A Unification-based Approach to Morpho-syntactic Parsing of Agglutinative and Other (Highly) Inflectional Languages

Gábor Prószéky
proszeky@morphologic.hu

Balázs Kis
kis@morphologic.hu

MorphoLogic
Késmárki u. 8.
Budapest, Hungary, H-1118
http://www.morphologic.hu

Abstract

This paper introduces a new approach to morpho-syntactic analysis through Humor 99 (High-speed Unification Morphology), a reversible and unification-based morphological analyzer which has already been integrated with a variety of industrial applications. Humor 99 successfully copes with problems of agglutinative (e.g. Hungarian, Turkish, Estonian) and other (highly) inflectional languages (e.g. Polish, Czech, German) very effectively. The authors conclude the paper by arguing that the approach used in Humor 99 is general enough to be well suitable for a wide range of languages, and can serve as basis for higher-level linguistic operations such as shallow parsing.

Introduction

There are several linguistic phenomena that are possible to process by means of morphological tools for agglutinative and other highly inflectional languages, while processing the same features requires syntactic parsers in case of other languages such as English. This paper provides a brief description of Humor 99 first presenting a general theoretical background of the system. This is followed by examples of the most recent applications (in addition to those listed earlier) where the authors argue that the approach used in Humor 99 is general enough to be well suitable for a wide range of languages, and can serve as basis for higher-level linguistic operations such as shallow or even full parsing.

1 Affix arrays rather than affixes

Segmentation of a word-form in Humor 99 is based on surface patterns, that is, typical sequences of separate suffix morphemes are analyzed as a whole. For example, the English nominal ending string *ers'* (NtoV+PL+POSS) is a complex affix handled as an atomic string in Humor 99[1]. The string *ers'* is generated from *er+s+'s* in an earlier development phase by a dedicated utility. The generator is able to make a finite set of affix sequences from an (even recursive) description[2]. Running this utility can be considered the learning phase of the algorithm. The resulting suffix combinations are stored in a compressed internal lexicon structure that guarantees very fast searching.[3] The entire algorithm shows features similar to the hypothesis according to which most segments of word-forms in agglutinative lan-

[1] We use mainly English examples in spite of the fact that English morphology is simpler than the morphologies of agglutinative and highly inflectional languages.

[2] Depth of the recursive process can be given as a parameter. The method is similar to the one of Goldberg & Kálmán (1992) used in the BUG system: the description is theoretically infinite, but there is a finite performance limit when running.

[3] The idea has something in common with the PC-Kimmo based analyzer of the University of Pennsylvania (Karp et al. 1992). Our compression ratio is around 20%.

guages are handled as "Gestalts" by native speakers, instead of parsing them on-line.[4]

This idea is not new in the literature: according to Bybee, "a psycholinguistic argument for treating (some) ending sequences as wholes comes from the observation that children acquiring inflectional languages seldom make errors involving the order of morphemes in a word." (Bybee 1985) Another source is Karlsson: "The endings and entries are often listed as wholes, especially in close-knit combinations.[5] Such combinations are often subject to bi-directional dependencies that are hard to capture otherwise" (Karlsson 1986).

2 Allomorphs rather than base forms

Karlsson (1986) shows several ways in which lexical forms of words may be constructed: full listing, minimal listing, methods with unique lexical forms and methods with phonologically distinct stem variants. Full listing does not need rules at all, but it is implausible for agglutinative languages. Minimal listings need a quite large rule system in case of highly inflectional languages, although their lexicons are relatively small. In methods based on unique lexical forms allowing diacritics and morpho-phonemes (Koskenniemi 1983, Abondolo 1988) paradigms are represented by a single base form[6]. Our approach is close to the minimal listing methods, but less rules are needed. Finally, the representation presented here regards *phonologically distinct bound variants* of a base form as separate *stems*.[7] There are two known important variants of this method: one using technical stems — that is, strings that linguists do not consider stem variants — and another using real allomorphs. The former was applied in the TEXFIN system of Karttunen (1981), the latter was used by Karlsson (1986). This is the method we have chosen for the Humor 99 system.

Humor 99 lexicons contain stem allomorphs (generated by the learning phase mentioned above) instead of single stems. Relations among allomorphs of the same base form (e.g. *wolf, wolv*) are, however, important for syntax, semantics, and the end-user. An online morphological parser needs not be directly concerned with the derivation of allomorphs from their base forms, for example, it does not matter how *happi* is derived from *happy* before *-ly*. This phenomenon – a consequence of the orthographical system – is handled by the off-line linguistic process of Humor 99, which makes the analysis much faster. This method is close to the lexicon compilation used in finite-state models.

3 Paradigm groups and paradigms

Concatenation of stem allomorphs and suffix allomorphs is licensed with the help of the following two factors: *continuation classes*[8] defined by paradigm descriptions, and classes of surface allomorphs. The latter is a cross-classification of the paradigms according to phonological and graphemic properties of the surface forms. Both verbal and nominal stem allomorphs can be characterized by sets of suffix allomorphs that can follow them. When describing the behavior of stems, all suffix combinations beginning with the same morpheme are considered equivalent because the only relevant pieces of information come from the suffix that immediately follows the stem. E.g. from the point of view of the preceding stem (*humid*) morpheme combinations

4 Psycholinguists are interested in testing this hypothesis with native speakers (Pléh, pers. comm.)

5 A good example is the linguistic tradition handling number and person combinations of Hungarian definite conjugation.

6 That is why it is very difficult to add new entries to the lexicons automatically in real NLP environments.

7 Actual two-level (and some other) descriptions apply similar methods in order to cope with morphotactic problems that cannot be treated phonologically in an elegant way.

8 Similar to the two-level descriptions' continuation classes (Koskenniemi 1983).

Example 1

Word-form	Humor's real-time segmentation	Humor's output segmentation
humidity	*humid + ity*	*humid + ity*
humidity's	*humid + ity's*	*humid + ity + 's*
humidities	*humid + ities*	*humid + iti + es*
humidities'	*humid + ities'*	*humid + iti + es'*

Example 2

			Stems[10]				
	Features= +/- Values		Cat=Nom				
			Subcat=N		Subcat=Adj		Subcat=Adv
		Morpheme	*fish*	*house*	*green*	*happy*	*far*
Affixes	Nbr=Pl	*s*	-	+	-	-	-
	Deriv=Adv	*ly*	-	-	-	+	-
	Deriv=Abstr	*ness*	-	-	+	+	-
	Deg=Comp	*er*	-	-	+	+	+
	Deg=Super	*est*	-	-	+	+	+

like *ity*+SG, *ity*+PL, *ity*+SG+GEN, *ity*+PL+GEN behave as *ity* itself (Example 1). Therefore, every affix array is represented by its starting affix[9]. Each equivalence class and each paradigm is given an abstract name, that is, each existing set of equivalence classes can have its own abstract name. Example 2 shows a simplified default paradigm of adjectives. For instance, the stem *green* belongs to the paradigm that can be described by the set {Deriv=Abstr, Deg=Comp, Deg=Super}, *er* is a suffix belonging to {Deg=Comp}, thus the word-form *greener* is morphotactically licensed by the unifiability of the two structures: the feature 'Deg' occurs in both with the same value. It is possible to construct a net – a partial ordering of paradigm sets – according to the degree and sort of defectivity. The subsumption hierarchy is useful in agglutinative languages where allomorph paradigms of various stem classes might behave the same way although they have been derived by different morphonological processes.

The scheme shown in Example 2 would better suit languages like Hungarian, but here we try to demonstrate constructing morphological classes without naming them. The (partial) paradigm net based on Example 2 can be the following:

$CLASS_{happy} > CLASS_{green} > CLASS_{far} > > CLASS_{fish}$

$CLASS_{house} > CLASS_{fish}$

This classsification might be used by traditional linguists for creating definitions (or rather naming conventions) of morpheme classes that are more precise than usual.

4 Unifiability without unification

Features used for checking appropriate properties of stems and suffixes are relevant attributes of morpho-graphemic behavior. Checking 'appropriateness' is based on unification, or, strictly speaking, checking unifiability of the adequate features of stems and suffixes. A phonologically and ortographically motivated allomorph-based variant of Example 3 is shown by Example 4.

---

9 There is an equivalence relation on the set of affix arrays.

10 **Nom** means nominal, **N, Adj** and **Adv** as usual. Some remarks to the sample words: *greens* does exist, but as a lexical noun. Some affixed forms, like *happily, happier, happiest, farther, farthest,* are influenced also by phonological and/or orthographical processes.

Example 3

	Features= +/- Values		Stem Allomorphs						
			Cat=Nom						
			Subcat=N		Subcat=Adj			Subcat=Adv	
		Allomorph	*fish*	*house*	*green*	*happy*	*happi*	*far*	*farth*
Affixes	**Lex=Base**	-	+	+	+	+	-	+	-
	Nbr=Pl	*s*	-	+	-	-	-	-	-
	Deriv=Adv	*ly*	-	-	-	-	+	-	-
	Deriv=Abstr	*ness*	-	-	+	+	-	-	-
	Deg=Comp	*er*	-	-	+	-	+	-	+
	Deg=Super	*est*	-	-	+	-	+	-	+

Features (morpho-phonological properties) are used to characterize both stem and suffix allomorphs. A list of **Feature=Value** pairs shows the morphological structure of the morphemes *green* and *er*:

green:
[Cat=Nom, Lex=Base, Subcat=Adj, Deriv =Abstr, Deg={Comp, Super}]

er:[Cat=Nom, Subcat={Adj,Adv}, Deg=C omp]They are unifiable, thus the word-form *greener* is also morpho-phonologically licensed[11]:

INPUT: *greener*

OUTPUT: *green*[A] + *er*[CMP]

The most important advantage of this feature-based method is that possible paradigms and morpho-phonological types need not be defined previously, only the classification criteria have to be clarified. Since the number of these criteria is around a few dozens (in case of a language with rather complicated morphology), the number of theoretically possible paradigm classes is several millions or more. According to our practice linguists choose about 10-20 orthogonal properties which produce 2^{10}-2^{20} possible classes, but, in fact, most of these hypothetical classes are empty in the language chosen.

The implemented morphological analyzer provides the user with more detailed category information (lexical, morpho-syntactic, semantic, etc.) according to the case illustrated by Example 4 (see next page).

Allomorphs *happy* and *ly* cannot be unified because of contradicting values of **Allom**, but *happi* and *ly* can. If the unifiability check is successful, the base form is reconstructed (according to the **Base** information: *happi* ⇨ *happy*) and the output information (that is, **Category** code in our case) is returned:

INPUT: happyly

OUTPUT: *happyly

INPUT: happily

OUTPUT: happy[A]=happi+ly[A2ADV]

As we have seen, lexical information has a central role in Humor, because only a single rule – unifiability-checking – is to be applied.

5 Controlling morpheme sequence recognition

Humor 99 is capable of much more than sketched above. For instance, there can be more than one concatenation points in a single word form. Therefore effective analysis requires an elegant

[11] Unifiability in Humor 99 is defined as follows:
An f feature of the D description can have either a *single value* or a *set of values*.
An f feature of the D description has *compatible values* in the E description iff one of the values of f can be found among the values of f in the E description.
D and E are *unifiable* iff every f feature of the E description has compatible values in the D description.

Example 4

Allomorph	Feature=Value	Base	Category
happy	Cat=Nom	0	[ADJ]
	Subcat=Adj		
	Deriv=Abstr		
	Allom=y		
	Lex=Base		
happi	Cat=Nom	__i->__y	[ADJ]
	Subcat=Adj		
	Deriv=Adv		
	Deg=Comp Deg=Super		
	Allom=i		
	Lex=NonBase		
ly	Cat=Nom	0	[ADV]
	Subcat=Adj		
	Deriv=Adv		
	Allom=i		
	Lex=NonBase		

way of handling compounding and adequate handling of derivational affixes.

Recent implementations of Humor 99 define the set of possible morpheme sequences by means of the so-called *meta-dictionary* (in fact, it's a finite-state automaton). This structure transforms Humor 99 into a representation where three independent types of conditions can be set (on different levels) to control which morphemes (and in what way) may be following each other. All of them were mentioned earlier; the list below is only a summary:

1. Morpheme sequence recognition is achieved through the meta-dictionary.

2. A continuation class matrix provides concatenation licensing based on paradigm descriptions.

3. A feature structure controls concatenation licensing based on surface allomorph classification by means of unifiability checking.

Earlier implementations of Humor used the following hard-coded scheme to control morpheme order where all parts except STEM1 were optional (Example 5).

Example 5

(PREFIX)	STEM1	(STEM2)	(DERIV. AFF.)	(INFL. AFF.)

Example 6 shows how a meta-dictionary can be drawn up to handle the above structure.[12]

Example 6

[% indicates the starting state; $ indicates ending (or accepting) states]

```
START:%
      PREFIX       -> STEM_REQUIRED
      STEM1        -> STEM1_PASSED
STEM_REQUIRED:
      STEM1        -> STEM1_PASSED
STEM1_PASSED:$
      STEM2        -> AFFIXES_POSSIBLE
      DERIV_AFF    -> INFL_AFF_POSSIBLE
      INFL_AFF     -> END
AFFIXES_POSSIBLE:$
      DERIV_AFF    -> INFL_AFF_POSSIBLE
      INFL_AFF     -> END
INFL_AFF_POSSIBLE:$
      INFL_AFF     -> END
END:$
```

Here is an example how Humor's analyzer reacts to a typical construction of an agglutinative language (Hungarian): *elszámítógépezgethettem.* ("I could use a computer to make fun for a while"):

INPUT:
elszámítógépezgethettem

INTERNAL SEGMENTATION:
el[PREFIX]+számító[STEM1]+gép[STEM2]+
+ezgethet[DERIV.AFF.]+tem[INFL.AFF]

OUTPUT:
el[VPREF]+számító[ADJ]+gép[N]+ez[N2V]+
+get[FREQ]+het[OPT]+tem[PAST-SG-1]

6 Comparison with other methods

There are only a few general, reversible morphological systems that are suitable for more than a single language. In addition to the well-known two-level morphology (Koskenniemi 1983) and its modifications (Karttunen 1993) it is worth mentioning the Nabu system (Slocum 1988). There are some morphological description systems showing some features in common with Humor 99 – like paradigmatic morphology (Calder 1989), or the Paradigm Description Language (Anick & Artemieff 1992) – but they don't have large-scale implementations. Two-level morphology is a reversible, orthography-based system that has several advantages from a linguist's point of view. Namely, the morpho-phonemic/graphemic rules can be formalized in a general and very elegant way. It also has computational advantages, but the lexicons must contain entries with extra symbols and other sophisticated elements in order to produce the necessary surface forms. Non-linguist users need an easy-to-extend dictionary into which words can be inserted (almost) automatically. The lexical basis of Humor 99 contains surface characters only – no transformations are applied –, while the meta-dictionary mechanism retains many advantages of the two-level systems. It means in the practice that users can add entries to the running system without re-compiling it.

The compilation time of a Humor 99 dictionary is usually 1–2 minutes (for 100,000 basic entries) on an average PC, which is another advantage (at least, for the linguist) when comparing it with other two-level systems. The result of the compilation is a compressed structure that can be used by any Humor 99 applications. The compression ratio is less than 20% in terms of lexicon size compared to the source material. The size of the dictionary has very little affect on the speed of the run-time system because the tree-based searching algorithm is enhanced with a special paging mechanism developed exclusively for this purpose.

7 Recent applications of the Humor 99 system

There are several applications of Humor 99 – most of them are fully implemented, some others are still in a planning phase. For the time being, our research focuses on two applications, both serving one larger goal: the improvement of translation support of morphologically complex languages. This paper does not cover industrial applications such as spelling checkers, hyphenators, thesauri etc., since these modules have

[12] The meta-dictionary shown in the example compiles with Humor's lexicon compiler without any changes.

been on the market for several years. The following sections briefly describe (1) linguistic stemming for searching purposes, (2) an enhancement to the Humor 99 morphological analyzer that can act as a shallow or full parser in translation support systems.

Linguistic stemming may be considered as a normalizer function which 'normalizes' word forms into canonic lexical forms, thus enabling searching systems to find any form of a specific word in an information base regardless of the word form entered in the search expression. In languages where a single lexical item can take thousands of possible forms, it is essential to have this normalization in electronic dictionaries used for translation support. However, it is these languages where linguistic stemming is impossible without morphological analysis – otherwise several billions of word forms would have to be included in a single database. Thus stemming is a combination of the morphological analysis and a post-processing phase where the actual stems (lexical forms) are extracted from the analysis results. Both the analysis and the extraction phase have to be very precise, otherwise false stems may be returned, and, in case of an electronic dictionary, wrong articles may be retrieved. In languages where words consist of several parts (i.e. productive compounding and/or sequences of derivative suffixes are possible), there might be a lot of possible stems of a single word form – the degree of disambiguity within a single word form can be much higher than in languages having less complex morphologies.

Extraction is based on the results of morphological analysis where the original word form is segmented into morphemes, with each morpheme having a category label and a lexical form. From the segmented results, this phase selects morphemes with stem categories (adjective, noun, verb etc.). Example 7 shows a typical stemming problem where the computer is not entitled to choose between the different possible stems. In these cases, all stems must be returned. Choice is a task of either the end-user or a disambiguator module that is based on the context of the word.

Example 7

There are two possible segmentations of the Hungarian word '*szemetek*':

szemetek = *szem*[N] + *etek*[Poss-P3]
in English: '*your eyes*' ('*you*' in plural)

szemetek = *szemét*[N]=*szemet* + *ek*[Pl]
in English: '*pieces of rubbish*'

The two possible stems are: '*szem*' (eye) and '*szemét*' (rubbish).

8 An enhancement: shallow and full parsing with HumorESK

HumorESK (Humor Enhanced with Syntactic Knowledge) is a twofold application of Humor 99 that is used for shallow and full parsing.[13] The first point of using the morphological analyzer in the parser is to get as much linguistic information about a single word form as possible. The second point is using the basic principles of the morphological analyzer to implement the parser itself. This means that we either collect or generate phrase patterns on different linguistic levels (noun phrases, prepositional phrases, verbal phrases etc.), and compile a Humor-like lexicon of them. On a specific linguistic level each atomic element of a pattern actually corresponds to a (more) complex structure on a lower linguistic level. Example 8 shows how a noun phrase pattern can be constructed from the result of the morphological analysis.

Example 8

Surface string:
the big bad wolves
Morphological analysis:
the[Det] big[Adj] bad[Adj]
wolf[N]=wolve+s[PL]
Noun phrase pattern:
[Det][Adj][Adj][N][PL]

[13] In our environment, shallow parsing of noun phrases – noun phrase extraction – is already implemented.

The example is quite simplified, and does not show an important aspect of the parser, namely, it retains the unification-based approach introduced in the morphological analyzer. This means that all atomic elements in a phrase pattern have three feature structures; two for the concatenation of two adjacent symbols, and one that describes the global ('phrase-wide') behavior of the symbol in question. After recognizing a phrase pattern (where recognition includes surface order licensing based on unifiability checking), another licensing step is performed, based on the global features of each phrase element. This step (1) may reflect the internal hierarchy of symbols within the phrase, (2) sometimes includes actual unification of feature structures. Thus a single higher-level symbol can be generated from the phrase pattern that inherits features from the lower levels. The parser is still in development, although there is an implementation that is being tested together with the dictionary system.

References

Abondolo, D. M. *Hungarian Inflectional Morphology*. Akadémiai, Budapest. (1988)

Anick, Peter & Susan Artemieff A High-level Morphological Description Language Exploiting Inflectional Paradigms. *Proceedings of COLING-92*, Nantes: 67–73. (1992)

Beesley, K. R. Constraining Separated Morphotactic Dependencies In Finite State Grammars. *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing:* 41–49 (1998)

Bybee, J. L. *Morphology. A Study of the Relation between Meaning and Form.* Benjamins, Amsterdam. (1985)

Calder, J. Paradigmatic Morphology. *Proceedings of 4th Conference of EACL 89*: 58-65 (1989)

Carter, D. Rapid Development of Morphological Descriptions for Full Language Processing Systems. *Proceedings of EACL 95:* 202–209 (1995)

Goldberg, J. & Kálmán, L. The First BUG Report. *Proceedings of COLING-92*: 945-949 (1992)

Jäppinen, H. and Ylilammi, M. Associative Model of Morphological Analysis: An Empirical Inquiry. *Computational Linguistics* 12(4): 257-252 (1986)

Karlsson, F. A Paradigm-based Morphological Analyzer. *Papers from the Fifth Scandinavian Conference of Computational Linguistics,* Helsinki: 95-112 (1986)

Karp, D. & Schabes, Y. A Wide Coverage Public Domain Morphological Analyzer for English. *Proceedings of COLING-92*: 950-955 (1992)

Karttunen, L., Root, R. and Uszkoreit, H. Morphological Analysis of Finnish by Computer. *Proceedings of the 71st Annual Meeting of the SASS.* Albuquerque, New Mexico. (1981)

Karttunen, L.Finite-State Lexicon Compiler. *Technical Report.* ISTL-NLTT-1993-04-02. Xerox PARC, Palo Alto, California (1993)

Koskenniemi, K. *Two-level Morphology: A General Computational Model for Word-form Recognition and Production.* Univ. of Helsinki, Dept. of Gen. Ling., Publications No.11. (1983)

Oflazer, K. Two-Level Description of Turkish Morphology. *Proceedings of EACL-93.* (1993)

Slocum, J. Morphological Processing in the Nabu System. *Proceedings of the 2nd Applied Natural Language Processing:* 228-234 (1988)

Voutilainen, A. Does Tagging Help Parsing? A Case Study on Finite State Parsing. *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing:* 25–36 (1998)

Zajac, R. Feature Structures, Unification and Finite-State Transducers. *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing:* 101–109 (1998)