A Multilingual FrameNet-based Grammar and Lexicon for Controlled Natural Language

Formalising the Swedish Constructicon in GF

Normunds Grūzītis
University of Gothenburg, Department of Computer Science and Engineering
University of Latvia, Institute of Mathematics and Computer Science

4th GF Summer School
Gozo, Malta, 13–24 July 2015
Agenda

• FrameNet
  – Aim and background
  – Extraction of semantico-syntactic verb valence patterns from FrameNet-annotated corpora
  – Generation of a FrameNet-based GF grammar and lexicon
  – Case study
  – Results

• Constructicon
  – Aim and background
  – Conversion of SweCcn into GF
  – Results
FrameNet (FN)

- A lexico-semantic resource based on the theory of frame semantics (Fillmore et al. 2003)
  - A semantic frame represents a cognitive, prototypical situation (scenario) characterized by frame elements (FE) – semantic valence
  - Frames are “evoked” in sentences by target words – lexical units (LU)
  - FEs are mapped based on the syntactic valence of the LU
    - The syntactic valence patterns are derived from FN-annotated corpora (for an increasing number of languages)
  - FEs are split into core and non-core ones
    - Core FEs uniquely characterize the frame and syntactically tend to correspond to verb arguments
    - Non-core FEs are not specific to the frame and typically are adjuncts
BFN and SweFN

- Our experiment is based on two FNs: the original Berkeley FrameNet (BFN) and the Swedish FrameNet (SweFN)
  - We consider only those frames for which there is at least one corpus example where the frame is evoked by a verb

- BFN 1.5 (2010) defines 1,020 frames of which 559 are evoked by 3,254 verb LUs in 69,260 annotated sentences

- A SweFN development version (Dec 2014) covers 995 frames of which 660 are evoked by 2,887 verb LUs in 4,400 sentences

- SweFN, like many other FNs, mostly reuses BFN frames, hence, BFN frames can be seen as a semantic interlingua
  - A linguistically motivated ontology
### Example frame

**Desiring**

<table>
<thead>
<tr>
<th>Definition:</th>
<th>An EXPERIENCER desires that an EVENT occur. In some cases, the EXPERIENCER is an active participant in the EVENT, and in such cases the EVENT itself is often not mentioned, but rather some FOCAL_PARTICIPANT which is subordinately involved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core FEs:</td>
<td>EVENT, EXPERIENCER, FOCAL_PARTICIPANT, LOCATION_OF_EVENT</td>
</tr>
<tr>
<td>Non-core FEs:</td>
<td>CAUSE, DEGREE, DURATION, MANNER, PLACE, PURPOSE_OF_EVENT, REASON, ROLE_OF_FOCAL_PARTICIPANT, TIME, TIME_OF_EVENT</td>
</tr>
</tbody>
</table>

Some valence patterns found in **BFN**

<table>
<thead>
<tr>
<th>Examples</th>
<th>Valence patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>40 (22)</td>
<td>EVENT EXPERIENCER VPto.Dep NP.Ext</td>
</tr>
<tr>
<td>14 (10)</td>
<td>EXPERIENCER FOCAL_PARTICIPANT NP.Ext NP.Obj PP[by].Dep NP.Ext</td>
</tr>
</tbody>
</table>

E.g. “[I] EXPERIENCER do n’t WANT [to deceive anyone] EVENT”  
| an embedded frame |

Some valence patterns found in **SweFN**

<table>
<thead>
<tr>
<th>Examples</th>
<th>Valence patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (1)</td>
<td>EVENT EXPERIENCER VB.INF.VG NN.SS</td>
</tr>
<tr>
<td>2 (1)</td>
<td>EXPERIENCER FOCAL_PARTICIPANT NN.SS NN.OO</td>
</tr>
</tbody>
</table>

E.g. “[Jag] EXPERIENCER KÄNNER FÖR [en tur på landet] FOCAL_PARTICIPANT”
FrameNet and GF

• Existing FNs are not entirely formal and computational
  – We provide a limited but computational FN-based grammar and lexicon

• Grammatical Framework:
  – Separates between an abstract syntax and concrete syntaxes
  – Provides a general-purpose resource grammar library (RGL)
    • Large mono- and multilingual lexicons (for an increasing number of languages)

• The language-independent layer of FrameNet (frames and FEs) – the abstract syntax
  – The language-specific layers (surface realization of frames and FEs; LUs) – concrete syntaxes

• RGL can be used for unifying the syntactic types used in different FNs and for the concrete implementation of frames
  – FrameNet allows for abstracting over RGL
Relation to CNL

- Kuhn (2014) defines Controlled Natural Language (CNL) as “a constructed language that is based on a certain natural language, being more restrictive concerning lexicon, syntax, and/or semantics, while preserving most of its natural properties”

- We deviate from this definition in two aspects:
  - Our intention is to produce a reusable grammar that covers a restricted subset of NL instead of a grammar of a predefined constructed language
  - We produce a currently bilingual but potentially multilingual grammar library which is therefore not based on exactly one NL but inherently has a shared semantic abstract syntax

- Thus, we do not provide a CNL as such but a high-level API for the facilitation of the development of CNL grammars, making them more flexible – easier to modify and extend

- In a sense, we aim at bridging the gap between CNL and NL
Specific aim (1)

• Provide a **semantic** API on top of RGL to facilitate the development of GF application grammars
  – In combination with the **syntactic** API of RGL
  – Hiding the comparatively complex construction of **verb phrases**

\[
\text{mkCl} \quad \text{person} \quad (\text{mkVP} \quad (\text{mkVP} \quad \text{live}_V) \quad (\text{mkAdv} \quad \text{in}_\text{Prep} \quad \text{place}))
\]

\[
\quad -- \quad \text{mkCl} : \text{NP} \rightarrow \text{VP} \rightarrow \text{Cl}
\quad -- \quad \text{mkVP} : \text{V} \rightarrow \text{VP}
\quad -- \quad \text{mkVP} : \text{VP} \rightarrow \text{Adv} \rightarrow \text{VP}
\quad -- \quad \text{mkAdv} : \text{Prep} \rightarrow \text{NP} \rightarrow \text{Adv}
\]

**Residence**

\[
\quad \text{person} \quad (\text{mkAdv} \quad \text{in}_\text{Prep} \quad \text{place})
\quad \text{live}_V\text{Residence}
\]

\[
\quad -- \quad \text{Residence} : \text{NP} \rightarrow \text{Adv} \rightarrow \text{V} \rightarrow \text{Cl}
\quad -- \quad \text{NP} \ (\text{Resident})
\quad -- \quad \text{Adv} \ (\text{Location})
\quad -- \quad \text{V} \ (\text{LU})
\]
### Specific aim (2)

- FN-annotated knowledge bases $\rightarrow$ multilingual verbalization

<table>
<thead>
<tr>
<th>Subject</th>
<th>Place</th>
<th>Time</th>
<th>Relatives</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Being born</td>
<td>Slokas pagasts</td>
<td>1933. gada 3. maijs</td>
<td>zvejnieka ģimene</td>
<td>Imants Ziedonis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Institution</th>
<th>Subject</th>
<th>Time</th>
<th>Place</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>Education teaching</td>
<td>Latvijas Universitāte</td>
<td>vēsture un filoloģija</td>
<td>1959. gads</td>
<td>Imants Ziedonis</td>
</tr>
<tr>
<td>Education teaching</td>
<td>Augstākais literārais</td>
<td>1964. gads</td>
<td>Maskava</td>
<td>Imants Ziedonis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Employment</th>
<th>Place_of_employment</th>
<th>Position</th>
<th>Time</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Being employed</td>
<td>izdevniecība Liesma</td>
<td>&gt; redaktors</td>
<td>1983. gads</td>
<td>Imants Ziedonis</td>
</tr>
<tr>
<td>Being employed</td>
<td>Latvijas raskstnieku</td>
<td>&gt; sekretārs</td>
<td>1972. gads</td>
<td>Imants Ziedonis</td>
</tr>
<tr>
<td>Being employed</td>
<td>AP tautas izglītības</td>
<td>&gt; loceklis</td>
<td>1998. gads</td>
<td>Imants Ziedonis</td>
</tr>
<tr>
<td>Being employed</td>
<td>Latvijas Institūts</td>
<td>&gt; padomnieks</td>
<td>1997. gads</td>
<td>Imants Ziedonis</td>
</tr>
<tr>
<td>Being employed</td>
<td>Jūrmalas 1. vidusskola</td>
<td>&gt; skolotājs</td>
<td>1991. gada barikāžu</td>
<td>Imants Ziedonis</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event</th>
<th>Time</th>
<th>Prize</th>
<th>Rank</th>
<th>Competition</th>
<th>Competitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Win_prize</td>
<td>1983. gads</td>
<td>Tautu draudzības</td>
<td></td>
<td>Imants Ziedonis</td>
<td></td>
</tr>
<tr>
<td>Win_prize</td>
<td>1972. gads</td>
<td>Nopelniem bagātais</td>
<td></td>
<td>Imants Ziedonis</td>
<td></td>
</tr>
<tr>
<td>Win_prize</td>
<td>1977. gads</td>
<td>Tautas dzejnieka goda</td>
<td></td>
<td>Imants Ziedonis</td>
<td></td>
</tr>
<tr>
<td>Win_prize</td>
<td>1991. gada barikāžu</td>
<td></td>
<td></td>
<td>Imants Ziedonis</td>
<td></td>
</tr>
</tbody>
</table>

**Imants Ziedonis ir dzimis 1933. gada 3. maijā Slokas pagastā.**

**Imants Ziedonis was born in Sloka parish on 3 May 1933.**
Outline

Semantic interlingua: shared frames and FE

- Berkeley FrameNet
  - BFN-annotated corpus

- Swedish FrameNet
  - SweFN-annotated corpus

- [...] FrameNet
  - [...]-annotated corpus

Shared semantico-syntactic valence patterns

- Eng concrete syntax
- Swe concrete syntax
- [...] concrete syntax

FN-based grammar: shared abstract syntax

- Syntactic interlingua: shared abstract syntax of GF RGL
  - Eng resource grammar
  - Swe resource grammar

- Application grammar: CNL interface to a KB
  - [...] resource grammar
Extraction of frame valence patterns

• Valence patterns that are **shared** between FNs (currently, BFN and SweFN)
  – Multilingual applications
  – Cross-lingual validation

• Currently, only core FEs that make the frames unique

• Example: some shared patterns of the frame *Desiring*
  – \textbf{Desiring}/V\textsubscript{Act} Experiencer/NP\textsubscript{Subj} Focal\_participant/Adv
    \[\text{e.g., } [\textit{Dexter}]_{\text{Experiencer}} \text{ [YEARNED]} \text{ [for a cigarette]}_{\text{Focal\_participant}}\]

  – \textbf{Desiring}/V2\textsubscript{Act} Experiencer/NP\textsubscript{Subj} Focal\_participant/NP\textsubscript{DObj}
    \[\text{e.g., } [\textit{she}]_{\text{Experiencer}} \text{ [WANTS]} \text{ [a protector]}_{\text{Focal\_participant}}\]

  – \textbf{Desiring}/VV\textsubscript{Act} Event/VP Experiencer/NP\textsubscript{Subj}
    \[\text{e.g., } [\textit{I}]_{\text{Experiencer}} \text{ \textit{would n’t}} \text{ [WANT]} \text{ [to know]}_{\text{Event}}\]

• The uniform patterns contain sufficient info for generating the grammar
1. Language- and FN-specific processing

- Different XML schemes, POS tagsets and syntactic annotations
- Rules and heuristics for generalizing to RGL types, and for deciding the syntactic roles
- A lot of automatic annotation errors → heuristic correction (partial)
2. Extracted sentence patterns (BFN)

Desiring Act Experiencer_NP.Subj Event_VP long.v
Desiring Act Experiencer_NP.Subj Event_VP Opt_Reason_Adv aspire.v
Desiring Act Experiencer_NP.Subj Opt_Time_Adv Event_VP fancy.v
Desiring Act Experiencer_NP.Subj Event_VP want.v
Desiring Act Experiencer_NP.Subj Event_VP yearn.v
Desiring Act Experiencer_NP.Subj Experiencer_NP.Subj Event_VP aspire.v
Desiring Act Experiencer_NP.Subj Event_NP.DObj want.v
Desiring Act Experiencer_NP.Subj Event_S desire.v
Desiring Act Experiencer_NP.Subj Focal_participant_Adv[after] yearn.v
Desiring Act Experiencer_NP.Subj Focal_participant_Adv[for] yearn.v
Desiring Act Experiencer_NP.Subj Focal_participant_Adv[for] yearn.v
Desiring Act Experiencer_NP.Subj Focal_participant_NP.DObj want.v
Desiring Act Focal_participant_NP.DObj Experiencer_NP.Subj crave.v
Desiring Act Focal_participant_NP.DObj want.v
Desiring Pass Focal_participant_NP.Subj Experiencer_NP.DObj desire.v
Desiring Pass Focal_participant_NP.Subj Experiencer_NP.DObj want.v
3. Summarized valence patterns (BFN)

Desiring : 288
  Act : 275
  Event_VP Experiencer_NP : 61
    Experiencer_NP.\textit{Subj} Event_VP : 59
    Event_VP Experiencer_NP.\textit{Subj} : 2

Experiencer_NP Focal\_participant_NP : 61
  Experiencer_NP.\textit{Subj} Focal\_participant_NP.\textit{DObj} : 55
  Focal\_participant_NP.\textit{DObj} Experiencer_NP.\textit{Subj} : 6

Experiencer_NP Focal\_participant\_Adv : 43
  Experiencer_NP.\textit{Subj} Focal\_participant\_Adv[\textit{for}] : 26
  Experiencer_NP.\textit{Subj} Focal\_participant\_Adv[\textit{after}] : 7
  Experiencer_NP.\textit{Subj} Focal\_participant\_Adv : 2
  ...

Pass : 13
  Experiencer\_NP Focal\_participant\_NP : 5
    Focal\_participant\_NP.\textit{Subj} Experiencer\_NP.\textit{DObj} : 5
    ...

- Normalized, ignoring the word order and prepositions (or cases)
- For the abstract syntax, we consider only the normalized patterns
- For the concrete syntax – the most frequent sentence pattern of each normalized pattern
4. Pattern comparison by subsumption

• To find a representative yet condensed set of shared patterns

• Pattern $A$ subsumes pattern $B$ if:
  – $A.frame = B.frame$
  – $\text{type}(A.LU) = \text{type}(B.LU)$
  – $A.voice = B.voice$
  – $B.FEs \subseteq A.FEs$ (incl. the syntactic types and roles)

• If $A$ subsumes $B$ and $B$ subsumes $A$ then $A = B$

• If a pattern of $\text{FN}_1$ is subsumed by a pattern of $\text{FN}_2$, it is added to the shared set (and vice versa)
  – In the final set, patterns that are subsumed by other patterns are removed

P1: $\text{Apply\_heat} \ V2 \ \text{Act} \ \text{Cook\_NP.\_Subj\_Food\_NP.\_DObj}$
P2: $\text{Apply\_heat} \ V2 \ \text{Act} \ \text{Cook\_NP.\_Subj\_Container\_Adv\_Food\_NP.\_DObj}$
P3: $\text{Apply\_heat} \ V2 \ \text{Act} \ \text{Food\_NP.\_DObj}$

P1 is subsumed by P2, P3 is subsumed by P1, P2; P1 and P3 are to be removed
Experiment series

- To roughly estimate the impact of various choices made in the extraction process, we have run a series of experiments

<table>
<thead>
<tr>
<th>Settings</th>
<th>Frames</th>
<th>Valence patterns</th>
<th>Sentence patterns</th>
<th>Corpus examples</th>
<th>Frames</th>
<th>Valence patterns</th>
<th>Sentence patterns</th>
<th>Corpus examples</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>total</td>
<td>per frame</td>
<td>total</td>
<td>per valence pattern</td>
<td>total</td>
<td>per sentence pattern</td>
<td>total</td>
</tr>
<tr>
<td>0.0</td>
<td>556</td>
<td>19905</td>
<td>36</td>
<td>25696</td>
<td>1.3</td>
<td>68653</td>
<td>2.7</td>
<td>637</td>
</tr>
<tr>
<td>2.0</td>
<td>555</td>
<td>16479</td>
<td>30</td>
<td>24491</td>
<td>1.5</td>
<td>66322</td>
<td>2.7</td>
<td>631</td>
</tr>
<tr>
<td>2.A</td>
<td>551</td>
<td>14135</td>
<td>26</td>
<td>22115</td>
<td>1.6</td>
<td>65008</td>
<td>2.9</td>
<td>625</td>
</tr>
<tr>
<td>2.B</td>
<td>551</td>
<td>5493</td>
<td>10</td>
<td>8696</td>
<td>1.6</td>
<td>65103</td>
<td>7.5</td>
<td>625</td>
</tr>
<tr>
<td>3.0</td>
<td>506</td>
<td>6381</td>
<td>13</td>
<td>14393</td>
<td>2.3</td>
<td>56224</td>
<td>3.9</td>
<td>273</td>
</tr>
<tr>
<td>3.A</td>
<td>502</td>
<td>5968</td>
<td>12</td>
<td>13948</td>
<td>2.3</td>
<td>56841</td>
<td>4.1</td>
<td>266</td>
</tr>
<tr>
<td>3.B</td>
<td>508</td>
<td><strong>3481</strong></td>
<td><strong>7</strong></td>
<td><strong>6684</strong></td>
<td><strong>1.9</strong></td>
<td><strong>63091</strong></td>
<td><strong>9.4</strong></td>
<td><strong>423</strong></td>
</tr>
</tbody>
</table>

0.0: Extract sentence patterns using FN-specific syntactic types ("baseline")
1.0: Skip examples containing few currently unconsidered syntactic types
2.0: Generalize syntactic types according to RGL
3.0: Skip once-used valence patterns (e.g., to reduce the propagation of annotation errors)

x.A: Skip repeated FEs
x.B: Skip non-core FEs and repeated FEs

- In the result, we have extracted a set of **869** shared semantico-syntactic valence patterns covering **483** frames

P.S. The SweFN numbers are based on the Feb 2014 version
FrameNet-based grammar: abstract

- **Frame** valence patterns are represented by functions
  - Taking one or more core FEs (A-Z) and one LU as arguments
  - Returning an object of type *Clause* whose linearization type is
    \{np: NP; vp: VP\}

\[
\begin{align*}
\text{fun} & \quad \text{Desiring}_V & : & \text{Experiencer}_NP \rightarrow \text{Focal/participant}_Adv \rightarrow V \rightarrow \text{Clause} \\
\text{fun} & \quad \text{Desiring}_V2 & : & \text{Experiencer}_NP \rightarrow \text{Focal/participant}_NP \rightarrow V2 \rightarrow \text{Clause} \\
\text{fun} & \quad \text{Desiring}_V2\_Pass & : & \text{Experiencer}_NP \rightarrow \text{Focal/participant}_NP \rightarrow V2 \rightarrow \text{Clause} \\
\text{fun} & \quad \text{Desiring}_VV & : & \text{Event}_VP \rightarrow \text{Experiencer}_NP \rightarrow VV \rightarrow \text{Clause}
\end{align*}
\]

- **FEs** are declared as semantic categories subcategorized by the syntactic RGL types
  - *NP, VP, Adv* (includes prepositional objects), *S* (embedded sentences), *QS*

\[
\begin{align*}
\text{cat} & \quad \text{Event}_VP & : & \text{Focal/participant}_NP \\
\text{cat} & \quad \text{Experiencer}_NP & : & \text{Focal/participant}_Adv
\end{align*}
\]
The mapping from the semantic FrameNet types to the syntactic RGL types is shared for all languages – Linearization types are of type *Maybe* to allow for optional (empty) FEs

```haskell
lincat Focal_participant_NP   = Maybe NP
lincat Focal_participant_Adv  = Maybe Adv
```

To implement the frame functions, RGL **constructors** are applied to the arguments depending on their types and syntactic roles, and the voice FrameNet-based grammar:

```haskell
lin Desiring_V2 experiencer focal_participant v2 = {
    np = fromMaybe NP experiencer ;
    vp = mkVP v2 (fromMaybe NP focal_participant)
}
lin Desiring_V2_Pass experiencer focal_participant v2 = {
    np = fromMaybe NP focal_participant ;
    vp = mkVP (passiveVP v2) (mkAdv by8agent_Prep (fromMaybe NP experiencer))
}```
FrameNet-based grammar: concrete

<table>
<thead>
<tr>
<th>Verb</th>
<th>Voice</th>
<th>Arguments</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>V2</td>
<td>Act</td>
<td>NP_{dobj} NP_{nsubj}</td>
<td>277</td>
</tr>
<tr>
<td>V</td>
<td>Act</td>
<td>Adv NP_{nsubj}</td>
<td>155</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>NP_{nsubj}pass</td>
<td>84</td>
</tr>
<tr>
<td>V2</td>
<td>Act</td>
<td>Adv NP_{dobj} NP_{nsubj}</td>
<td>80</td>
</tr>
<tr>
<td>V</td>
<td>Act</td>
<td>NP_{nsubj}</td>
<td>78</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>Adv NP_{nsubj}pass</td>
<td>34</td>
</tr>
<tr>
<td>VS</td>
<td>Act</td>
<td>NP_{nsubj} S</td>
<td>29</td>
</tr>
<tr>
<td>VV</td>
<td>Act</td>
<td>NP_{nsubj} VP</td>
<td>21</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>NP_{dobj} NP_{nsubj}pass</td>
<td>19</td>
</tr>
<tr>
<td>V</td>
<td>Act</td>
<td>NP_{dobj}</td>
<td>17</td>
</tr>
<tr>
<td>VQ</td>
<td>Act</td>
<td>NP_{nsubj} QS</td>
<td>10</td>
</tr>
<tr>
<td>V2</td>
<td>Act</td>
<td>Adv NP_{dobj}</td>
<td>9</td>
</tr>
<tr>
<td>V</td>
<td>Act</td>
<td>Adv S</td>
<td>8</td>
</tr>
<tr>
<td>V2V</td>
<td>Act</td>
<td>NP_{dobj} NP_{nsubj} VP</td>
<td>5</td>
</tr>
<tr>
<td>VS</td>
<td>Pass</td>
<td>S</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Verb</th>
<th>Voice</th>
<th>Arguments</th>
<th>Freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>V</td>
<td>Act</td>
<td>Adv Adv Adv NP_{nsubj}</td>
<td>2</td>
</tr>
<tr>
<td>V2</td>
<td>Act</td>
<td>Adv Adv NP_{dobj} NP_{nsubj}</td>
<td>2</td>
</tr>
<tr>
<td>V3</td>
<td>Act</td>
<td>NP_{obj} NP_{nsubj}</td>
<td>2</td>
</tr>
<tr>
<td>VQ</td>
<td>Act</td>
<td>QS</td>
<td>2</td>
</tr>
<tr>
<td>VS</td>
<td>Act</td>
<td>Adv NP_{nsubj} S</td>
<td>2</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>Adv Adv NP_{nsubj}pass</td>
<td>2</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>Adv NP_{dobj} NP_{nsubj}pass</td>
<td>2</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>NP_{dobj}</td>
<td>2</td>
</tr>
<tr>
<td>V2</td>
<td>Act</td>
<td>Adv Adv NP_{dobj}</td>
<td>1</td>
</tr>
<tr>
<td>V2S</td>
<td>Act</td>
<td>NP_{dobj} NP_{nsubj} S</td>
<td>1</td>
</tr>
<tr>
<td>V2S</td>
<td>Act</td>
<td>NP_{dobj} S</td>
<td>1</td>
</tr>
<tr>
<td>V2V</td>
<td>Act</td>
<td>NP_{dobj} VP</td>
<td>1</td>
</tr>
<tr>
<td>VS</td>
<td>Act</td>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>VV</td>
<td>Act</td>
<td>VP</td>
<td>1</td>
</tr>
<tr>
<td>V2</td>
<td>Pass</td>
<td>Adv</td>
<td>1</td>
</tr>
<tr>
<td>VS</td>
<td>Pass</td>
<td>NP_{nsubj}pass S</td>
<td>1</td>
</tr>
</tbody>
</table>

The **869** semantico-syntactic valence patterns reuse **32** syntactic patterns

- **32 RGL-based code templates** are used to generate the implementation
- Most templates are derived on the fly from few basic templates
  - E.g., adverbial modifiers are added by recursive calls of the \textit{mkVP} constructor
  - Note: the order of \textit{Adv} FEs can differ across languages
FrameNet-based lexicon: abstract

- All the distinct LUs from the sentence patterns that belong to the shared valence patterns
  - BFN: 2,831 LUs resulting in \(3,432\) lexical functions
    - 1.21 functions per LU due to alternative verb types
  - SweFN: 1,844 LUs, \(1,899\) functions (1.03 per LU)
    - \(~1.5\) corpus examples per LU vs. \(~20\) per LU in BFN

- Verb types: \(V, V2, V3, VV, VS, V2V, V2S\)
- To distinguish between different types and senses of LUs, the verb type and the frame name is appended to the function identifiers
  - The LU-frame mapping, however, is not restricted (apart from the verb type)

```
fun hunger_V_Desiring : V
fun yearn_V_Desiring : V
fun want_V2_Desiring : V2
fun want_VV_Desiring : VV
fun yearn_VV_Desiring : VV
fun längta_V_Desiring : V
fun känna_för_V2_Desiring : V2
fun känna_för_VV_Desiring : VV
fun vilja_VV_Desiring : VV
fun känna_sig_V_Feeling : V
fun känna_V2_Familiarity : V2
```
Verb constructors are extracted from various RGL modules:
- L/DictL (6,034 for English, 7,324 for Swedish)
- translator/DictionaryL (6,037 for English, 2,430 for Swedish)
- L/LexiconL (98 for English, 96 for Swedish)
- L/IrregL (173 for English, 182 for Swedish)
- L/StructuralL (2 for English, 4 for Swedish)

For each lexical function, generate its linearization based on the corresponding verb constructor, taking into account particles and reflexive pronouns (MWEs), and the verb type

\[
\begin{align*}
\text{lin } \text{want}_V^2_{\text{Desiring}} &= \text{mkV}^2 (\text{regV } "\text{want}"
) \\
\text{lin } \text{känna}_V^V_{\text{for}}_{\text{Desiring}} &= \text{mkVV} (\text{partV} (\text{irregV } "\text{känna}" "\text{kände}" "\text{känt}" )"\text{för}"
) \\
\text{lin } \text{känna}_V^V_{\text{sig}}_{\text{V}}_{\text{Feeling}} &= \text{reflV} (\text{irregV } "\text{känna}" "\text{kände}" "\text{känt}"
)
\end{align*}
\]

Linearization: 3,350 (98%) Eng entries and 1,789 (94%) Swe entries

Simple, fixed multi-word units (MWU):
- 98 for English – ~3% of all entries and ~84% of all MWU entries
- 465 for Swedish – ~25% of all entries and ~85% of all MWU entries
FrameNet-based lexicon: alignment

- Based on the multilingual RGL dictionaries (translator/DictionaryL)

Eng: lin feel$_V$ = IrregEng. feel$_V$
Swe: lin feel$_V$ = mkV "könnan" "kännde" "känt"

Eng: lin want$_V$2 = mkV2 (mkV "want")
Swe: lin want$_V$2 = mkV2 IrregSwe. vilja$_V$

Eng: lin yearn$_V$ = mkV "yearn" "yearns" "yearned" "yearning"
Swe: lin yearn$_V$ = mkV "trängtar"

feel$_V$$_V$ Desiring = känna$_V$$_V$ Desiring
want$_V$$_V$ Desiring = vilja$_V$$_V$ Desiring

- Result: 703 BFN entries (21%) aligned with 900 SweFN entries (47%)
  - Still promising (there is a clear space for improvement)
FrameNet-based API to GF Resource Grammar Library
A tool for cross-lingual comparison of FrameNet-annotated corpora

Desiring

Desiring_V : Experiencer_NP → Focal_participant_Adv → V → Clause

- **Eng:** [They] Experiencer [ASPIRED] [towards the Chelsea shore, where, in the early 1960s many thousands lived with sensible occupations and adequate amounts of money] Focal_participant

- **Swe:** [Roberte] Experiencer [LÄNGTADE] [hem till Tyskland] Focal_participant

Desiring_V2 : Experiencer_NP → Focal_participant_NP → V2 → Clause

- **Eng:**
  - covet_V2_Desiring : V2
  - crave_V2_Desiring : V2
  - desire_V2_Desiring : V2
  - [!] Experiencer neither [DESIRE] [this house] Focal_participant

  - fancy_V2_Desiring : V2
  - feel_like_V2_Desiring : V2
  - want_V2_Desiring : V2
  - yearn_V2_Desiring : V2

- **Swe:** [Jag] Experiencer [KÄNNER FÖR] [en tur på landet] Focal_participant

Desiring_VV : Event_VP → Experiencer_NP → VV → Clause

- **Eng:** [He] Experiencer [LUSTING] [to tear the alien apart and eat of its lurid vitals, so as to comprehend something of its strange nature] Event

http://grammaticalframework.org/framenet/
FrameNet-based API to Grammatical Framework

Version 0.9.7

Introduction

The aim of this project is to make existing FrameNet (FN) resources computationally accessible for multilingual natural language generation and controlled semantic parsing via a shared semantico-syntactic grammar and lexicon API.

We provide a currently bilingual but potentially multilingual FN-based grammar and lexicon library implemented in Grammatical Framework (GF) on top of GF Resource Grammar Library (RGL). The API of the FN-based library represents a shared set of automatically extracted semantico-syntactic verb valence patterns from 66,918 annotated sentences in Berkeley FrameNet (BFN 1.5) and 4,267 sentences in Swedish FrameNet (SweFN, a snapshot taken in

https://github.com/GrammaticalFramework/gf-contrib
Case study: Phrasebook

- Apart from idiomatic phrases, many can be constructed by applying the generated frame functions

  **ALive**: Person $\rightarrow$ Country $\rightarrow$ Action
  - $Residence\_V : Location\_Adv \rightarrow Resident\_NP \rightarrow V \rightarrow Clause$
    - *I live in Sweden* (Eng)
    - *jag bor i Sverige* (Swe)

  **AWantGo**: Person $\rightarrow$ Place $\rightarrow$ Action
  - $Desiring\_VV : Event\_VP \rightarrow Experiencer\_NP \rightarrow VV \rightarrow Clause$
    - *we want to go to a museum* (Eng)
    - *vi vill gå till ett museum* (Swe)

- No changes needed in the Phrasebook abstract syntax
  - Frame functions are not part of Phrasebook abstract syntax trees...

- The re-engineered grammar generates equal phrases
Case study: Phrasebook

• Before:

```
Case study: Phrasebook

lin Alive p co =
  mkCl
  p.name
  (mkVP
     (mkVP (mkV "live"))
     (mkAdv in_Prep co))

lin AWantGo p pl =
  mkCl
  p.name
  want_VV
  (mkVP
    IrregEng.go_V
    pl.to)
```

• After:

```
lin Alive p co = let cl : Clause =
  mkCl
  name
  Residence_V
    (Just Adv (mkAdv in_Prep co))
    (Just NP p.name)
  Live_V_Residence
  in mkCl cl np cl vp

lin AWantGo p pl = let cl : Clause =
  Desiring_VV
    (Just VP -- Event
      Motion_V_2
        (Just Adv pl.to) -- Goal
        (Nothing' Adv) -- Source
        (Nothing' NP) -- Theme
        go_V_Motion
      ).vp)
    (Just NP p.name) -- Experiencer
    want_VV_Desiring
  in mkCl cl np cl vp
```
Case study: Paintings

• Verbalizes descriptions of museum objects stored in an ontology

• A set of triples describing the artwork *Bacchus*:
  – <Bacchus> <createdBy> <Leonardo_da_Vinci>
  – <Bacchus> <hasDimension> <Bacchus_ImageDimesion>
  – <Bacchus> <hasCreationDate> <Bacchus_CreationDate>
  – <Bacchus> <hasCurrentLocation> <Musee_du_Louvre>
  – <Bacchus_ImageDimesion> <lengthValue> 115
  – <Bacchus_ImageDimesion> <heightValue> 177
  – <Bacchus_CreationDate> <timePeriodValue> 1510

• Triples are combined by the grammar to generate a coherent text
  – DPainting : Painting -> Painter -> Year -> Size -> Museum -> Description

    • Eng: *Bacchus* was **painted by** Leonardo da Vinci in **1510**. It **measures** 115 by 177 cm. This work is **displayed** at the Musée du Louvre.

    • Swe: *Bacchus* **målades av** Leonardo da Vinci **år 1510**. Den **mäter** 115 gånger 177 cm. Det här verket är **utställt** på Louvre.

• The re-engineered grammar generates semantically equiv. descriptions
  – In Swedish, the use of the main verb *mäta* is imposed instead of the copula
Case study: Paintings

lin DPainting
painting painter year size museum =
let
s1 : Text = mkText (mkS
  pastTense (mkCl painting (mkVP
    (mkVP (passiveVP paint_V2)
      (mkAdv by8agent_Prep
        painter.Long)) year.s))) ;

s2 : Text = mkText
  (mkCl it_NP (mkVP (mkVP
    (mkVPSlash measure_V2)
     (mkNP (mkN "")) size.s))) ;

s3 : Text = mkText
  (mkCl (mkNP this_Det painting)
    (mkVP (passiveVP display_V2)
      museum.s))
in mkText s1 (mkText s2 s3) ;

* Currently not available out-of-the-box

lin DPainting
painting painter year size museum =
let
cl1 : Clause =
  Create_physical_artwork_V2_Pass*
  (Just NP painter.long) -- Creator
  (Just NP painting) -- Representation
  paint_V2_Create_physical_artwork ;

cl2 : Clause = Dimension_V2*
  (Just NP (mkNP emptyNP size.s)) -- Measurement
  (Just NP it_NP) -- Object
  measure_V2* ;

cl3 : Clause = Placing_V2_Pass
  (Just Adv museum.s) -- Goal
  (Just NP (mkNP this_Det painting)) -- Theme
  display_V2*

in mkText (mkText (mkS pastTense
  (mkCl cl1.np (mkVP cl1.vp year.s))) -- Time
  (mkText (mkCl cl2.np cl2.vp)
    (mkText (mkCl cl1.np cl3.vp)))) ;
Evaluation

• **Intrinsic**
  – The number of examples in the source corpora that belong to the set of shared frames and are *covered* by the shared valence patterns
  – Corpus examples are judged by the sentence patterns that represent them, disregarding non-core FEs, word order, and prepositions
    • The syntactic roles and the grammatical voice are considered
  – **BFN**: 57,615 examples (90%) belong to the shared set of 483 frames, and *77.5%* of them are covered by the shared patterns
  – **SweFN**: 3,348 examples (80%), *77.5%* are covered
  – The *shared lexicon* covers *25.1%* of BFN sentences and *35.8%* of SweFN

• **Extrinsic**
  – The number of constructors used to linearize functions in the original vs. the re-engineered grammar (comparison of *code complexity*)
    • In Paintings, the number of constructors is reduced by 38% while in Phrasebook only by 20–27%
Summary and future work

• Despite the small SweFN corpus, the set of extracted shared valence patterns is **concise** and already provides a **wide coverage**
  – The relatively small number of patterns allows for manual checking
  – The numbers are not stable and vary across releases but illustrate the tendency

• Include shared **non-core** FEs; generate missing **passive voice** functions

• Separate LU-governed **prepositional objects** from adverbial modifiers (*Adv* vs. *NP*; probability); differentiate syntactic roles of *VP* FEs (object vs. *Adv*)

• Add more languages (looking for cooperation)
  – Intersection of all languages vs. union of intersections of language pairs
  – **ExtraL** modules

• Towards FrameNet-based semantic parsing in GF
  – First, frame labelling
    • As an **embedded grammar**
    • Restrict LUs to frames by using GF **dependent types**
  – Later, semantic role labelling (SRL)
Constructicon

- A collection of conventionalized (learned) pairings of form and meaning (or function), typically based on principles of Construction Grammar, CxG (Fillmore et al. 1988, Goldberg 1995)
  - Semantics is associated directly with the surface form
  - LUs in FrameNet: pairings of word and meaning (frame)
    - Including fixed MWUs

- Each construction (cx) contains at least one variable element
  - Often at least one fixed element as well
  - Somewhere in-between the syntax and the lexicon

- An example from FrameNet Constructicon: *make one’s way* (WAY_MEANS)
  - Structure: \{Motion verb [Verb] [PossNP]\}
  - Evokes: MOTION
    - [Theme They] \{hacked their way\} [Source out] [Goal into the open].
    - [Theme We] \{sang our way\} [Path across Europe].
Towards a multilingual constructicon

- **Berkeley/FrameNet** Constructicon (BCxn)
  - A pilot project (~70 cx)

- **Swedish** Constructicon (SweCcn)
  - An ongoing project (nearly 400 cx so far), inspired by BCxn

- **Brazilian Portuguese** Constructicon
  - An ongoing project, inspired by BCxn

- ...  

- Allows for non-compositional translation in a compositional way
  - e.g. some constructions are covered by L/ConstructionL in RGL

- Constructions with a referential meaning may be linked via FrameNet frames, while those with a more abstract grammatical function may be related in terms of their grammatical properties

Jag behöver mat till festen.

I need food to the party.

http://sprakbanken.gu.se/eng/swecncn
SweCcn

- Partially schematic multi-word units/expressions
- Particularly addresses constructions of relevance for second-language learning, but also covers argument structure constructions
- Descriptions are manually derived from corpus examples
- Construction elements (CE):
  - Internal CEs are a part of the cx
  - External CEs are a part of the valency of the cx
  - Described in more detail by attribute-value matrices specifying their syntactic and semantic features
- A central part of cx descriptions is the free text definitions
  - ‘eat himself full’ vs. ‘feel himself tired’ (ätta sig mätt vs. känna sig trött)

<table>
<thead>
<tr>
<th>Name</th>
<th>REFLEXIV_RESULTATIV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Category</td>
<td>VP</td>
</tr>
<tr>
<td>Frame</td>
<td>CAUSATION</td>
</tr>
<tr>
<td>Definition</td>
<td>[Someone/something]NP performs/undergoes [an action]Activity that leads (or is supposed to lead) the [actor/theme]Pn, expressed by reflexive, to [a state]Result.</td>
</tr>
<tr>
<td>Structure</td>
<td>NP [V Pn_refl AP]</td>
</tr>
<tr>
<td>Internal</td>
<td>Activity: {cat=V, role=Activity}</td>
</tr>
<tr>
<td></td>
<td>Pn: {cat=Pn_refl, role=Actor</td>
</tr>
<tr>
<td></td>
<td>Result: {cat=AP, role=Result}</td>
</tr>
<tr>
<td>External</td>
<td>NP: {cat=NP, role=Actor</td>
</tr>
<tr>
<td>Example</td>
<td>PeterNP [ätterActivity sig mättPn Result]</td>
</tr>
</tbody>
</table>
SweCcn → GF

• Task: convert the semi-formal SweCcn into a computational CxG

• Why GF?
  – There is no formal distinction between lexical and syntactic functions in GF – fits the nature of constructicons
  – The potential support for multilinguality
  – Based on RGL / an extension to RGL / an embedded grammar
  – An extension to the FrameNet-based grammar and lexicon

• Goals:
  – From the linguistic point of view
    • New insights on the interaction between the lexicon and the grammar
    • Allows for testing the linguistic descriptions of constructions
  – From the language technology point of view:
    • Facilitates language processing in both mono- and multilingual settings (e.g. IE, MT)
  – Useful in second-language learning
    • Linguistic or technology point of view?
Conversion steps

• Preprocessing:
  – **Automatic** normalization and **consistency** checking
  – **Automatic** rewriting of the original structures in case of optional CEs and alternative types of CEs, so that each combination has a separate GF function
    • Does not apply to alternative LUs (either free variants or should be split into alternative constructions, or the CE should be made more general)
  – **Automatic** conversion of SweCcn **categories** to RGL categories
    • May result in more rewriting

• **Automatic** generation of the **abstract** syntax
  – By systematically applying the high-level RGL constructors
    • And limited low-level means

• **Manual** verification and completion (ToDo)
  – Requires a good knowledge and linguistic intuition of the language
Preprocessing examples

• \( \text{behöva} \ NP_1 \text{ till} \ NP_2 | \text{VP} \rightarrow \)
  \( \text{behöva}_v \ NP_1 \text{ till}_{\text{Prep}} \ NP_2 \mid \text{behöva}_v \ NP \text{ till}_{\text{Prep}} \text{ VP} \)

• \( \text{snacka} | \text{prata} | \text{tala} \ NP_{\text{indef}} \rightarrow \)
  \( \text{snacka}_v | \text{prata}_v | \text{tala}_v \ a_{\text{Sg,Det}} \text{ CN} \mid \)
  \( \text{snacka}_v | \text{prata}_v | \text{tala}_v \ a_{\text{Pl,Det}} \text{ CN} \mid \)
  \( \text{snacka}_v | \text{prata}_v | \text{tala}_v \ \text{CN} \)

• \( V \ av \ Pn_{\text{refl}} \ (\text{NP}) \rightarrow \)
  \( V \ av_{\text{Prep}} \text{ refl}_{\text{Pron}} \text{ NP} \mid V \ av_{\text{Prep}} \text{ refl}_{\text{Pron}} \)

• \( N | \text{Adj}+\text{städa} \rightarrow \)
  \( N + \text{städa}_v \mid A + \text{städa}_v \)
Abstract syntax

• Each **construction** is represented by **one or more functions** depending on how many **alternative structures** are produced in the preprocessing steps

• Each **function** takes **one or more arguments** that correspond to the **variable CEs** of the respective alternative construction

  • `behöva något till något VP_1 : NP -> NP -> VP`
  
  • `behöva något till något VP_2 : NP -> VP -> VP`

  • `snacka NP_1 : CN -> VP`
  
  • `snacka NP_2 : CN -> VP`
  
  • `snacka NP_3 : CN -> VP`

  • `verba_av_sig_transitiv_1 : V -> NP -> VP`
  
  • `verba_av_sig_transitiv_2 : V -> VP`

  • `x_städa_1 : N -> VP`
  
  • `x_städa_2 : A -> VP`
Concrete syntax

- Many constructions can be implemented by systematically applying the high-level RGL constructors
  - A parsing problem: which constructors in which order?

<table>
<thead>
<tr>
<th>Construction</th>
<th>Elements</th>
<th>Patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td>behöva_något_till_något_VP_1</td>
<td>behöva_V NP_1 till_Prep NP_2</td>
<td>{V} NP {Prep} NP</td>
</tr>
<tr>
<td>behöva_något_till_något_VP_2</td>
<td>behöva_V NP_1 till_Prep VP</td>
<td>{V} NP {Prep} VP</td>
</tr>
</tbody>
</table>

**Code template**

1. mkVP (mkVP (mkV2 mkV) NP) (mkAdv mkPrep NP)
2. The parser failed at token VP

**Final code (by automatic post-processing)**

\[
\text{lin behöva_något_till_något_VP_1 np_1 np_2 = mkVP (mkVP (mkV2 (mkV "behöver") np_1) (SyntaxSwe.mkAdv (mkPrep "till") np_2));}
\]
Code-generating grammar

fun mkV2: V -> V2  
fun mkVP__V2_NP: V2 -> NP -> VP  
fun mkVP__VP_Adv: VP -> Adv -> VP  
fun mkAdv: Prep -> NP -> Adv  
fun __mkV__: V  
fun __mkPrep__: Prep  
fun __NP__: NP

A simplified fragment of the abstract syntax

```
parse -cat=VP "{V} {Prep} NP"

mkVP__V2_NP  
(mkV2__V (partV __mkV__V  
(toStr__Prep__mkPrep_))) __NP__

mkVP__VP_Adv  
(mkVP__V__Prep  
__mkV__V__mkPrep__) __NP__

mkVP__VP_Adv  
(mkVP__V__mkV__V)  
(mkAdv__mkPrep____NP__)
```

A simplified fragment of the concrete syntax

```
param Voice = Act | Pass

lincat
  V, V2 = Voice => Str
  VP, NP, Adv, Prep = Str
lin
  mkV2 v = \voice => v ! voice
  mkVP__V2_NP v2 np = v2 ! Act ++ np
  mkVP__VP_Adv vp adv = vp ++ adv
  mkAdv prep np = prep ++ np

  __mkV__ = table {
    Act => "{V}"
    Pass => "{V_pass}"  
  }

  __mkPrep__ = "{Prep}"
  __NP__ = "NP"
```
This is GF version 3.6.10-darcs.
663 recorded changes since RELEASE-3.6
Last recorded change: Thu Apr  9 12:18:41 CEST 2015  hallgren@chalmers.se
Built on darwin/x86_64 with ghc-7.6, flags: interrupt
License: see help -license.
Bug reports: http://code.google.com/p/grammatical-framework/issues/list

Languages:
> 1 CxnSweCnc.gf
linking ... OK

Languages: CxnSweCnc
3298 msec
CxnSweAbs> p "jag behöver något till något"
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (DetNP someSg_Det))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (something_NP))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 something_NP (DetNP someSg_Det))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 something_NP (DetNP someSg_Det))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (deponens_reciprok_VP (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (DetNP someSg_Det))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (deponens_reciprok_VP (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (something_NP))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (deponens_reciprok_VP (behöva_något_till_något_VP_1 something_NP (DetNP someSg_Det))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (deponens_reciprok_VP (behöva_något_till_något_VP_1 something_NP (DetNP someSg_Det))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (DetNP someSg_Det))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 (DetNP (someSg_Det) (something_NP))))))
UseCL (TTAnt TPres ASimul) PPos (PredVP (UsePron i_Pron) (behöva_något_till_något_VP_1 something_NP (DetNP someSg_Det))))

235 msec
CxnSweAbs>
Results

• In the current experiment, we have consider only the 96 VP constructions which resulted in 127 functions
  – Dominating in SweCcn; have the most complex internal structure

• Given the 127 functions, we have automatically generated the implementation for 98 functions (77%) achieving a 70–90% accuracy
  – There is clear space for improvement

• Manual completion postponed because of the active development of SweCcn (changes → synchronization)

• https://github.com/GrammaticalFramework/gf-contrib (SweCcn)

• A methodology on how to systematically formalise the semi-formal representation of SweCcn in GF, showing that a GF construction grammar can be, to a large extent, acquired automatically

• Consequence: feedback to SweCcn developers on how to improve the annotation consistency and adequacy of the original construction resource
Publications

• Normunds Grūzītis, Pēteris Paikens, Guntis Bārzdiņš. **FrameNet Resource Grammar Library for GF.** CNL 2012

• Dana Dannélls, Normunds Grūzītis. **Extracting a bilingual semantic grammar from FrameNet-annotated corpora.** LREC 2014

• Dana Dannélls, Normunds Grūzītis. **Controlled natural language generation from a multilingual FrameNet-based grammar.** CNL 2014

• Normunds Grūzītis, Dana Dannélls, Benjamin Lyngfelt, Aarne Ranta. **Formalising the Swedish Constructicon in Grammatical Framework.** GEAF 2015

• Normunds Grūzītis, Dana Dannélls. **A Multilingual FrameNet-based Grammar and Lexicon for Controlled Natural Language.** Journal of LRE (in progress)