# An implementation of a Latvian resource grammar in Grammatical Framework

**Pēteris Paikens, Normunds Grūzītis**
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia
E-mail: peterisp@ailab.lv, normundsg@ailab.lv

**Abstract**

This paper describes an open-source Latvian resource grammar implemented in Grammatical Framework (GF), a programming language for multilingual grammar applications. GF differentiates between concrete grammars and abstract grammars: translation among concrete languages is provided via abstract syntax trees. Thus the same concrete grammar is effectively used for both language analysis and language generation. Furthermore, GF differentiates between general-purpose resource grammars and domain-specific application grammars that are built on top of the resource grammars. The GF resource grammar library (RGL) currently supports more than 20 languages that implement a common API. Latvian is the 13th official European Union language that is made available in the RGL. We briefly describe the grammatical features of Latvian and illustrate how they are handled in the multilingual framework of GF. We also illustrate some application areas of the Latvian resource grammar, and briefly discuss the limitations of the RGL and potential long-term improvements using frame semantics.

**Keywords:** computational grammar, language generation, Grammatical Framework

## 1. Introduction

The long-term research behind this paper is aimed at semantic parsing of Latvian and natural language generation in Latvian. While our former focus has been on developing language resources and tools that can be primarily used for language analysis, in this paper, we describe a recent open-source implementation of a Latvian resource grammar that can be effectively used for both language *analysis* and language *generation*. We have implemented this resource grammar in Grammatical Framework (GF), a toolkit and formalism for rapid development of *multilingual* grammar applications (Ranta, 2011).

Latvian is an Indo-European language, a member of the Baltic language group, one of the official EU languages. In terms of speakers, it is a relatively small language (about 1.5 million native speakers and about 0.5 million non-native speakers). It uses a Latin-based alphabet that in almost all cases provides a one-to-one mapping between letters and phonemes. The general grammatical characteristic of Latvian is that it is a highly inflective language with a relatively free word order.

Large annotated language resources, such as treebanks and parallel corpora of various domains that would facilitate statistical parsing and generation, are scarce for Latvian – reusability of the developed computational grammars across general and domain-specific use-cases and across languages is very important.

A fairly successful attempt developing a robust, wide coverage partial parser of Latvian has been in lines with the dependency grammar approach (Bārzdiņš et al., 2007; Pretkalniņa et al., 2011). Other computational grammars of Latvian have been crafted for the needs of various machine translation systems (Skadiņa et al., 2007; Greitāne, 1997) and grammar checking tools (Deksne & Skadiņš, 2011). However, there has been no general-purpose wide-coverage computational grammar available for generating Latvian sentences.

Although dependency-based grammars allow for robust and effective parsing they lack the potential of language generation. This is the strength of phrase structure grammars, e.g. categorial grammars that link the surface structure with the underlying semantic representation. Among other features, GF essentially is an effective implementation of the categorial grammar approach.

## 2. Grammatical Framework

GF facilitates reusability by splitting the grammar development in two levels:

1. a general purpose *resource grammar* that covers a wide range of morphological features and syntactic structures,
2. and domain specific *application grammars* defining semantic structures and the subset of natural language that is used within a particular domain.

This allows developing and testing of the morphological and syntactic complexity once, which can be afterwards reused in multiple domains and in different usage scenarios without in-depth knowledge about the particular language and without the need to implement a large list of nuanced exceptional cases. The use-cases are ranging from controlled languages (e.g. dialogue systems and interfaces to formal languages) to domain-specific machine translation applications (e.g. speech-to-speech travel assistants).

GF differentiates not only between general-purpose resource grammars and domain-specific application grammars, but also between abstract syntax and concrete syntax. The abstract syntax captures the semantically relevant structure of language, defining grammatical categories and functions for building trees (Ranta, 2011). Concrete syntax defines the linearization of the abstract tree structures at the surface level. Translation among languages (concrete grammars) is provided via abstract syntax trees.

Note that in the GF grammar development there is no

concept of a language pair or a translation direction. Also there is no common semantic interlingua. Instead there are many application- and domain-specific interlinguas, and the concrete syntax can be built (but not necessarily) on top of the common resource grammar API.

The GF resource grammar library (Ranta, 2009), or RGL for short, currently supports more than 20 languages[1] that implement the common API. Latvian is among 13 (out of 23) official EU languages that are supported.

The common API specifies about 60 hierarchical grammatical categories and nearly 500 syntactic construction functions (including structural words and parameters used in the abstract trees)[2]. The large number of functions is still manageable from the application grammar developer perspective: due to extensive overloading, most of the functions are arranged in about 35 overload groups. Apart from the syntactic functions, there are also about 15 groups of lexical construction functions (the exact number of overloaded paradigms varies among languages; see Table 1 for a simplified example).

## 3. Morphology

Morphology plays an important part in grammatical analysis of Latvian, as there are many[3] inflected wordforms possible for each lemma: about 10 noun/pronoun forms, about 40 verb forms (excluding about 160 participle forms whose syntactic function is that of adjectives), and more than 100 adjective forms. Still, a lot of analytical wordforms are also used (e.g. analytical verb forms and prepositional phrases).

We have developed a GF morphology module for the full Latvian language by transforming and improving a previously developed morphological analyzer (Paikens, 2007) to the GF language, taking into account the language generation aspects. In particular, we have implemented a set of functions that detail the lemmatization and palatalization that occurs in Latvian, and an exhaustive list of word ending tables used in each paradigm. In the result, the Latvian GF morphology module and the analyzer by Paikens (2007) are quite different from the application point of a view. The latter one is designed as a highly robust analyzer for maximum coverage of an unrestricted text and is not appropriate for the generation needs as it suffers from overgeneration. However, the GF module is designed for high precision within a known lexical domain.

In Table 1, a simplified inflectional paradigm for Latvian nouns of the 5th declension is given along with the corresponding tiny fragment from the abstract grammar. A similar approach has been used for implementing morphology in GF for other inflective languages, e.g. Russian[4] (Khegai, 2006).

All the possible wordforms (linearizations) of a particular Latvian noun are given in Table 2 along with possible linearizations of the corresponding English noun.

Note that in the public API, the specific internal functions (operations) that deal with the lexical paradigms are hidden by overloaded functions (e.g. `mkN` in the case of nouns).

Common abstract grammar: categories

```
cat N ;
```

Latvian resource grammar: the morphology module

```
param
  Number = Sg | Pl ;
  Gender = Masc | Fem ;
  Case = Nom | Gen | Dat | Acc | Loc ;
  Declension = D1 | D2 | D3 | D4 | D5 | ... ;

oper
  Noun : Type = {
    s : Number => Case => Str ;
    g : Gender
  } ;

  mkNoun : Str -> Noun = \lemma ->
    let decl : Declension = case lemma of {
      ...
      s + "e" => D5 ; -- usually
      ...
    } in mkNoun_Decl lemma decl ;

  mkNoun_Decl : Str -> Declension -> Noun =
    \lemma,decl -> case decl of {
      ...
      D5 => mkNoun_D5 lemma ;
      ...
    } ;

  mkNoun_D5 : Str -> Noun = \lemma ->
    let stem : Str = cutStem lemma
    in {
      s = table {
        Sg => table {
          Nom => stem + "e" ;
          Gen => stem + "es" ;
          Dat => stem + "ei" ;
          Acc => stem + "i" ;
          Loc => stem + "ē"
        } ;
        Pl => table {
          Nom => stem + "es" ;
          Gen => palatalize stem + "u" ;
          Dat => stem + "ēm" ;
          Acc => stem + "es" ;
          Loc => stem + "ēs"
        }
      } ;
      g = Fem
    } ;
```

Latvian resource grammar: API

```
oper mkN = overload {
  mkN : (s : Str) -> N = \n -> lin N (mkNoun n) ;
  mkN : (s : Str) -> Declension -> N = \n,d ->
    lin N (mkNoun_Decl n d) ;
} ;
```

Table 1: A simplified fragment of RGL.

[1] http://www.grammaticalframework.org/lib/doc/status.html

[2] http://www.grammaticalframework.org/lib/doc/synopsis.html

[3] If compared to analytical languages like English or Scandinavian languages.

[4] In terms of grammar, the Slavic language group is the closest branch to the Baltic language group.

| Domain-specific lexicon: abstract |
|---|
| `fun sun_N : N ;` |
| Domain-specific lexicon: Latvian |
| `lin sun_N = mkN "saule" ;` |
| Domain-specific lexicon: English |
| `lin sun_N = mkN "sun" ;` |
| Parsing into the abstract categories |
| `>> parse -lang=Lav "sauļu"`<br>`sun_N`<br><br>`>> parse -lang=Eng "suns'"`<br>`sun_N` |
| Generating the full inflectional paradigms |
| `>> linearize -lang=Lav -table sun_N`<br>`s Sg Nom : saule`<br>`s Sg Gen : saules`<br>`s Sg Dat : saulei`<br>`s Sg Acc : sauli`<br>`s Sg Loc : saulē`<br>`s Pl Nom : saules`<br>`s Pl Gen : sauļu`<br>`s Pl Dat : saulēm`<br>`s Pl Acc : saules`<br>`s Pl Loc : saulēs`<br><br>`>> linearize -lang=Eng -table sun_N`<br>`s Sg Nom : sun`<br>`s Sg Gen : sun's`<br>`s Sg Acc : sun`<br>`s Pl Nom : suns`<br>`s Pl Gen : suns'`<br>`s Pl Acc : sun` |

Table 2: A sample domain lexicon (a part of an application grammar): its definition and usage.

## 4. Syntax

We have implemented grammar rules for all the common phrase structures in the conventional style of categorial grammars, basically: noun phrases with agreement rules for adjectives and other modifiers, adjective phrases, and verb phrases with the relevant complements. On one hand, the implemented rules cover only the most common (neutral) ways of expressing these phrases (in terms of word order), excluding several alternative word orderings that are occasionally used for special emphasis (e.g. to indicate the given vs. new information) or for poetic reasons. On the other hand, the grammar includes syntactic construction rules for a full range of dependent clauses and participle clauses used in Latvian language, thus ensuring a wide coverage for generating natural, complex sentences.

In essence, this approach models a subset of the full natural language which relies on rich lexical information about words in a specific domain and on a grammatically correct standard language, gaining high precision while accepting a lower recall rate if analyzing an unrestricted text. From the language generation point of a view, the design goal is that it should be possible to express every valid structure in the most common way, i.e., in the natural/neutral word order – but not necessarily in all the possible word orderings, as there is no well-defined model (for Latvian) for the exact semantic nuances transferred by alternative word order in a more or less unrestricted text[5].

### 4.1 Clauses

We treat clauses as elements that specify actions – a verb with its arguments – but leaves unspecified the way in which the actions are described. Traditional Latvian linguistics describes clauses in terms of moods and tenses. There are infinitive, indicative, relative[6], debitive[7] and imperative moods, as well as few subtypes of some of them and several types of participles. The relative and debitive moods are Latvian-specific and are used to express the reported speech and necessity or requirement accordingly.

In general, every action can be expressed in any of these moods by using different synthetic verb forms. In the case of a perfect tense, analytical verb forms are used. We have implemented the full set of mood, tense and polarity combinations used in Latvian language, some examples of which are illustrated in Table 3.

| Parameters | Example | Translation |
|---|---|---|
| `Indicative Present` | zāle ir zaļa | grass is green |
| `Indicative Past` | zāle bija zaļa | grass was green |
| `Indicative Anterior Present` | zāle ir bijusi zaļa | grass has been green |
| `Relative Simultaneous Present` | zāle esot zaļa | [one says that] grass is green |
| `Debitive Simultaneous Present` | zālei jābūt zaļai | grass has to be green |
| `Conditional Simultaneous` | zāle būtu zaļa | grass would be green |
| `Relative Anterior Negated` | zāle neesot bijusi zaļa | [one says that] grass has not been green |

Table 3: Examples of mood, tense and polarity variation in Latvian.

The basic abstract (language-independent) syntax used in GF RGL is based on a narrow view of tenses (present, past, future and conditional). This limits the easily (synthetically) available variety in generation of Latvian sentences. In a standard resource grammar, at the sentence level, the verb phrases are used in the indicative mood, however, keeping the other types of moods integrated allows us to reuse the same verb phrase

[5] Although, in the case of a highly controlled Latvian, there is a deterministic model defined by Grūzītis (2010).

[6] http://www.isocat.org/datcat/DC-3836

[7] http://www.isocat.org/datcat/DC-3835

constructing functions in application grammars that need the additional means of expression. This helps also when translating specific (structural) verbs such as 'must', 'might' or 'said' – in a proper translation to Latvian it is often necessary to modify the mood of the dependent clause governed by these verbs instead of including the literal translation of these verbs.
The API interface provided by the resource grammar is as follows:

1. Function `mkCl` (make clause), parameterised by the subject, core verb and any appropriate complements. For example, "`mkCl John_N give_V2 key_N Mary_N`" generates clauses that correspond to all combinations of tense and polarity for using in different kinds of sentences: "*John gives a key to Mary*", "*John has not given a key to Mary*", "*will John give a key to Mary*" etc.
2. Functions to apply such clauses – parameterised by tense, anteriority and polarity. For example, by applying "`mkS pastTense simultaneousAnt positivePol`" to the previously defined clause the specific declarative sentence "*John gave a key to Mary*" is generated.
3. Helper functions for building incomplete clauses that may be needed to form questions, imperative sentences or subclauses.

This structure enforces a clean separation between the actual predicate that is described, and the way in which it is described in a narrative. For example, an application may need to refer to the same action multiple times: first, (hypothetically) to request a confirmation from a user, and afterwards to refer to it as a completed action, requiring a completely different syntactic structure.
In the practical development of user interfaces in Latvian this is almost always done in an unsophisticated way, using simple declarative sentences where the correct wordform can be built easily by regular expressions or similar methods. This results in sentence structures that look clumsy to users, because humans would commonly use a more complicated structure with subclauses.
Such a resource grammar allows applications to express a particular clause once in a standardised way, and then use it in various forms or combine it in complex sentence structures without dealing with the rather complex rules of inflection, agreement and structural changes when using it as a subclause.

## 4.2. Verb phrases

The agreement rules for complements of multi-argument verbs are implemented by specifying the syntactic valences of each verb – the case or preposition that the relevant complement must or may have.
This presents a challenge for implementing a practical system for Latvian in relatively unrestricted language domains with large lexicons, as currently there is no publicly available syntactic valence dictionary for Latvian, and thus all such verbs would need to be defined manually instead of importing them from some database of verbs with appropriate morpho-syntactic information. However, if (application) grammar users define syntactic valences of verbs that are appropriate to the specific domain, it gives an opportunity to specify (at the same time) also semantic valences, so that the role of each complement can be obtained from the case (or preposition) used, allowing to integrate the grammar with frame semantics, e.g. with the data of FrameNet (Fillmore et al., 2003), or to map the verb valences to domain-specific predicate parameters.
In any case, this lexical information is necessary to ensure correct analysis or synthesis, as verb complement roles (both syntactic and semantic) are mainly defined by their case or preposition. In Table 4 we illustrate this valence mapping of semantic and syntactic roles for three related verbs.

(a) *saņemt* (to receive):

| Sem. role | Latvian | English |
|---|---|---|
| `Recipient` | `Nominative` | `Subject` |
| `Theme` | `Accusative` | `Object-1` |
| `Donor` | `"no" ++ Genitive` | `"from" ++ Object-2` |

*Mērija saņem atslēgu no Jāņa – Mary receives a key from John*

(b) *vajadzēt* (to need):

| Sem. role | Latvian | English |
|---|---|---|
| `Recipient` | `Dative` | `Subject` |
| `Theme` | `Accusative` | `Object-1` |
| `Donor` | `"no" ++ Genitive` | `"from" ++ Object-2` |

*Mērijai vajag atslēgu no Jāņa – Mary needs a key from John*

(c) *dot* (to give):

| Sem. role | Latvian | English |
|---|---|---|
| `Donor` | `Nominative` | `Subject` |
| `Theme` | `Accusative` | `Object-1` |
| `Recipient` | `Dative` | `"to" ++ Object-2` |

*Jānis dod atslēgu Marijai – John gives a key to Mary*

Table 4: Syntactic and semantic role mappings

Note that the examples given in Table 4 correspond to the neutral word order, but the other possible orderings that preserve the same morphological features are also valid in Latvian: "*Mērija no Jāņa saņem atslēgu*", "*no Jāņa atslēgu Mērija saņem*" etc. They convey virtually the same meaning, but the information structure (topic and focus) is different, affecting the further discourse analysis (Grūzītis, 2010).
The syntactic information specific to each of the (a), (b) and (c) verbs in Table 4 is necessary both to choose the proper complement wordform in language generation, and to determine the subject while parsing a sentence. This also means that in the case of verbs that are classified as three-place verbs some complements can be (and often are [8]) omitted while still keeping clear valences.

[8] Preliminary corpus analysis of Latvian verb valences indicates that in about 30% cases one or multiple frame elements are omitted.

This property is relevant to other languages as well[9], and the current GF approach of classifying verbs according to the number of arguments is not sufficient in the long term, especially in the multilingual environment where the syntactic realization of the same verb (concept) can be different across languages.

### 4.3 Noun and adjective phrases

Noun and adjective phrases are implemented in a straightforward manner as it is typical for inflective languages – the phrase constituent relations are determined from agreement of morphological features.

The treatment of determiners is somewhat interesting: definite and indefinite articles are not used in Latvian, and, in general, there is no difference between definite and indefinite noun phrases (at the surface level). A noun phrase might include an indefinite or demonstrative pronoun, or an adjective that have distinct definite and indefinite forms, however, the given and new information is often indicated implicitly – by rather systematic changes in the neutral word order (Grūzītis, 2010). These formal features can be exploited to ensure the proper translation in a multilingual application. In this regard, the definiteness property is tracked in noun phrases in order to determine the agreement between a noun and an adjective or a participle.

In Latvian, an attribute of a noun can be easily transformed into a (comma-delimited) attributive subclause or vice versa (in most cases). The resource grammar includes full support for deep nesting of such subclauses as they are typically used, for example, in legal texts.

## 5. Applications

GF has been used for a logic-based Latvian-English application grammar even before the Latvian resource grammar was available, creating a prototype for authoring and verbalizing OWL ontologies in controlled Latvian via Attempto Controlled English and its readily available infrastructure (Grūzītis & Bārzdiņš, 2011; Fuchs et al., 2008). Now it is possible to extend this research on the basis of the resource grammar library and on the basis of the work by Angelov & Ranta (2010).

However, the provided resource grammar is suitable also for significantly less controlled applications if the interpretation is left to the user, e.g. for tourist phrasebooks as demonstrated by Ranta et al. (2012).

Language generation facilities can be used to easily construct grammatically correct and natural sentences (or even a text) in various end-user interfaces: from simple use-cases like proper handling of named entities up to automatic verbalization of database query results or in hybrid machine translation systems (see the deliverables of the MOLTO project[10] for an example).

It should be emphasised that the limitations that are imposed by the RGL API are present only if we want to exploit the readily available multilingual parsing and generation facilities. For single-language applications it is possible to extend the resource grammar without preserving full compatibility with the shared API. For instance, the current system could be adapted for parsing texts in a weakly controlled language, e.g. legal documents. Furthermore, Angelov (2011) has demonstrated the potential of the current GF resource grammar library in statistical partial parsing of unrestricted texts.

Our future work is aimed at adaptation of the Latvian resource grammar and at creation of a reusable Latvian GF lexicon in order to enable semantic parsing of multi-domain texts. I.e., we are aiming at integration of the current approach with the frame semantics approach so that the semantic valences of a verb would be taken into account[11]. However, this would require significant modifications not only in the Latvian resource grammar, but also in the abstract syntax and to the current principles of building GF lexicons.

## 6. Conclusion

We have implemented a computational grammar for Latvian that works equally well for parsing and language generation. It is available as an open-source distribution in the GF release 3.3.3 and is available for download from the GF source code repository or as a part of binary packages[12]. Compiled GF application grammars are suitable for inclusion in third-party applications on various platforms.

For the developers of GF RGL modules for other languages, it may be interesting to note the discrepancies between the current resource grammar API and its implementation for Latvian. While the morphological layer is completely language-dependent, the sharing of common syntactic structures to some extent limits the resource grammar development and applicability in order to ensure the compatibility (transferability) among the languages. Our impression is that the current language-independent API is still rather biased towards peculiarities of English, and that it may be worthwhile to summarize the issues for all language implementations to identify the common limitations.

While we lack the knowledge to summarize the situation for all languages supported by the RGL, our experiments with Latvian-English-Russian parallel grammars suggest that development of accurate robust multilingual systems will eventually require including additional details in the abstract syntax layer of the RGL. Notably, we would recommend to replace the 'n-place' verb classification with more structured valence data, and to extend the common tense and mood system.

## Acknowledgements

This work has been supported by the European Regional Development Fund under the project No. 2011/0009/

[9] For example, Khegai (2006) mentions similar issues.

[10] http://www.molto-project.eu/

[11] There is an ongoing work developing a valence dictionary for the most frequently used verbs in Latvian (Nešpore & Saulīte, 2012).

[12] http://www.grammaticalframework.org/

2DP/2.1.1.1.0/10/APIA/VIAA/112. The authors would like to thank Aarne Ranta for his helpful hints on the implementation details, and the anonymous reviewers for their suggestions on how to improve this paper.

## References

Angelov, K. (2011). *The Mechanics of the Grammatical Framework*. PhD Thesis. Chalmers University of Technology and University of Gothenburg.

Angelov, K., Ranta, A. (2010). Implementing controlled languages in GF. In N.E. Fuchs (Ed.), *Controlled Natural Language (CNL 2009)*, Lecture Notes in Computer Science, Vol. 5972, Springer, pp. 82–101

Bārzdiņš, G., Grūzītis, N., Nešpore, G., Saulīte, B. (2007). Dependency-Based Hybrid Model of Syntactic Analysis for the Languages with a Rather Free Word Order. In *Proceedings of the 16th Nordic Conference on Computational Linguistics (NODALIDA 2007)*, Tartu, pp. 13–20

Deksne, D., Skadiņš, R. (2011). CFG Based Grammar Checker for Latvian. In *Proceedings of the 18th Nordic Conference on Computational Linguistics (NODALIDA 2011)*, Riga, pp. 275–278

Fillmore, C.J., Johnson, C.R., Petruck, M.R.L. (2003). Background to FrameNet. *International Journal of Lexicography*, 16, pp. 235–250

Fuchs N.E., Kaljurand K., Kuhn T. (2008). Attempto Controlled English for Knowledge Representation. In *Proceedings of the 4th International Reasoning Web Summer School*, Lecture Notes in Computer Science, Vol. 5224, Springer, pp. 104–124

Greitāne, I. (1997). Mašīntulkošanas sistēma LATRA (The Machine Translation System LATRA). *Proceedings of the Latvian Academy of Sciences*, Section A, 51 (3/4), pp. 1–6

Grūzītis, N., Bārzdiņš, G. (2011). Towards a More Natural Multilingual Controlled Language Interface to OWL. In *Proceedings of the 9th International Conference on Computational Semantics (IWCS 2011)*, Oxford, pp. 335–339

Grūzītis, N. (2010). Word Order Based Analysis of Given and New Information in Controlled Synthetic Languages. In *Proceedings of the Workshop on the Multilingual Semantic Web* (at WWW 2010), Raleigh, CEUR Workshop Proceedings, Vol. 571, pp. 29–34

Khegai, J. (2006). GF parallel resource grammars and Russian. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL 2006)*, Sydney, pp. 475–482

Nešpore, G., Saulīte, B. (2012). Verbu valences apraksta iespējas latviešu valodā. In *Valoda: nozīme un forma. Teorija un metodoloģija latviešu valodniecībā*, Rīga: LU Akadēmiskais apgāds (to appear)

Paikens, P. (2007). Lexicon-Based Morphological Analysis of Latvian Language. In *Proceedings of the 3rd Baltic Conference on Human Language Technologies (Baltic HLT 2007)*, Kaunas, pp. 235–240

Pretkalniņa, L., Nešpore, G., Levāne-Petrova, K., Saulīte, B. (2011). A Prague Markup Language Profile for the SemTi-Kamols Grammar Model. In *Proceedings of the 18th Nordic Conference on Computational Linguistics (NODALIDA 2011)*, Riga, pp. 303–306

Ranta, A., Enache, R., Détrez, G. (2012). Controlled Language for Everyday Use: the MOLTO Phrasebook. In N.E. Fuchs, M. Rosner (Eds.), *Proceedings of the 2nd Workshop on Controlled Natural Language (CNL 2010)*, Lecture Notes in Computer Science, Vol. 7175, Springer (to appear)

Ranta, A. (2011). *Grammatical Framework: Programming with Multilingual Grammars*. Stanford: CSLI Publications

Ranta, A. (2009). The GF Resource Grammar Library. *Linguistic Issues in Language Technology*, 2 (2)

Skadiņa, I., Skadiņš, R., Deksne, D., Gornostaja, T. English/Russian-Latvian Machine Translation System. In *Proceedings of the 3rd Baltic Conference on Human Language Technologies (Baltic HLT 2007)*, Kaunas, pp. 287–295