VISL & CG-3: Constraint Grammar on the Move An application-driven paradigm[1]

Eckhard Bick
Institute of Language and Communication
University of Southern Denmark
eckhard.bick@gmail.com

Abstract

With its vastly increased versatility and expressive power, CG-3 has become a de-facto standard for Constraint Grammar-based research and applications. Rooted in the VISL project's focus on multi-lingual, treebank-based CALL and corpus tools, it supports not only classical morphosyntactic annotation with static tags, but also substitutions, regular expressions, string and flow variables, rewriting rules, numerical conditions and tokenization changes, as well as structural annotation with dependency links and arbitrarily named relations (e.g. anaphora, rhetorical structure). Today, there are CG-3 parsers for over 30 languages, covering all levels of linguistic annotation, with a strong focus on semantics, such as NER, word senses, semantic roles and frames. This chapter provides an overview of the methodological possibilities and presents a number of research and development applications, such as genre adaptation (social media, speech, historical data), machine translation and proofing tools.

Key Words: *Constraint Grammar, CG-3,ICALL, Corpus Linguistics, Treebanks, Machine Translation, Semantic annotation, Rule-based proofing tools*

1 Introduction

Constraint Grammar (CG) is a rule-based, incremental method for linguistic annotation of running text, originally conceived (Karlsson 1990) for the contextual disambiguation of multiple morphological input from finite state transducers (FST) and subsequent mapping of shallow syntactic function tags. Though there have been efforts of FST-style "simultaneous" application of the rules in a CG grammar (Voutilainen 1997, Yli-Jyrä 2017), ordinary CG applies its rules sequentially and iteratively, never discarding the last possible reading, hereby allowing for time/space-manageable compiler implementation as well as implicit robustness, even in the face of contradictory rules or input errors. CG comes in several flavors that differ not only in the way rules are compiled (programming

[1] **Ref:** Bick, Eckhard. 2023. VISL & CG-3: Constraint Grammar on the Move: An application-driven paradigm . In: Arvi Hurskainen, Kimmo Koskenniemi, and Tommi Pirinen (eds.), Rule-Based Language Technology. NEALT Monograph Series, 2:112-140.
https://dspace.ut.ee/handle/10062/89595

framework, speed, efficiency), but also - and more importantly - in the power and scope of the supported rule types. The first implementation, CG-1, was originally compiled in LISP, soon followed by the slightly more powerful, and much faster CG-2 (Tapanainen), the first, academic parsers being written for English (Karlsson et al. 1995), Swedish (Birn 1995), Finnish[2] and French (Chanod & Tapanainen 1994), followed by other Nordic languages and Portuguese (Bick 1996). The two variants both spawned commercial companies - LingSoft, using CG-1, and Conexor, with its CG-2-derived Finite Dependency Grammar (FDG, Tapanainen, 1997) and "Machinese" parsers.

Though there were experiments ongoing with different approaches, such as machine-learning (ML) for the design and ordering of (simplified) CG rules (e.g. Marques & Rodriguez 1995, Padró 1996, Lindberg & Eneborg 1998, Lager 1999), the CG-2 rule formalism became a de-facto standard, not least because it was re-implemented by a new company, GrammarSoft ApS, in an open source framework for the VISL project ("Visual Interactive Syntax Learning", Bick 2001), an ICALL initiative at the University of Southern Denmark (then Odense University) that used CG for a wide range of languages to support grammar teaching tool and games and to build treebanks for pedagogical and teaching purposes. This version, VISL-CG[3], was backward compatible with CG-2, but introduced one major improvement, substitute rules. These were originally meant to allow system hybridization, contextually correcting and adapting input from probabilistic taggers, notably TreeTagger (Schmid 1994), that would then be fed into a higher-level, syntactic CG pipeline. In 2006/2007, GrammarSoft's open source CG-3 (Bick and Didriksen 2015), the main topic of this chapter, replaced VISL-CG with an ambitious agenda of improvements and additions to the Constraint Grammar formalism, vastly expanding the expressive power of CG rules. Over the years, CG-3 has become a de-facto standard and implemented a large number of features to support the needs of various research projects and practical applications, such as corpus annotation, treebanking, ICALL, spell- and grammar-checking and Machine Translation. Today, there are CG-3 grammars for over 30 languages, ranging from mature, full-scale multi-level systems to small task-oriented grammars in hybrid systems.

2 Limitations of classical Constraint Grammar

Constraint Grammar is in its essence a task-driven, methodological paradigm. The goal of a CG grammar is efficient, flexible and robust annotation of running text, harnessing linguistic knowledge in a piecemeal fashion without the need of either training data (probabilistic taggers, machine learning) or a complete description of what constitutes correct language (e.g. HPSG, LFG). From such a methodological perspective, the strength of classical CG was its simple, reductionist approach to disambiguation, out-performing both probabilistic and PSG taggers with simple contextual `REMOVE`, `SELECT` and `MAP` rules (cf. CG chapter, this volume).

[2] https://web.archive.org/web/20110722011002/https://victorio.uit.no/langtech/trunk/kt/fin/src/fin-dis.cg1 . Originally written by Fred Karlsson in CG-1.
[3] https://visl.sdu.dk/visl/vislcg-doc.html . The compiler for VISL-CG was written by Martin Carlsen.

However, the formalism only supported topological/positional context conditions, in the sense of referring to contextual tags with a relative position (±1, ±2 ... ±n) or to a directed sentence scope left or right (*±1, *±2 ... *±n). This limitation makes it all but impossible to handle relational annotation such as dependency trees, anaphora or discourse labeling. A second shortcoming concerned rule writing efficiency, specifically the way of how to refer to tags or sets of tags, where classical CG only allowed literal tag and set references, and only in contexts, not targets, leading to entire rule batches reiterating various combinations of feature-attribute pairs, but basically expressing only a single linguistic truth. This is relevant for all unification-based rules and becomes more acute for higher linguistic levels because of the increase in tag inventory, where e.g. semantic or domain tag sets tend to be orders of magnitude large than a language's POS inventory[4].

Classical CG is quite efficient at addressing ambiguities, where each alternative is relatively frequent. Very rare readings, however, once introduced by a morphological analyzer or (e.g. semantic) lexicon look-up, tend to get overrepresented in automatic annotation[5], as long as disambiguation is based at feature level tags. This can be prevented only by using lexeme-level rules, at the expense of considerable grammar and workload increase. General-domain probabilistic taggers, on the other hand, tend to *under*-represent rare readings for the obvious reason that they may be absent in the training data. In principle, Constraint Grammar can achieve the best of both worlds by adding heuristic, frequency-based conditions that will limit the selection and contextual use of rare readings (Bick 2009). The necessary information can be harvested from automatically CG-annotated corpora, because even including false positives, rare readings will still have a lower count than the ordinary readings for a given lexeme. In classical CG, the limitation to literal tags and the absence of numerical conditions, this could only be approximated through the use of RARE sets[6] listing rare lexeme-tag combinations.

With a formal limitation of having to express context conditions with a position-based strategy, a grammarian may want to refer to a neighbouring np rather than a neighbouring noun, because there may be interfering prenominal words such as articles. An unbounded search does not achieve the same thing because it risks ignoring words that cannot be prenominal, such as verbs and conjunctions. To solve this problem, classical CG works with `BARRIERS` and NON-sets. For instance, the context condition `'*1 N BARRIER NON-PRE-N'`, combined with a set definition of `'NON-PRE-N = V PRP CS'`, will look for a noun to the right with no interfering verbs, prepositions or subordinating conjunctions, but allow articles, determiners and adjectives. What is needed, is a way of positionally referring to np's or other constituents as a whole, in all kinds of positional

[4] Even at the morphological level, feature set can be very complex if compound tags are used, such as NFPACC (noun female plural accusative) or VP1SIND (verb present tense 1. person indicative). While common in English-inspired probabilistic taggers, CG convention reduced this complexity by using individual tags for all features, separating POS, gender, number, case, tense etc.

[5] This does not normally translate into a large overall performance decrease, because most lexemes do not have this problem. But it may lead to a high error percentage for individual lexemes and thus become an irritant to corpus users, or a problem for applications such as MT.

[6] Frequency information could then be approximated by working with frequency ranges expressed as different sets, RARE-0, RARE-1, RARE-2 etc.

contexts, or even as a `BARRIER`. Even ordinary dependency relations would not solve this issue, because they are blind to word order and adjacency. As a solution, Karlsson et al. (1995) suggested so-called templates - labels for tag-defined word sequences. Templates would allow to refer to what is known as constituents in phrase structure grammars, but the idea was never fully implemented before CG3.

Finally, a grammarian using classical CG has no or very little control about the interplay between a CG and its input text. For instance, pre-processed tokenization cannot be changed, and while sentence segmentation is subject to delimiter definition, delimiters cannot be removed or added. At the same time, rule scope is limited to individual sentences, and cannot be controlled by the grammarian, be it at the grammar level or the level of individual rules. Nor is it possible to add, remove, change or move tokens, as might be relevant for applications such as grammar-checking and machine translation. Also, classical CG is blind to genre, domain and other textual characteristics[7], as it is not possible to pass text-level variables, or for that matter, variables at all.

3 The CG-3 formalism

3.1 General philosophy

Constraint Grammar frameworks typically use one common label for the combination of a rule formalism and its compiler implementation, with a descriptive standard more loosely associated. While it comes with a reasonably fast C++ compiler and has undergone extensive optimization testing, the focus of CG-3 development is on improving the rule formalism, addressing all the issues mentioned in section 2, as well as many minor options motivated by individual projects. Descriptively, CG-3 advocates the cross-language VISL standard for grammatical categories and tag abbreviations, with many large CG grammars adhering to it, or at least being inspired by it. However, the rule formalism as such is kept as theory-neutral as possible, lending support to many of the major linguistic frameworks, either directly or with a plethora of associated notational filtering programs. The intended benefit to the linguistic community is two-fold: First, CG3 - and with it, the robustness and high accuracy of Constraint Grammar - can be used to produce (filtered) output that is notationally equivalent to one's desired descriptional brand of corpus annotation or teaching materials. In other words, output should contain the type and granularity necessary for either dependency or constituent treebanks, or for a topological field grammar. Second, it should be possible for a linguist to think up CG rules using the conceptual approach most familiar to him, formulating a given linguistic truth (or constraint) in terms of either dependency links, rewriting templates, variable unification or topological context. Thus, CG-3 has fully implemented the creation and use of dependency links and variable unification, while at the same time allowing rewriting rules for templates and enhancing the original, topological "field" rules.

[7] One work-around was to add genre or domain tags, or markers for slang or spoken language to individual lexical entries for the use in CG rules. With the advent of VISL-CG's `SUBSTITUTE` operator, it was also possible to propagate such tags to other words in the sentence, where they could be used, for instance, to select appropriate translation equivalents (Bick, 2007).

3.2 Increased Boolean power and scope management

The lowest-level and most straightforward innovation in CG-3 is the generalization of "Boolean" operators beyond the limitations of classical CG. First, set operations (+, - and OR) are allowed throughout, i.e. not just tags or sets, but also contexts. Second, a new negation operator, `NEGATE`, negation scopes can be entire contexts rather than just position-linked tags, and these negations can be nested. Third, the C condition (= "true for all cohort readings") is extended to barriers (`CBARRIER`) and relational contexts (ALL and NONE).

Similarly close to the core machinations of CG is scope management. Here, innovations include (a) the double-asterisk operator (**), extending an unbounded search in the presence of failed LINK contexts, (b) the S-suffix for including the self-position (e.g. `*1S`) and (c) the bidirectional search (`*0`). The default sentence scope can be expanded with a W-suffix (e.g. `*1W`), which as a default expands the working window to ± 2 sentences, but may be set arbitrarily to larger (or smaller) scopes.

3.3 Superseding literal tags: Regular expressions and variable unification

CG-3 allows the use of regular expressions for all tag types (wordform, lemma, POS etc.) in both `LIST` definitions and rule contexts. The feature allows the grammarian to address open class lexical tags (e.g. `".*i[zs]e"r V` in a set of mono-transitive verbs) or to generalize over systematically constructed ontology tags (e.g. `"H[a-z]*"r N` for `+HUMAN` noun, lumping tags like `<Hprof>` (profession), `<Hideo>` (ideology supporter), `<Hnat>` (nationality) etc., even in the face of tag inventories that are still being amended (e.g. with `<Hsick>`).

In a bid to construct resource-free, so-called "bare-bones" grammars for small, under-resourced languages, Bick (2011a) demonstrated how the regular expression feature can be used to perform morphological analysis and even some valency and semantic typing within a CG module (rather than using input from an FST or full-form lexicon file). Here, endings and affixes were combined with dummy .* stems in so-called `APPEND` rules, which add readings to un-analyzed wordforms (or existing cohorts). For closed-class function words, irregular tense forms and the like, similar `APPEND` rules can be used to add a shared reading for `LIST`ed word batches.

CG3 supports the capture and unification of string variables. Regular expressions can therefore be used to "harvest" variable parts of a tag (e.g. slot filler conditions in a frame template tag) and either match them to tags in other cohorts (e.g. arguments of a frame carrying predicate), or to add new tags on other cohorts based on a harvested string (e.g. semantic class propagation along anaphora links)[8]. In the bare-bones scenario, variables can be used to craft a lemma tag from (parts of) the wordform, e.g. for English adjectives with certain affixes:

```
APPEND ("$1"v ADJ) TARGET ("<(.*(ic|oid|ous))>"r) ;
```

[8] Tags containing regular expressions are marked with an 'r', tags containing a variable match are marked with a 'v'.

Another use scenario is POS disambiguation based on prepositional valency:

```
SELECT (V) (1 ("(.+)"r PRP) (0 (<$1^vp>v V)) ;

SELECT (N) (1 ("(.+)"r PRP) (0 (<$1^vp>v N)) ;
```

The methodologically most important use of variables is feature unification. In CG3, to-be-unified sets are $$-prefixed (e.g. `$$POS`). The most obvious use is gender/number/case-disambiguation of noun phrases, but also coordination-based disambiguation can profit from unification rules. In the example below, it is semantic roles (agent, patient, theme and location) that are disambiguated in this fashion:

```
LIST ROLE = §AG §PAT §TH §LOC ;

SELECT $$ROLE (-1 KC) (-2C $$ROLE) ;
```

Simple unification of tag LISTs will not, however, work with under-specified "Portmanteau" tag, e.g. unifying `'nC'` - no-case with a real case such as `NOM` (nominative) or `ACC` (accusative). For this purpose, CG3 uses so-called *top-level set unification*, with a &&-prefix, instead. Where `$$-` (list-) unification unifies "terminal" set members, &&-unification unifies subsets belonging to a superset. Two contexts will &&-unify if they have tags sharing the same subset. In the example below, `N-SEMS` (semantic super-types) is defined as a superset, with `N-SEM` (semiotic product super-type) as one of the subsets.

```
LIST N-SEM = <sem> <sem-l> <sem-r> <sem-w> <sem-c> <sem-s> <sem-e>
<coll-sem> <sem-nons> <system> <system-h> ;

SET N-SEMS = N-HUM OR N-LOC ... OR N-SEM ... OR N-SUBSTANCE ;

REMOVE @SUBJ> (0 $$@<ARG LINK 0 &&N-SEMS) (*-1 KC BARRIER NON-PRE-
N/ADV LINK *-1C $$@<ARG BARRIER CLB-ORD OR &MV OR @ARG/ADVL> LINK
0 &&N-SEMS) ; # ... offered the reader detailed notes and instructions on most of the
questions …
```

The example sentence has an ambiguous coordination, where it is not clear if 'and' starts a new clause, and the task of the `REMOVE` rule is to exclude a subject reading for 'instructions' (tagged `<sem-s>`) by semantically aligning it with 'notes' (tagged `<sem-r>`) because both `<sem-r>` and `<sem-s>` are part of the `N-SEM` subset of the `&&N-SEMS` superset, - and by checking if both nouns also have matching left-pointing argument readings (`$$@<ARG`), in this case `@<ACC` (direct objects).

3.4 Manipulating input-level information: New readings and new tokens

In a classical CG annotation pipeline, input tokenization is fixed, and lexical tokens have a fixed readings cohort that can only be whittled down, not created or appended. A CG-3 grammar, on the other hand, can change its own input. While VISL-CG's `SUBSTITUTE` operator only enables changes in existing readings, CG-3's new `APPEND` operator can add entire new reading lines in a context-dependent way. Apart from the bare-bones scenario described in section 3.3., it can be used for FST-analysis failures in the same manner. Other examples of `APPEND` uses are proper noun classification based on word parts, or correction alternatives for misspelled words or real-word errors.

Arguably more radical are changes affecting the input tokens themselves (rather than their tags). For this purpose, CG-3 offers the `ADDCOHORT` and `REMCOHORT` operators that can for instance be used to add or remove punctuation, insert zero-constituents (e.g. surface subjects) or to delimit chunk with named opening and closing brackets for later XML-conversion (Bick 2013a). In an MT pipeline, `ADDCOHORT` permits the insertion of prepositions based on NP-case and of pronouns based on verb inflection (Bick 2022), and the `MOVE` operator will change the order of input cohorts at the syntactic transfer stage (ibd.). In both cases, a `WITHCHILD` attribute can be used to address entire dependency treelets rather than individual cohorts, listing the types of dependents that should be included in the operation.

CG-3 can also change tokenization by fusing or splitting tokens. This is relevant where there is an ambiguity between a contraction and a single word reading, or for the creation or resizing of multi-word expressions (MWEs) in Named Entity Recognition (NER). In both cases, CG may do a better job than a - context-agnostic - lexical preprocessor. One way of implementing MWE fusion is to create the MWE using `ADDCOHORT`, with wordform, lemma and tagging harvested from the original, to-be-fused cohorts, and then remove the individual cohorts with `REMCOHORT`. Similarly, a cohort-split would require 2 or more additions and one removal. With the newer `SPLITCOHORT` and `MERGECOHORT`, the same can be achieved in a single rule.

3.5 Dependency and relational tags

CG-3 is the first/only brand of CG with direct support for structural information, first and foremost dependency structure, that earlier had to be treated with add-on modules. Dependency relations are added with the `SETPARENT ... TO` or `SETCHILD ... TO` operators and marked as `#n->m` tag, where n is a running self-id, and m the head id. Both the `SET` and the `TO` targets can have their own context conditions. Each token can only have one (primary) dependency head, and unless specifically allowed context instantiation avoids dependency loops.

Thus, for the sentence *"He thought **that** the flowers **looked** rather nice on her."*

```
SETPARENT @FS-<ACC (*-1 (”that” KS) BARRIER CLB) TO (**-1 <mv>
LINK 0 V-COG)  (NEGATE *1 @<ACC CBARRIER NON-PRE-N);
```

will link a finite object clause (underlined, marked `@FS-<ACC` on *'looked'*) with a that-conjunction to the main verb (`<mv>`, *'thought'*) anywhere to the left (`**-1`) if the latter is a cognitive verb (`V-COG`) and is not (`NEGATE`) followed by an ordinary (i.e. np) direct object (`@<ACC`) without (`CBARRIER`[9]) interfering prenominals (`NON-PRE-N`).

In the VISL parsers, dependency assignment is prepared by using attachment direction arrows (>, <) on function labels (e.g. `@SUBJ>` and `@<SUBJ`) that can be used by the dependency rules to identify the correct head and to establish crossing branch `BARRIER`s. Once established, dependency links can be used in contexts instead of position markers. 'p' refers to the parent/head, 'c' to a child/dependent and 's' to a sibling token in the dependency tree. Instead of C and NOT conditions, ALL and NONE are used with 'c' and 's', and there are a number of other refinement, such as 'deep scan' (*), self reference (S) and left/right(most) conditions (l, r, ll, rr) that can be attached to the link letters themselves. For the sentence *"Tesla has again designed a great car.",*

```
ADD (§AG) TARGET @SUBJ (p V-HUM LINK c @ACC LINK 0 N-THING) ;
```

will add the semantic role of 'agent' (§AG) to subjects (@SUBJ) with a human verb as head (p) if that in turn has a non-human (N-THING) direct object (@ACC). For existing dependency trees, annotation conventions can often be changed with a single rule, e.g. when changing between parallel coordination attachment and conjunct-on-conjunct attachment, or between syntactic and semantic dependencies. Thus

```
SETPARENT (@P<) TO (p PRP LINK p (*) ;
SETPARENT (PRP) TO (*1 @P<) ;
```

will raise the attachment of the argument of a preposition (`@P<`) to the head (p) of the preposition (PRP), then attach the preposition to its former dependent using a positional context (*1) rather than the (now deleted) head-dependent link. Depending on the language, semantic dependencies may hold between *morphemes* rather than words. Thus, a Greenlandic dependency tree has to extend below the word level to achieve structural equivalence with a European MT target language such as Danish. In this scenario, morphologically based retokenization will allow the use of an ordinary CG dependency module (Bick 2019b).

At a more general level, *named relations* can be used for structural annotation other than primary dependency. These are tagged *ID:n R:relname:m,* where *n* and *m* are the id's of the from- and to-tokens of the relation. A low-level, syntactic use is for secondary dependencies, as between subject complement (`@SC`) and subject (`@SUBJ`):

```
SETRELATION (attribute) TARGET (@SC) TO (s @SUBJ) ;
```

[9] The use of CBARRIER (unambiguous barrier) rather than BARRIER is necessary because of the NEGATE scope, which might otherwise fail unnecessarily and thereby allow the rule to be used, although it maybe wouldn't have been used had the BARRIER context already been disambiguated.

Other, more advanced, usage examples are anaphora (Bick 2010a), discourse structure (Bick 2018) and framenet annotation with semantic roles (Bick 2011b, 2012a and 2022b). For bidirectional relations, both ends of the relation can be labelled simultaneously by the same rule. For instance,

```
SETRELATIONS (wife_of) (husband_of) TARGET @SUBJ OR @OBJ (0 <fem>)
TO (p ("marry" V) LINK c @SUBJ OR @OBJ LINK NOT 0 <masc>) ;
```

will extract (non-LGBTQ) family relations from classical literature such as *"Princess Leonora married a beautiful knight."*

3.6 Templates: Approximating constituent structure

Approximating the then dominant descriptive paradigm of phrase structure grammars (PSGs) was a kind of holy grail of NLP in the early era of Constraint Grammar, and for treebanking - at least for non-Slavic languages - CG's native shallow syntax had to be post-processed to achieve constituent structure (cp. treebanking section 5.1.2). Rather than using external converters, this would today also be possible using `ADDCOHORT` to add PSG chunking brackets. CG-3 can also, however, directly address constituents, using so-called *templates,* first proposed by Karlsson et al. (1995). Templates are basically pre-defined sequences of tokens, POS or function tags that can be referred to as a whole in rule contexts, or even in other templates. The linguistic motivation is to support direct reference to (complex) constituents rather than individual tokens, and to allow a grammarian to *think* in terms of generative grammar. For instance, an np can be defined as:

```
(a) TEMPLATE np = ([ART, N]) OR ([ART,ADJ,N]) ;
(b) TEMPLATE np = (? ART LINK 1 N) OR (? ART LINK 1 ADJ LINK 1
N) ;
(c) TEMPLATE np = ? ART LINK *1 N BARRIER NON-PRE-N ;
```

and then used in ordinary rules with a T:-prefix

```
(*1 VFIN LINK *1 T:np).
```

Wile (a) may feel more "generative", (b) and (c) harness the full power of CG contexts[10], allowing unbounded references and LINKed conditions. While it is possible to think of templates simply as shorthand for complex contexts, they can still achieve things that are very difficult or impossible to express otherwise, for instance when using a template as a barrier or a -1 left context (which would otherwise require a complete inversion/rewriting of the "templated" context). And they do allow proper PSG rewriting rules. Thus, the following mini-PSG:

[10] Compiler-internally, both template types are processed in a similar way, which is why constituent templates have question marks or 0-positions as place holders for an external position marker, which will be inserted into the template by the compiler at run-time (”position override”).

```
np = art? adjp? n pp? ;
adjp = adv? adj ;
pp = prp np ;
```

could in CG3 be expressed as:

(a)
```
TEMPLATE np = (? N)
      OR (? ART LINK 1 N) OR (? ART LINK N LINK 1 T:pp)
      OR (? T:adjp LINK 1 N) OR (? T:adjp LINK 1 N LINK 1 T:pp)
      OR (? ART LINK 1 T:adjp LINK 1 N)
      OR (? ART LINK 1 T:adjp LINK 1 N LINK 1 T:pp) ;
TEMPLATE adjp = (? ADJ) OR (? ADV LINK 1 ADJ) ;
TEMPLATE pp = ? PRP LINK 1 T:np ;
```

or, in a more hybrid fashion:

(b)
```
TEMPLATE np = (? ART) OR (? ADV OR ADJ) LINK *1 N BARRIER (*) -
ADJ - ADV LINK (0 (*)) OR (1 T:pp) ;
```

which will match *"a very long winter with a lot of snow"* , by first recognizing the core np *"a very long winter",* either directly (b) or after first mounting the adjp *"very long"* (a), then mounting the nested pp's *"with (a lot (of snow))"* and finally rewriting the large np to accommodate the pp postnominal.

3.7 Shades of gray: Numerical tags

CG-3's numerical tags are secondary `<vn:num>` tags consisting of a variable name (vn) and a value (num), separated by a colon. Rules can make mathematical reference to these tags, matching values (`<vn=x`) or setting thresholds as '`<vn>[=]x>`' (greater than [or equal]) or '`<vn<[=]x>`' (smaller than [or equal]), or by referring to the highest (`<vn=MAX>`) or lowest (`<vn=MIN>`) values in a given cohort. The original motivation for introducing numerical tags were heuristic `REMOVE` rules that would harness corpus frequencies in a more fine-grained fashion than grammar-defined `<Rare>` sets. For this purpose, VISL parsers use relative in-cohort frequencies between 0 and 100 (<fr:num>) alongside in-corpus frequencies (`<f:num>`) per 10 million words. For noun-verb ambiguities,

```
REMOVE (<fr<5> N) (0 (<fr>65> VFIN)) (-1C N) ;
```

will heuristically discard rare (`fr<5`) noun readings in the face of a competing frequent (>65) finite verb reading, if there is a safe (C) noun immediately to the left (-1). Adding such heuristic cut-off rules with gradually more relaxed thresholds in the higher/later rule sections, instead of "absolute" heuristic rules (e.g. `REMOVE N (0 V`)) can measurably increase performance of an existing grammar (Bick 2009).

Obviously, frequency can also be used in less heuristic rules, as one condition of several, especially in `NOT` or `NEGATE` contexts. Frequency can also be made safer my

directly using ngram frequency tags rather than unigram frequencies, and acting on them depending on ±1 and ±2 contexts. For the rule above, that would mean using "`fr-NaN`" (N after N) and "`fr-VaN`" (V after N) rather than just "fr". For Danish, this would prevent the rule from applying after type or quantity nouns that take noun complements (e.g. 'en *flaske* vand' - a bottle [of] milk). Similarly, the Portuguese PALAVRAS parser (Bick, 2014) uses verb tags like `<fSUBJ/H:45>` and `<fACC/deverbal:51>` meaning that the verb in question has a 45% probability that its subject is a person, and a 51% likelihood that its object is a deverbal noun (action/activity/process/event).

Finally, numerical tags can pass on application dependent information from a non-CG module. In a spellchecking context (cp. section 5.4), this could be confidence values for correction alternatives, based on letter and sound similarities. For the latter, the simplified rule

```
SELECT (<PHONSIM=MAX> N) IF (-1 ART OR DET) ;
```

will select the phonetically most likely noun (N) alternative, if the left context is an article or a determiner.

3.8 Grammar-level parameters and flow control

CG-3 supports several means of grammar-text interaction absent in older CGs, one being the addition or removal of tokens (`ADDCOHORT`, `REMCOHORT`) and the above-mentioned scope control, where grammars or individual rules can resize their context window across sentence boundaries, using the W-suffix for unbounded contexts. And by defining matching bracket pairs, a grammar can be allowed a first pass on simplified text with removed bracket content, where tokens left and right of a parenthesis can "see" each other as ±1 neighbors, rather than risking being `BARRIER`-isolated by material within the parenthesis.

Another important feature is the possibility of using parameter variables that can be set or unset in the data stream or dynamically-contextually controlled by the grammar itself. Variables may, for instance, indicate the applicative context (e.g. spellchecking) or a genre (e.g. recipe). Rules can then react to the former (`VAR:spelling`) by relaxing e.g. np agreement rules, or to the latter (`VAR:recipe`) by not discarding sentence-initial imperatives. While an application-based parameter will be fixed throughout, a genre parameter may have to be changed, either drawn from a corpus or newspaper section header up-front, or "concluded" by a CG rule base on a bag-of-words trigger in the window of analysis - for the recipe scenario e.g. food items, units or cooking verbs. The genre information will then be visible for other rules and further sentences until unset or changed to a different genre.

Genre adaptation adds to the inherent robustness of CG systems, and often comparably few changes will lead to marked improvement for a new genre. In this, CG has a methodological advantage compared to ML systems[11], because it is easy to identify

[11] The rationale for this is that an ML system basically is a snapshot of the linguistic knowledge contained in its training data, and therefore will need new training data for each new genre in order to perform optimally.

the rule culprit for an error once identified, without the need of completely new training data.

Finally, CG-3 grammars are - to a certain degree - self-organizing. Unlike earlier CG compilers, CG-3 applies rules strictly sequentially, and each rule is run on all cohorts in a window before the next rule is tried. This makes rule tracing more predictable, but it also facilitates grammar self-organization. Thus, we allow context-triggered `JUMP`s to rule `ANCHOR`s, or to `INCLUDE` additional rules from a file or to call `EXTERNAL` programs. For instance, rule sections can be used or ignored depending on a genre parameter, or an early rule in the POS section can scan the window for verbo-nominal ambiguities, and if there are none, bypass the rule section in question, increasing efficiency.

3.9 Efficiency and hybridization

The CG-3 compiler has been optimized for efficiency and maintains the same speed as VISLCG despite the added computational complexity represented by regular expressions, numerical tags and variables. On an ordinary desktop machine, a full morphosyntactic grammar (5-10,000 rules) will annotate about 1000 words per second. Though the above-mentioned complexities, as well as the fixed rule order, prevent the use of ordinary FST technology, hybrid compiler techniques may allow substantial performance improvements. Thus, Yli-Jyrä (2011) showed that considerably higher speeds are possible for those rules that remain compatible with VISLCG, using a double finite-state representation, where rule conditions are matched against a string of feature vectors that summarize compact representations of local ambiguity. Therefore, by lumping rules by type and declaring some sections VISLCG-compatible and order-insensitive, a grammarian could allow the rule compiler to apply FST techniques to those sections and restricting current CG-3 compilation to the remaining sections, improving overall performance.

While compiler hybridization has not been implemented, CG-3 already supports some hybridization at the grammar level. Thus, rules with the `EXTERNAL` operator can call an external program to process the active window, triggered by a target and making use of context conditions in the usual way.

4. CG-3 parsers

Today, most actively developed CG parsers use the CG-3 formalism, and many older CG's have been re-implemented in CG-3 It should be noted, however, that there are considerable differences as to which features of CG are used, depending on the scope and purpose of the individual parsers. For instance, in re-implementations of older systems, the bulk of the rules will still `be REMOVE, SELECT, ADD` and `MAP` rules, but will now make use of CG-3's increased Boolean and regular expressions to make the rules more efficient. MT grammars will have a greater need for the `MOVE` operator (for syntactic transfer), NER grammars and format filtering grammars (e.g. UD adaptation of treebanks) may exploit the tokenization features, named relations are useful for Framenet tagging, and spell- and grammar-checkers need `SUBSTITUTE` rules and possibly confidence weighting in the form of numerical tags.

Another difference are descriptive conventions and category definitions, that vary somewhat between research groups. For instance, the POS classification and subdivision of function words differ depending on teaching tradition, and there is a gray zone as to which participles get "promoted" to being adjectives or adverbs, whether ordinals are numerals or adjectives, or what constitutes a named entity. At higher linguistic levels, differences get more pronounced. Thus, the syntactic tag inventories are more diverse than those for POS, and less diverse than semantic tags. Notationally, Latin category abbreviations are quite homogeneous, but some systems use different vernacular abbreviations depending on the language being parsed, and some - e.g. both Norwegian research groups (Oslo/Tekstlab and Tromsø/Giellatekno - have switched from Karlsson's original upper-casing of primary tags to lower case for both primary and secondary tags.

In terms of multi-lingual unified annotation systems, and running a certain risk of over-simplification, one can say that CG-3 parsers come in three main flavors, VISL (SDU & GrammarSoft), Giellatekno (UiT, Moshagen, Pirinen & Trosterud, 2013) and Apertium (Forcada, 2011 and Forcada & Tyers, 2016), with the latter focused on machine translation, and the first two covering a broader spectrum of purposes. In addition, CG-3 is used in a number of further research or development initiatives, such as SALAMA (Swahili and other languages, Hurskainen, this volume) and Estonian CG (EstCG[12]) at Tartu University.

In terms of language range, VISL (12 CG languages) covers mainly the Germanic and Romance languages, almost all with mature grammars (thousands of rules), dependency and often semantic annotation. Giellatekno (17 CG languages[13]) focuses on the Sámi and neighbouring languages (e.g. Finnish and Faroese), but it also has experimental CGs for various small Uralic languages. In addition, it functions as a repository for external projects such as Irish and Cornish CGs. Apertium (20 languages[14]) mainly uses CG-3 for morphosyntactic disambiguation in its MT pipeline and most of its 15 from-scratch grammars are small (< 300 rules), but it also re-uses five larger open-source grammars from Giellatekno and other sources.

With Giellatekno and Apertium covered in separate chapters in this volume, we will here focus on the VISL initiative. Table 1 shows the linguistic annotation levels for each VISL language and lists major applicational areas. All systems feature mature grammars and enjoy extensive lexicon support (Table 2).

[12] https://github.com/EstSyntax/EstCG

[13] North, South and Lule Sámi (with dep.), Inari Sámi, Norwegian Bokmål (dep.), Finnish, Russian. Uralic languages in the making (5): Estonian, Hill Mari, Komi, Komi-Permyak, Meadow Mari.
Other (5): Cornish, Faroese (with dependency), Finnish Kalo, Greenlandic (with dependency), Irish. [Source: https://giellatekno.uit.no/index.eng.html, 16 Aug 2022]

[14] External: Oslo-Bergen tagger (Bokmål & Nynorsk), Giellatekno (North-Sámi, Faroese) and Finnish (Fred Karlsson). From-scratch: Estonian, Danish, Icelandic, Spanish, Catalan, Irish, Welsh, Breton, Russian, Serbo-Croatian, Macedonian, Turkish, Kazach, Tatar [Source: https://wiki.apertium.org/wiki/Constraint_Grammar, 16 Aug 2022]

Table 1: VISL CG-3 parsers, linguistic overview

System (language)	Morph. Syntax	Dep.	NER	Sem. roles	Framenet	Applications
PALAVRAS (Portuguese)	+	+	+	+	+	Corpus, CALL, QA, IE
DanGram (Danish)	+	+	+	+	+	Corpus, CALL, MT, Proofing, WikiTrans
EngGram (English)	+	+	+	+	+	Corpus, CALL, MT
GerGram (German)	+	+	+	+	+	Corpus, CALL, MT, Proofing
SweGram (Swedish)	+	+	+			Corpus, CALL, MT, WikiTrans
NorGram (Bokmål)	+	+	+			Corpus, CALL, MT
NedGram (Dutch)	+	+	+			CALL,
EspGram (Esperanto)	+	+		+	+	Corpus, CALL, MT, WikiTrans
HISPAL (Spanish)	+	+		+		Corpus, CALL
FraG (French)	+	+				Corpus, CALL
ItaGram (Italian)	+					Corpus, CALL
GCG (Greenlandic)[15]	+	+				Corpus, MT

Table 2: VISL CG-3 parsers, lexicon and grammar sizes

System (language)	CG rules all levels, analysis [+ applications[16]]	lexicon size (general)	lexicon size (valency and/or semantics)
PALAVRAS (Portuguese)	16,275 [+982]	85,000	~ all
DanGram (Danish)	17,035 [+4750]	144,000	~ all
EngGram (English)	9,514 [+2136]	307,000	290,000
GerGram (German)	7,607 [+1715]	712,000	682,000
SweGram (Swedish)	11,353 [+200]	141,000	62,000
NorGram (Bokmål)[17]	15,450 [+740]	80,800	~ all
NedGram (Dutch)	3,948	58,800	16,600
EspGram (Esperanto)	3,356 [+855]	58,600	57,200
HISPAL (Spanish)	7,153	132,000	63,600
FraG (French)	2,065	180,000	100,000
ItaGram (Italian)	3,653	77,800	39,400
GCG (Greenlandic)[18]	7,376+1,365 [+500]	97,000	41500

[15] The Greenlandic VISL pipeline is an adapted MT pipeline, containing both VISL-normalized modules (1,365+500 rules, e.g. dependency, additional disambiguation and semantics) and Oqaasileriffik's original morphosyntactic CG (https://oqaasileriffik.gl/en/langtech/live/, 7,376 rules), all of which use CG3.

[16] In addition, there are application-oriented CGs, for instance transfer CGs for machine translation, or spell- / gramma-checkers

[17] After morphological analysis, the Norwegian pipe converts Norwegian lemmas to Danish lemmas, then runs (most of) the Danish CGs and finally reinserts the Norwegian lemmas.

More than half of the VISL parsers perform annotation beyond morphosyntax and dependency trees, adding semantic information such as NER, semantic prototypes, frame structures and semantic roles. Here, CG is used for a variety of tasks, such as semantic disambiguation and the (multi-word) tokenization and classification of named entities. Development was often driven by cooperative international projects. Thus, DanGram was used in the Nordic *Nomen Nescio* NER initiative (Bondi Johannesen et al. 2004) and the Nordic Treebank Network[19], and PALAVRAS participated in the shared Portuguese HAREM NER evaluations, performing at the top of the field (Bick 2006). The framenets[20] were developed specifically for CG use, exploiting the parsers' dependency links, valency tags and semantic noun classes for frame and role mapping and disambiguation (Bick 2011b and Bick 2012a). On top of semantic dependencies, CG-3's named-relation feature is used for frame arguments with multiple heads (e.g. relative clauses and infinitive clauses). Another type of high-level annotation performed by some of the parsers are anaphora (Portuguese, Bick 2010a) and rhetorical structure (English, Bick 2018).

Most of the parsers achieve F1-scores of 99% for POS/morphology and 95-96% for syntactic function, if evaluated on standard written language data such as news or literature. Errors will propagate into the dependency module, affecting performance, but not dramatically, e.g. F=92.0 and F=94.2 for labeled and unlabeled attachment, respectively, for German Twitter data (Bick 2020a), as opposed to 93.6 for syntactic function alone. Error rates for semantic annotation are higher. Performance is best, if semantic roles and verb senses are co-tagged with full lexicon support for frames and semantic ontologies. Thus, in recent evaluations of framenet annotation, GerGram achieved an F-score of 93.6% for German verb frame senses (Bick, 2022b). Corresponding results for Portuguese were F=92.2 for verb frame senses and F=96.1 for argument roles (Bick 2022c).

In some projects, the parsers were augmented to handle non-standard input. Thus, EngGram, GerGram and DanGram were used for CMC corpora (computer-mediated communication), where performance ranged from 93% (POS) and 89% (syntactic functions) for English gaming chat (Bick 2010b) to 97% (full morphology) and 92-94% (syntax/dependency) for the above-mentioned German Twitter data. The Portuguese parser (PALAVRAS, Bick 2014a) was repeatedly used for the annotation of historical and speech data. For the former, it achieved F-scores between 95.4 and 97.2 for POS/morphology and between 91.5 and 94.3 for syntactic function (Bick & Módolo 2010c). For transcribed speech, performance was almost on par with written text, with F-scores of 98.6% for part of speech, 95% for syntactic function and 99% for lemmatization (Bick 2012b).

---

[18] The Greenlandic VISL pipeline is an adapted MT pipeline, containing both VISL-normalized modules (e.g. dependency, additional disambiguation and semantics) and Oqaasileriffik's original morphosyntactic CG (https://oqaasileriffik.gl/en/langtech/live/), all of which use CG3.

[19] https://cl.lingfil.uu.se/~nivre/research/nt.html

[20] cp. httpd://framenet.dk for a category inventory and definitions. The framenets are highly compatible, using the same frame inventory, and achieve almost full coverage of the verb lexicon. Noun frames are treated for deverbal nouns and for nouns with a valency entry in the lexicon.

5. CG-3 applications

5.1 Corpus annotation

All VISL parsers have been used for extensive corpus annotation. On its own corpus site, CorpusEye (http://corp.hum.sdu.dk), VISL offers graphical, menu-based search access to about 80 different corpora in 13 languages, covering very different genres and ranging from small specialized data sets to multi-billion-word general corpora, most of them password-free. The search interface allows queries involving all existing tag types (wordform, lemma, POS, syntax, semantic type, role or frame) or combinations thereof, with boolean operators and regular expressions. Searches can involve chains of tokens and dependency relations. Resulting concordances can be inspected, exported and evaluated statistically. It is even possible to build sub-corpora on the fly before the actual search.

Sometimes, however, a search pattern is difficult or cumbersome to implement, or simply counter-intuitive in a cross-language perspective. Good examples are complex (auxiliary-based) tenses, aspect or passive that may be realized as simple inflection in another language, or definiteness, which is marked with inflection or enclitics in some languages (e.g. Danish, Swedish), but distributed across the noun phrase in others (e.g. German, English). In these cases, corpus use can be simplified by post-processing the annotated corpus with a small "search CG"[21] using sentence context to add "local" tense or aspect tags on the main verb, and a tag for definite/indefinite on np heads (or for that matter, on all parts of an np). Such a "search CG" can also be used to propagate semantic information from nouns to anaphorical pronouns, or to introduce zero-constituent tokens, depending on the needs of the corpus user.

Another specialized corpus interface is DeepDict (http://deepdict.com, Bick 2010d), a relational corpus-based lexicon, which for a given search word will show graphical overviews of dependents or heads with the highest co-occurrence strength (e.g. adjective-noun, verb-subject, verb-object), computed as mutual information, i.e. by normalizing co-occurrence frequency for the individual frequencies of head and dependent in a background corpus. DeepDict was built from CG annotated dependency corpora and provides a kind of semantic and usage overview for a given lexeme, making visible implicit corpus information in ways an edited, active dictionary would. This has obvious lexicographic and teaching uses. For instance, the differences between the English adjectives 'big', 'high' and 'tall' become obvious from the kind of nouns they combine with, indicating that the first is used for general size or importance, the second for measurable features and the third for height. In addition, DeepDict picks up fixed idiomatic expressions such as "big bang" or "tall story".

The inherent robustness of the CG reductionist approach means that a general parser can be used for a specialized genre with few rule adaptations, as long as there is some lexical support for the genre in question. Typical project work for genre adaptation will thus consist of lexicographical part - or finding a heuristic solution to lexical variation -

[21] The method was used extensively for the Portuguese Linguateca corpora (http://linguateca.pt), and specifically the Portuguese *Floresta Sintá(c)tica* treebank.

on the one hand, and tracing and correcting rules that under-perform on the new data. For instance, the Portuguese parser (PALAVRAS, Bick 2000) could be made to handle historical data (Bick & Modolo 2010c) by crafting a preprocessor that normalized[22] old and Spanish-style orthography and tokenization (e.g fusion of clitics, prepositions and other function words), while at the same time adapting a number of rules with too strict word order conditions. Similarly, PALAVRAS was adapted for the annotation of transcribed speech in the C-ORAL-Brasil (Bick 2012b) and NURC (Bick 2019) projects. Here, a lexicon-extension file was used for spoken variants, additional interjections etc. The parser itself could be made to work with almost normal accuracy by tagging and CG-disambiguating non-verbal units (laughter, cough) and pauses of different lengths as either commas, periods or "non-breaking". This provided the CG grammar with the necessary delimiters and prevented rules with unbounded context conditions from overshooting and making errors.

The arguably most ambitious genre-adaptation of any VISL parser was carried out for CMC data (computer-mediated communication) for the XPEROHS project (Baumgarten et al. 2019) that investigated hatespeech in a large Twitter- and Facebook-corpus of German and Danish. Here, automatic normalization of orthographical errors, abbreviations and interjections was used to provide recognizable input to the standard parsers (Bick 2022). As a side effect of ordinary disambiguation, the existing CGs would also resolve ambiguous normalization, for instance in the case of wrongly lower-cased German nouns. Since the data proved to contain many new word creations, compound analysis was improved in order to prevent out-of-vocabulary (OOV) problems for the parsers and to provide contextual CG rules with semantic information weaned from the individual parts of a compound. Finally, sentiment mark-up was refined by annotating emoticons and emojis as adverbs[23] (Bick 2020b) with 10 different "lemmas" (e.g. happy, love, laughter, skeptical, sad, angry etc.).

5.2 Treebanking

In a sense, all VISL parsers support treebanking, because they are used for the live generation of syntactic trees for VISL's visualizers and grammar teaching tools, and for dependency annotation of the CorpusEye corpora. But there are also a number of "real" treebanks - in the sense of manually revised sentence repositories - that have been built using VISL parsers. The biggest and most refined ones are the *Floresta Sintá(c)tica* for Portuguese (Afonso et al. 2002) and the Arboretum treebank for Danish (Bick 2003), created by post-editing output from the PALAVRAS and DanGram parsers, respectively, and both repeatedly post-processed with higher-level linguistic information. *Floresta sint(á)ctica* also added a semi-revised section *(Selva)* to the fully revised news section *(Bosque),* as well as unedited parser output in treebank format (Freitas et al. 2008), including the *Floresta Virgem* (news) and *Amazônia* (mixed text). A third treebank,

[22] In a kind of two-level annotation, the original wordforms were maintained as <O:...> tags throughout the program pipe. In the final output, original and normalized wordforms would then switch places.

[23] The category of adverb was chosen, because it best matched the word order distribution of emojis, and because it interfered least with various BARRIER conditions in CG rules.

L'Arboratoire, was built for French (Salmon-Alt et al. 2004), using output from the FraG parser, also with a revised section *(Ananas)* and a "jungle" section (Europarl data). For Estonian, a small treebank *(Arborest,* Uibo & Bick 2008*)* was created in VISL constituent format using data from the Estonian CG corpus. Later, a large dependency treebank (EDT, Muischnek et al. 2016) was created at Tartu university using CG-3.

Table 3: VISL-style treebanks with a parser base

System (language)	**Size (tokens)**	**Dep.**	**Const ituent**	**Genre**	**Special annotation on top of morphosyntax and tree structure**
Floresta Sintá(c)tica, *Bosque* (Portuguese)	212,700	+	+	news (pt/br)	NER, complex categories
Floresta Sintá(c)tica, *Selva* (partially revised)	300,000	CG	+	news (pt/br)	NER, complex categories
Arboretum (Danish)	423,600	+	+	mixed	NER, semantic ontology
L'Arboratoire[24] (French), *Ananas*	76,700	CG	+	news	-
EDT (Estonian)	437,800	+		mixed	co-references
Arborest (Estonian)	2400	CG	+	mixed	

In order to build tree structures, morphosyntactic CG output requires further grammatical processing. For constituent trees, originally the dominant treebank paradigm and used in the VISL grammar teaching tools, a special type of phrase structure grammars (PSGs[25]) were crafted, where terminals are not tokens, but higher-level CG categories such as POS and syntactic function. This made it possible to create full, bracketed tree structures with relatively small grammar sizes (~ 200 rules). In order to resolve certain structural ambiguities, CG input was enhanced with special tags for e.g. close and long postnominal attachment and coordination. For dependency, Perl-compiled external attachment grammars (~ 300 rules) were built (Bick 2005), using CG syntactic function tags with attachment direction markers[26]. Ultimately, the creation of dependency treebanks proved to be more efficient, even as an intermediate step for later constituent conversion, not least after CG-3 introduced direct CG support for dependencies (`SETPARENT`, `SETCHILD` operators and p(arent), c(hild), s(ibling) contexts). Another method to add constituent information to CG is chunking. For this, CG-3 `ADDCOHORT` feature can be used to add opening and closing brackets (Bick 2013). With nested chunking, full PSG trees can also be achieved.

Both *Floresta, Arboretum* and *EDT* have been converted to non-native formats such as PENN tree format, MALT xml and TIGER xml. VISL offers downloadable Perl scripts[27]

[24] https://corp.hum.sdu.dk/tgrepeye_fr.html
[25] The PSGs used a tailor-made in-house C++ compiler.
[26] The arrows in @SUBJ> and @<SUBJ, for instance, indicate whether the main verb head is to be found right (>) or left (<).
[27] https://visl.sdu.dk/treebanks.html#Transformation_tools

for such conversions. Conversion between dependency and constituent formats is also supported, but risks introducing errors because the dependency format tends to under-specify coordination and ellipsis, while the PSG format doesn't specify heads. To minimize such errors, preparatory CG enrichment may be necessary.

With the advent of the Universal Dependencies (UD) initiative (McDonald et al. 2013), most treebanks offer a converted UD version, e.g. Rademaker et al. (2017) for the *Floresta* treebank. However, depending on UD guidelines, this process may involve structural and category changes as well as tree-flattening and other information loss. Addressing these issues, Bick (2016) shows how CG-3 can be used to perform the necessary retokenization (MWE and contraction splitting), reattachment (semantic dependencies, copula- and auxiliary flattening) and category renaming or even contextual category re-tagging (especially function words).

5.3 ICALL

ICALL was the original motivation, in 1997, for the introduction of rule-based NLP tools for visual interactive syntax learning (VISL) in a large, 3-year digitalization project at the Institute of Language and Communication (ISK, Odense University). The purpose of the project at the time was to build visual interactive online tools for grammar teaching in general, and syntax in particular. To this end, small treebanks were to be built for the languages taught at ISK (English, German, French, Spanish and Danish), to provide an annotated sentence base for the teaching tools. Inspired by an existing Portuguese parser (Bick 1996), which supported interactive category checking for live sentences (Bick 1997), Constraint Grammar was adopted as the method of choice to not only facilitate manual treebanking, but also allow live, realistic and user-centered input to the visualizers and other teaching tools. While the parsers were either built from scratch (most languages), or licensed and augmented (first versions for English and German)[28], the VISL team also set down to standardize a cross-language grammatical category set[29] for both POS, morphology and syntax, including a systematic convention for abbreviations and subcategories, as well as a graphical mnemonic color- and symbol-coding[30], progressing from the original 6 languages to the 10 languages of the Danish secondary education, and ultimately involving 28 languages with treebanks of varying sizes 20 years before the advent of Universal Dependencies. This unified tagset allowed direct typological comparisons between the languages and facilitated teaching synergy across study lines. In an effort to expand this synergy from the university level to the

[28] LingSoft's ENGCG (morphosyntax) and GERCG (POS/morphology only).
[29] Principles and category sets are explained both in general and in the individual language sections at visl.sdu.dk, e.g. (e.g. https://visl.sdu.dk/treebanks.html and http://visl.sdu.dk/sentencelab_gym.html). For a university-level overview, see https://visl.sdu.dk/lecture_notes.html. For a story-line introduction targeting primary schools, see (Bick 2002) or https://visl.sdu.dk/visl/light.
[30] Colours were used for POS, e.g. stable blue for nouns, energetic red for verbs. The symbols were used for syntax, an expansion of the Danish cross-circle-triangle system for subject-predicator-object.

entire Danish school system[31], age- or school-type-specific treebanks were constructed, while also limiting and simplifying the grammatical categories used at each level of teaching. The actual teaching tools included a tree-manipulator for syntactic trees and about 10 games[32] exploiting data from both the treebanks and live CG annotation (e.g. WordFall and SynTris, Tetris-like games for POS and syntactic chunking & labeling, respectively).

Between 2002 and 2006, VISL spear-headed the Nordic PaNoLa[33] cooperation (Bick 2004), a cross-language initiative to build or enhance Constraint-Grammar parsers for the Nordic languages, and to integrate other Nordic languages with the existing Danish VISL teaching site. PaNoLa, addressing all Nordic languages, was funded by the Nordic Council of Ministers and had participating institutions in all Nordic countries.

5.4 Machine Translation (MT)

The CG-3 brand of Constraint Grammar has become a methodologically important paradigm in rule-based machine translation (RBMT). Conversely, all major CG initiatives have an interest in MT. Thus, there is work on Sámi and Finnish MT at Giellatekno, while VISL/GrammarSoft offers MT for Germanic languages, Portuguese and Esperanto. Both approaches graft MT on top of mature CG systems with a pronounced semantic element, whereas most Apertium systems to date only use CG-3 for low-level disambiguation. The oldest system, Dan2Eng (Bick 2007) was launched by GrammarSoft one year before Google started to offer Danish-English MT. Dan2Eng, Eng2Dan and its sister systems for German, Swedish and Esperanto introduced CG-derived lexical transfer rules in their MT dictionaries, selecting one or other translation equivalents based on CG labels[34] on the word itself, as well as neighbouring or dependency-linked tokens. CG proper is used for preparing the transfer stage, propagating morphological feature and adding or disambiguating categories necessary for the target language (TL), but absent or under-specified in the source language (SL). GrammarSoft's MT systems originally handled syntactic transfer with Perl-compiled external movement grammars, but newer systems, notably the Greenlandic-Danish system[35] (Bick 2022), use the CG-3 `MOVE` operator for this purpose, operating on constrained dependency treelets (sub-trees) rather than individual words. Both methods have equivalent results in straight-forward cases, but the `MOVE` operator has greater expressive power because it allows restrictions on the inclusions of dependency daughters in the treelets and because it harnesses the full power of CG-3 for the contextual conditions. In addition to movements, CG-3's `ADDCOHORT`

[31] For an overview of school target groups, see "teaching projects" at http://visl.sdu.dk
[32] Most tools were programmed as Java applets, a fact that has become a problem for VISL today, as most browser version have ceased to support the Java platform for security reasons. For Chrome, the extension CheerpJ Applet Runner is a possible solution.
[33] https://visl.sdu.dk/panola.html
[34] These rules can make full use of regular expressions, and matches can be optional, negated or linked.
[35] https://nutserut.gl/ (official site) or https://visl.sdu.dk/visl/gl/tools/translation.html (VISL academic version)

and REMCOHORT operators are used at the syntactic transfer stage to insert or remove pronouns, prepositions and particles that have no equivalent in the other language.

In an effort to make Wikipedia content in large languages available (and searchable!) in smaller languages, VISL/GrammarSoft launched the WikiTrans initiative (http://wikitrans.net). Here, one system translates English into Esperanto (Bick 2011c)[36], another Swedish into Danish (Bick 2014b). Both target languages have relatively small Wikipedias (~200,000 articles), and also smaller articles than Swedish and English (millions of articles). Wikipedia html is parsed and reconstituted after translation, relinking visual and other non-textual content. In both cases translations have a very high quality and may appear untranslated at a quick glance. To keep the system up-to-date, new and change articles are continually retranslated using a dedicated 32-core computer cluster at SDU.

5.5 Proofing tools

CG has a long history of supporting spell- and grammar checkers. Thus, LingSoft supplied tailor-made CG-1 tools for Microsoft Word for many years, e.g. for Swedish (Arppe 2000, Birn 2000), and Giellatekno offers an FST- and CG-3-based spell- and grammar-checker for Sámi language(s), Divvun (e.g. Wiechetek et al. 2019, for North-Sámi).

Also using CG-3, GrammarSoft has built a Danish spell- and grammar-checker (Bick 2015b) that is distributed as *RetMig* (http://retmig.dk) for online use and as a Word- or browser plugin, an as *DanProof* for command-line use in a language technology (LT) context. In addition, there is a CG-based comma-checker *(Kommaforslag[37])*, a spell- and comma-checker for German *(Kommatroll[38])* and an English comma-checker *(Commatizer[39])*. The Danish spell-checking component starts out with a ranked list of correction suggestions with confidence values, computed from Levenshtein-distances for both written words and phonetic transcriptions, as well as frequencies and keyboard features.

The DanGram parser is then used for both annotation and disambiguation, where one module is spell-checking only, while rules in the other (ordinary) modules may refer to the presence and confidence value of spellchecker suggestions[40]. Optimally, DanGram will weed out suggestions where POS or other tags are in conflict with the sentence context. At the same time APPEND rules in the spellchecking module will address possible real word errors (e.g. 'og' for 'at'), and MAP or ADD rules will add grammatical error tags such as @vfin/@inf (finity errors), @sg/@pl (number inflection errors), @def/idf (definiteness inflection errors) and so on. The spelling module is run three

[36] In the case of English-Esperanto, Wikipedia users can also post-edit translations or chose to see links to a WikiTrans article within the original Esperanto Wikipedia, if no original article exists for the search term.
[37] http://kommaforslag.dk
[38] http://kommatroll.com
[39] http://commatizer.com
[40] The correction suggestions are listed as secondary <R:...> tags alongside ordinary CG tags, and are as such "visible" to the CG rules.

times, at increasingly higher levels of the DanGram pipe. This way, early error mappings can, for instance, address agreement errors *before* they can trigger a wrong POS disambiguation base on rules operating on feature-attribute unification. At the same time, more difficult or risky error mappings can wait for DanGram to establish syntactic tags or relations. The last module is a morphological generator that will create a correctly inflected correction suggestion from the grammatical error tags, in the same format as the graphical/phonetic spellchecking suggestions. All in all, there are over 40 error types. In addition, the grammar recognizes proper nouns and compounds as such with a high accuracy, even if they are not listed in the lexicon, allowing the user to ignore OOV words that are likely real words.

For comma checking, a similar method is used, marking wrong commas or adding tags for missing commas on the subsequent word. Again, tag names specify the type of error (e.g. subclause comma or apposition), and a distinction is made between optional and obligatory commas. In order to capture stray commas in unconventional places, DanGram removes many commas before the comma-checker run, reinserting them afterwards. This way, commas can be tagged as wrong, if the comma-checker doesn't mark them as missing, if removed.

Comma and grammar correction are arguably among the LT tasks that are most difficult to address with machine learning, and - conversely - profit most from the exact structure of a sentence. For instance, marking subclause commas is fairly straightforward given a complete sentence tree and a markup for types of subclauses, but it is a hard task if only surface wordforms can be used, because most language only have lexical markers for the start, not the end of a subclause, an even those (conjunctions, relatives, interrogatives) may be ambiguous or missing, as they often do in Danish. However, our rule-based error-mapping setup has another, pedagogical advantage that is relevant for both grammar and comma checkers: Since errors are not only marked, but classified, it is possible for the editing interface to explain the errors and to provide usage examples.

6. Export filters and notational conversion

In section 5.1 we have discussed parser adaptations for specific genres or corpora. Another scenario for parser adaptation is notational adaptation. Thus, output filtering, i.e. removing unwanted tags or renaming tags depending on the linguistic preferences of a user, is often necessary for external projects, but relatively straightforward for native CG output, because all tagging information is on one line with the annotated token, so it can be changed with a line-by-line substitution script, for instance changing a participle into an adjective if it carries a prenominal syntactic function tag. Sometimes such a line-based conversion has to be prepared with a CG rule, because the necessary information is tagged on another token. For instance, one may want to convert an adjective into a noun, if it occurs as head of an np ("The fast and the furious")[41]. In this case, a neighbouring prenominal or an incoming dependency link can be used by a CG rule to map an <n> tag ("noun-like") on the adjective.

[41] In German, such adjectives are written in upper case, like nouns, so the information is present on the same line.

Structural notational conversion is more difficult, but more systematic across languages. For VISL CG output, there are standard conversion programs for xml (a refined version of MALT xml), the VISL treebank format (constituent trees) and the TIGER treebank format (either constituent trees or dependency trees), as well as tailor-made XML-converters that maintain and enrich existing xml mark-up of individual projects (e.g. NURC, Bick 2019a).

Most difficult is structural descriptional conversion, which may involve information loss (easier) or - conversely - demand additional information, not explicit in the input. Treebank conversion between dependency and constituent trees (see section 5.1.2) is a classical example of this. Another example is retokenization, where VISL output has to be simplified to the space-based, non-linguistic tokenization often used in ML research. VISL's native tokenization i linguistically motivated, so it splits, for instance, preposition-article contractions common in Romance languages, so the subsequent np can "get" its article (or determiner). This split has to be undone by the converter, using the parser's markers for first and second part. Conversely, certain multi-word expressions (MWEs), e.g. names and multi-part conjunctions, need to be split. In CG3, this can be done with cohort-manipulating rules (`ADD-, REM-, SPLIT-, MERGE-COHORT`). In this process, non-trivial information has to be added, e.g. assigning POS and syntactic tags to newly split token-parts, as well as adding or removing outgoing dependency links and reattaching incoming ones.

7. ML grammar optimization

Because the application of a given CG rule will change the grammatical context for all subsequent rules, and because of section-based rule iteration, any Constraint Grammar of realistic size will be subject to complex rule interactions. Though instance-level errors can be traced and corrected, it is practically impossible for the grammarian to meaningfully rearrange the grammar as a whole, or to systematically constrain or relax under-performing rules. In a hybrid approach, however, machine learning (ML) can be used for such optimization, as has been shown for simple ML-learned template rules (Lager 1999) and induced `REMOVE` rules (Lindberg & Eineborg 1998), but also for existing, human-made rules, as shown in (Bick 2013b) for the Danish DanGram parser. In this setup, the grammar was iteratively changed and measured against a gold corpus. Rules were "killed" (removed from the grammar) if they made more errors than correct changes, while rules with an error percentage below a (experimentally optimized) threshold were moved one section up, otherwise 1 section down. Kill-operations were the most effective and prioritized recall, while promoting and demoting improved precision. Other methods examined were resectioning, sorting all rules in a section according to their error percentage or changing the strictness of a rule (adding or removing the C [unambiguous] condition to contexts and `BARRIER`s). On the original, mature grammar, ML optimization reduced errors by 7%, corresponding to an F-score improvement of 0.41 percentage points. Interestingly, the effect was twice as large with a reduced grammar, where 50% of the rules had been removed, indicating the usefulness of the method for small grammars or genre adaptation, where only part of the rules is applicable due to lexical or grammatical reasons. Similarly, ML optimization can be used for porting

grammars between related languages, by identifying and optimizing the (Danish) rules that do work for English[42] (Bick 2014c). Thus, when used for English data, an ML-tuned DanGram outperformed (F=92 for POS) both the statistical baseline (F=85.7) and the unmodified grammar (F=86.1).

8 Conclusion

With its methodological rather than linguistic theory-based approach, Constraint Grammar has always been a paradigm on the move, adapting to the needs of a project and the task at hand. In this vein, CG-3 has adopted and adapted methods from various other parsing strategies (unification grammar, PSG, dependency grammar, statistical approaches). The VISL parsers and other CG-3 systems have proven their accuracy and robustness in hundreds of corpus and application projects. Last not least, CG-3 and VISL have shown how an open source compiler can support mixed-resource infrastructures with both academic/public and commercial/private elements and uses.

References

Järvinen, Timo and Pasi Tapanainen, 1997:
A Dependency Parser for English. *Technical Reports*, No. TR-1. Department of General Linguistics, University of Helsinki.

Afonso, Susana, Eckhard Bick, Renato Haber & Diana Santos, 2002:
"Floresta sintá(c)tica": a treebank for Portuguese. In Manuel González Rodríguez & Carmen Paz Suárez Araujo (eds.), *Proceedings of LREC 2002*. ELRA, 2002, pp.1698-1703.

Arppe, Antti, 2000:
Developing a grammar checker for Swedish. In: Nordgård, T. (ed.) *Nodalida'99 Proceedings*. Department of Linguistics, University of Trondheim. pp. 13-27.

Baumgarten, Nicole; Bick, Eckhard; Klaus, Geyer; Iversen, Ditte Aakær; Kleene, Andrea; Lindø, Anna vibeke; Neitsch, Jana; Niebuhr, Oliver; Nielsen, Rasmus; Petersen, Esben Nedenskov, 2019:
Towards Balance and Boundaries in Public Discourse: Expressing and Perceiving Online Hate Speech (XPEROHS). In: Mey, Jacob; Holsting, Alexandra; Johannessen, Christian (ed.): *RASK - International Journal of Language and Communication*. Vol. 50., pp. 87-108. University of Southern Denmark.

Bick, Eckhard, 1996:
Automatic Parsing of Portuguese. In: García, Laura Sánchez (ed.), *Anais / II Encontro para o Processamento Computacional de Português Escrito e Falado*. Curitiba: CEFET-PR.

Bick, Eckhard, 1997:
Internet Based Grammar Teaching. In: Christoffersen, Ellen & Music, Bradley (eds.), *Datalingvistisk Forenings Årsmøde 1997 i Kolding, Proceedings*, pp. 86-106. Kolding: Institut for Erhvervssprog og Sproglig Informatik, Handelshøjskole Syd.

[42] Obviously, category-based targets and contexts are more relevant for the new language than lexeme- or lemma-constrained rules.

Bick, Eckhard, 2000:
The Parsing System Palavras - Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Famework, Aarhus: Aarhus University Press.
Bick, Eckhard, 2001:
The VISL System: Research and applicative aspects of IT-based learning. In: Anna Sågvall Hein (ed*.), Proceedings of the 13thNordic Conference of Computational Linguistics (NODALIDA 2001)*. ACL Anthology W01-174.
Bick, Eckhard, 2002:
Grammy i Klostermølleskoven - "VISL light": Tværsproglig sætningsanalyse for begyndere (Teaching Manual), Århus: Mnemo.
Bick, Eckhard, 2003:
Arboretum, a Hybrid Treebank for Danish, in: Joakim Nivre & Erhard Hinrich (eds.), *Proceedings of TLT 2003* (2nd Workshop on Treebanks and Linguistic Theory, Växjö, November 14-15, 2003), pp.9-20. Växjö University Press.
Bick, Eckhard, 2004:
PaNoLa: Integrating Constraint Grammar and CALL. In: Henrik Holmboe (red.), *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram 2000-2004* (Yearbook 2003). pp.183-190, Copenhaguen: Museum Tusculanum.
Bick, Eckhard, 2005:
Turning Constraint Grammar Data into Running Dependency Treebanks, In: Civit, Montserrat & Kübler, Sandra & Martí, Ma. Antònia (red.), *Proceedings of TLT 2005* (4th Workshop on Treebanks and Linguistic Theory, Barcelona, December 9th - 10th, 2005). pp.19-27.
Bick, Eckhard, 2006:
Functional Aspects in Portuguese NER. In: Renata Vieira et al. (eds.), *Proceedings of PROPOR-2006* (Itatiaia), pp. 80-89. Springer. pp. 80-89.
Bick, Eckhard, 2007:
Dan2eng: Wide-Coverage Danish-English Machine Translation. In: Bente Maegaard (ed.), *Proceedings of Machine Translation Summit XI*, 10-14. Sept. 2007, Copenhagen, Denmark. pp. 37-43
Bick, Eckhard, 2009:
Introducing Probabilistic Information in Constraint Grammar Parsing. In Michaela Mahlberg, Victorina González-Díaz, Catherine Smith (eds.), *Proceedings of Corpus Linguistics 2009*, Liverpool, UK. Electronically published at: ucrel.lancs.ac.uk/publications/cl2009/
Bick, Eckhard, 2010a:
A Dependency-based Approach to Anaphora Annotation. In: (eds.) *Extended Activities Proceedings, 9th International Conference on Computational Processing of the Portuguese Language* Apr. 27-30. Porto Alegre, Brazil.
Bick, Eckhard, 2010b:
Degrees of Orality in Speech-like Corpora: Comparative Annotation of Chat and E-mail Corpora, In: Ryo Otoguru et al. (eds.), *Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation* (Tohoku University, 4-7 November, 2010). pp. 721-729.
Bick, Eckhard & Marcelo Módolo, 2010c:
Letters and Editorials: A grammatically annotated corpus of 19th century Brazilian Portuguese. In: Dermeval da Hora & Camilo Rosa Silva (eds): *Para a História do Português Brasileiro: Abordagens e perspectivas*, Vol. VIII, pp. 29-36. João Pessoa: Idéia Editora.

Bick, Eckhard, 2010d:
DeepDict - et korpusbaseret relationelt leksikon. In: Ruth Vatvedt Fjeld & Henrik Lorentzen (eds.), *Lexico Nordica 17 - 2010, Leksikografi og språkteknologi i Norden.* pp. 17-34 . LexicoNordica.

Bick, Eckhard, 2011a:
A Barebones Constraint Grammar. In: Helena Hong Gao & Minghui Dong (eds), *Proceedings of the 25th Pacific Asia Conference on Language, Information and Computation* (Singapore, 16-18 December, 2011). pp. 226-235.

Bick, Eckhard, 2011b:
A FrameNet for Danish. In*: Proceedings of NODALIDA 2011*, May 11-13, Riga, Latvia. NEALT Proceedings Series, Vol 11, pp.34-41. Tartu: Tartu University Library. (ISSN 1736-6305)

Bick, Eckhard, 2011c:
WikiTrans, The English Wikipedia in Esperanto. In: *Constraint Grammar Applications, Workshop at NODALIDA 2011*, Riga, Latvia. (forthcoming: NEALT Proceedings Series, Vol. 14, pp. 8-16. Tartu: Tartu University Library.

Bick, Eckhard, 2012a:
Towards a Semantic Annotation of English Television News - Building and Evaluating a Constraint Grammar FrameNet, In*: Proceedings of the 26th Pacific Asia Conference on Language, Information and Computation* (Bali, 7-10 November, 2012). pp. 60-69. Faculty of Computer Science, Universitas Indonesia.

Bick, Eckhard, 2012b:
A anotação gramatical do C-ORAL Brasil. In: Tommaso Raso & Heliana Mello (eds), *C-ORAL-Brasil I - Corpus de referência do português brasileiro falado informal*. Chapter 6, pp.223-254. Belo Horizonte: Editora UFMG.

Bick, Eckhard, 2013a:
Using Constraint Grammar for Chunking. In: S. Oepen, K. Hagen & J. B. Joannessen (Eds), *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013),* May 22-24, Oslo, Norway. Linköping Electronic Conference Proceedings Vol. 85, pp. 13-26. Linköping: LiU Electronic Press.

Bick, Eckhard, 2013b:
ML-Tuned Constraint Grammars. In: *Proceedings of the 27th Pacific Asia Conference on Language, Information and Computation*, pp. 440-449. Taipei: Department of English, National Chengchi University.

Bick, Eckhard, 2014a:
PALAVRAS, a Constraint Grammar-based Parsing System for Portuguese. In: Tony Berber Sardinha & Thelma de Lurdes São Bento Ferreira (eds.), *Working with Portuguese Corpora*, pp 279-302. London/New York:Bloomsburry Academic.

Bick, Eckhard, 2014b:
Translating the Swedish Wikipedia into Danish. In: *Accepted Abstracts of The 5th Swedish Language Technology Conference. SLTC 2014*, Uppsala University, 13-14 Nov 2014.

Bick, Eckhard, 2014c:
ML-Optimization of Ported Constraint Grammars. In: Calzolari, Nicoletta et al. (eds.), *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC2014* (Reykjavik, May 28-30, 2014). pp. 3382-3386.

Bick, Eckhard & Tino Didriksen, 2015a:
CG-3 - Beyond Classical Constraint Grammar. In: Beáta Megyesi: *Proceedings of NODALIDA 2015*, May 11-13, 2015, Vilnius, Lithuania. pp. 31-39. Linköping: LiU Electronic Press. ISBN 978-91-7519-098-3

Bick, Eckhard, 2015b:
DanProof: Pedagogical Spell and Grammar Checking for Danish. In: Galia Angelova, Kalina Bontcheva & Ruslan Mitkov: *Proceedings of RANLP 2015* (Hissar, Bulgaria, 7-9 Sept. 2015). pp. 55-62.
Bick, Eckhard, 2016:
Constraint Grammar-based Conversion of Dependency Treebanks. In: *Proceedings of the 13th International Conference on Natural Language Processing (ICON 2016*, Varanasi, India, 17-20 Dec 2016). pp 109-114. NLP Association of India (NLPAI).
Bick, Eckhard, 2018:
Towards an Automatic Mark-up of Rhetorical Structure in Student Essays. In: *Proceedings of CICLing 2018 - 19th International Conference on Computational Linguistics* (Hanoi, March 2018). (online, forthcoming in print)
Bick, Eckhard, 2019a:
Anotação Gramatical do Projeto NURC Digital. Chapter in: *Miguel Oliveira Jr.: NURC - 50 anos*. pp. 195-216. Ipipiranga: Parábola Editorial.
Bick, Eckhard, 2019b:
Dependency Trees for Greenlandic. In: *Proceedings of the 15th Conference on Natural Language Processing (KONVENS 2019)*. German Society for Computational Linguistics & Language Technology". pp 140-148.
Bick, Eckhard, 2020a:
An Annotated Social Media Corpus for German. In: Calzolari, Nicoletta et al. (eds.), *Proceedings of the 12th International Conference on Language Resources and Evaluation, LREC2020* (Marseille, May 2020). pp. 6129-6137. ACL / ELRA.
Bick, Eckhard, 2020b:
Annotating Emoticons and Emojis in a German-Danish Social Media Corpus for Hate Speech Research. In: Mey, Jacob; Holsting, Alexandra; Johannessen, Christian (Eds.): *RASK - International Journal of Language and Communication*. Vol. 52, pp. 1-20. University of Southern Denmark.
Bick, Eckhard, 2022a:
A Modular Machine Translation Pipeline for Greenlandic. In: *Proceedings of The International Conference on Agglutinative Language Technologies as a Challenge of Natural Language Processing (ALTNLP 2022)*. Forthcoming.
Bick, Eckhard, 2022b:
A Framenet and Frame Annotator for German Social Media. In: Nicoletta Calzolari et al. (eds.): *Proceedeings of LREC 2022* (Marseille). pp. 3942-3949.
Bick, Eckhard, 2022c:
PFN-PT: A Framenet Annotator for Portuguese. In: Heliana Mello & Fernanda Farinelli: *The computational treatment of Brazilian Portuguese*. Domínios de Linguagem. [S. l.], v. 16, n. 4, pp. XX-XX, 2022. (In print).
Bick, Eckhard, 2022d:
Lemma Hunting: Automatic Spelling Normalization for CMC Corpora, In: *Proceedings of KONVENS 2022* (Potsdam). Potsdam University. (Forthcoming).
Birn, Jussi, 1995:
A Syntax-Geared Approach to Determiners and Pronouns in Swedish Constraint Grammar. Department of General Linguistics. University of Helsinki.
Birn, Jussi, 2000:
Detecting grammar errors with Lingsoft's Swedish grammar checker. In: Nordgård, T. (ed.) *Nodalida'99 Proceedings*. Department of Linguistics, University of Trondheim. pp. 28-40.
Bondi Johannessen, J., Bick, E., Hagen, K., Hansen, D. H., Haaland, Å., Jónsdottir, A. B., Kokkinakis, D., Meurer, P., & Nøklestad, A. 2004:

The Nomen Nescio Project - Scandinavian Named Entity Recognition. In*: Proceedings of ALLC/ACH 2004*. pp. 1-5.

Chanod, Jean-Pierre & Tapanainen, Pasi, 1994:
Tagging French - comparing a statistical and a constraint- based method. Adapted from: Statistical and Constraint- based Taggers for French, *Technical report MLTT-016*, Rank Xerox Research Centre, Grenoble.

Forcada, L et al. 2011:
Apertium: a free/open-source platform for rule-based machine translation. *Machine translation*, 25(2):127–144.

Forcada, Mikel L. and Francis M. Tyers, 2016:
Apertium: a free/open source platform for machine translation and basic language technology. In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation: Projects/Products*, Riga, Latvia. Baltic Journal of Modern Computing.

Freitas, Cláudia & Rocha, Paulo & Bick, Eckhard, 2008:
Floresta Sintá(c)tica: Bigger, Thicker and Easier. In: António Teixeira et al. (eds.) *Computational Processing of the Portuguese Language* (Proceedings of PROPOR 2008, Aveiro, Sept. 8th-10th, 2008), pp.216-219. Springer.

Karlsson, Fred et al. 1995:
Constraint Grammar: A Language-Independent System for Parsing Unrestricted Text. Berlin: Mouton de Gryuter.

Karlsson, Fred, 1990:
Constraint grammar as a framework for parsing running text. In: Karlgren, Hans (ed.*), Proceedings of 13th International Conference on Computational Linguistics*, volume 3, pp. 168-173, Helsinki, Finland.

Lager, Torbjörn, 1999:
"The μ-TBL System: Logic Programming Tools for Transformation-Based Learning". In: *Proceedings of CoNLL'99,* Bergen.

Lindberg, Nikolaj & Martin Eineborg, 1998:
Learning Constraint Grammar-style disambiguation rules using Progol. In: *Lecture Notes in Computer Science, 1998*, Volume 1446/1998, 116-124, DOI: 10.1007/BFb0027315

Marques, Lluis & Horacio Rodriguez, 1995:
Towards Learning a Constraint Grammar from Annotated Corpora Using Decision Trees. ESPRIT BRA-. 7315 Acquilex II, Working Paper.

McDonald, Ryan, Joakim Nivre, Yvonne Quirmbach-Brundage, Yoav Goldberg, Dipanjan Das, Kuzman Ganchev, Keith Hall, Slav Petrov, Hao Zhang, Oscar Täckström, Claudia Bedini, Núria Bertomeu Castelló, and Jungmee Lee, 2013:
Universal Dependency Annotation for Multilingual Parsing. In *Proceedings of ACL.*

Moshagen, Sjur N., Tommi A. Pirinen and Trond Trosterud, 2013:
Building an open-source development infrastructure for language technology projects. In: *Proceedings of the 19th Nordic Conference of Computational Linguistics (NODALIDA 2013*). Volume 16 of NEALT Proceedings Series. (May 22–24 2013).

Muischnek, Kadri, Kaili Müürisep, and Tiina Puolakainen, 2016:
Estonian Dependency Treebank: from Constraint Grammar tagset to Universal Dependencies. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16),* pages 1558–1565, Portorož, Slovenia. European Language Resources Association (ELRA).

Padró, Luís, 1996:
POS Tagging Using Relaxation Labelling. In: *Proceedings of COLING '96*. Copenhagen, Denmark.

Rademaker, Alexandre & Fabricio Chalub & Livy Real & Cláudia Freitas & Eckhard Bick & Valeria de Paiva, 2017:
Universal Dependencies for Portuguese. In: Simonetta Montemagni & Joakim Nivre (Eds.): *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017*). Linköping Electronic Conference Proceedings No. 139. ACL Anthology W17-65. pp. 197-206.

Salmon-Alt, Susanne & Bick, Eckhard & Romary, Laurent & Pierrel, Jean-Marie, 2004:
La FReeBank: Vers une base libre de corpus annotés, In: Bernard Bel & Isabelle Marlien (eds), *11th Conference on Natural Language Processing* (TALN 2004, April 19-22, 2004). pp. 419-428. Fès: ATALA.

Schmid, Helmut, 1994:
Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*. Manchester, UK.

Tapanainen, Pasi, 1996:
The Constraint Grammar Parser CG-2. Publications No. 27. Department of General Linguistics, University of Helsinki.

Tapanainen, Pasi, 1997:
A Dependency Parser for English. Technical Reports No TR1. Department of Linguistics, University of Helsinki.

Uibo, Heli & Bick, Eckhard. 2005. Treebank-based Research and E-Learning of Estonian Syntax. In *Proceedings of Second Baltic Conference on Human Language TechnologiesTallinn*, April 4-5, 2005. Editors: M. Langemets, P. Penjam. Pp. 195-200.

Voutilainen, Atro. 1997. Designing a (Finite State) Parsing Grammar. In: Roche and Schabes (Eds), *Finite State Language Processing*. The MIT Press.

Wiechetek, Linda and Sjur Nørstebø Moshagen and Børre Gaup and Thomas Omma. 2019. Many shades of grammar checking - Launching a Constraint Grammar tool for North Sámi In *Proceedings of the NoDaLiDa 2019 Workshop on Constraint Grammar - Methods, Tools and Applications*, Turku, Finland.

Yli-Jyrä, Anssi Mikael. 2011. An Efficient Constraint Grammar Parser based on Inward Deterministic Automata. In: *Proceedings of the NODALIDA 2011 Workshop Constraint Grammar Applications*, pp. 50-60 NEALT Proceedings Series , vol. 14.