10000000 00000000 00000000 00000000
    {
    Morpheme:     "Ör[S_FN]=ör"
    Required:     ........ ........ ........ ........ #0
    Error:        ======== ======== ======== ======== ! (#999)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "ö[S_FN|BETU]"
    Required:     ........ ........ ........ ........ #0
    Error:        ======== ======== ======== ======== ! (#999)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    }
  Morpheme:     "t[D=MN_MIB]"
  Required:     11.0.... ........ ........ ........ H
  OK:           (first)
  Features:     10100110 11111111 00000000 00000000 #524
  Error:        transition not found
  -------------------------------------------------------
  Morpheme:     "t[D=MN_MIB]"
  Required:     11.10... ........ ........ ........ H
  OK:           (first)
  Features:     10110110 11111111 00000000 00000000 #524
  Error:        transition not found
  -------------------------------------------------------
  Morpheme:     "t[D=MN_MIB]"
  Required:     11.11... ........ ........ ........ H
  OK:           (first)
  Features:     10111110 11111111 00000000 00000000 #524
  Error:        transition not found
  -------------------------------------------------------
  Morpheme:     "t=a[D=FN_IF=tA]=t"
  Required:     11.0.... ......1. ........ ........ #44
  OK:           (first)
  Features:     10100101 11001111 00000000 00000000 #963
  Error:        transition not found
  -------------------------------------------------------
  Morpheme:     "t=e[D=FN_IF=tA]=t"
  Required:     11.1.... ......1. ........ ........ #44
  OK:           (first)
  Features:     10110111 11001111 00000000 00000000 #963
  Error:        transition not found
  -------------------------------------------------------
  Morpheme:     "T_trícium[S_FN|ROV|VEGY]=t"
  Required:     ........ ........ ........ ........ #0
  OK:           (first)
  Features:     10000101 11001100 10011000 00000100 #999
  Categ. Name:  acron_!cmpd
  -------------------------------------------------------
  Looking up:   "t|->|ör"
  State Name:   N2u
  Flags:        10000000 00000000 00000000 00000000
    {
    Morpheme:     "Ör[S_FN]=ör"
    Required:     ........ ........ ........ ........ #0
    Error:        ======== ======== ======== ======== ! (#999)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    Morpheme:     "ö[S_FN|BETU]"
    Required:     ........ ........ ........ ........ #0
    Error:        ======== ======== ======== ======== ! (#999)
    Error:        not matching to previous morpheme or dialect condition
    -------------------------------------------------------
    }
  }
-------------------------------------------------------
End of processing
-------------------------------------------------------
Analysis of "tör":
        tör[S_IGE]+[I_e3]
```