A Dependency Constraint Grammar for Esperanto

Eckhard Bick
Institute of Language and Communication
University of Southern Denmark
eckhard.bick@mail.dk

Abstract

This paper presents a rule-based formalism for dependency annotation within the Constraint Grammar framework, implemented as an extension of the open source CG3 compiler. As a proof of concept we have constructed a complete dependency grammar for Esperanto, building on morphosyntactically annotated input from the EspGram parser. The system is described and evaluated on a test corpus. With a 4% error rate, and most errors caused by simple error propagation from the morphosyntactic input module, our system has proven robust enough to be integrated into real life applications, such as the Lingvohelpilo spell- and grammar-checker.

1 Introduction

Traditionally, Constraint Grammar (Karlsson et al. 1995) as a descriptive system, has regarded syntax as an extension of morphology, with a shallow syntax based on function tags built on case markers, word order and contextual constraints. This approach to syntax efficiently exploits lexico-morphological clues, and the tag-based annotation allows the grammarian to treat syntax as a disambiguation technique similar to the one used for morphological disambiguation. However, function is only an indirect marker for the relation between words, and it is difficult to express the structural relations of deeper syntax in this fashion. As a first approximation, dependency direction markers were used for the dependents in noun phrases (e.g. @N> or @>N), adjective phrases (@A> or @>A) and prepositional phrases (@P<), a descriptive principle later generalized to clause level functions and subclauses (Bick 2000). In this convention, some obvious underspecifications arise, such as the distinction between short and long attachment in np's, and the scope of coordinators. Nevertheless, two different methods were developed to create full syntactic trees from shallow CG function tags. The first (Bick 2003) uses higher level phrase structure grammars with function tags as terminals, and resolves underspecifications in a generative way. The second, and more robust (Bick 2005), uses ordinary CG rules to add secondary attachment markers (e.g. <np-close>, <np-long>, <co-acc>, <cjt-first>) to resolve underspecification, and creates dependency trees through successive attachment rules. However, the method used an external formalism, with a specially designed dependency rule compiler that also handled issues like uniqueness, circularity and coordination chains.

This paper describes an effort to move this last, tree-building step into the realm of Constraint Grammar proper, thus allowing the user to exploit CG's powerful contextual methodology in the process, to better integrate dependency and functional syntax and to achieve some control over dependency interaction not fully implementable in an the external formalism. The new CG extension was then used to create a dependency CG grammar for Esperanto, and it is this grammar that will be described and evaluated here. The module deep linguistic processing, and can thus be seen as facilitation stepping stone both for further, syntax-dependent annotation (e.g. anaphora, semantic roles) and for various applicative purposes such as machine translation. Currently, the grammar is used in the newly-developed Esperanto grammar checker, Lingvohelpilo (http://lingvohelpilo.ikso.net/), where it provides important contextual information for the checking of accusative/nominative case endings and transitivity affixes, as well as for the identification of long-distance agreement errors, e.g. between subject and subject complement.

2 The formalism

In order to accommodate for dependency, 2 new operators, SETPARENT and SETCHILD, were introduced to GrammarSoft's open-source CG3 compiler (Didriksen 2007), establishing dependency arcs from daughter to mother, or mother to daughter, respectively, addressing one in the SETPARENT/SETCHILD field and the other in a TO field. Both fields of the rule can be independently conditioned with CG contexts in the usual way. The first field works like the TARGET of a MAPping rule, while the TO-end of the dependency is specified by a context condition itself – as seen from the TARGET position. In the case of a LINKed condition, the attachment point can be marked (with a special A operator) as any of the individual contexts checked and "passed". As a default, the dependency arc will attach to the *last* condition of the LINK chain if it can be instantiated. As in the older, external dependency compiler, dependency arcs are expressed as number tokens of the type #n->m, where n is the token ID of the daughter and m the token ID of the mother. Internally, the CG3 compiler uses unique, running IDs (necessary for cross-sentence relations such as anaphora or discourse relations), but in standard dependency output, sentence windows boundaries are respected, using relative IDs. The notation is information equivalent to constituent tree structures, and has been successfully converted into various exchange formats, such as TIGER xml and the VISL cross-language format (constituent trees), as well as MALT xml and CoNNL field format (dependency).

The rule below is an example of a dependency-creating rule for prenominal dependents (@>N), attaching to np-heads (@NP-HEAD) or nouns in the nominative (N NOM), to the right (*1).

```
(a) SETPARENT @>N TO (*1 @NP-HEAD
    OR (N NOM) BARRIER PRP) ;
```

Once established, dependency arcs can be used by later rules – even by other dependency-mapping rules – using three types of dependency relators: p (parent), c (child) and s (sibling). The p-, c- and s-relators replace what would otherwise be position markers in a traditional CG context. Thus, rule (a) exploits semantic prototype roles to select +HUM subjects in the presence of cognitive verbs, while (b) implements the syntactic uniqueness principle for direct objects (@ACC).

```
(a) SELECT (%hum) (0 @SUBJ) (p <Vcog>)
(b) SELECT (@ACC) (NOT s @ACC)
(c) ... (*-1 N LINK c DEF)      -> definite np
    recognized through dependent
(d) ADD (§AG) TARGET @SUBJ (p V-HUM
    LINK c @ACC LINK 0 N-NON-HUM) ;
```

Rule (c) is an example of a rule context used to recognize a definite np through its determiner, and (d) assigns the semantic role tag of agent (§AG) to subjects of "human" verbs with a non-human direct objects.

3 The Esperanto grammar

The preposition barrier (PRP) in the np rule in the last section is a sensible safety measure for English and French, but fails to account for pre-nominal pp's as they do occur in e.g. Esperanto and German. The next rule therefore allows prenominals to search right (**1) *across* the first np-head to a later one that is not part of a prenominal pp (as implied by @P<). Note that the SET target has its own condition excluding targets that already have a parent (using the (*) convention for "any tag"). Since rule application order supersedes token order, this will have the effect of not undoing the pp-free prenominal attachments already mapped by the first rule.

```
SETPARENT @>N (NOT p (*))
    TO (**1 @NP-HEAD OR (N NOM))
    (NOT 0 @P<) ;
```

At the clause level, it is a fair assumption that all left-pointing functions attach to the closest main verb (&MV), unless an intervening subclause ending is marked by punctuation (CLB):

```
SETPARENT @<FUNC
        TO (*-1 &MV BARRIER CLB) ;
```

For right-pointing functions (@FUNC>), the blocking condition is a subclause "complementizer" (relative/interrogative pronoun or a subordinating conjunction), which – unlike English - is an obligatory feature in Esperanto. In a subsequent rule, long-distant attachment across relative clauses can be performed for still unattached subjects (NOT p (V)), by linking to the next main verb that does not already have a subject (NOT c @SUBJ>):

```
SETPARENT @SUBJ> (NOT p (V))
    TO (**1 &MV)
    (*-1 NON-V LINK NOT 1 PCP)
    (NOT c @SUBJ>)
```

Note the additional context condition in the TO field that identifies the first verb in a possible verb chain and conditions it as not being a participle – since participle clauses don't have left subjects.

In our grammar, coordination is handled as "parallel" attachment, not chained Mel'cuk-style, and in the absence of uniqueness-demanding contexts, ordinary attachment rules will therefore handle coordination, too. However, the clause boundary barrier discussed before poses a problem where a chain of conjuncts contains not only a coordinator, but also commas. Therefore, a somewhat more complicated rule becomes necessary to attach comma-isolated conjuncts:

```
SETPARENT $$@FUNC (NOT p (V))
    TO (*-1 IT BARRIER NON-PRE-N/ADV
    LINK *-1 $$@FUNC BARRIER @FUNC
    LINK p (V)) ;
```

This rule exploits the new uniqueness feature in CG3 to attach any as yet unattached function if the *same* function ($$@FUNC) can be found to the left of an immediately adjacent (BARRIER NON-PRE-N/ADV) iterator (IT = coordinator or comma), with no other functions in between (BARRIER @FUNC). The dependency head will be the mother (p V) of the same-function antecedent found. Further rules, not discussed here, attach the coordinator token itself, and assign secondary conjunct tags to all conjuncts, in order to distinguish between first and later conjuncts should the need for a Mel'cuk-style transformation arise.

4 Evaluation

Compared to the complexity of morphological and syntactic CGs, our dependency CG module is strikingly rule efficient, achieving robust annotation with just 66 rules, compared to the thousands of rules in lower-level CGs, and the couple of hundred rules in a CG-based PSG. Of course, it has to be born in mind, that our rules rely heavily on syntactic functions and attachment direction markers introduced by preceding CG modules. Also, at the time of writing, we have not yet incorporated the distinction between close and long postnominal attachment, ellipsis and quoted sentences which will unavoidably add to the number of rules.

Speedwise, CG-dependency is also quite efficient. A 75.000 word corpus consisting of 50% news magazine text and 50% classical texts, was analyzed with the EspGram tagger (Bick 2007) at the syntactic-functional level, and the annotated corpus was then tagged with our dependency CG on a 2.4 GHz laptop. In this experiment, the analysis chain up to the syntactic function level ran at 72 words/s, while the dependency level alone ran at 6336 words/s, using 10.2 % of overall processing time. Compared to the external dependency system (608 words/s), this implies a speed improvement by almost one order of magnitude.

A rough inspection of annotation results for a sample of 1000 words indicate an overall error rate for the dependency annotation of about 4%. Of these, about half were attachment failures (no mothernode for non-topnode functions), half were wrong attachments (wrong daughter-mother relation). With most errors being caused by syntactic-function errors in the input, the error rate of the dependency module itself was very low, under 1%.

5 Conclusion and outlook

Given the necessary formal changes to the CG compiler software, it appears to be feasible, even with a relatively small set of rules, to handle the creation of dependency tree structures for CG-analyzed input within the CG formalism itself. Our experiments with such a grammar for use in an Esperanto spell- and grammar-checker produced robust results, both quantitatively and qualitatively. In particular, the dependency module proved to be considerably more robust than the syntactic function module, inheriting most of its errors from the former. We therefore believe that CG dependency modules can be created with comparatively little effort, to turn existing CG function annotations into dependency treebanks without substantial loss of information. Future research should allow us to shed light on the question to what degree our dependency grammar, given a compatible set of morphological and syntactic input tags, is language independent - as the size and simple nature of our rule set indicates.

References

Bick, Eckhard (2000), "The Parsing System PALAVRAS - Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework", Aarhus: Aarhus University Press

Bick, Eckhard. (2003) A CG & PSG Hybrid Approach to Automatic Corpus Annotation, In: Kiril Simow & Petya Osenova (eds.), Proceedings of SProLaC2003 (at Corpus Linguistics 2003, Lancaster), pp. 1-12

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 (2007), Tagging and Parsing an Artificial Language: An Annotated Web-Corpus of Esperanto, In: *Proceedings of Corpus Linguistics 2007, Birmingham, UK*. Electronically published at (http://ucrel.lancs.ac.uk/publications/CL2007/, Nov. 2007)

Didriksen, Tino (2003). "Constraint Grammar Manual", http://beta.visl.sdu.dk/cg3/single/

Karlsson, Fred et al. (1995): Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text. Natural Language Processing, No 4. Berlin & New York: Mouton de Gruyter.

Appendix: Annotation sample

Post 12 jaroj da reformoj, la efikeco de la ĉeĥa ekonomio ne signife transpaŝas la nivelon atingitan en la jaro 1989.
(Ater 12 years of reforms, the efficiency of the chech economy has not significantly surpassed the level reached in [the year of] 1989,)

Post	[post] <*> PRP @ADVL> #1->14
12	[12] <card> <cif> NUM P @>N #2->3
jaroj	[jaro] <dur> <per> N P NOM @P< #3->1
da	[da] PRP @N< #4->3
reformoj	[reformo] <sem-c> <act> N P NOM @P< #5->4
la	[la] ART @>N #6->7
efikeco	[efikeco] <f> N S NOM @SUBJ> #7->14
de	[de] PRP @N< #8->7
la	[la] ART @>N #9->11
cxehxa	[cxehxa] <jnat> ADJ S NOM @>N #10->11
ekonomio	[ekonomio] <domain> N S NOM @P< #11->8
ne	[ne] <amod> <setop> ADV @>A #12->13

```
signife      [signife] ADV @ADVL> #13->14
transpasxas  [transpasxi] <mv> <vt>V PR @FS-STA
             #14->0
la           [la] ART @>N #15->16
nivelon      [nivelo] <ac> N S ACC @<ACC #16->14
atingitan    [atingi] <mv> <vt> V PCP PAS IMPF ADJ
             S ACC @ICL-N< #17->16
en           [en] PRP @<ADVL #18->17
la           [la] ART @>N #19->20
jaro         [jaro] <dur> <per> N S NOM @P<
             #20->18
1989         [1989] <year> <card> <cif> NUM S @N<
             #21->20
$.
```

The following fields are used in the annotation scheme, and expressed as feature attribute pairs in xml: wordform, [base form/lemma], <semantics>, @syntactic_function, #dependency-link

(*part of speech tags:* N=noun, V=verb, ADJ=adjective, ADV=adverb, PRP=preposition, ART=article, NUM=numeral; *inflexion:* S=singular, P=plural, NOM=nominative, ACC=accusative, PCP=participle, PAS=passive, PR=present tense, IMPF=past tense; *syntactic function:* @SUBJ=subject, @ADVL=adverbial, @ACC=direct object, @>N=pre-nomina modifier, @N<=post-nominal modifier, @P<=argument of preposition, @ICL=non-finite clause, @FS=finite clause, @STA=statement; *semantic prototypes:* <dur> duration, <ac> abstract countable, <domain> domain, <sem-c> semantic product, <act> action, <f> feature, <jnat> nationality, <mv> main verb; *valency:* <vt> transitive)