A New Form of Humor
Mapping Constraint-Based Computational Morphologies to a Finite-State Representation

Attila Novák

MTA-PPKE Hungarian Language Technology Research Group and
Pázmány Péter Catholic University, Faculty of Information Technology
50/a Práter street, 1083 Budapest, Hungary
novak.attila@itk.ppke.hu

Abstract

MorphoLogic's Humor morphological analyzer engine has been used for the development of several high-quality computational morphologies, among them ones for complex agglutinative languages. However, Humor's closed source licensing scheme has been an obstacle to making these resources widely available. Moreover, there are other limitations of the rule-based Humor engine: lack of support for morphological guessing and for the integration of frequency information or other weighting of the models. These problems were solved by converting the databases to a finite-state representation that allows for morphological guessing and the addition of weights. Moreover, it has open-source implementations.

Keywords: morphological analysis, finite-state morphology, morphological guessing

1. Introduction

MorphoLogic's Humor ('High speed Unification MORphology') morphological analyzer engine (Prószéky and Kis, 1999) has been used for the development of several high quality computational morphologies. These include ones for agglutinating languages, such as Hungarian and other members of the Uralic language family: Komi, Udmurt, Lowland Mari, Northern Mansi and two Khanty dialects. Furthermore, it has been used to create morphologies for several Indo-European languages including Polish, English, German, French and Spanish. Most of these resources were created using a morphological description development environment that can generate the resources used by the analyzer engine from a feature-based human-readable description that contains no redundant information and is thus easy to maintain (Novák, 2008).

However, 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.

The structure of this paper is as follows. First we introduce the Humor morphological analyzer and the morphological grammar formalism that we used to create and maintain Humor morphological databases. This is followed by a brief description of the Xerox finite-state tools and formalism, which we used as a target representation when converting our morphologies. Then we describe how various components of Humor morphological databases were converted to a finite-state representation. We show that significant analysis speed benefits can be obtained by converting the original Humor databases to finite-state transducers, although this also results in a significant increase of run-time (and compile-time) memory requirements. The paper is concluded by a short section on the implementation of morphological guessing and a final section on further problems to solve.

2. The Humor morphological analyzer

The Humor analyzer performs a classical 'item-and-arrangement' (IA)-style analysis, where the input word is analyzed as a sequence of morphs. Each morph is a specific realization (an allomorph) of a morpheme.

The word is segmented 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.

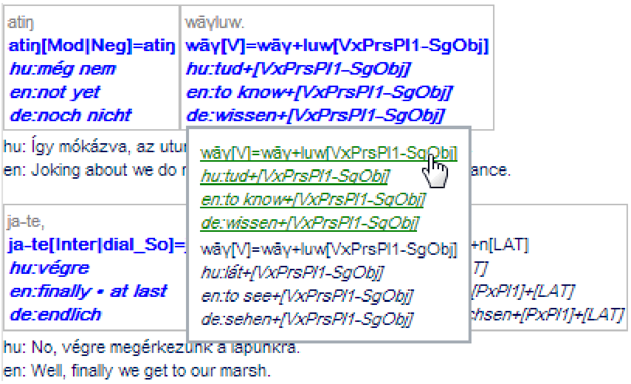
The following analyses of the Hungarian word form *Várnának* contain two morphs each, a stem and an inflectional suffix, delimited by a plus sign.

```
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 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 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*.

The same formalism was used to distinguish all homonymous/polysemic stems in the morphological databases created for the other Uralic languages. These tags could be used to assign glosses to each stem unambiguously. Thus the output of the morphology could be used directly to create word sense disambiguated corpora with an interlinear annotation in a semi-automatic manner. The automatic annotation of the texts was checked and corrected manually. A snapshot of the web-based manual annotation tool can be seen in Figure 1, with the fragment of a Northern Mansi text with trilingual glossing.



atiŋ
atiŋ[Mod|Neg]=atiŋ
hu:még nem
en:not yet
de:noch nicht

wāyluw.
wāγ[V]=wāγ+luw[VxPrsPl1-SgObj]
hu:tud+[VxPrsPl1-SgObj]
en:to know+[VxPrsPl1-SgObj]
de:wissen+[VxPrsPl1-SgObj]

hu: Így mókázva, az utu
en: Joking about we do r
ance.

wāγ[V]=wāγ+luw[VxPrsPl1-SgObj]
hu:tud+[VxPrsPl1-SgObj]
en:to know+[VxPrsPl1-SgObj]
de:wissen+[VxPrsPl1-SgObj]
wāγ[V]=wāγ+luw[VxPrsPl1-SgObj]
hu:lát+[VxPrsPl1-SgObj]
en:to see+[VxPrsPl1-SgObj]
de:sehen+[VxPrsPl1-SgObj]

ja-te,
ja-te[Inter|dial_So]=
hu:végre
en:finally • at last
de:endlich

+n[LAT]
T]
[PxPl1]+[LAT]
hsen+[PxPl1]+[LAT]

hu: No, végre megérkezünk a lapunkra.
en: Well, finally we get to our marsh.

Figure 1: Snapshot of a manual disambiguation tool used for morphosyntactic and lexical semantic disambiguation. The fragment is in Northern Mansi with Hungarian–English–German glossing and Hungarian and English translations.

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 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 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. A sample of Humor representation of morphs can be seen in Figure 2 below.

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.

The word grammar automaton used to check overall word structure may have extra binary state variables in addition to its main state variable, 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 superlative prefix without the comparative suffix licensing it later within the word is ill-formed[1]. 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. This fragment of the Hungarian word grammar shows the non-final state N2 of the automaton

[1] There are a few stems that license the superlative prefix themselves, such as *utolsó* 'last': *leg+utolsó* 'the very last' is a well-formed Hungarian word.

```
kutya,#599,1000000100100100,#48,.0..............,kutya,S_N
kutyá,#916,1000000100100100,#48,.0..............,kutya,S_N

t,    #738,0010000000000000,#34,10......0.......,t,  I_ACC
at,   #738,0010000000000000,#37,10.0..1.1.......,at, I_ACC
et,   #738,0010000000000000,#37,10.1..1.1.......,et, I_ACC
ot,   #738,0010000000000000,#37,10.0..0.1.......,ot, I_ACC
öt,   #738,0010000000000000,#37,10.11.0.1.......,öt, I_ACC
```

Figure 2: Humor representation of the allomorphs of the Hungarian stem morpheme *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

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. 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. When encountering a comparative ndercmp[2] or a verbalizer suffix vder, the respective flag (sup or vpfx) is cleared.

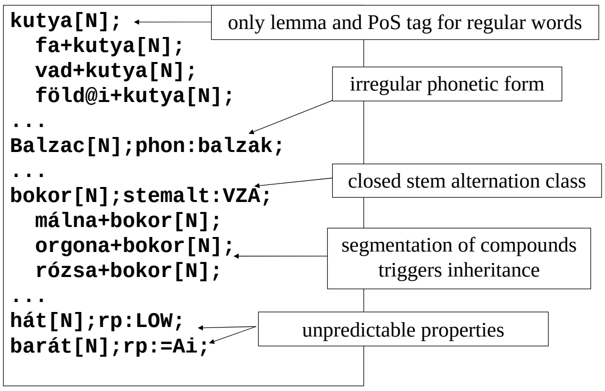
```
#right compound member encountered
N2:
inf ->      END ?{0.0}  #?{!sup !vpfx}
119sfx ->   ADJ ={1...} #={lcase}
nder-119 -> N2  ={1...} #={lcase}
ndercmp ->  ADJ ={10..} #={lcase !sup}
nder2_adj ->ADJ
nder2 ->    N2
vder ->     V   ={1..0} #={lcase !vpfx}
```

Figure 3: Fragment of the Hungarian word grammar automaton – non-final state N2.

In addition, the morphological database of the Humor analyzer contains a mapping 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. 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 vector of the currently looked up morph is mapped. At the end of the word, the automaton must be in final state for the current analysis to be acceptable.

The database would be 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, 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.

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 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 PoS tag for regular words
irregular phonetic form
closed stem alternation class
segmentation of compounds triggers inheritance
unpredictable properties

Figure 4: Entries in the high-level stem database.

Figure 5 below 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 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 there is no lengthening). Then it generates an allomorph that is identical to the lemma and another one with the final vowel lengthened. The 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.

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 de-

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

```
#final vowel lengthening:
#kutya -> kutyá, eke ->eké
  root:/[ae]\+⋆$/&&phon:/[ae]\+⋆$/
  +;!FVL;;
  +/a(?=\+⋆$)/á/;FVL;;
  +/e(?=\+⋆$)/é/;FVL;;
```

Figure 5: Fragment of the Hungarian rule grammar: a rule generating allomorphs of Hungarian final vowel lengthening stems.

scription. 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.

In the following sections, we describe our approach to open up new possibilities for using the linguistic resources created in the Humor formalism by converting them to a finite-state representation.

3. Finite-state morphologies

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 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.

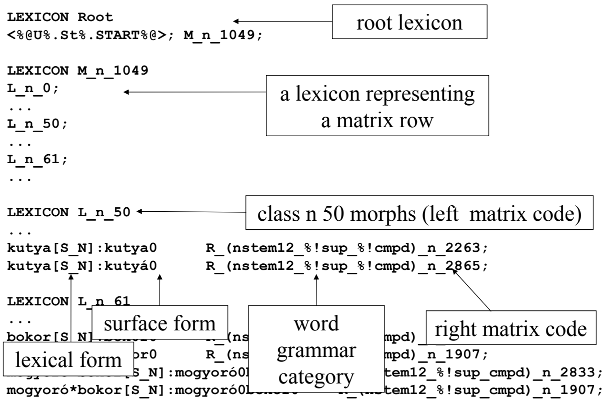
However, the Xerox tools, although made freely available for academic and research use in 2003 with the publication of (Beesley and Karttunen, 2003), do not differ from Humor in two significant respects: a) they are closed-source and b) cannot handle weighted models. Luckily, a few years later quite a few open-source alternatives to xfst were developed. One of these open-source tools, Foma (Hulden, 2009), can be used to compile and use morphologies written using the lexc/xfst 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.

4. The Humor-to-lexc conversion

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 matrix name and code along with its word grammar category. Figure 6 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, the word *kutya* 'dog' has a different left and right continuation class code than in the Humor lexicon fragment shown in Figure 2. 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.



```
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]              R_(ns               mpd)_
              or0       R_(ns               mpd)_n_1907;
        S_N]:mogyoró0                      tem12_%!sup_cmpd)_n_2833;
mogyoró*bokor[S_N]:mogyoró0                tem12_%!sup_cmpd)_n_1907;
```

Figure 6: Fragment of the lexc representation of converted Humor data structures: a row of a continuation matrix and stem allomorphs

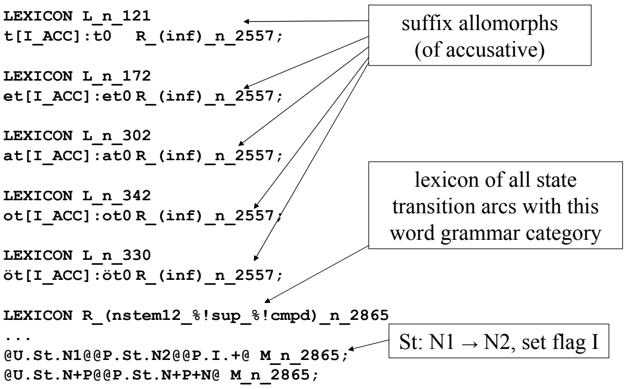
The easiest way to map the Humor word grammar to a finite-state representation is using flag diacritics. The main state variable of the automaton is mapped to one flag (called St), while the extended binary state variables to one additional generated flag each. The exact set of flag diacritics arcs attached to the representation of each morph is determined by the word grammar category of the morph. The sublexicon fragment at the bottom of Figure 7 illustrates how this is implemented. Figure 7 also shows the converted representation of the allomorphs of the Hungarian accusative suffix.

Elimination of the word grammar state flag St is possible to improve the speed of lookup on the transducer. However, it may result in a considerable growth of the state space.

Table 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
runtime mem	3.3 MB	20.6 MB	38.5 MB
lookup speed	4700 w/s	12500 w/s	33333 w/s

Table 1: Comparison of the original Humor and xfst-compiled equivalents of a 144000-morph Hungarian lexicon



```
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 7: 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*

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

5. Implementing a morphological guesser

The original Humor analyzer engine is not capable of analyzing word forms the stem of which is neither included in the stem lexicon nor is covered by the compounding and derivational model implemented in the morphology. However, the grammar defined by the rules is capable of generating a lexicon containing the proper allomorphs and properties of a set of properly formulated pseudo-stems. This can be transformed to a guesser lexicon by substituting a regular expression at the beginning of the pseudo-stems and combining it with the regular affix inventory. The usual strategy of doing lookup with the regular lexical transducer first then defaulting to the guesser transducer can then be used to return possible analyses for all word forms in the input.

6. Problems

A problem that currently impedes achieving all the goals set forth in the introduction is that although xfst correctly compiles all lexc lexicons generated from the Humor lexicon sources, and they also work correctly using the Xerox lookup utility, the open-source Foma tool fails to compile many of them. In other cases, even if the Foma lexc compilation itself succeeds, further finite-state operations fail. This seems not to be due to bugs in the morphology conversion but rather to bugs and limitations in Foma. Errors range from segmentation faults to messages like *Stack full*. Nevertheless, we hope that these problems will not be too difficult to overcome.

Another problem is that even when conversion of a lexicon is successful and we also succeed in extracting the lower language of the transducer that could be used e.g. for correction candidate generation in a spelling correction task, the presence of flag diacritics seems to be a problem for the Foma med (minimum error distance) algorithm, which can normally be used to generate correction candidates. The problem is that the med algorithm considers flag diacritics arcs regular arcs and does not handle them properly. On the other hand, an attempt to eliminate all relevant flags results in unfeasibly big transducers or automata.

7. Summary

In this paper, we described an approach to open up new possibilities for using the linguistic resources created in the Humor morphological formalism by converting them to a finite-state representation. The morphological databases were converted to a representation that can be compiled and used by finite-state morphological tools, among them ones with open-source implementation. The finite-state representation can be used for suffix-based morphological guessing, and it provides a natural means for introducing frequency data, also making composition with weighted error models possible. Although this latter goal could not yet be achieved due to problems with the finite-state compiler tool we used, we are confident that these problems can be overcome.

Acknowledgment

This work was partially supported by the grants TÁMOP-4.2.1./B-11/2/KMR-2011-002 and TÁMOP-4.2.2./B-10/1-2010-0014.

8. References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In Jan Holub and Jan Zdárek, 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.

Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications.

Mans Hulden. 2009. Foma: a finite-state compiler and library. In Alex Lascarides, Claire Gardent, and Joakim Nivre, editors, *Proceedings of EACL 2009*, pages 29–32. The Association for Computer Linguistics.

Krister Lindén, Miikka Silfverberg, Erik Axelson, Sam Hardwick, and Tommi Pirinen, 2011. *HFST—Framework for Compiling and Applying Morphologies*, volume Vol. 100 of *Communications in Computer and Information Science*, pages 67–85.

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*, pages 27–32.

Gábor Prószéky and Balázs Kis. 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*, pages 261–268. Association for Computational Linguistics.