A Morphological Parser for Estonian

Richard Johansson
d99rj@efd.lth.se

Abstract

This report describes how a prototype morphological parser for Estonian, a language with a relatively complex morphology, was implemented. The implementation uses a two-level model which was hand-coded in Prolog. Some background material on the Estonian language and the two-level model is presented.

1 Introduction

A morphological parser is a computer tool that accepts a given inflected word (or sequence of words) and determines which its root word is and which inflections it has undergone. It can be used alone, as a kind of intelligent dictionary for the learner of a language, or as a morphological backend for a high-level natural language tool such as a dialogue system.

Morphology has been a neglected area of research in computational linguistics, mainly due to the fact that English has a trivial morphology. A simple morphology eliminates the need for morphological parsers since the inflectional forms are few enough to be stored in the dictionary along with their root words. But in a language with a complex morphology, where a noun or verb can take hundreds of shapes, an automatic tool is needed to extract the root word and interpret the inflectional information.

This report describes the implementation of a morphological parser for Estonian, a Finno-Ugric language with a relatively complex morphology. Although quite incomplete, the morphological parser is currently running for everyone to try at its web interface at `http://www.df.lth.se/~richardj/parser`.

The report is organized as follows: In Section 2 a short introduction to the Estonian language is given. The implementation of the parser is described in Section 3. In the following section, Section 4, the parser is evaluated and compared to existing tools on the web, and in Section 5 we discuss why the result is imperfect. Finally, a conclusion is given in Section 6.

Linguistic terms are for the sake of brevity only explained only when they are specific to Estonian. When using linguistic terminology, I have tried to stick to the Glossary of Linguistic Terms by SIL (`http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/`).

2 An Introduction to the Estonian Language

2.1 Background

Estonian is spoken by approximately 1100000 people in the world. Most of them live in Estonia, located on the east side of the Baltic Sea in the north of Europe, but there is also a large diaspora. The language is a Finno-Ugric language closely related to Finnish and some nearly extinct languages in the St. Petersburg area in Russia.
Some characteristics of the Finno-Ugric languages compared to the Indo-European are:

- absence of gender (the same pronoun for both he and she),
- absence of definite-indefinite articles (a and the in English),
- long words due to the structure of the language,
- numerous noun cases,

- postpositions rather than prepositions, and
- no syntactic equivalent of the verb “to have”.

Estonian has been more influenced by Indo-European languages (above all German) than Finnish has been, which is visible not only in the numerous loan words but also in the German-influenced word order and the use of genitive forms instead of possessive suffixes. Another striking feature of Finnish, the vowel harmony, has also disappeared in Estonian.

When the nationalist movement in Estonia emerged, and Estonian needed to be remade from a peasant’s language into a language for all levels of society, the language was artificially reconstructed grammatically and many new words were invented to purge the language from unwanted loan words and to cover concepts for which Estonian earlier had no suitable words.

A readable and thorough Estonian language course (written in Swedish) is [8]. A linguistic overview can be found in [1]. The dictionary data and inflection tables used when implementing the morphological parser were taken from *Õigekeelsussõnaraamat* [2] (from now on called the *ÕS*), the standard reference dictionary used in Estonia. The best Estonian-English dictionary is [7].

2.2 Letters and Sounds

Estonian uses the Latin alphabet with the following non-ASCII additions: *ä*, *ö*, *ü*, *õ*, *š* and *ž*. The letters *c*, *f*, *q*, *w*, *x*, *y*, *z*, *š* and *ž* are only used in words of foreign origin. In computer systems various encodings are used. This morphological parser, like most other computer systems using Estonian, uses the ISO-8859-1 encoding.

One interesting feature of Estonian phonology is that there are three degrees of quantity (length) of vowels and consonants. To indicate the first degree of quantity, a single letter is used. Unfortunately, the distinction between the second and third degree is not always indicated in the spelling. Here are two examples where this leads to ambiguity:

s*aa*da	(second)	send!	li*nn*a	(second)	of the town
s*aa*da	(third)	to get	li*nn*a	(third)	into town

Beside quantity, there is another source of ambiguity in Estonian spelling: palatalization of consonants. This means that dental consonants are pronounced with the tongue in the position of an *i*-sound, slightly similar to the sound *ñ* in Spanish. The letters *d*, *t*, *l*, *s* and *n* can all denote an unpalatalized as well as a palatalized sound.

Except for the quantity and the palatalization, Estonian spelling is simple. One letter generally stands for one sound. A single letter denotes a short sound and a doubled a long (second or third quantity).

The pronunciation of letters in Estonian is usually similar to the pronunciation in German, its greatest influence. There are although some differences: *b*, *d* and *g* are distinguished from *p*, *t* and *k* mainly in quantity (the former are pronounced with the first degree of quantity and the latter with the second) rather than voicing. The letter *v* is always pronounced voiced and *s* always voiceless. The umlauted letters *ö* and *ü* are pronounced like their German counterparts, but *ä* is open like the *a* in the English word *fan*. The letter *õ* is an indescribable sound specific to Estonian, but it can be approximated by the *u* in the Swedish word *Mumintrollen* with a deep Finland-Swedish pronunciation. The letters *z*, *ž* and *š*, which are used in loan words only, denote sounds similar to the English *z*oo, mea*s*ure and *sh*oe, respectively.

2.3 General Picture of Morphology

Originally Estonian, like its neighbour language Finnish still is, was a highly agglutinative language where separately identifiable and independent morphemes were added to the word stem. But due to influence from Indo-European languages and phonological attrition, morphemes in word-endings are not always separable in present-day Estonian. For example, the present tense person and number endings are today different from the past tense endings. But Estonian still has some agglutinative features, like most of the noun case endings (which are simply added to the genitive endings) and the clitic *-gi/-ki*.

Central to the morphology of Estonian is the concept of gradation. This is a process where a word changes phonetically, usually in the form of a syllable shortening or loss of a stop consonant, when the word is put into certain inflections. For example, the verb *õppima*, "to study", (where *pp* signifies a third degree stop) becomes *õpin* (with a second degree stop) in the present tense first person singular, and the noun *lind*, "bird" (with a short stop at the end) becomes *linnu* in the genitive singular (the stop is lost).

2.4 Morphology of Verbs

The verb morphology is more complex in Estonian than in English but comparable in structure and complexity to Romanic languages like French or Spanish. The finite forms of the verbs are constructed according to the following morphological features:

- *Mood*: Indicative, conditional, imperative or oblique (reported speech),
- *Voice*: Active or passive,
- *Tense*: Present or past (perfect and pluperfect are constructed with participle forms),
- *Polarity*: Affirmative or negative,
- *Person*: First, second or third,
- *Number*: Singular or plural.

As one can see, the Estonian verb features are rougly the same as in many Indo-European languages, with the main difference being that the verb has an affirmative – negative polarity feature. In the negative polarity, the verbs lose their person and number endings (except for in the imperative mood). The oblique mood, which tells what one is said to or supposed to do, is also notable.

There are also nonfinite forms such as participles, a gerund and two kinds of infinitive. The dictionary form for verbs is the first infininitive.

In Appendix A we can see an example verb *minema*, which is the most irregular verb of the Estonian language, conjugated in most forms.

2.5 Morphology of Nouns and Adjectives

While the verb morphology of Estonian is relatively similar to those of some Indo-European languages, the morphology of nouns is radically different. As mentioned above, Estonian has a wealth of noun cases. From a morphological point of view, the cases can roughly be divided as follows:

- *Fundamental*: Nominative (denoting the subject of the sentence), genitive (denoting possessor or the object of the sentence) and partitive (denoting a quantity or the object of the sentence),
- *Nonfundamental*: Illative (into what?), inessive (in what?), elative (from the inside of what?), allative (onto what?), adessive (on what?), ablative (from the outside of what?), translative (into the state of what?), terminative (until what?), essive (as what?), abessive (without what?) and comitative (with what?).

The nonfundamental cases are usually formed by adding their respective ending to the genitive form. For example, if we wish to construct the inessive plural we add the ending *-s* to the genitive plural form.

Adjectives in Estonian are from a morphological point of view just nouns which can form the comparative or superlative forms. The dictionary form for nouns and adjectives is the nominative singular. The nouns and adjectives are declined in singular and plural. Many alternate forms are possible.

In Appendix B we can see the nouns *raamat* and *mägi* in all inflectional forms. For *raamat*, alternate plural forms are available for many cases – the case ending may be added both to the genitive plural and to the partitive plural (which is considered somewhat archaic). For the other example noun, *mägi*, we see that a short form of the illative case is possible. If the short form is available, it is generally preferred.

Nouns in the nominative singular or genitive singular or plural may form a compound with another noun.

2.6 Syntax

As mentioned above, Estonian word order is influenced by the German one. But that is more a question of preferred style than a rigid rule, and generally speaking the word order is free for the speaker to vary according to taste and style or for emphasis or clarity.

One of the fundamental concepts in Estonian syntax, which is also notable from a morphological point of view, is the distinction between total and partial objects, which rougly means that the object was affected "totally" or "partially". This manifests itself in the use of the genitive or nominative case for the total object and partitive case for the partial object. For example:

Ma lugesin raamatut.	I read (or was reading) a book.
Ma lugesin raamatu läbi.	I read the book through.

Here *raamat*, "book", is in the partitive case in the first example and in the genitive case in the second. The word *läbi*, "through", is one of a number of particles used to emphasize that the action is complete. These particles have come into use relatively recently and the reason they are used is that for many words in present-day Estonian, the genitive and the partitive forms coincide. For example:

Koer sõi vana muna.	The dog ate (or was eating) an old egg.
Koer sõi vana muna ära.	The dog ate the old egg (up).

Here *vana*, "old", and *muna*, "egg", have identical genitive and partitive forms.

3 Implementation

3.1 Goals

The following goals of the parser project were set:

- The parser should be able to parse all inflected forms of nouns, pronouns, adjectives and verbs,
- it should be reasonably fast,
- it should be able to handle all possible compounds, not just the most frequent ones,
- it should be able to handle the most regularly appearing derivations.

3.2 Two-level Morphology

The model used in this implementation is the two-level model introduced by Kimmo Koskenniemi, which presently is the most widespread one for implementations of morphology systems. This model is useful because it is conceptually very simple and because it easily can be implemented efficiently using finite-state techniques. In [4], an introduction to the subject is given by its original inventor.

In the two-level model, a word is thought to have a *surface form* – what appears to the speaker – and an *underlying form* – the theoretical form consisting of root word and inflection information. The relations between these two forms are called *two-level rules.* The two-level rules are bidirectional – one can go from the surface form to the underlying form or vice versa.

The two-level rules usually operate symbol by symbol. To handle the case where a symbol is present in the underlying form but not in the surface form (or vice versa), a *zero symbol* is inserted.

Here is an example of a two-level rule for a group of Estonian nouns. For these nouns, which consist of two vowels separated by *b*, *d* or *g*, the surface genitive singular form is formed from the underlying nominative form by removing the separating consonant and mutating the vowels in the following way: *i* becomes *e*, *u* becomes *o* and *ü* becomes *ö*. Here we can see how a two-level rule could relate the surface form *sea* ("of the pig") to the underlying form *siga* ("pig") Singular Genitive.

Surface form	`s`	`e`	`0`	`a`	`0`	`0`
Underlying form	`s`	`i`	`g`	`a`	`S`	`G`

3.3 Finite-state Transducers

The two-level rules are usually implemented using finite-state transducers, which are nondeterministic finite-state automata where the arcs are labelled with pairs of surface-underlying symbols.

In [6] and [5], there are introductions to the theory of finite-state transducers, with theorems about their basic properties presented.

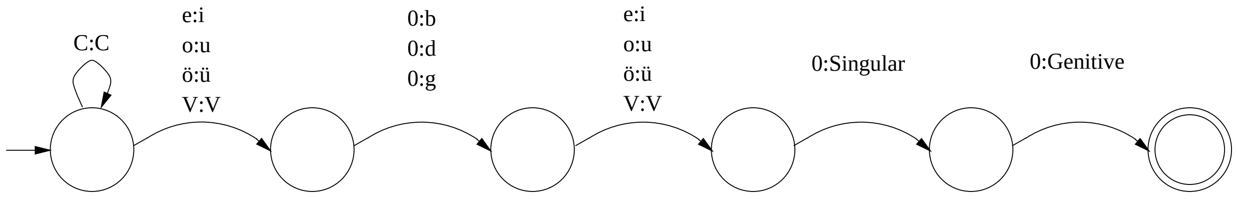

Figure 1: A finite-state transducer

In figure 1 we see a transducer which implements the two-level rule for forming the genitive for the noun group mentioned in Subsection 3.2. Here, C means any consonant and (for the sake of brevity) V means any vowel except *i*, *u* or *ü*.

To make a transduction from the surface string *sea* to the underlying string *siga* Singular Genitive, we first follow the arc from the start state back to itself, while consuming the *s* in both the surface and the underlying form. Then we jump to the next state, this time consuming the *e* in the surface form and the *i* in the underlying form. We follow arcs and consume symbol pairs in this way until there are no more symbols left, and if we are then in a final state (marked in the figure as a double circle), the transduction was successful.

Note that in this transducer, the transduction from surface form to underlying form is not a function, that is the underlying form is not uniquely determined by the surface form. In this example five additional underlying forms, for example the nonexistent **seba* Singular Genitive, are produced by the transducer. When trying to find correct underlying forms, the results need to be checked with a dictionary.

3.4 Implementation of Transducers

The transducers for this morphological parser were hand-written in Prolog. Using Prolog is particularly convenient from a programming point of view: the nondeterminism of the transducers and their bidirectional nature can both be implemented almost without effort.

In Appendix C a Prolog code example implementing the described transducer is given.

The reason that no existing two-level morphology system was used instead of hand-writing was mainly the time constraints – there was simply no time to learn how to use for example the Kimmo system [3].

3.5 The Dictionary

As mentioned above, the transducers produce lots of hypothetical forms which usually are incorrect. This means that for every word, many dictionary lookup need to be made. An efficient dictionary data structure is therefore essential.

The dictionary was implemented with a *letter tree* or *trie*. In a letter tree the words are stored as paths in a tree, with the nodes representing letters. In the leaf nodes, grammatical information such as conjugation pattern numbers could be stored. Two words which begin with the same letters share a common entry path. This leads to a compact dictionary data structure, where the search time is proportional to the number of letters of the word. In figure 2 a letter tree with two words is shown.

The dictionary used for the morphological parser is the online version of the *ÕS*. Since the transducers should be able to construct any possible compound word, those compound words which were already present in the dictionary could be removed. This reduced the size of the dictionary from about 120000 words to 50087

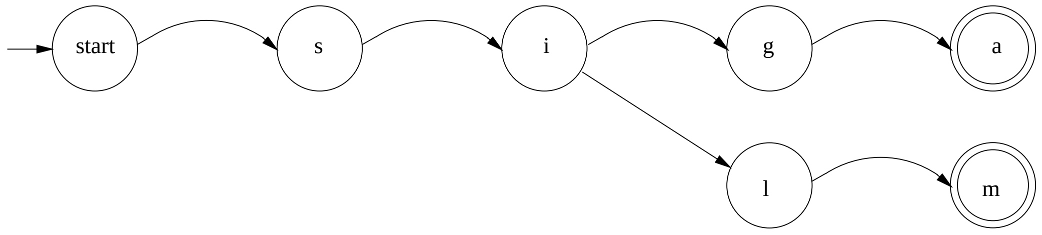

Figure 2: A letter tree

words.

3.6 Other Implementation Details

The parser system consists of a client containing the transducer code and a server handling the dictionary. This division was made for development reasons since the startup of the dictionary takes several minutes, and while developing and debugging the transducers the program frequently needs to be restarted.

The executables are built as compiled Prolog code and currently run on an UltraSPARC-II 300 MHz machine.

4 Evaluation

To evaluate the quality of the implementation, the parser was compared to two existing parsers on the web: Filosoft's at `http://www.filosoft.ee` and the one at Estonian Language Institute, `http://kiisu.eki.ee`. The prototype was run once with compound words enabled and once with compound words disabled. The 100-word test data used can be seen in Appendix D. The results were the following:

- When compound word handling was enabled the parser produced a result after 1 minute and 40 seconds with 69 percent correct parses.
- When compound word handling was disabled the result was produced in 20 seconds with 63 percent correct parses.
- The parsers at Filosoft and at Estonian Language Institute both parse the test data in a couple of seconds.

The benefit of compound words is questionable since it may lead to an explosion of possible constituent words of the compound. For example, when the word *patsiendiorganisatsiooniks* ("patient organization" in the translative case, which is just a two-word compound) was input, the parser produced no output at all (at least the report author did not have the time to wait for it).

5 Improvements

5.1 Program Speed

The main problem with the parser is, as we have seen in the previous section, its speed. There are a number of reasons for this:

The client-server division of the program should not have been done between the transducers and the dictionary, since all the dictionary lookups will lead to excessive process communication. This is currently not a big issue since the processes run on the same machine (and time measurements have shown that communication is only a small part of the execution time), but making another division could lead to a more efficient program. But as mentioned previously, the reason for this division was the development cycle time and this should only be used as a last resort.

A less attractive implementation improvement could be to abandon Prolog as implementation program language and fine-tune the implementation in a low-level language such as C. The current implementation uses Prolog features such as atom unification and backtracking extensively. Abandoning Prolog would however surely increase the development time drastically.

The parser behaves acceptably when compound word handling is disabled, but when compound words are handled the output may be produced after many minutes. The naïve implementation of compound word handling used here (just connecting nominative and genitive final states to the start of the noun transducer) is obviously not a good strategy. Limiting the number of allowed constituents is a strategy taken in the implementation of the two parsers which were compared to the prototype – they both allow a maximum of four nouns to be compounded. Memoization is obviously also a necessity when handling compound words. Another simple option is to disallow arbitrary compound words and just use the ones given in the $\tilde{O}S$.

5.2 Maintainability

Another obvious problem seen when inspecting the code is that it is unreadable, which is the result of the decision to hand-write the transducers in Prolog. This leads to code that is difficult to change and debug. It is obvious that such code could be generated by an automatic tool (for example a graph-drawing program), which could be produced in relatively little time. Another option could be to use existing rule-based two-level morphology systems such as Kimmo [3].

6 Conclusion

6.1 Goals Met

A prototype of the parser was implemented and set up to run with a web user interface. The verb morphology was completely implemented (except for some nonstandard forms), but due to time constraints the noun morphology was only partially implemented. The most common personal pronouns are handled, but no adjective features (comparative and superlative). Derivations are not handled except for the verbal noun forms (like "the eating"). The clitic *-gi/-ki* is not handled. The success rate of the prototype the test data in Appendix D was 63 percent when compound words were disabled.

As we saw in the Section 4, the parser prototype is acceptably fast when compound words are disabled but still much slower than the other ones tested. Compound words can be handled but expose a severe implementation problem, as the number of possible constituent words of the compound becomes enormous for long words.

6.2 Final Words

The parser project has been relatively successful and its goals have to a certain degree been met. But to overcome the speed and maintainability problems and produce a high-quality product such as the two existing parsers compared to the prototype, much work would need to be done.

A A Verb Example: *minema*, "to go"

A.1 Finite forms

Mood	*Voice*	*Tense*	*Affirmative polarity*	*Negative polarity*
			1st Person Singular, ..., 3rd Person Plural	
Indicative	Active	Present	lähen, lähed, läheb, läheme, lähete, lähevad	lähe
		Past	läksin, läksid, läks, läksime, läksite, läksid	läinud
	Passive	Present	minnakse	minda
		Past	mindi	mindud
Conditional	Active	Present	läheksin, läheksid, läheks, läheksime, läheksite, läheksid	läheks
		Past	läinuksin, läinuksid, läinuks, läinuksime, läinuksite, läinuksid	läinuks
Imperative			mingu, mine, mingu, mingem, minge, mingu	mingu, mine, ...
Oblique	Active	Present	minevat	minevat
	Passive	Present	mindavat	mindavat

A.2 Nonfinite forms

First infinitive	minema
Second infinitive	minna
Progressive form	minev
Gerund	minnes
Active past participle	läinud
Passive past participle	mindud

B Two Noun Examples

B.1 *raamat*, "book"

	Singular	*Plural*	
Nominative	raamat	raamatud	
Genitive	raamatu	raamatute	
Partitive	raamatut	raamatuid	
Illative	raamatusse	raamatutesse	raamatuisse
Inessive	raamatus	raamatutes	raamatuis
Elative	raamatust	raamatutest	raamatuist
Allative	raamatule	raamatutele	raamatuile
Adessive	raamatul	raamatutel	raamatuil
Ablative	raamatult	raamatutelt	raamatuilt
Translative	raamatuks	raamatuteks	raamatuiks
Terminative	raamatuni	raamatuteni	raamatuini
Essive	raamatuna	raamatutena	raamatuina
Abessive	raamatuta	raamatuteta	
Comitative	raamatuga	raamatutega	

B.2 *mägi*, “mountain”

	Singular	*Plural*
Nominative	mägi	mäed
Genitive	mäe	mägede
Partitive	mäge	mägesid
Illative	mäesse | mäkke	mägede
Inessive	mäes	mägedes
Elative	mäest	mägedest
Allative	mäele	mägedele
Adessive	mäel	mägedel
Ablative	mäelt	mägedelt
Translative	mäeks	mägedeks
Terminative	mäeni	mägedeni
Essive	mäena	mägedena
Abessive	mäeta	mägedeta
Comitative	mäega	mägedega

C Prolog Code Example

```
arc(start, start, C, C) :- consonant(C).

arc(start, state1, i, e).
arc(start, state1, u, o).
arc(start, state1, ü, ö).
arc(start, state1, V, V) :- member(V, [a, e, o, y, ä, ö, õ]).

arc(state1, state2, b, 0).
arc(state1, state2, d, 0).
arc(state1, state2, g, 0).

arc(state2, state3, i, e).
arc(state2, state3, u, o).
arc(state2, state3, ü, ö).
arc(state2, state3, V, V) :- member(V, [a, e, o, y, ä, ö, õ]).

arc(state3, state4, 'Singular', 0).
arc(state4, final, 'Genitive', 0).

final_state(final).

transduce(Start, Final, [U | UnderlyingString], SurfaceString) :-
        arc(Start, Next, U, 0),
        U \== 0,
        transduce(Next, Final, UnderlyingString, SurfaceString).
transduce(Start, Final, UnderlyingString, [S | SurfaceString]) :-
        arc(Start, Next, 0, S),
        S \== 0,
        transduce(Next, Final, UnderlyingString, SurfaceString).
transduce(Start, Final, Under, Surface) :-
        arc(Start, Next, 0, 0),
        transduce(Next, Final, Under, Surface).
transduce(Start, Final, [U | UnderlyingString], [S | SurfaceString]) :-
        arc(Start, Next, U, S),
        U \== 0,
        S \== 0,
        transduce(Next, Final, UnderlyingString, SurfaceString).
transduce(Final, Final, [], []) :-
        final_state(Final).
```

D Test Data

The following test data was used in the performance tests. The text consists of 100 words and was taken from the website of the Estonian daily *Eesti Päevaleht* (`http://www.epl.ee`).

```
Kahju kui see nii läheb ütles Arstide Liidu eestseisuse liige Andres Lehtmets
kelle sõnul on patsiente esindav organisatsioon demokraatliku
ühiskonna üheks tunnuseks
Ühingul on laiem ja tasakaalustav roll kui ainult üksikisikute abistamine
lisas Lehtmets Mingil juhul pole see arstide vastane ühing vaid aitab välja
tuua kitsaskohti meditsiinis
Patsiendil kaob sõltumatu abi Praegu on esindusühingul kabinetid neljas linnas
Kahju on mitte inimeste koondamisest vaid sellest et patsiendil puudub edaspidi
abiline ja võimalus kaebuste korral nõu saada praegugi on meil mitu juhtumit
pooleli märkis Ilves kelle sõnul on ühing juhtumi lahendamisel inimesi
tõhusalt aidanud
Tappo kinnitusel pakkus ministeerium ühe lahendusena et esindusühing
```

References

[1] Aavik, Johannes. 2000. Introduction. In Saagpakk, Paul F., *Estonian-English Dictionary.* Koolibri.

[2] Erelt, T., R. Kull, V. Põlma, K. Torop. 1976. *Õigekeelsussõnaraamat.* Eesti NSV TA Keele ja Kirjanduse Instituut. (`http://ee.www.ee/QS`)

[3] PC-KIMMO. A Morphological Parser. `http://www.sil.org/pckimmo`

[4] Koskenniemi, Kimmo. 1997. Representations and Finite-State Components in Natural Language. In Roche, E. and Y. Schabes, editor, *Finite State Natural Language Processing.* MIT Press, pages 99–116.

[5] Mohri, Mehryar. 1997. On the Use of Sequential Transducers in Natural Language Processing. In Roche, E. and Y. Schabes, editor, *Finite State Natural Language Processing.* MIT Press, pages 355–382.

[6] Roche, Emanuel and Yves Schabes. 1997. Introduction. In Roche, E. and Y. Schabes, editor, *Finite State Natural Language Processing.* MIT Press, pages 1–65.

[7] Saagpakk, Paul F. 2000. *Estonian-English Dictionary.* Koolibri.

[8] Tuldava, Juhan. 1997. *Lärobok i estniska.* Pangloss.