# 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**) – <u>semantic valence</u>
  - Frames are “evoked” in sentences by **target words** – **lexical units** (**LU**)
  - FEs are mapped based on the <u>syntactic valence</u> 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 <u>arguments</u>
    - Non-core FEs are not specific to the frame and **typically** are <u>adjuncts</u>

# 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 <u>motivated</u> ontology

# Example frame

| Desiring | |
|---|---|
| Definition: | 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. |
| Core FEs: | EVENT, EXPERIENCER, FOCAL_PARTICIPANT, LOCATION_OF_EVENT |
| Non-core FEs: | CAUSE, DEGREE, DURATION, MANNER, PLACE, PURPOSE_OF_EVENT, REASON, ROLE_OF_FOCAL_PARTICIPANT, TIME, TIME_OF_EVENT |

Introduced in **BFN**, reused in SweFN

*want*.v..6412

*känna_för*.vb..1

| Examples | Valence patterns | |
|---|---|---|
| 40 | EVENT | EXPERIENCER |
| (22) | VPto.Dep | NP.Ext |
| 14 | EXPERIENCER | FOCAL_PARTICIPANT |
| (10) | NP.Ext | NP.Obj |
| (1) | PP[*by*].Dep | NP.Ext |

Some valence patterns found in **BFN**

| Examples | Valence patterns | |
|---|---|---|
| 1 | EVENT | EXPERIENCER |
| (1) | VB.INF.VG | NN.SS |
| 2 | EXPERIENCER | FOCAL_PARTICIPANT |
| (2) | NN.SS | NN.OO |

Some valence patterns found in **SweFN**

e.g. "[*I*]$_{\textbf{Experiencer}}$ *do n't WANT* [*to deceive anyone*]$_{\textbf{Event}}$"

|

an embedded frame

e.g. "[*Jag*]$_{\textbf{Experiencer}}$ *KÄNNER FÖR* [*en tur på landet*]$_{\textbf{Focal\_participant}}$"

# 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 <u>subset</u> of NL instead of a grammar of a predefined <u>constructed</u> language
  - We produce a currently bilingual but potentially multilingual grammar library which is therefore not based on exactly one NL but inherently has a <u>shared</u> 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

```
mkCl person (mkVP (mkVP live_V) (mkAdv in_Prep place))
    -- mkCl : NP -> VP -> Cl
    -- mkVP : V -> VP
    -- mkVP : VP -> Adv -> VP
    -- mkAdv : Prep -> NP -> Adv
```

```
Residence                           -- Residence : NP -> Adv -> V -> Cl
    person                          -- NP (Resident)
    (mkAdv in_Prep place)           -- Adv (Location)
    live_V_Residence                -- V (LU)
```

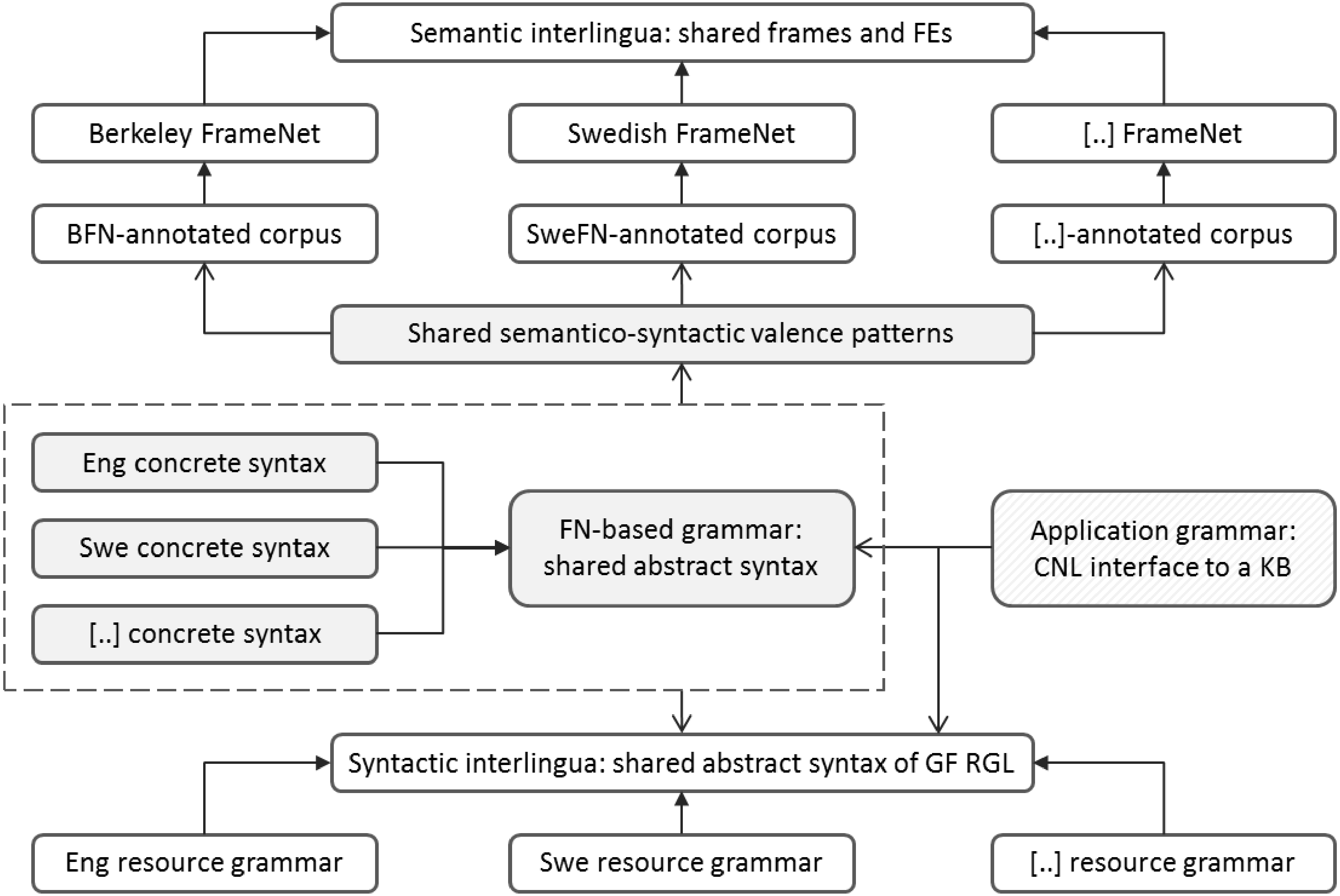
# Specific aim (2)

- FN-annotated knowledge bases → multilingual verbalization

| | | | | | | |
|---|---|---|---|---|---|---|
| | | Time | Place | Relatives | **Child** | |
| **Being_born** | dzimt.v | 1933. gada 3. maijs | Slokas pagasts | zvejnieka ģimene | Imants Ziedonis | |
| | | | | | | |
| | | **Institution** | **Subject** | Time | Place | **Student** |
| **Education_teaching** | absolvēt.v | Tukuma 1. vidusskola | | 1952. gads | Tukums | Imants Ziedonis |
| **Education_teaching** | beigt.v | Latvijas Universitāte | vēsture un filoloģija | 1959. gads | | Imants Ziedonis |
| **Education_teaching** | beigt.v | Augstākais literārais [..] | | 1964. gads | Maskava | Imants Ziedonis |
| | | | | | | |
| | | **Employer** | **Place_of_employment** | **Position** | Time | **Employee** |
| **Being_employed** | redaktors.n | izdevniecība Liesma | | > redaktors | | Imants Ziedonis |
| **Being_employed** | sekretārs.n | Latvijas rakstnieku [..] | | > sekretārs | | Imants Ziedonis |
| **Being_employed** | loceklis.n | AP tautas izglītības [..] | | > loceklis | | Imants Ziedonis |
| **Being_employed** | loceklis.n | Latvijas Institūts | | > loceklis | 1998. gads | Imants Ziedonis |
| **Being_employed** | padomnieks.n | | | > padomnieks | 1997. gads | Imants Ziedonis |
| **Being_employed** | skolotājs.n | Jūrmalas 1. vidusskola | | > skolotājs | | Imants Ziedonis |
| | | | | | | |
| | | Time | **Prize** | Rank | Competition | **Competitor** |
| **Win_prize** | apbalvot.v | 1983. gads | Tautu draudzības [..] | | | Imants Ziedonis |
| **Win_prize** | piešķirt.v | 1972. gads | Nopelniem bagātais [..] | | | Imants Ziedonis |
| **Win_prize** | piešķirt.v | 1977. gads | Tautas dzejnieka goda [..] | | | Imants Ziedonis |
| **Win_prize** | saņemt.v | | 1991. gada barikāžu [..] | | | Imants Ziedonis |

*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 FEs
Berkeley FrameNet
Swedish FrameNet
[..] FrameNet
BFN-annotated corpus
SweFN-annotated corpus
[..]-annotated corpus
Shared semantico-syntactic valence patterns
Eng concrete syntax
Swe concrete syntax
[..] concrete syntax
FN-based grammar: shared abstract syntax
Application grammar: CNL interface to a KB
Syntactic interlingua: shared abstract syntax of GF RGL
Eng resource grammar
Swe resource grammar
[..] 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*
  - **Desiring**/$V_{Act}$ Experiencer/$NP_{Subj}$ Focal_participant/Adv
    e.g., [*Dexter*]$_{\text{Experiencer}}$ [*YEARNED*] [*for a cigarette*]$_{\text{Focal\_participant}}$
  - **Desiring**/$V2_{Act}$ Experiencer/$NP_{Subj}$ Focal_participant/$NP_{DObj}$
    e.g., [*she*]$_{\text{Experiencer}}$ [*WANTS*] [*a protector*]$_{\text{Focal\_participant}}$
  - **Desiring**/$VV_{Act}$ Event/VP Experiencer/$NP_{Subj}$
    e.g., [*I*]$_{\text{Experiencer}}$ *would n't* [*WANT*] [*to know*]$_{\text{Event}}$
- The uniform patterns contain sufficient info for generating the grammar

# 1. Language- and FN-specific processing

```
<sentence ID="732945">
 <text>Traders in the city want a change.</text>
 <annotationSet><layer rank="1" name="BNC">
  <label start="0" end="6" name="NP0"/>
  <label start="20" end="23" name="VVB"/>
  <label start="25" end="25" name="AT0"/>
 </layer></annotationSet>
 <annotationSet status="MANUAL">
  <layer rank="1" name="FE">
   <label start="0" end="18" name="Experiencer"/>
   <label start="25" end="32" name="Event"/>
  </layer>
  <layer rank="1" name="GF">
   <label start="0" end="18" name="Ext"/>
   <label start="25" end="32" name="Obj"/>
  </layer>
  <layer rank="1" name="PT">
   <label start="0" end="18" name="NP"/>
   <label start="25" end="32" name="NP"/>
  </layer>
  <layer rank="1" name="Target">
   <label start="20" end="23" name="Target"/>
  </layer>
 </annotationSet>
</sentence>
```

```
<sentence id="ebca5af9-e0494c4e">
 ...
 <w pos="VB" ref="3" deprel="ROOT">skulle</w>
 <element name="Experiencer">
  <w pos="PN" ref="4" dephead="3" deprel="SS">
   jag
  </w>
 </element>
 <element name="LU">
  <w msd="VB.AKT" ref="5" dephead="3" deprel="VG">
   vilja
  </w>
 </element>
 <element name="Event">
  <w msd="VB.INF" ref="6" dephead="5" deprel="VG">
   ha
  </w>
  <w pos="RG" ref="7" dephead="8" deprel="DT">
   sju
  </w>
  <w pos="NN" ref="8" dephead="6" deprel="OO">
   sångare
  </w>
 </element>
</sentence>
```

- 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_Adv | want.v |
| Desiring | Act | Experiencer_NP.*Subj* Focal_participant_NP.*DObj* | want.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.Subj Event_VP : 59
            Event_VP Experiencer_NP.Subj : 2

        Experiencer_NP Focal_participant_NP : 61
            Experiencer_NP.Subj Focal_participant_NP.DObj : 55
            Focal_participant_NP.DObj Experiencer_NP.Subj : 6

        Experiencer_NP Focal_participant_Adv : 43
            Experiencer_NP.Subj Focal_participant_Adv[for] : 26
            Experiencer_NP.Subj Focal_participant_Adv[after] : 7
            Experiencer_NP.Subj Focal_participant_Adv : 2
            ...
        ...
    Pass : 13
        Experiencer_NP Focal_participant_NP : 5
            Focal_participant_NP.Subj Experiencer_NP.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*
  - *type*(A.*LU*) = *type*(B.*LU*)
  - A.*voice* = B.*voice*
  - B.*FEs* ⊆ A.*FEs* (incl. the syntactic types and roles)
- If *A* subsumes *B* and *B* subsumes *A* then *A* = *B*
- If a pattern of $FN_1$ is subsumed by a pattern of $FN_2$, it is added to the **shared** set (and vice versa)
  - In the final set, patterns that are subsumed by other patterns are removed

```
P1: Apply_heat  V2  Act  Cook_NP.Subj Food_NP.DObj
P2: Apply_heat  V2  Act  Cook_NP.Subj Container_Adv Food_NP.DObj
P3: Apply_heat  V2  Act  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

| Settings | BFN | | | | | | | SweFN | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Frames | Valence patterns | | Sentence patterns | | Corpus examples | | Frames | Valence patterns | | Sentence patterns | | Corpus examples | |
| | | total | per frame | total | per valence pattern | total | per sentence pattern | | total | per frame | total | per valence pattern | total | per sentence pattern |
| 0.0 | 556 | 19905 | 36 | 25696 | 1.3 | 68653 | 2.7 | 637 | 3395 | 5 | 3426 | 1.0 | 3688 | 1.1 |
| 2.0 | 555 | 16479 | 30 | 24491 | 1.5 | 66322 | 2.7 | 631 | 2811 | 4 | 2912 | 1.0 | 3393 | 1.2 |
| 2.A | 551 | 14135 | 26 | 22115 | 1.6 | 65008 | 2.9 | 625 | 2704 | 4 | 2812 | 1.0 | 3275 | 1.2 |
| 2.B | 551 | 5493 | 10 | 8696 | 1.6 | 65103 | 7.5 | **625** | **1965** | **3** | **2089** | **1.1** | **3298** | **1.6** |
| 3.0 | 506 | 6381 | 13 | 14393 | 2.3 | 56224 | 3.9 | 273 | 392 | 1 | 493 | 1.3 | 974 | 2.0 |
| 3.A | 502 | 5968 | 12 | 13948 | 2.3 | 56841 | 4.1 | 266 | 381 | 1 | 489 | 1.3 | 952 | 1.9 |
| 3.B | **508** | **3481** | **7** | **6684** | **1.9** | **63091** | **9.4** | 423 | 630 | 1 | 754 | 1.2 | 1963 | 2.6 |

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}`

```
fun Desiring_V       : Experiencer_NP -> Focal_participant_Adv -> V -> Clause
fun Desiring_V2      : Experiencer_NP -> Focal_participant_NP -> V2 -> Clause
fun Desiring_V2_Pass : Experiencer_NP -> Focal_participant_NP -> V2 -> Clause
fun Desiring_VV      : Event_VP -> Experiencer_NP -> VV -> Clause
```

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

```
cat Event_VP                 cat Focal_participant_NP
cat Experiencer_NP           cat Focal_participant_Adv
```

# **FrameNet-based grammar:** concrete

- 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

```
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

```
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

| Verb | Voice | Arguments | Freq. | Verb | Voice | Arguments | Freq. |
|---|---|---|---|---|---|---|---|
| V2 | Act | $NP_{dobj}$ $NP_{nsubj}$ | 277 | V | Act | Adv Adv Adv $NP_{nsubj}$ | 2 |
| V | Act | Adv $NP_{nsubj}$ | 155 | V2 | Act | Adv Adv $NP_{dobj}$ $NP_{nsubj}$ | 2 |
| V2 | Pass | $NP_{nsubjpass}$ | 84 | V3 | Act | $NP_{iobj}$ $NP_{nsubj}$ | 2 |
| V2 | Act | Adv $NP_{dobj}$ $NP_{nsubj}$ | 80 | VQ | Act | QS | 2 |
| V | Act | $NP_{nsubj}$ | 78 | VS | Act | Adv $NP_{nsubj}$ S | 2 |
| V2 | Pass | Adv $NP_{nsubjpass}$ | 34 | V2 | Pass | Adv Adv $NP_{nsubjpass}$ | 2 |
| VS | Act | $NP_{nsubj}$ S | 29 | V2 | Pass | Adv $NP_{dobj}$ $NP_{nsubjpass}$ | 2 |
| VV | Act | $NP_{nsubj}$ VP | 21 | V2 | Pass | $NP_{dobj}$ | 2 |
| V2 | Pass | $NP_{dobj}$ $NP_{nsubjpass}$ | 19 | V2 | Act | Adv Adv $NP_{dobj}$ | 1 |
| V2 | Act | $NP_{dobj}$ | 17 | V2S | Act | $NP_{dobj}$ $NP_{nsubj}$ S | 1 |
| V | Act | Adv Adv $NP_{nsubj}$ | 16 | V2S | Act | $NP_{dobj}$ S | 1 |
| VQ | Act | $NP_{nsubj}$ QS | 10 | V2V | Act | $NP_{dobj}$ VP | 1 |
| V2 | Act | Adv $NP_{dobj}$ | 9 | VS | Act | S | 1 |
| V | Act | Adv | 8 | VV | Act | VP | 1 |
| V2V | Act | $NP_{dobj}$ $NP_{nsubj}$ VP | 5 | V2 | Pass | Adv | 1 |
| VS | Pass | S | 3 | VS | Pass | $NP_{nsubjpass}$ S | 1 |

The **869** semantico-syntactic valence patterns reuse **32** <u>syntactic</u> 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 *mkVP* constructor
    - Note: the order of *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
```

# **FrameNet-based lexicon**: concrete

- Verb constructors are extracted from various RGL modules:
  - `L/`**`Dict`**`L` (6,034 for English, 7,324 for Swedish)
  - `translator/`**`Dictionary`**`L` (6,037 for English, 2,430 for Swedish)
  - `L/`**`Lexicon`**`L` (98 for English, 96 for Swedish)
  - `L/`**`Irreg`**`L` (173 for English, 182 for Swedish)
  - `L/`**`Structural`**`L` (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

```
lin want_V2_Desiring = mkV2 (regV "want")
lin känna_för_VV_Desiring = mkVV (partV (irregV "känna" "kände" "känt") "för")
lin känna_sig_V_Feeling = reflV (irregV "känna" "kände" "känt")
```

- 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/`**`Dictionary`**`L`)

```
Eng: lin feel_V = IrregEng.feel_V
Swe: lin feel_V = mkV "känna" "kände" "känt"

Eng: lin want_V2 = mkV2 (mkV "want")
Swe: lin want_V2 = mkV2 IrregSwe.vilja_V

Eng: lin yearn_V = mkV "yearn" "yearns" "yearned" "yearned" "yearning"
Swe: lin yearn_V = mkV "trängtar"
```

⇓

```
feel_like_VV_Desiring = känna_för_VV_Desiring

want_VV_Desiring = vilja_VV_Desiring
```

- Result: 703 BFN entries (21%) aligned with 900 SweFN entires (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

Frames | Verbs

vers. 0.9.7

REMU

| | |
|---|---|
| Court_examination | (1) |
| Create_physical_artwork | (1) |
| Create_representation | (2) |
| Creating | (2) |
| Criminal_investigation | (1) |
| Cutting | (2) |
| Damaging | (2) |
| Daring | (2) |
| Death | (1) |
| Deciding | (1) |
| Delimitation_of_diversity | (1) |
| Delivery | (3) |
| Deny_permission | (1) |
| Departing | (1) |
| Deserving | (1) |
| Desiring | (3) |
| Destroying | (4) |
| Detaching | (2) |
| Detaining | (2) |
| Differentiation | (3) |
| Discussion | (1) |
| Dispersal | (1) |
| Distinctiveness | (1) |
| Dodging | (1) |
| Dominate_competitor | (1) |
| Dominate_situation | (1) |
| Dressing | (2) |
| Drop_in_on | (1) |
| Dunking | (1) |
| Duplication | (2) |
| Duration_relation | (2) |

## 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

[I]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 ground his teeth together , [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/

# Source code

branch: **master** gf-contrib / **framenet** / +

Minor updates to the documentation

**normundsg** authored 14 minutes ago latest commit f41e84a5a6

..

| | | |
|---|---|---|
| examples/phrasebook | Version 0.9.7 | 7 months ago |
| lib | Version 0.9.7 | 7 months ago |
| README.md | Minor updates to the documentation | 14 minutes ago |

**README.md**

# 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 -> Country -> Action`
  - **`Residence`**`_V : Location_Adv -> Resident_NP -> V -> Clause`
    - *I live in Sweden* (Eng)
    - *jag bor i Sverige* (Swe)

- **`AWantGo`** `: Person -> Place -> Action`
  - **`Desiring`**`_VV : Event_VP -> Experiencer_NP -> VV -> Clause`
  - **`Motion`**`_V_2 : Goal_Adv -> Source_Adv -> Theme_NP -> V -> 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**:

```
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
      (mkVP IrregEng.go_V)
      pl.to)
```

- **After**:

```
lin ALive p co = let cl : Clause =
    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 <u>in 1510</u>. It **measures** 115 by 177 cm. This work is **displayed** at the Musée du Louvre.*
    - Swe: *Bacchus **målades av** Leonardo da Vinci <u>år 1510</u>. Den **mäter** 115 gånger 177 cm. Det här verket är **utställt** på Louvren.*
- 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) ;
```

```
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))) ;
```

* Currently not available out-of-the-box

# 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: {$^{\text{Motion verb}}$ [$^{\text{Verb}}$] [$^{\text{PossNP}}$]}
  - Evokes: MOTION
    - [$_{\textbf{Theme}}$*They*] {*hacked their way*} [$_{\textbf{Source}}$*out*] [$_{\textbf{Goal}}$*into the open*].
    - [$_{\textbf{Theme}}$*We*] {*sang our way*} [$_{\textbf{Path}}$*across Europe*].

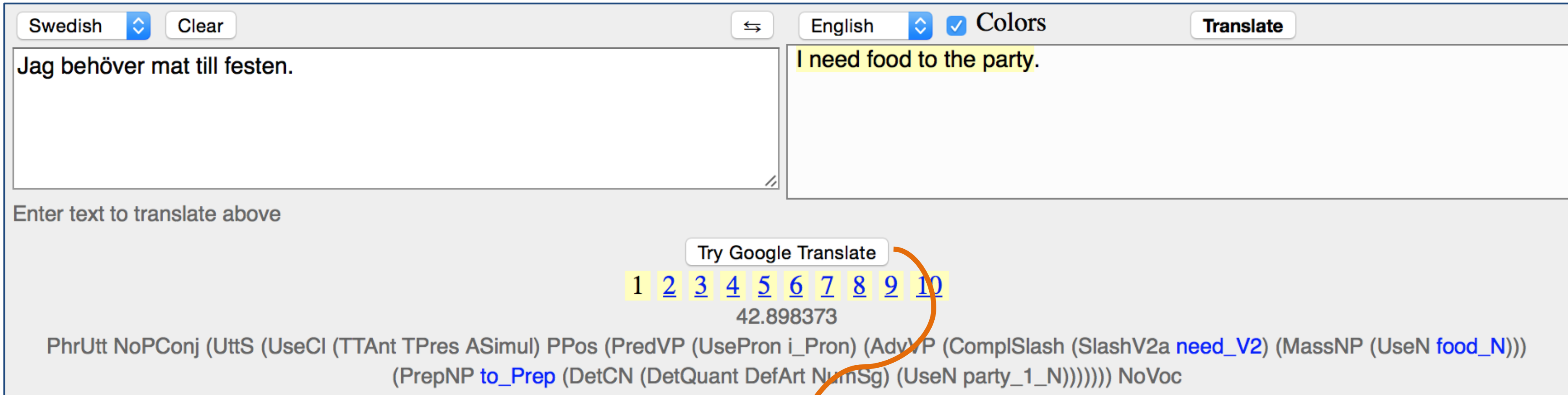
# Towards a multilingual constructicon

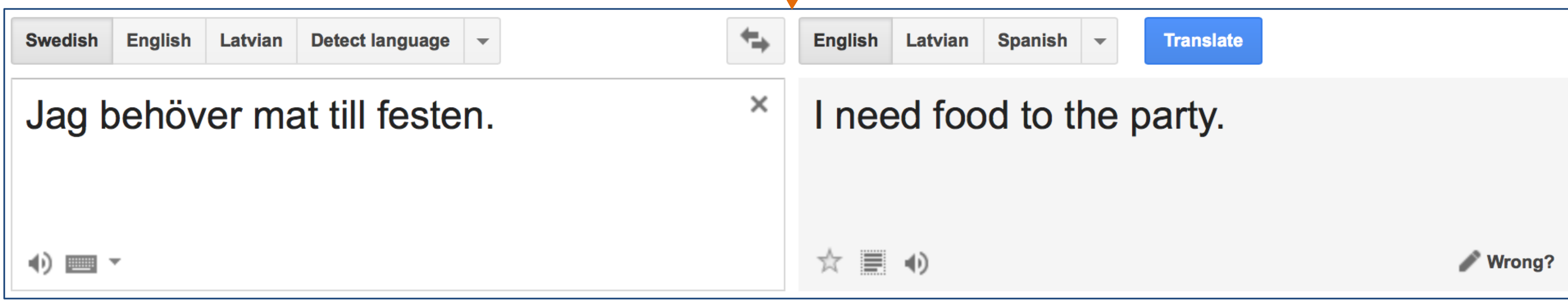
- **Berkeley/FrameNet** Constructicon (BCxn)
  - A pilot project (~70 cx)
- **Swedish** Constructicon (SweCcn)
  - An ongoing project (nearly 400 cx so far), inspired by BCxn
- **Brazilian Portuguese** Constructicon
  - An ongoing project, inspired by BCxn
- ...

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

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

  [Bäckström L., Lyngfelt B., Sköldberg E. (2014) Towards interlingual constructicography]




behöva_något_till_något - *behöver mat till festen*

| | |
|---|---|
| **category** | VP |
| **FrameNet** | Needing |
| **structure** | [behöva1 NP$_1$ till1 NP$_2$ \| VP ] |

http://spraakbanken.gu.se/eng/sweccn

# 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' (*äta sig mätt* vs. *känna sig trött*)

| | |
|---|---|
| **Name** | REFLEXIV_RESULTATIV |
| **Category** | VP |
| **Frame** | CAUSATION |
| **Defintion** | [Someone/something]$_{\text{NP}}$ performs/undergoes [an action]$_{\text{Activity}}$ that leads (or is supposed to lead) the [actor/theme]$_{\text{Pn}}$, expressed by reflexive, to [a state]$_{\text{Result}}$. |
| **Structure** | `NP [V Pn`$_{\texttt{refl}}$ `AP]` |
| **Internal** | Activity: {`cat=V, role=Activity`}<br>Pn: {`cat=Pn`$_{\texttt{refl}}$`, role=Actor|Theme`}<br>Result: {`cat=AP, role=Result`} |
| **External** | NP: {`cat=NP, role=Actor|Theme`} |
| **Example** | *Peter*$_{\text{NP}}$ [*äter*$_{\text{Activity}}$ *sig*$_{\text{Pn}}$ *mätt*$_{\text{Result}}$] |

# 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
- **Automatic** generation of the concrete 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

- *behöva* $NP_1$ *till* $NP_2$|VP →
  *behöva*$_V$ $NP_1$ *till*$_{Prep}$ $NP_2$ | *behöva*$_V$ NP *till*$_{Prep}$ VP

- *snacka*|*prata*|*tala* $NP_{indef}$ →
  *snacka*$_V$|*prata*$_V$|*tala*$_V$ aSg_Det CN |
  *snacka*$_V$|*prata*$_V$|*tala*$_V$ aPl_Det CN |
  *snacka*$_V$|*prata*$_V$|*tala*$_V$ CN

- V *av* $Pn_{refl}$ (NP) →
  V *av*$_{Prep}$ refl$_{Pron}$ NP | V *av*$_{Prep}$ refl$_{Pron}$

- N|Adj+*städa* →
  N + *städa*$_V$ | A + *städa*$_V$

# Abstract syntax

- Each **construction** is represented by <u>one or more functions</u> depending on how many <u>alternative structures</u> are produced in the preprocessing steps
- Each **function** takes <u>one or more arguments</u> that correspond to the <u>variable CEs</u> 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?

| Construction | Elements | Patterns |
|---|---|---|
| behöva_något_till_något_VP_1 | behöva_V NP_1 till_Prep NP_2 | {V} NP {Prep} NP |
| behöva_något_till_något_VP_2 | behöva_V NP_1 till_Prep VP | {V} NP {Prep} VP |

| Code template |
|---|
| 1. mkVP (mkVP (mkV2 **mkV**) **NP**) (mkAdv **mkPrep** **NP**) |
| 2. The parser failed at token VP |

A simple GF grammar

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

```
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**

```
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"
```

A simplified fragment of the **concrete syntax**

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

mkVP__V2_NP
  (mkV2__V (partV _mkV___V
  (toStr__Prep _mkPrep_))) _NP_

**mkVP**__V2_NP **(mkV2**__V_Prep
  **_mkV_**__V **_mkPrep_) _NP_**

mkVP__VP_Adv (mkVP__V _mkV___V)
  (mkAdv _mkPrep_ _NP_)

# Running example

out — gf — 149×45

```
         *  *  *
      *           *
    *               *
   *
   *
   *        * * * * * *
   *        *         *
    *       * * * *  *
      *     *      *
         *  *  *

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:
> i 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 something_NP))
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 something_NP)))
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 something_NP))
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 something_NP)))

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)