

Morphological regularity and paradigms of Latvian Towards two-level morphology

Latvian is inflective, mainly suffixing language. Nouns inflect for number and case. A nominal word-form usually consists of a stem plus inflectional ending. Endings may be polysemous (e.g. gender + number + case) and homonymous (e.g., the same ending stands for both SG GEN and PL NOM (D5, D6)). Due to inflectional classes (declensions), there are also synonymous endings (e.g. masculine NOM endings *-s*, *-š*, *-is*, *-us*) [1].

Each noun has fixed grammatical gender, either masculine or feminine. For sake of simplicity and clarity we are presenting a robust model of Latvian noun morphology (Fig.1), excluding some subdeclensions, exceptions etc.

No.	Case	Declensions [‡]							
		D1	D2	D3	D4	D5	D6	D7	D8
SG	NOM	-s	-š	-is	-us	-a	-e	-s	-
	GEN	-a	-a	-a*	-us	-as	-es	-s	-
	DAT	-am	-am	-im	-um	-ai	-ei	-ij	-
	ACC	-u	-u	-i	-u	-u	-i	-i	-
	LOC	-ā	-ā	-ī	-ū	-ā	-ē	-ī	-
PL	NOM	-i	-i	-i*	-i	-as	-es	-is	-
	GEN	-u	-u	-u*	-u	-u	-u*	-u*	-
	DAT	-iem	-iem	-iem*	-iem	-ām	-ēm	-īm	-
	ACC	-us	-us	-us*	-us	-as	-es	-is	-
	LOC	-os	-os	-os*	-os	-ās	-ēs	-īs	-

Fig.1 — inflectional paradigms of nouns. Declension D8 (formed by loanwords) lacks inflections; stem ends with *-a*, *-ā*, *-e*, *-ē*, *-i*, *-o*, *-u*, or *-ū*, which traditionally are called non-inflectional endings.

Legend of Fig.1:

- “vertical” ambiguity: inflectional polysemy inside of one declension;
- * — morphophonological rules for stem-internal alternations have to be taken into account (see Fig.4);
- ‡ — to facilitate computation the division into declensions in our table slightly differs from the one traditional in Latvian linguistics.

In order to build a morphological parser, we need at least the following: lexicon, morphotactics, and orthographic/spelling rules [2].

Computational lexicon

Inflectional paradigm and gender determines the declension: D1–D4 stand for masculine (M), D5–D7 mostly represents feminine (F), and D8 mostly represents masculine. We will assume that gender is inferable from the declension.

Below are given some sample nouns with their English translations.

<i>Declension</i>	<i>Examples</i>
D1	gald-s ‘table’, kaln-s ‘mountain’, rag-s ‘horn’
D2	ceļ-š ‘road’, vēj-š ‘wind’
D3	cāl-is ‘chicken’, cel-is ‘knee’, kaķ-is ‘cat’
D4	liet-us ‘rain’, med-us ‘honey’
D5	galv-a ‘head’, liet-a ‘thing’, pann-a ‘frying-pan’
D6	lell-e ‘doll’, lent-e ‘type’, pas-e ‘passport’
D7	aus-s ‘ear’, nakt-s ‘night’
D8	kino ‘cinema’, ragū ‘ragout’, ateljē ‘atelier’

Taking into account the knowledge of morphological regularity and paradigms of Latvian nouns, as well as bearing in mind similarities in other parts-of-speech (see Fig.2) we are aiming to create very simple and flat, though relatively flexible data model for encoding lexicon to be used in morphological processing.

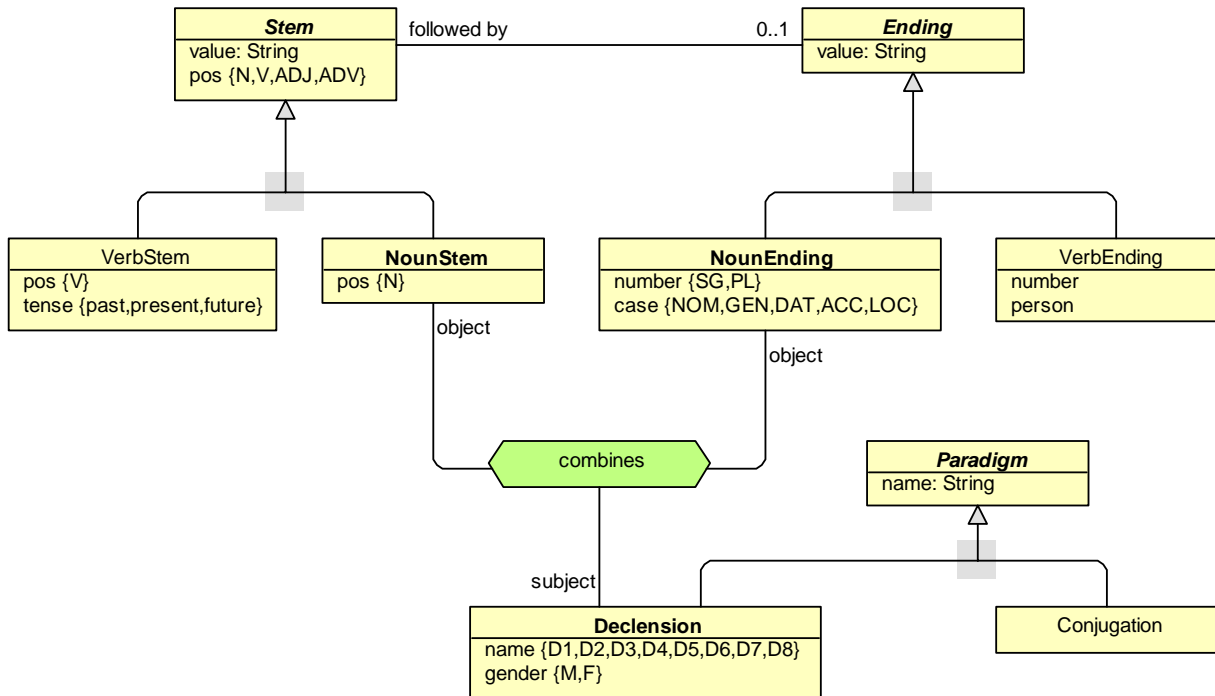


Fig.2 — conceptual class diagram representing relations among morphological features and grammatical description of word forms.

Schema of the lexicon (noun subset) can be derived from the Fig.2 as follows:

```

NounStem(string, pos, declension) .
NounEnding(string, declension, number, case) .
Declension(name, gender) .
    
```

It should be noted that while endings and declensions are countable sets and can/have to be provided manually, list of stems is theoretically infinite. Thus stem description (lexicon entry) has to have minimal attribute set to facilitate automatic acquisition of entries. In our model it can

be achieved easily exploiting any typical machine readable dictionary of Latvian, where part-of-speech and inflectional paradigm is given or can be inferable for each head word.

Morphotactics

Architectural requirements of a two-level morphological analyzer are sketched out in Fig.3. Analysis (parsing) is a mapping of surface type to the lexical type via finite-state transducer(s) — mapping rules (synthesis is a reverse process). In other words it is about how to combine units (entries) from the lexicon and how they can be combined on a surface.

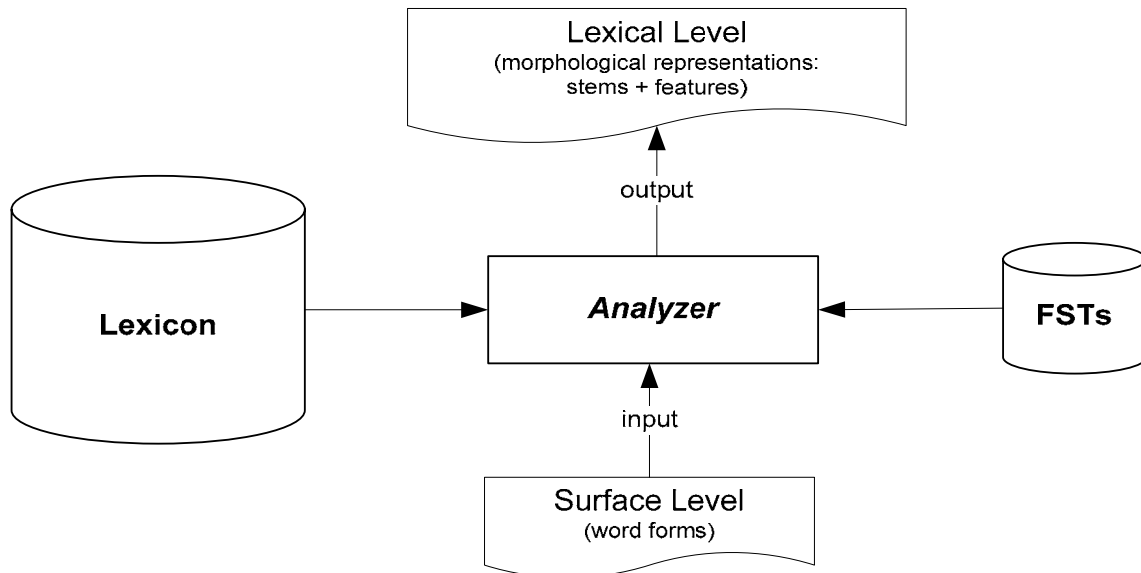


Fig.3 — overall architecture of a two-level morphological parser.

Input (surface level) alphabet:

$$I = \{a, \bar{a}, b, c, \check{c}, d, e, \bar{e}, f, g, \acute{g}, h, i, \bar{i}, j, k, \check{k}, l, \downarrow, m, n, \grave{n}, o, p, r, s, \check{s}, t, u, \bar{u}, v, z, \check{z}\}$$

Output (lexical level) alphabet:

$$O = I \cup \{N, D1, D2, D3, D4, D5, D6, D7, D8, M, F, SG, PL, NOM, GEN, DAT, ACC, LOC\}$$

For a general transducer for Latvian nominal inflection, please, see project's web site [3], page "morphotactics"; it is too large to include it here. Note: in Latvian an ending expresses following set of morphological features at a time: {gender, case, number}.

Morphophonology

In some declensions palatalization occurs in certain forms (marked with * in Fig.1), which changes the end of the stem. The possible changes are shown in Fig.4.

Palatalization can be solved using FSTs at the intermediate level (between surface and lexical levels).

Type	Palataliz.	Example	Formalization
J-insertion rule (lack of correspondent soft consonant)	b > bj m > mj p > pj v > vj	gulb-is > gulbj-a ‘swan’ kurm-is > kurmj-a ‘mole’ krup-is > krupj-a ‘toad’ ziv-s > zivj-u ‘fish’	$\varepsilon \rightarrow j / \left\{ \begin{matrix} b \\ m \\ p \\ v \end{matrix} \right\} \wedge \left\{ \begin{matrix} D3 \text{ SG GEN} \\ D3 \text{ PL GEN} \\ \dots \\ D7 \text{ PL GEN} \end{matrix} \right\} \#$
Consonant interchange rule (exists correspondent soft consonant)	c > č l > ļ n > ņ s > š t > š d > ž z > ž	lāc-is > lāč-a ‘bear’ cel-is > ceļ-a ‘road’ tauren-is > taureņ-a ‘butterfly’ trus-is > truš-a ‘rabbit’ zalkt-is > zalkš-a ‘adder’ bried-is > briež-a ‘deer’ āzi-is > āž-a ‘goat’	$\left\{ \begin{matrix} c \\ l \\ \dots \\ z \end{matrix} \right\} \rightarrow \left\{ \begin{matrix} \check{c} \\ \check{l} \\ \dots \\ \check{z} \end{matrix} \right\} / \left\{ \begin{matrix} c \\ l \\ \dots \\ z \end{matrix} \right\} \wedge \left\{ \begin{matrix} D3 \text{ SG GEN} \\ D3 \text{ PL GEN} \\ \dots \\ D7 \text{ PL GEN} \end{matrix} \right\} \#$
Double interchange (softened consonant palatalizes previous consonant as well)	ll > ļļ ln > ļņ nn > ņņ sl > šļ sn > šņ zl > žļ zn > žņ	lell-e > leļļ-u ‘doll’ aln-is > aļņ-a ‘moose’ pinn-e > piņņ-u ‘acne’ kausl-is > kaušļ-a ‘rowdy’ krāsn-s > krāšņ-u ‘oven’ zisl-is > zižļ-a ‘wand’ zvaigzn-e > zvaigžņ-u ‘star’	Cascade of previous and analogical rule/transducer

Fig.4 — orthographic/phonetic rules, which may affect ending boundary of a stem.

Implementation

Flexible architecture of a two-level morphological analyzer, which allows to plug-in FST(s) for word form recognition, can be achieved via one of the following approaches:

- formal representation of FST(s) using state transition tables;
- exploitation of ND-RECOGNIZE and state-space search algorithms in parsing and searching for solutions.

OR

- FST representation by set of regular expressions (any *regex* can be transformed to a finite state automaton, if it doesn't use memory [2]);
- combinations (e.g., cascades) of rules (e.g., due to palatalization) → combinations of regular expressions;
- exploitation of an existing regular expression engine (*regex* engines typically compile given expressions into FSAs and thus are using FSA processing algorithms [4]).

For sake of implementation simplicity we have chosen **regex approach**. From the assertion above follows that any FSA can be transformed to regular expression. In case of FST we will need pairs of IF-THEN expressions (patterns): IF will serve as recognizer (surface level), but THEN — as transducer (lexical level). Guidelines how to write down *regex*s from an automaton:

- if an arc reads from surface type, then the input label is appended to IF rule;
- if an arc writes to lexical type, then the output label is appended to THEN rule;
- if input equals output and both are variables, then input label is enclosed into parenthesis (capturing group), but output label is replaced with \$ + consecutive number of the capturing group — reference;

- if there are alternative labels on an arc, then comma mark is replaced with |.

A side effect of applying these guidelines: we found out that initial FST can be re-arranged in a more systematic way providing also slightly compact regular expressions and uniform transducer output. The new transducer as it is can be easier exploited for synthesizing as well.

We can also extend now our schema of the lexicon (Fig.2) as follows:

```
RegExRule (order, if, then) .
```

where *order* indicates at which stage to apply the rule — depending on its type: whether it deals with morphophonology or with morphotactics. Inside a stage the order of rules is not important — it have to be taken into account while writing the \mathbb{F} patterns.

Current implementation doesn't fully meet requirements of a language-neutral architecture of the analyzer. We tended to believe that truly flexible, universal parser can be created only by creating a metamodel of the morphology domain: describing what is a morphological feature, what is a morpheme, what are lexicon and its components etc. + **relations** (and their semantics) that stand among the concepts in a generic way (e.g., what are relations between morphotactics and schema of lexicon). Next is to derive a model for each specific language + to develop a set of FSTs + to populate entries of a lexicon and plug-in all the components into a generic analyser/synthesizer of word forms of any given language and part-of-speech. Apart from other issues one of essential is a language for morphological knowledge representation. The idea presented in DATR [5] could be considerable mechanism, especially, because of “built-in” facilities for defining of FSTs; however, we are sceptic about facilities required for metamodelling and modelling phases.

But there is **significant objection**: if lexicon itself is implemented as an FST (at the char level) and combined together with morphotactics, then there is no demand for metamodelling because of no separation (in term of model between lexical layer and rule layer), ie. the model is common. More over, it is more natural and therefore easier to incorporate morphophonological rules in a transducer-based lexicon than to apply them to “static” morphemes (stems).

Lesson learned: it is hard to define and implement stem-internal alternations in a language neutral, even in a part-of-speech neutral way, using finite state techniques, if stems are atomic units in the model of the lexicon. In general, one shouldn't reinvent the wheel developing yet another FST machine (if we are not considering technical or efficiency issues) — the task instead is to encode all the morphological knowledge (including lexicon) of a language into a form of finite-state transducer and to exploit an existing machine to apply the model.

More information and the experimental application can be found at the **project's web site** [3].

References

1. Nau Nicole. Latvian (Languages of the world: Materials; 217). — München; Newcastle: LINCOM Europa, 1998.
2. Jurafsky Daniel and Martin James H. Speech and Language Processing. New Jersey: Prentice Hall, 2000.
3. <http://www.ailab.lv/Normunds/Morphology>
4. Friedl Jeffrey E. F. Mastering Regular Expressions. Sebastopol: O'Reilly, 2002.
5. Evans Roger and Gerald Gazdar. 1996. DATR: A language for lexical knowledge representation. *Computational Linguistics* 22(2): 167-216.