Human Language Technologies – The Baltic Perspective
K. Muischnek and K. Müürisep (Eds.)

doi:10.3233/978-1-61499-912-6-47

Estonian Morphology in the Giella Infrastructure

Heiki-Jaan KAALEP [a,1], Sjur Nørstebø MOSHAGEN [b] and Trond TROSTERUD [b]
[a] *Tartu Ülikool, Tartu, Estonia*
[b] *UiT Norgga Árktalaš universitehta, Giela ja kultuvrra instituhtta, Romssa, Norga*

Abstract. Years ago, a dedicated infrastructure for facilitating language technology development was set up by the Divvun language technology development group at the University of Troms.

We present a case study of using the infrastructure while developing a morphological description for Estonian. We are interested in how the environment helps a linguist to focus on the actual linguistics, and supports reuse of code, by which we mean reuse of dedicated scripts and make-files to manipulate the language-specific source material into FST.

The paper's concern is what is the best practice to use with different FST programming language files lexc, twol and xfst - how one should write the expressions, and what conventions should be followed. In many cases there are alternative ways for describing the phenomena, and the choice will prove to be good or bad only after a considerable amount of effort has been put into following it.

Keywords. morphology, finite state transducers, Estonian, infrastructure,

1. Introduction

We present a case study of developing a morphological description for Estonian in the Giella infrastructure[2], developed at the University of Tromsø by [1]. It is important to note that:

1. Estonian is a well-described language. e.g. [2], [3], [4], [5] are recent sources describing its morphology from different perspectives.
2. Several wide-coverage morphological analyzers for Estonian existed prior to this attempt, e.g. [6], [7], [8].

The motivation for yet another implementation of Estonian was partly practical, partly theoretical. Although much of the existing morphology-related software for Estonian is open source[3], the code has not been recently forked or developed further by either the linguistic or the computational community. We take this as evidence that these im-

[1]Corresponding Author: Heiki-Jaan Kaalep, Tartu Ülikool, Tartu, Estonia; E-mail: Heiki-Jaan.Kaalep@ut.ee.

[2]`https://victorio.uit.no/langtech/trunk/`

[3]`https://github.com/Filosoft/vabamorf`, `http://www.eki.ee/tarkvara/`, `https://github.com/jjpp/plamk`

plementations are somewhat uncomfortable to enhance. From the theoretical viewpoint, [9] has argued for some revisions in the way Estonian morphology is described. The aim was to check whether the arguments bear out when applied consistently to the whole vocabulary, and whether the resulting system would be without unwelcome complications. Thus, the case study is about what helps (and what does not) a theoretically-inclined linguist to formulate his or her ideas in the finite-state transducer (FST) framework.

The paper's concern is what is the best practice to use with different FST programming language files *lexc*, *twol* and *xfst* - how one should write the expressions, and what conventions should be followed. In many cases there are alternative ways for describing the phenomena, and the choice will prove to be good or bad only after a considerable amount of effort has been put into following it. Note that the issue here is the human-readable form, analogous to the issue of readability of program code in computer science. It took a long time before the programming language community understood that the GOTO statement is harmful and should be avoided; can we find similar suggestions for FST?

2. Giella infra for inflectional morphology

Judging by the majority of papers on computing and morphology, building a morphological analyzer for a language is commonly viewed as an exercise in coding, rather than one in linguistic research or development. (There are notable exceptions, though, especially when working on lesser-studied languages e.g. [10, p. 212-213], [11, p. 294]) Existing open-source analyzers available for one to mimic abound, and the field of computational morphology (using FST technology) is mature enough to have dedicated courses and textbooks. (The most comprehensive guide for a practitioner in FST, covering basic technology as well as implementation tips, is [12] with its accompanying web-site `https://web.stanford.edu/~laurik/fsmbook/home.html`, `https://web.stanford.edu/~laurik/.book2software/twolc.pdf`.)

Moreover, a dedicated infrastructure for facilitating language technology development has been set up by the Divvun language technology development group at the University of Tromsø [1]. Giella infra is a pre-defined directory structure for populating with the appropriate language-related data, and a set of modifiable pre-fabricated configuration and make files for turning the language data into transducers. These pre-defined elements reflect the previous experience of creating FSTs for different languages and purposes. They also guide a new user's thoughts to follow the same paths of reasoning and abstraction. For example, there are directories called *stems* and *affixes*, reflecting the view that separating these is good abstraction.

The vision of [1] is the following.

Every language has a a core FST that embodies the morphology and word formation of the language, i.e. covers inflection, derivation and compounding. Concrete applications, e.g. speller, machine translation, language learning aids, use this core FST, modifying it as necessary. The core, in turn, is the result of computationally modelled morphotactics and morphophonology.

[1] characterizes the desirable properties of this infrastructure in the following way: "it separates between language-independent and language-specific data and constructs; it works the same for all languages, thereby letting the linguists focus on the actual linguis-

```
LEXICON NOUNS
          kott+N:kott TAUD ;

LEXICON TAUD
          +Sg+Gen:>i+WeakGrade # ;
```

Figure 1. Declination as a continuation lexicon for *kott*

```
 Lexicon transducer:            k o t t +N +Sg +Gen
                                k o t t  >  i  +WeakGrade
 Morphophonological transducer: k o t t  >  i  +WeakGrade
                                k o t 0  0  i  0
```

Figure 2. Cascade of two FST-s for mapping inflectional categories to surface form `kott+N+Sg+Gen:koti`

tics, and ease comparison and cooperation across languages; it uses standard and widely available tools as far as possible; it uses and depends on open-source LT tools, either exclusively or as an alternative; it is platform neutral (but requires un*x for compilation); it supports building a large number of scientific and end-user tools, both normative and descriptive; it supports dialect variation in a standardized way; it integrates with or exports end-user tools for different platforms (e.g. spellers, hyphenators, grammar checkers, for both open-source and closed-source host systems and applications); and finally: it supports reuse of code as much as possible."

An FST is a finite state automaton that reads symbols from input tape and writes to output tape, thus mapping the input and output pairs. Figure 2 shows a cascade of two transducers, resulting from an operation called composition (lexicon.fst .o. morphophonological.fst), where the output of one serves as the input for the other; depending on the direction (downwards or up), the result is either the generation or the analysis of word-form *koti* (bag). Using and combining these two transducers is a standard way in the Giella infra, implemented for many languages.

The lexicon transducer is for paring the base form and grammatical categories (on the upper-side) with the possible stems and final-form-related, more or less abstract surface strings and morphophonological triggers (e.g. the weak grade trigger on figure 2). The lexicon transducer is described using `lexc`, language that facilitates dictionary-style descriptions. The conjugations and declinations are expressed via a set of continuation lexicons, c.f. figure 1.

The morphonological transducer is for paring the abstract strings with the final orthographic form. It is used for phenomena that act similarly across different inflectional classes, like consonant gradation (e.g. delete one *t* in the presence of weak grade trigger on figure 2). The lexicon transducer is described in `twolc`, language that facilitates expressing phonological and orthographic re-write rules that operate in parallel.

3. Estonian inflectional morphology

The implementation of Estonian inflectional morphology fits into Giella infra well. Actually, at present there are two versions: `https://victorio.uit.no/langtech/`

`trunk/langs/est` hosts a re-implementation of [8], a descendant of [2] and [13], and `https://victorio.uit.no/langtech/trunk/experiment-langs/est` hosts the version described in this article.

This new computational implementation of Estonian is based on previous theoretical and computational treatments. A theoretical starting point was a series of articles [14] [15] [9] [16] that argued for some revisions in the theoretical treatment of the Estonian case system (and which were very much influenced by [17]), and of verb categories. Combined with [2], [3] [7] and [8], they are responsible for many practical decisions in the way Estonian FST-s are described.

The current implementation has the following properties:

- The implementation has a lexicon of 68,000 entries containing tens of thousands of frequent derived and compounded words in addition to simplex ones. It is a re-make of the lexicon of the open source analyzer and generator by Filosoft[4]. The rules for derivation and compounding are largely re-engineered from the same source. Morphophonological rules are largely based on [3], [13] and [18].

- The nomenclature of inflectional classes is slightly different from all previous descriptions.

- The possibility of gradation alternations in inflections is determined by the inflectional class, but whether it realizes per concrete word, depends on the phones the word consists of. For example, *kott* (bag), *president* and *seminar* belong to the same inflectional class with weakening gradation - singular genitive is *koti*, *presidendi* and *seminari* - but *seminar* has no long phones, so there is nothing that could possibly shorten (as opposed to *kott* and *president* with a final long stop). This is different from the conventional descriptions; e.g. [2] and [5] put *seminar* into a different class.

- Unproductive phone alternations are marked with special symbols in the lexicon entries, productive ones with conventional orthography. This is a way to limit certain two-level morphophonological rules to apply only to a limited set of irregularly inflected words.

- In case of synonymous inflectional forms of a word, the less-preferred alternative, i.e. the less frequent one, is explicitly marked with a tag +Use/Rare. The treatment of rare forms is then dependent upon subsequent processing, described in [19].

The guiding principles when implementing Estonian morphology were:

1. The complex nature of the system results from an interplay of simple rules.

1.1. Inflectional classes and morphophonological rules: e.g. compare the Sg Gen of *sugu* (kin, gender) and *pugu* (crop): *sugu*:*soo* vs *pugu*:*pugu*. Despite their different Sg Gen, *sugu* and *pugu* might be classified to the same declination – all the rules of inferral (for the stems) and the agglutinatively added inflections to stems are the same for both words. It is the underlying phonological structure *suG1u* that makes its Sg Gen different. There are two morphophonological rules that operate on all words with *G1*: first, *G1* surfaces as *g* as a default, but disappears in weak grade forms, and second, a high vowel (*i*, *u*, *ü*) lowers to (*e*, *o*, *ö*) once *G1* has disappeared. We thus get *suG1u* - *su0u* - *soo*.

1.2. Instead of many possible surface realizations of one morphophoneme, conditioned by complex context conditions, we postulate that the underlying phonological structure has more different morphophonemes, each having a few surface realizations with simple context conditions. We thus increase the complexity of the lexical items,

[4] `https://github.com/Filosoft/vabamorf`

```
read regex ~[?* A l p i d "+N" "+Sg" ?*] .o. simplexwords.fst
```

Figure 3. `xfst` expression for blocking the singular forms of *Alpid* from a transducer for simplex words

achieving simplification of the rules in return. E.g. for Sg Nom:Sg Gen *härg*:*härja* (ox, bull) we assume that there is no realization of *G1* as *j*, but that *J1* which by default surfaces as *j*, is suppressed next to a voiceless consonant. The phonological structure is *härJ1G1*, with *g* and *j* never surfacing simultaneously. Incidentally, *J1* in *härJ1G1* indicates palatalization that is missing in contemporary Estonian, but was present in the 19th century, according to [20].

2. Despite being seemingly unpredictable, the membership of an Estonian word w.r.t. its declination can be determined from its derivational and phonological structure. For example, consider *leiB1* (bread) that inflects *leib*:*leiva* and *seib* (washer) that inflects *seib*:*seibi*. *Seib* belongs to class that contains many more words, is productive, i.e. grows by incorporating new words (e.g. loanword *beib* (babe)), and is considered by the speakers to follow the default, normal way of inflecting. In contrast, *leib* belongs to a class with few members (and shrinking), and its inflectional behaviour has to be remembered by the speakers (because it cannot be inferred from the phonological structure).

Words that belong to unproductive classes are naturally encoded with more and/or special symbols (*leiB1*), mimicking the need of a human speaker to remember them for their special inflectional behaviour. This reasoning has led to the practical decision to use only standard orthography to encode the stems of words of productive classes; the special symbols are used in stems of un-productive classes, and in continuation classes that represent conjugations and declinations, i.e. the constellations of stems, inflectional endings and morphological category tags, where some extra knowledge is natural to expect from a speaker. This is an attempt to move towards the best practice for morphophonemic symbols and rules, trying to set principles instead of ”a matter of taste” [21, p. 149]. An additional benefit would be that a lexicon where only exceptions need special knowledge for writing them down is easier to update and maintain.

In addition to morphotactics, phonology and orthography that are well captured by the cascade of lexicon and morphonological transducers, there are some semantic and pragmatic features that determine the possible well-formed words of a language. For example, pluralia tantum words like *Alpid* (the Alps) have no singular forms. The most natural way to define this constraint is to subtract the singular forms from the full paradigm, using a metaphor of filtering by [12, p. 249-255]. That is, first create an over-generating FST, then an FST for singular forms of *Alpid*, and finally subtract the latter from the first FST. This subtraction can be performed via composition by `xfst`, a dedicated program for the Finite-State Calculus (figure 3).

In Estonian, filtering is also used for handling derivation. First, a derivational affix can only attach to a designated inflectional form, Sg Gen usually. Second, Estonian nouns, adjectives, numerals and pronouns form a uniform group of declinable words – a declination is typically not specific to any of those finer categories. However, a derivational affix changes the word class, e.g. from noun to adjective, and is hence applicable only to nouns. The way to define transducers for derivations is first to allow all affixes attach to any word-class, and then use a word-class-based filter to prune the result.

The need for various filters is so pervasive that there are dedicated directories for filters (both general and language-specific) in the Giella infra.

4. Compounding

Compounding is a process where word-forms are concatenated to form a new word. In the Giella infra, the tradition has been to describe it as morphotactics, via continuation lexicons: the continuation has to point to some stem lexicon. This turns the system into a cyclical one. In order to impose restrictions on the process (e.g. allowing only some adjectives to precede a noun), some initially defined sub-lexicons should be split. What is troubling here is that compounding as a process is different from inflection – it is guided by different type of information for the language users, and described in different terms by linguists – but here this essential difference is ignored on the description level. Cycles and partitioned sub-lexicons result in an intricately intertwined system.

The strategy is different for Estonian. Instead of an intertwined network of continuation classes, presented in `lexc` language, we first define FST-s which correspond to common linguistic elements (e.g. an FST for simplex words, another for numbers), and then filter and combine these blocks with `xfst`.

Technically it means that the configuration and make files are notably different from what was initially assumed to be universal, language independent. The Estonian-specific features are a result of specific description strategies and reasoning patterns, not the result of the peculiarities of the Estonian language.

Estonian compounding is very productive (derived words and compounds may form new compounds), but limited in that not every word-form may be concatenated with any other, and the number of word-forming stems cannot exceed five. In fact, there are only a handful of attested five-stem compounds, and they contain lexicalised units, e.g. *all+maa+raud+tee+jaam* contains the same three lexicalised components as its English translation, *sub+way rail+way station*. So a natural restriction would be to limit the number of components to three, given the fact the lexicon already contains many lexicalised compounds.

The simplest solution would be to concatenate three simplex word transducers. Unfortunately, this would result in a huge FST (that also takes ages to compile), and every restriction on the set of possible combinations of word-forms will increase the size some more times. This happens because of the nature of FST-s: a state in FST has no information about the path that has been traversed to reach it, i.e. it has no memory. Restrictions of compounding, however, refer to word class or semantic properties of the components, i.e. long distance (counted as traversed symbols) dependencies. Moreover, limiting the number of consecutive components means that the final states of previous components must be ”remembered”. The only way to make a state in FST remember its previous path is to ensure that this path is unique to this state, i.e. there are no alternative incoming transitions for the intermediate states. This means that many intermediate states and transitions cannot be shared with other paths.

A solution would be to use *Flag diacritics* (FD) [12, p. 339–373], a confusing term for symbols that are best thought of as Boolean variables in a traditional programming language, variables that are set to some value and checked by the program that traverses the FST. FD are useful when one wants to block a path in FST, and information for this decision comes from several distant parts of this path. A typical case is allowing an adjectival prefix to a verb stem that has been turned into an adjective by a suffix, and prohibiting this prefix if there is no such suffix.

For counting components of a compound, we start by concatenating the FST for simplex words with itself an unspecified number of times, i.e. perform a Kleene star

operation. It means the resulting FST has a transition from every end state to the starting state, the traversing routine may cycle over the FST indefinitely, and accept any number of concatenated simplex words.

However, we modify the simplex word FST by adding an FD that changes its value once some final state has been passed. Finally, one of its values blocks the path, and the cycle is stopped.

A similar FD mechanism is used for representing info about word class, semantic properties of the stem etc. A traversing program checks the values of FD-s and blocks some unsuitable paths, e.g. one that contains a noun, followed by a personal pronoun.

5. Concluding remarks (hypothetical impact, possibly wishful thinking)

[10, p. 212-213] criticizes theoretical morphologists for ignoring computational linguistics and re-inventing their own theoretical wheels - representation formalisms - instead of adopting an implementation of finite-state transducers. For a case study, he shows how the verb morphology of Lingala, a Bantu language, can be described using *xfst* more elegantly than with the formalism invented by [22] for the same purpose.

A possible way to bring theorists and computational practitioners closer together could be via good FST-descriptions of languages, presented in human-readable form. Traditionally, a theoretical general linguist first gets his facts (knowledge about a language) from articles, grammar books and dictionaries, written by fellow linguists, and then formulates his own theories. Now if the knowledge source about a language would be a comprehensive, human-readable formal description instead, then perhaps the general theory would somehow also reflect this, and position itself to it.

The result would be a discussion about the strengths and weaknesses of different descriptions and theories, unhindered by spurious differences in formalism.

This would help the practitioners as well as provide the theorists with sounder facts. After all, it is the computational tools that use morphological descriptions that are used for processing real texts, these tools thus acting as a test-bed for the adequacy and completeness of their underlying morphological descriptions. Done in this way, linguistics would become a falsifiable science.

Our experience with Estonian shows that the Giella infra imposes certain conceptualization patterns on those who use it. Breaking away from some of these (while still staying in the environment) proved to be doable, but it required understanding of the whole architecture of the infra, as well as the u*nix make and configuration files. Hopefully, our activity has explicated some implicit assumptions on how linguistic knowledge might and should be encoded in FST. We hope that different description strategies will result in best practice convergence.

Acknowledgements

This work has been supported by the Estonian Ministry of Education and Research grant IUT 20-56 (Computational Models for Estonian), by the Norwegian-Estonian Research Cooperation Program grant EMP160 (SAMEST Sami-Estonian language technology cooperation similar languages, same technologies), and by the European Union through the European Regional Development Fund (Centre of Excellence in Estonian Studies).

References

[1] S.N. Moshagen, T.A. Pirinen and T. Trosterud, Building an open-source development infrastructure for language technology projects, in: *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013); May 22-24; 2013; Oslo University; Norway. NEALT Proceedings Series 16*, Linköping University Electronic Press, 2013, pp. 343–352.

[2] Ü. Viks, *Väike vormisõnastik: Sissejuhatus & grammatika*, Vol. 1, Eesti teaduste akadeemia, Keele ja kirjanduse Instituut, 1992.

[3] M. Erelt and R. Kasik, *Eesti keele grammatika: Morfoloogia sõnamoodustus*, Eesti TA Keele ja Kirjanduse instituut, 1995.

[4] M. Erelt (ed.), *Estonian language*, Tallinn: Eesti Teaduste Akadeemia Kirjastus, 2003.

[5] T. Erelt, T. Leemets, S. Mäearu and M. Raadik, *Eesti õigekeelsussnaraamat*, Eesti Keele Sihtasutus, Tallinn, 2013.

[6] H.-J. Kaalep, An Estonian morphological analyser and the impact of a corpus on its development, *Computers and the Humanities* **31**(2) (1997), 115–133.

[7] Ü. Viks, Eesti keele avatud morfoloogiamudel, in: *Arvutuslingvistikalt inimesele*, T. Hennoste, ed., Tartu Ülikooli üldkeeleteaduse õppetooli toimetised., Vol. 1, Tartu Ülikooli Kirjastus, 2000, pp. 9–36.

[8] J. Pruulmann-Vengerfeldt, Praktiline lõplikel automaatidel põhinev eesti keele morfoloogiakirjeldus, Master's thesis, Tartu likool, 2010.

[9] H.-J. Kaalep, Eesti käänamisssteemi seaduspärasused, *Keel ja kirjandus* (2012), 418–449.

[10] L. Karttunen, Computing with Realizational Morphology, in: *Computational Linguistics and Intelligent Text Processing. CICLing 2003*, A. Gelbukh, ed., Lecture Notes in Computer Science, Vol. 2588, Springer, 2003.

[11] T. Trosterud, Grammatically based language technology for minority languages, in: *Lesser-Known Languages of South Asia: Status and Policies, Case Studies and Applications of Information Technology*, A. Saxena and L. Borin, eds, Trends in Linguistics. Studies and Monographs [TiLSM], De Gruyter, 2006, pp. 293–315. ISBN ISBN 9783110197785.

[12] K.R. Beesley and L. Karttunen, *Finite-state morphology*, CSLI, Stanford, 2003.

[13] H. Uibo, Eesti keele morfoloogia modelleerimisest, in: *Keel ja arvuti*, M. Koit, R. Pajusalu and H. Õim, eds, Tartu Ülikooli üldkeeleteaduse õppetooli toimetised., Vol. 6, Tartu Ülikooli Kirjastus, 2006, pp. 13–35.

[14] H.-J. Kaalep, Kuidas kirjeldada ainsuse lhikest sissetlevat kasutamisandmetega koosklas, *Keel ja kirjandus* (2009), 411–425.

[15] H.-J. Kaalep, Mitmuse osastav eesti keele käändesüsteemis, *Keel ja kirjandus* (2010), 94–111.

[16] H.-J. Kaalep, Eesti verbi vormistik (Estonian Verb Paradigm), *Keel ja kirjandus* (2015), 1–15.

[17] W.U. Wurzel, System-dependent morphological naturalness in inflection, in: *Leitmotifs in Natural Morphology*, W.U. Dressler, W. Mayerthaler, O. Panagl and W.U. Wurzel, eds, Studies in Language Companion Series, Vol. 10, John Benjamins Publishing Company, Amsterdam/Philadelphia, 1987, pp. 59–95.

[18] K. Koskenniemi, Two-level Morphology: A General Computational Model for Word-Form Recognition and Production, PhD thesis, University of Helsinki, 1983.

[19] H.-J. Kaalep, Parallel Forms in Estonian Finite State Morphology, in: *Proceedings of the Fourth International Workshop on Computatinal Linguistics of Uralic Languages*, 2018, pp. 141–155.

[20] F.J. Wiedemann, *Ehstnisch-deutsches Wrterbuch*, Kaiserliche Akademie der Wissenschaften, St. Peterburg, 1869.

[21] T. Trosterud and H. Uibo, Consonant Gradation in Estonian and Sámi: Two-Level Solution, in: *Inquiries into Words, Constraints and Contexts. Festschrift for Kimmo Koskenniemi on his 60th Birthday*, A. Arppe, L. Carlson, K. Lindén, J. Piitulainen, M. Suominen, M. Vainio, H. Westerlund and A. Yli-Jyrä, eds, CSLI, Stanford, CA, 2005, pp. 136–150.

[22] G.T. Stump, *Inflectional morphology: A theory of paradigm structure*, Vol. 93, Cambridge University Press, 2001.