Syntactic Sugar for Syntactic Queries: Sequential Representations for Dependency Queries

Niklas Deworetzki[1] and Arianna Masciolini[2]

[1]Department of Computer Science and Engineering; Chalmers University of Technology and University of Gothenburg
[2]Språkbanken Text; Department of Swedish, Multilingualism, Language Technology; University of Gothenburg

Motivation

Syntactic query languages such as **Grew** and **dep_search** make it easy to search for syntactic patterns, but are not supported in large-scale corpus management tools such as **Corpus Workbench**, **SketchEngine** and **Korp**.

Grew

```
pattern {
   aux[word="ska"];
   inf[msd=re"VB\.INF.*"];
   aux-[VG]->inf; }
```

dep_search

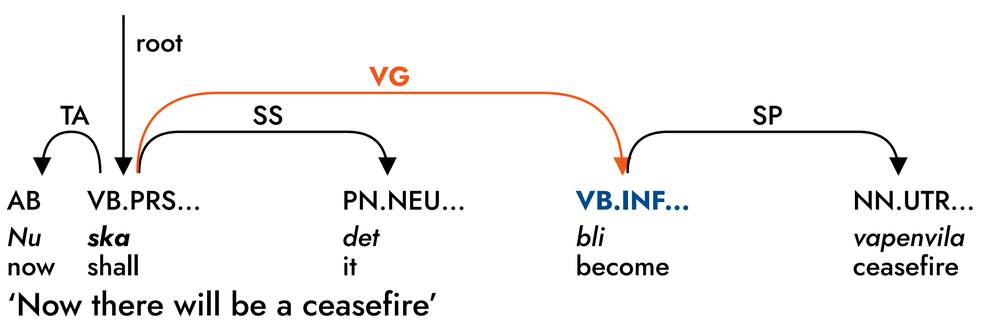
```
ska >VG msd="VB\.INF.*"
```

CQL

```
aux:[(word="ska")] []* inf:[(msd="VB\.INF.*") & (deprel="VG")
& dephead=aux.ref] | inf:[(msd="VB\.INF.*") & (deprel="VG")]
[]* a:[(word="ska") & inf.dephead=ref]
```

How it Works

To turn a syntactic query into a sequential one, all possible arrangements for constituents have to be found. In general, there are $n!$ possible arrangements for n constituents and dependency constraints need to be adapted for each arrangement.



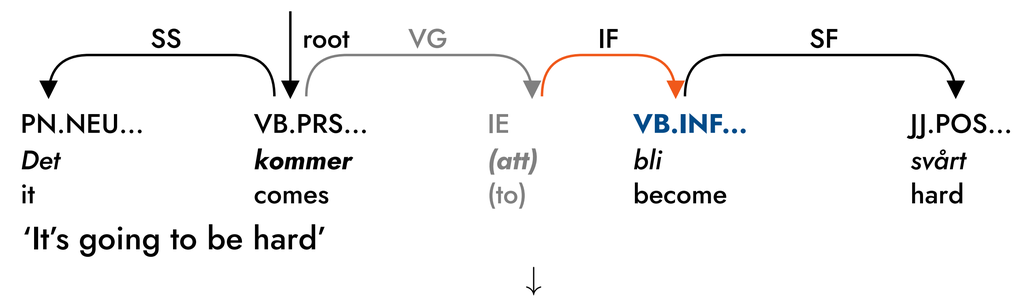
'Now there will be a ceasefire'

↓

```
aux:[(word="ska")] []* inf:[(msd="VB\.INF.*") & (deprel="VG") &
dephead=aux.ref] | inf:[(msd="VB\.INF.*") & (deprel="VG")] []*
aux:[(word="ska") & inf.dephead=ref]
```

CQP Recipes

Not all syntactic queries can be translated into a single sequential query. When this is the case, the output is a *CQL recipe*, i.e. sequence of CQL queries and set operations on their results.



'It's going to be hard'

↓

1. search for **all** *kommer* + infinitive constructions:
   `A=[word="kommer"] []* [msd="VB\.INF.*"]`
2. search for *kommer* + infinitive constructions **with** the optional infinitival marker *att*:
   `B=[word="kommer"] []* a:[word="att"] []* [msd="VB\.INF.*" & dephead=a.ref]`
3. remove the latter from the former: `diff A B`

Try CQP/Tree

as a web application

or:

- as a command-line tool
- as a Python library

Supported Technologies

CQP/Tree supports translation from **Grew**, **dep_search**, **deptreepy** and even **CoNLL-U*** for the corpus systems **Corpus Workbench**, **Korp** and **(No)SketchEngine**.* Just dependency annotations are required!

Frontends:
- Grew
- dep_search
- deptreepy
- CoNLL-U*

Backends:
- Corpus Workbench
- Korp
- (No)Sketch Engine*

*work in progress

Evaluation

Translated queries can "get stuck" on the wrong constituents, which leads to under-reporting of matches. We generate and execute randomized queries to verify our translation and quantify the impact of this restriction.

Tokens	Queries	Correct [%]	Δ_{rel} [%]
1	2228	99.96	0.00
2	1789	99.78	0.12
3	511	67.47	8.07
4	402	55.47	14.57
5	7	100.00	0.00
All	4937	92.93	2.07

Future Work

- Support for more query language constructs
- Complete (No)SketchEngine integration
- "What You See Is What You Get" editor for tree queries

CHALMERS | GÖTEBORGS UNIVERSITET | SPRÅKBANKENTEXT