CONSTRAINT GRAMMAR AS A FRAMEWORK FOR PARSING RUNNING TEXT

Fred Karlsson

University of Helsinki
Department of General Linguistics
Hallituskatu 11
SF-00100 Helsinki
Finland
e-mail: KARLSSON@FINUH.bitnet

1. Outline

Grammars which are used in parsers are often directly imported from autonomous grammar theory and descriptive practice that were not exercised for the explicit purpose of parsing. Parsers have been designed for English based on e.g. Government and Binding Theory, Generalized Phrase Structure Grammar, and Lexical-Functional Grammar. We present a formalism to be used for parsing where the grammar statements are closer to real text sentences and more directly address some notorious parsing problems, especially **ambiguity**. The formalism is a linguistic one. It relies on transitional probabilities in an indirect way. The probabilities are not part of the description.

The descriptive statements, **constraints**, do not have the ordinary task of defining the notion 'correct sentence in L'. They are less categorical in nature, more closely tied to morphological features, and more directly geared towards the basic task of parsing. We see this task as one of inferring surface structure from a stream of concrete tokens in a basically bottom-up mode. Constraints are formulated on the basis of extensive corpus studies. They may reflect absolute, rule-like facts, or probabilistic tendencies where a certain risk is judged to be proper to take. Constraints of the former rule-like type are of course preferable.

The ensemble of constraints for language L constitute a Constraint Grammar (CG) for L. A CG is intended to be used by the Constraint Grammar Parser **CGP**, implemented as a Lisp interpreter.

Our input tokens to CGP are morphologically analyzed word-forms. One central idea is to maximize the use of morphological information for parsing purposes. All relevant structure is assigned directly via lexicon, morphology, and simple mappings from morphology to syntax. The task of the constraints is basically to **discard as many alternatives as possible**, the optimum being a fully disambiguated sentence with one syntactic reading only.

The second central idea is to treat morphological disambiguation and syntactic labelling by the same mechanism of discarding improper alternatives.

A good parsing formalism should satisfy many requirements: the constraints should be declarative rather than procedural, they should be able to cope with any real-world text-sentence (i.e. with running text, not just with linguists' laboratory sentences), they should be clearly separated from the program code by which they are executed, the formalism should be language-independent, it should be reasonably easy to implement (optimally as finite-state automata), and it should also be efficient to run. The CG formalism adheres to these desiderata.

2. Breaking up the problem of parsing

The problem of parsing running text may be broken up into six subproblems or 'modules':

- preprocessing,
- morphological analysis,
- local morphological disambiguation,
- morphosyntactic mapping,
- context-dependent morphological disambiguation,
- determination of intrasentential clause boundaries,
- disambiguation of surface syntactic functions.

The first four of these modules are executed sequentially, optimally followed by parallel execution of the last three modules which constitute 'syntax proper'. We have a **five-stage parsing-process**.

In this general setting, CG is the formalism of the fifth stage, syntax proper. The same CG constraint formalism is used to disambiguate morphological and syntactic ambiguities, and to locate clause boundaries in a complex sentence. Parts of the CG formalism are used also in morphosyntactic mapping.

Real texts are full with idiosyncracies in regard to headings, footnotes, paragraph structure, interpunctuation, use of upper and lower case, etc. Such phenomena must be properly normalized. Furthermore several purely linguistic phenomena must be somehow dealt with prior to single-word morphological analysis, especially idioms and other more or less fixed multi-word expressions. (It would e.g. make no sense to subject the individual words of the expression *in spite of* to plain morphological analysis.) The existence of an adequate preprocessor is here simply taken for granted.

We concentrate on morphological analysis, clause boundary determination, morphological disambiguation, and syntactic function assignment. Viewing the problem of parsing in turn from one or another of these angles clarifies many intricacies. The subproblems take more manageable proportions and make possible a novel type of modularity.

Morphological analysis is relatively independent. CGP is always supplied with adequate morphological input. The morphological analyzers are designed according to Koskenniemi's (1983) two-level model. Currently our Research Unit has morphological analyzers available for English (41,000 lexicon entries), Finnish (37,000 entries), and Swedish (42,000 entries). Below are two morphologically analyzed English word-forms, *a* has one **reading**, *move* four. The set of readings for a word-form we call a **cohort**. All readings in a cohort have the base-form initially on the line. Upper-case strings are morphological features except for those containing the designated initial character "@" which denotes that the string following it is the name of a syntactic function, here emanating from the lexicon. "@DN>" = determiner as modifier of the next noun to the right, "@+FMAINV" = finite main verb, "@-FMAINV" = non-finite main verb as member of a verb chain, "@<NOM-FMAINV" = non-finite main verb as postmodifier of a nominal:

```
*a*
 a " DET CENTR ART INDEF @DN> "
*move*
 move " N NOM SG "
 move " V SUBJUNCTIVE @+FMAINV "
 move " V IMP @+FMAINV "
 move " V INF @-FMAINV @<NOM-FMAINV "
```

Here, disambiguation refers to reduction of morphological ambiguities, optimally down to cohort size = 1. Sense disambiguation is not included (presently our lexical items have no sense descriptions).

The subproblems of morphosyntactic mapping, morphological disambiguation, clause boundary location, and syntactic function determination are interrelated. E.g., for disambiguation it is useful to know the boundaries of the current clause, and to know as much as possible about its syntactic structure. An important aspect of the general problem is to work out the precise relations between these modules.

3. Local disambiguation

Morphological ambiguities may be due to intersection of forms of the same or of different lexical entries, or to intersection of recursive compound paths. The latter phenomenon arises if productive compound formation in e.g. Finnish, German, and Swedish is described by recursive links back to the main lexicon. Consider the cohort of the Swedish word-form *frukosten* ("_" = compound boundary, *frukost* 'breakfast', *fru* 'mrs', *kost* 'nutrition', *ko* 'cow', *sten* 'stone'):

```
*frukosten*
 frukost " N UTR DEF SG NOM "
 fru_kost " N UTR DEF SG NOM "
 fru_ko_sten " N UTR INDEF SG NOM "
```

By 'local disambiguation' we refer to constraints or strategies that make it possible to discard some readings just by local inspection of the current cohort, without invoking any contextual information. The present cohort contains three readings. An interesting local disambiguation strategy can now be stated: "Discard all readings with more than the smallest number of compound boundaries occurring in the current cohort". This strategy properly discards the readings "fru_kost" and "fru_ko_sten". I have found this principle to be very close to perfect.

A similar principle holds for derivation: "Discard readings with derivational elements if there is at least one non-derived reading available in the cohort".

Other local disambiguation strategies compare multiple compound readings in terms of how probable their part of speech structure is (NNN, ANN, NVN, AAV, etc.).

Local disambiguation is a potent module. The Swedish morphological analyzer was applied to a text containing some 840,000 word-form tokens. The following table shows cohort size *N(r)* in the first column. The second and third columns show the number of cohorts with the respective number of readings before (a) and after (b) local disambiguation. E.g., before local disambiguation there were 3830 word-forms with 6 readings but after local disambiguation only 312.

N(r)	*(a)*	*(b)*
0	13957	13957
1	440035	487994
2	253779	236298
3	55857	44782
4	38062	29053
5	24135	18911
6	3830	312
7	9551	8913
8	541	23
9	232	47
10	72	2
11	46	5
12	124	-
13	15	-
14	28	-
15+	33	-

Out of roughly 1,5 million readings assigned by morphology (1.8 readings/word-form), local disambiguation discards more than 100,000. Especially dramatic the drop is for highly ambiguous words.

4. Morphosyntactic mapping

After local disambiguation, each word in the sentence undergoes morphosyntactic mapping, i.e. it is assigned at least one syntactic label, perhaps several if a unique label is not possible to assign. This mapping will be discussed in connection with the syntactic constraints in section 7.

5. Context-dependent disambiguation constraints

The CG formalism will first be illustrated by context-dependent disambiguation constraints. **Sets** of grammatical features are needed in the constraints for the purpose of generalization. Each set declaration consists of a set name followed by the elements of that set. The elements are (strings of) features and/or base-forms occurring in readings:

```
(DET "DET")
(N "N")
(TO "to")
(PREMOD "A" "DET")
(NOMHEAD "N NOM" "PRON NOM")
(VFIN "V PRES" "V PAST" "V IMP" "V SUBJUNC-
TIVE")
```

Each **constraint** is a quadruple consisting of domain, operator, target, and context condition(s). An example:

```
(@w =0 "PREP" (-1 DET))
```

stating that if a word (@w) has a reading with the feature "PREP", this very reading is discarded (=0) iff the preceding word (i.e. the word in position -1) has a reading with the feature "DET".

The **domain** points out some element to be disambiguated, e.g. (the readings of) a particular word-form. The designated domain *@w* is a variable over any word-form, used when the target reading is picked by feature(s) only.

The **target** defines which reading the constraint is about. The target may refer to one particular reading, such as "V PRES -SG3", or to all members of a declared set, such as VFIN.

The **operator** defines which operation to perform on the reading(s). There are three disambiguation operators, here treated in order of decreasing strength. The operator '=!!' indicates that the target reading is the correct one iff all context conditions are satisfied; all other readings should be discarded. If the context conditions are not satisfied, the target reading itself is discarded. The operator '=!' indicates that the target reading is the correct one iff all context conditions are satisfied, all other readings are discarded. The operator '=0' discards the target reading iff the context conditions are satisfied, it leaves all other readings. The operators are here defined in the procedural mode as performing operations. Conceptually they just express constraints.

The **context conditions** are defined relative to the target reading in position 0. Position 1 is one word to the right of 0, -3 three words to the left of 0, etc. (Such straightforward positions we call **absolute**.)

Each context condition is a triple consisting of **polarity, position**, and **set**.

'Polarity' is either NOT or nothing (i.e. positive), 'position' is a legal position number, and 'set' is a declared set name.

An asterisk "*" (functionally and mnemotechnically reminiscent of the Kleene star) prefixed to position number *n* refers to some position rightwards of *n* (if *n* is positive), or some position leftwards of *n* (if *n* is negative), in both cases including *n*, up to the next sentence boundary (or clause boundary, if enforced in clause boundary mode, cf. below). The asterisk convention thus enables the description of **unbounded dependencies**.

Examples: (1 N) requires there to be a reading with the feature "N" for the next word-form. (NOT *-1 VFIN) states: nowhere leftwards in this sentence is there a reading with any of the feature combinations defining finite verbs. The condition ensemble (1 PREMOD) (2 N) (3 VFIN) requires there to be a reading with either "A" or "DET" in position 1, with "N" in position 2, and with one of the VFIN readings in position 3. Here are two more context-dependent disambiguation constraints for English:

```
(@w =0 VFIN (-1 TO))
("that" =! "<Rel>" (-1 NOMHEAD) (1 VFIN))
```

The first one discards all finite verb readings immediately after the base-form *to* (itself either a preposition or an infinitive mark). VFIN is a declared set. The constraint is applicable to all strings declared to belong to this set.

The second constraint states that the proper reading of the word *that* is relative pronoun (i.e. a reading containing the string "<Rel>", itself an inherent feature emanating from the lexicon) immediately after a nominal head and immediately before a finite verb.

There is also a mechanism available for expressing **relative position** with reference to variable positions established via unbounded dependencies. Let condition (*1 VFIN) be satisfied at absolute position 5, i.e. at the fifth word to the right. Then (L-1 N) would require there to be a feature "N" in absolute position 4, (L* N) would establish a second unbounded dependency somewhere left of position 5 (but right of position 0), i.e. looking for satisfaction at one of positions 4,3,2,1.

Often context conditions work on ambiguous cohorts, i.e. one reading satisfies the condition, but this reading perhaps is not the correct one in the first place. If so, should a risk be taken? The CG formalism makes this a matter of deliberate choice. All constraints so far treated allow the context conditions to be satisfied by ambiguous context cohorts. By appending the character *C* to the position number, one requires the respective condition to be satisfied only if the cohort being tested is itself unambiguous. This is called **careful mode**, e.g.:

(@w =0 VFIN (-1C TO))

For many constraints it is necessary to require that they do not apply over clause boundaries. This **clause boundary mode** is effected by appending either of the atoms **CLB (ordinary mode) or **CLB-C (careful mode) after the last context condition. Clause boundary mode is typically used in conjunction with unbounded contexts.

A **template** mechanism is available for expressing partial generalizations. E.g., a template "&NP" could be declared to contain the alternatives ((N)), ((A) (N)) ((DET) (N)) ((DET) (A) (N)), etc. Then the template &NP could be used in the context part of any constraint. At run-time all alternative realizations of &NP would be properly considered.

Every constraint embodies a true statement. Occasionally the constraints might seem quite down-to-earth and even 'trivial', given mainstream conceptions of what constitutes a 'linguistically significant generalization'. But the very essence of CG is that low-level constraints (i) are easily expressible, and (ii) prove to be effective in parsing.

6. Constraints for intrasentential clause boundaries

Clause boundary constraints establish locations of clause boundaries. They are important especially for the formulation of proper syntactic constraints. E.g., the syntactic constraint "there is only one finite predicate in a simplex non-coordinated clause" presupposes that clause boundary locations are known.

Clause boundaries occur i.a. as the inherent feature "<**CLB>" in the input stream. E.g. subjunctions are lexically marked by this feature. But many boundaries must be spotted by specific constraints. Clause boundary constraints have the special operator "=**CLB" stating that there is a clause boundary before the word specified by the target.

E.g., given that conjunctions are lexically marked by the inherent feature "<Conj>", the constraint:

(@w =**CLB "<Conj>" (1 NOMHEAD) (2 VFIN))

states that there is a clause boundary before conjunction instances that precede a NOMHEAD followed by a finite verb (e.g., before the conjunction in a sentence such as *John eats and Bill drinks*).

7. Syntactic constraints

CG syntax is based on **dependency** and should assign **flat, functional, surface labels**, optimally one to each word-form. The labels are roughly the classical repertoire of heads and modifiers. CG syntax maps morphological categories and word order information onto syntactic labels.

The designated syntactic subsets of verb chain elements, head labels, and modifier labels should be established. For English, these include e.g.:

- *verb chain members*: @+FAUXV (finite auxiliary V), @-FAUXV (non-finite auxiliary V), @+FMAINV (finite main V), @-FMAINV (non-finite main V), ...
- *nominal heads*: @SUBJ, @OBJ, @I-OBJ, @PCOMPL-S (subj. pred. compl.), @PCOMPL-O (obj. pred. compl.), @ADVL (adverbial), ...
- *nominal modifiers*: AN> (adjective as premodifier to N), DN> (determiner as premodifier to N), <NOM (postmodifier to nominal), A> (premodifier to A), <P (postmodifier to P), ...

A verb chain such as *has been reading* gets the labels @+FAUXV @-FAUXV @-FMAINV. In the sentence *She bought the car*, *she* is @SUBJ and *car* @OBJ.

Certain verb chain and head labels may occur maximally once in a simplex clause. This restriction we call the **Uniqueness Principle**. At least @+FAUXV, @+FMAINV, @SUBJ, @OBJ, @I-OBJ, @PCOMPL-S, and @PCOMPL-O obey this restriction. Many constraints may be based on consequences of the Uniqueness Principle. E.g., if a morphologically and syntactically unambiguous @SUBJ has been identified in a clause, all other instances of @SUBJ occurring in syntactically ambiguous readings of that clause may be discarded.

Modifier and complement labels **point** in the direction (right ">", left "<") of the respective head which is identified by its part-of-speech label. E.g., the label @<P is assigned to a prepositional complement such as *park* in *in the park*. Our analysis of modifier and complement labels is more delicate than in traditional grammar, cf. the premodifiers AN>, DN>, NN>, GN> (genitival).

In Constraint Grammar, syntactic labels are assigned in three steps. The basic strategy is: Do as much as possible as early as possible.

The first step is to provide as many syntactic labels as possible in the **lexicon** (including morphology). For entries having a reduced set of labels (compared to what that morphological class normally has), those labels will be listed in the lexicon. Thus, output from lexicon and morphology will indicate that *he* is @SUBJ, that *him* is either @OBJ, @I-OBJ, or @<P (NB: a considerably reduced subset of all nominal head functions), that *went* is @+FMAINV, etc.

The second step is **morphosyntactic mapping**. For all readings that remain after local disambiguation and do not yet have any syntactic function label, simple mapping statements tell, for each relevant morphological feature, or combination of features, what its range of syntactic labels is. This may be compared to traditional grammar book statements such as "the syntactic functions of nouns are subject, object, indirect object, ...".

CG contains one enrichment of this scheme. A mapping statement may be constrained by the context condition mechanism specified in section 5.

Thus, a mapping statement is a triple <morphological feature(s), context condition(s), syntactic function(s)>. The first element is a feature string occurring in a morphological reading, the second is either NIL (no conditions) or a list of sublists each of which is a legal context condition. Finally the requisite grammatical function label(s) are listed. Here are some mapping statements without context conditions, providing a maximal set of labels:

("PRON GEN" NIL GN> @PCOMPL-S @PCOMPL-O)

("A" NIL AN> @PCOMPL-S @PCOMPL-O @SUBJ @OBJ @I-OBJ)

("N NOM" NIL @SUBJ @OBJ @I-OBJ @PCOMPL-S @PCOMPL-O @APP @NN> @<P)

A pronoun in the genitive case is either prenominal genitival modifier, subject predicate complement, or object predicate complement. An adjective is prenominal adjectival modifier, predicate complement, subject, object, or indirect object (the last three functions refer to occurrences of adjectives as 'nominalized heads'), etc.

Often morphosyntactic mappings may be considerably constrained by imposing context conditions :

("N NOM" ((1 N)) @NN>)
("N NOM" ((-1 PREP)) @<P)
("INF" ((-2 N) (-1 TO)) @<NOM-FMAINV)

These state that a noun in the nominative case premodifies (@NN>) a subsequently following noun (in compounds, cf. *computer screen*), that a noun in the nominative case after a preposition is @<P, and that an infinitive preceded by a noun + *to* postmodifies that noun.

In this way, the task of syntactic analysis is simplified as much as possible, as early as possible. Superfluous alternatives are not even introduced into the parsing of a particular clause if it is clear at the outset, i.e. either in the lexicon or at the stage of morphosyntactic mapping, that certain labels are incompatible with the clausal context at hand.

There may be several mapping statements for the same morphological feature(s), e.g. "N NOM". Mapping statements with more narrowly specified contexts have precedence over more general statements. In the present implementation of CGP, the mapping statements apply in plain **linear order**. The last mapping statement for a particular feature provides the worst case, i.e. the maximal assortment of function labels for that feature.

Every word-form will have at least one syntactic label after morphosyntactic mapping, and all possible syntactic ambiguities have also now been introduced.

In step three, **syntactic constraints** reduce syntactic ambiguities where such exist due either to lexical information (cf. the infinitive *move* above), or to morphosyntactic mapping. Syntactic constraints discard the remaining superfluous syntactic labels. Syntactic constraints differ from context-dependent disambiguation constraints only by having one of the syntactic operators '=s!', or '=s0' (where *s* indicates that the constraint is a syntactic one). Their semantics is identical to that of the disambiguation constraints:

(@w =s0 "@+FMAINV" (*-1 VFIN))
(@w =s0 "@+FMAINV" (*1 VFIN))
(@w =s! "@SUBJ" (0 NOMHEAD) (NOT *1 NOMHEAD) (*1 VFIN) (NOT *-1 NOMHEAD))

The first two constraints discard @+FMAINV as a syntactic alternative if there is a unique finite main verb either to the left or to the right in the same clause. The third constraint prescribes that @SUBJ is the correct label for a noun or pronoun (NOMHEAD in target position, i.e. position 0), with a finite verb somewhere to the right in the same clause and no similar noun or pronoun either left or right (-- *woman* -- *laughed* --).

Maximal profit is extracted from the Uniqueness Principle. At each syntactic step (before mapping, after mapping, and after the application of a syntactic constraint that affects the labels obeying the Uniqueness Principle), each clause is checked for eventual violations of this principle. In this way many ambiguous primary labels may be safely discarded.

Here is an example sentence, fully analyzed and unambiguous in all respects but the one syntactic ambiguity remaining for the word *in*:

Bill
Bill " <Proper> N NOM SG " @SUBJ
saw
see " <SVO> V PAST " @+FMAINV
the
the " DET " @DN>
little
little " A ABS " @AN>
dog
dog " N NOM SG" @OBJ
in
in " PREP " @<NOM @ADVL
the
the " DET " @DN>
park
park " N NOM SG " @<P

There is no non-semantic way of resolving the attachment ambiguity of the adverbial *in the park*. This ambiguity is therefore properly unresolved.

In CGP, all ambiguities 'are there' after morphosyntactic mapping and require no additional processing load. Notice in passing that CGP makes an interesting prediction which might be relevant from the viewpoint of mental language processing. Disambiguation, i.e. finding a unique interpretation by applying constraints, requires 'more effort' than leaving all or many ambiguities unresolved (in which case constraints were not applied). Parsers based on autonomous grammars tend to work in the opposite way (the more ambiguities, the more rules to apply and trees to construct).

In CGP, there is precisely one output for each sentence regardless of how many unresolved ambiguities there might be pending in it. This output is an annotated linear, flat string of word-forms, baseforms, inherent features, morphological features, and syntactic function labels, all of the same formal type. The dependency structure of the sentence is expressed by the pointers and parts of speech of the syntactic labels.There is no proliferation of parse trees, often encountered in other types of parsers, even if morphological and/or syntactic ambiguities are left unresolved.

8. Implementation

I have written an interpreter in strict Common Lisp for parsing with constraint grammars. This is what we call the Constraint Grammar Parser (CGP). CGP currently runs on Unix workstations under Lucid Common Lisp and Allegro Common Lisp. A PC version with the same functionality runs under mu-Lisp on ordinary XT/AT machines.

CGP takes two inputs, a **constraint file** with set declarations, mapping statements, context-dependent disambiguation constraints, syntactic constraints, etc., and a **text file** with morphologically analyzed word-forms (cf. section 2).

The optimal implementation of constraint grammar parsing would be in terms of finite-state machines (cf. Kimmo Koskenniemi, COLING-90 Proceedings, Vol. 2).

9. Discussion

The CG formalism has so far been extensively applied only to English context-dependent disambiguation and syntax.

Presently some 400 context-dependent disambiguation constraints have been formulated for English by Atro Voutilainen, Juha Heikkilä, and Arto Anttila. These constraints prune 95-97 % of all morphological ambiguities in running English text, depending upon the complexity of the text. This level (which is not final as work on the most recalcitrant problems proceeds) has been achieved using plain morphological information, i.e. information present in the cohorts of neighbouring words. No syntactic functions have been used. No enormous amounts of disambiguation constraints will thus be needed, the 'final' order of magnitude might be some 500. We consider this number surprisingly small. It shows the descriptive power of low-level morphology-based constraints.

The most successful achievements so far in the domain of large-scale morphological disambiguation of running text have been those for English reported by Garside, Leech, and Sampson (1987), on tagging the LOB corpus, and Church (1988), on assigning part-of-speech labels and parsing noun phrases. Success rates ranging between 95-99% are reported, depending on how 'success' is defined. These approaches are probabilistic and based on transitional probabilities calculated from extensive pretagged corpora.

As for morphological disambiguation, CGP has achieved almost the same success rate. In comparison, we first note that CG provides a formalism, based on ordinary linguistic concepts, that is applicable to any language. Work on Finnish and Swedish is in progress. Second, CG fully integrates morphology and surface syntax within the same formalism.

Our present success rate for syntax, with some 220 mapping statements and 25 syntactic constraints, is slightly above 90% (words with unique syntactic labels). The remaining words have more than one syntactic label (one of which is correct).

References

Church, K. 1988. "A Stochastic Parts Program and Noun Phrase Parser for Running Text." *Second Conference on Applied Natural Language Processing, Proceedings of the Conference*, ACL 1988, pp.136-143.

Garside, Roger, Leech, Geoffrey, and Sampson, Geoffrey 1987 (eds.), *The Computational Analysis of English.* Longman, London and New York.

Karlsson, Fred 1989. "Parsing and Constraint Grammar". Manuscript, Research Unit for Computational Linguistics, University of Helsinki.

Koskenniemi, Kimmo 1983. Two-Level Morphology. A General Computational Model for Word-Form Recognition and Production. Department of General Linguistics, University of Helsinki, Publications No. 13.

Acknowledgements

This research was supported by the Academy of Finland in 1985-89, and by the Technology Development Centre of Finland (TEKES) in 1989-90. Part of it belongs to the ESPRIT II project SIMPR (2083). I am indebted to Kimmo Koskenniemi for help in the field of morphological analysis, and to Atro Voutilainen, Juha Heikkilä, and Arto Anttila for help in testing the formalism.