Parsing with Category Coocurrence Restrictions

James KILBURY
Universität Düsseldorf
Seminar für Allgemeine Sprachwissenschaft
Universitätsstraße 1, D-4000 Düsseldorf 1
Federal Republic of Germany

Abstract

This paper summarizes the formalism of Category Cooccurrence Restrictions (CCRs) and describes two parsing algorithms that interpret it. CCRs are Boolean conditions on the cooccurrence of categories in local trees which allow the statement of generalizations which cannot be captured in other current syntax formalisms. The use of CCRs leads to syntactic descriptions formulated entirely with restrictive statements. The paper shows how conventional algorithms for the analysis of context free languages can be adapted to the CCR formalism. Special attention is given to the part of the parser that checks the fulfillment of logical well-formedness conditions on trees.

1. The CCR formalism

The CCR formalism, introduced in Kilbury (1986, 1987), has its origin in Generalized Phrase Structure Grammar (GPSG; cf Gazdar et al. 1985, henceforth 'GKPS') and is a special type of unification grammar (cf Shieber 1986). It achieves a more homogeneous structure than GPSG by assigning a central role to cooccurrence restrictions. The immediate dominance rules, linear precedence statements, and metarules of GPSG are eliminated in favor of Category Cooccurrence Restrictions (CCRs), which are Boolean conditions on the cooccurrence of categories in local trees (LTs, i.e. trees of depth one). The elimination of metarules (cf Kilbury 1986), which lead to particular difficulties in parsing with GPSGs, was in fact the immediate motivation for developing and implementing the formalism, but its main justification lies in its capacity to express generalizations which cannot be captured in GPSG or other present formalisms.

While reducing the number of descriptive devices in GPSG, the formalism with CCRs retains the restrictiveness of GPSG. Although it is not intended as a vehicle for the implementation of other grammar formalisms, it is 'tool-oriented' in the sense of Shieber (1986) in that it provides a clearer picture of the relation GPSG has to other formalisms (cf Kilbury 1987).

The motivation for CCRs is analogous to that for distinguishing Immediate Dominance (ID) and Linear Precedence (LP) rules in GPSG (cf GKPS, pp. 44-50). A context free (CF) rule binds information of two kinds in a single statement. By separating this information in ID and LP rules, GPSG is able to state generalizations of the sort "A precedes B in every LT which contains both as daughters," which cannot be captured in a CF grammar.

Just as ID and LP rules capture generalizations about sets of CF rules (or equivalently, about LTs), CCRs can be seen as stating more abstract generalizations about ID rules, which in turn are equivalent to generalizations of the following sort about LTs:

(1) Any LT with S as its root must have A as a daughter.

(2) No LT with C as a daughter also has D as a daughter.

Statements such as (1) and (2) are CCRs. CCRs are expressions of first order predicate logic using two primitive predicates, R(α, t) 'α is the root of LT t' and D(α, t) 'α is a daughter in LT t'. [1] CCRs have one of the three forms

$$\forall\, t\colon \mu \supset \omega \qquad \forall\, t\colon \omega \qquad \forall t\colon \omega \supset \mu$$

where μ is a disjunction of positive literals R(α, t) and ω is a schema, whereby the notion of a possible schema is defined as follows:

(a) D(α, t) is of form ω;
(b) if ψ is of form ω, then $(\sim\psi)$ is of form ω;
(c) if ψ and χ are both of form ω, then $(\psi \varkappa \chi)$ is of form ω, where $\varkappa \in \{\vee, \wedge, \supset, \equiv\}$;
(d) these are all expressions of form ω.

Furthermore, all predicates within a CCR have the variable t as second argument; all first arguments are constants designating categories. Parentheses may be omitted following the usual conventions in predicate logic. CCRs will be normally be written in the three forms

$$\mu' \;|[\, \omega']|\qquad\qquad |[\omega']|\qquad\qquad |[\omega']|\; \mu'$$

where each occurrence of D(α, t) in ω is replaced simply by α in ω', and each occurrence of R(α, t) in μ by α in μ'. Using this notation, (1) and (2) may be restated as (3) and (4), respectively:

(3) S |[A]|
(4) |[C $\supset$ ~D]|

Let us consider the illocutionary force of grammatical statements. The standard phrase-structure rules of CF grammar and similar formalisms expressly permit or license structures; in the terminology of Toulmin (1958) such statements constitute *warrants* that say "Xs are allowed" (viewed declaratively) or "You can build Xs" (viewed procedurally) for corresponding structures of type X. Grammatical *restrictions*, in contrast, say "Xs are forbidden" or "You may not build Xs." Warrants and restrictions used together characterize the possible and necessary features of linguistic structures and thus introduce a modal dimension in the language of grammatical description.

Pure CF grammars consist solely of warrants, while GPSG is a mixed system. GPSG categories are defined restrictively with Feature Cooccurrence Restrictions and Feature Specification Defaults after the *space* of categories under consideration is defined by the language-independent notion of category itself; properties of LTs, however, are defined with both warrants (in the form of ID rules and metarules) and with restrictions (in the form of LP statements and the Feature Instantiation Principles). [2]

In analogy to the definition of categories in GPSG, the formalism of CCRs defines a language-independent space of LTs and then provides all descriptive statements in the form of restrictions. If we abandon the formal distinction between categories and LTs and represent the information of both with functional descriptions (i.e. feature structures) as in Functional Unification Grammar (cf Kay 1985) we obtain a single type of informational element which in the descriptions of particular natural languages is defined solely with restriction. This paper retains the distinction between LTs and categories and treats the latter as atoms purely for simplicity of presentation.

CCRs and the LTs to which they apply may formally be regarded as the same kind of informational element. For the corresponding notion of unification and for the analysis of LP statements as CCRs see Kilbury (1987); the reformulation of LP statements will be omitted in the following sections.

2. An example of CCRs

GKPS (pp. 47-49) examines sets of simple CF rules and then proposes strongly equivalent descriptions in ID/LP format. The first resulting ID/LP grammar is given in (5):

(5)

ID rules	LP rules
S → NP, VP	AUX < NP
S → AUX, NP, VP	V < NP
VP → AUX, VP	NP < VP
VP → V, VP	
VP → V, NP	
VP → V, NP, VP	

The ID rules of (5) admit LTs with the following *branches* (i.e. mother-pairs pairs):

(6) ⟨S, NP⟩, ⟨S, VP⟩, ⟨S, AUX⟩,
⟨VP, V⟩, ⟨VP, VP⟩, ⟨VP, AUX⟩, ⟨VP, NP⟩

It is clear that a daughter category can cooccur only with certain mother categories. Given this set of branches, three generalizations can be formulated as CCRs:

(7) CCR 1: $[\![\text{NP} \vee \text{VP} \vee \text{AUX}]\!] (\text{S} \vee \text{VP})$
CCR 2: $[\![\text{V}]\!] \text{VP}$
CCR 3: $[\![\sim\text{S}]\!]$

CCR 1 states that any LT with NP, VP, or AUX as daughter must have S or VP as its root category, while CCR 2 requires LTs with V as a daughter to have VP as root. CCR 3 prevents S from occurring as daughter in a LT.

Now consider the Cartesian product of the set of phrasal categories {S, VP} -- excluding NP, which is not expanded in the ID/LP grammar of (5) -- with the set of phrasal and lexical categories {S, VP, NP, V, AUX}. CCR 2 excludes the branch ⟨S, V⟩ while CCR 3 excludes ⟨S, S⟩ and ⟨VP, S⟩. Thus, the set of legal branches remaining after the Cartesian space is filtered by the CCRs of (7) is exactly the set of branches specified in (6).

Proceeding in turn from a given root category, corresponding restrictions appy to the cooccurrence of daughter categories. The LTs with S as root can be covered by a single CCR:

(8) CCR 4: $\text{S} \Rightarrow [\![\text{NP} \wedge \text{VP}]\!]$

CCR 4 states that NP and VP are obligatory in any LT with S as its root. Since ⟨S, AUX⟩ is also a branch, AUX may optionally occur as daughter in such a tree.

Given the VP expansions of (5), an elementary logical technique employing truth tables (cf Kilbury 1986) allows us to construct the three CCRs of (9), which taken together with (7) and (8) admit the same set of LTs as the ID rules of (5):

(9) CCR 5': $\text{VP} \Rightarrow [\![\text{AUX} \equiv \sim\text{V}]\!]$
CCR 6': $\text{VP} \Rightarrow [\![\text{AUX} \supset (\text{VP} \wedge \sim \text{NP})]\!]$
CCR 7': $\text{VP} \Rightarrow [\![\text{V} \supset (\text{VP} \vee \text{NP})]\!]$

The CCRs of (9) have been formulated only on the basis of LTs with VP as root, however, and therefore fail to capture generalizations that apply to all LTs. Any daughter AUX must have a VP as sister, so CCR 6' may be restated as two simpler CCRs, CCR 5 and CCR 7, where CCR 5 does not depend on the root category. Furthermore, CCR 7' can be rewritten as CCR 8 since V cannot be a daughter of S. The following final set of CCRs thus emerges:

(10) CCR 1: $[\![\text{NP} \vee \text{VP} \vee \text{AUX}]\!] (\text{S} \vee \text{VP})$
CCR 2: $[\![\text{V}]\!] \text{VP}$
CCR 3: $[\![\sim\text{S}]\!]$
CCR 4: $\text{S} \Rightarrow [\![\text{NP} \wedge \text{VP}]\!]$
CCR 5: $[\![\text{AUX} \supset \text{VP}]\!]$
CCR 6: $\text{VP} \Rightarrow [\![\text{AUX} \equiv \sim\text{V}]\!]$
CCR 7: $\text{VP} \Rightarrow [\![\text{AUX} \supset \sim\text{NP}]\!]$
CCR 8: $[\![\text{V} \supset (\text{VP} \wedge \text{NP})]\!]$

The grammar can be extended to cover a VP dominating a single V by simply eliminating CCR 8. Moreover, CCR 6 can be logically conjoined with CCR 7 and CCR 3 with CCRs 5 and 8 to form single, complex CCRs.

3. Parsing with the CCR formalism

3.1 An Earley-based chart parser

The CCR formalism can be interpreted by a chart parser based on Shieber's (1984) adaptation of the Earley (1970) algorithm to the ID/LP format of GPSG. Modifications here involve the CCR formalism and details of Earley's predictor. As noted above, categories are treated as atoms here for simplicity of presentation.

Items, corresponding to edges of a chart, have the form ⟨i, j, A, α, β, σ⟩, where i and j are integers designating the beginning and end of the edge, A is the root category of the LT, α is the string of categories [3] already identified as daughters of A, β is the set of categories not in α which may be immediately dominated by A, and σ is a set of clauses expressing conditions that must be fulfilled in order for the LT to be accepted. A sentence is recognized if its analysis produces an item of the form ⟨0, n, S, α, β, Φ⟩, where n is the sentence length and Φ is the empty clause set.

The algorithm uses two basic procedures, the predictor and the completer, in addition to the scanner, which reads the next word of the input string and calls the other two procedures. The function of the predictor is to introduce new active edges in the chart. Its efficiency has been increased through the use of a first relation F and its transitive closure F^+ [4] , which are derived with the LP rules so that ⟨A, B⟩ ∈ F iff B can be the left-most daughter of A in some LT. Given an inactive item ⟨j, k, B, γ, τ, Φ⟩, an active item ⟨j, k, A, B, β, σ⟩ with ⟨A, B⟩ ∈ F is introduced iff there is some C such that ⟨C, A⟩ ∈ F or else C = A and

there is an active item ⟨i, j, D, α, δ, σ'⟩ such that C ∈ δ and σ' can be *reduced* (see below) with respect to C.

The predictor first constructs the set ψ of categories that may be immediately dominated by A, subtracts {B} from it, and then further subtracts the set χ of those categories which B may not precede, giving β in the new active item. [6] A clause set is then constructed as the conjunction of all CCRs (stated in simplified conjunctive normal form) applying to the members of ψ in a LT with A as root. Finally, this clause set is reduced (see below) with respect to B and the negation of each member of χ giving σ.

The completer extends active edges. Given the inactive edge ⟨j, k, B, γ, τ, Φ⟩ and the active edge ⟨i, j, A, α, β, σ⟩ such that B ∈ β and β' = β \ {B}, it subtracts the subset β'' of categories in β' that B cannot precede from β' giving β''', reduces σ with respect to B and the negation of each member of β'', giving σ', and then introduces the new item ⟨i, k, A, α∩B, β''', σ'⟩.

Reduction of a clause set σ with respect to a literal α tests the consistency of α with respect to the conditions stated by σ and amounts simply to the One-Literal Rule in the procedure of Davis and Putnam (cf Chang and Lee 1973, pp. 63-64). A clause set σ is reduced with respect to α by reducing each clause π ∈ σ with respect to α. The latter is accomplished as follows:

a) If α ∈ π, then π is removed from σ.
b) If β ∈ π and either α = ~β or ~α = β, then if β = π reduction fails and α is inconsistent with σ, and otherwise β is removed from π.

The following PROLOG program, which is called with

reduce_cnf(Literal, CNF, [], RedCNF),

implements reduction of a clause set in conjunctive normal form (CNF):

```
:- op(60, fx, ~).    /* '~' binds more strongly than ','. */

reduce_cnf(_, [], CNF, CNF).

reduce_cnf(Literal, [Clause | Clauses], OldCNF, RedCNF) :-
      reduce_clause(Literal, Clause, [], NewClause),
      append(OldCNF, NewClause, NewCNF),
      reduce_cnf(Literal, Clauses, NewCNF, RedCNF).


reduce_clause(_, [], [], [] ) :- !.

reduce_clause(_, [], Clause, [Clause] ).

reduce_clause(Literal, [Literal | _ ], _, [] ) :- !.

reduce_clause(~Cat, [Cat], [], _ ) :- !, fail.

reduce_clause(Cat, [~Cat], [], _ ) :- !, fail.

reduce_clause(~Cat, [Cat | Lits], OldClause, RedClause) :-
      !, reduce_clause(~Cat, Lits, OldClause, RedClause).

reduce_clause(Cat, [~Cat | Lits], OldClause, RedClause) :-
      !, reduce_clause(Cat, Lits, OldClause, RedClause).

reduce_clause(Lit1, [Lit2 | Lits], OldClause, RedClause) :-
      append(OldClause, [Lit2], NewClause),
      reduce_clause(Lit1, Lits, NewClause, RedClause).
```

3.2 A left-corner parser

PROLOG implementations of the chart parser described above suffer from inefficiency stemming from manipulations of the knowledge base during parsing. Such knowledge-base manipulations can be more easily avoided in a left-corner parser (cf Aho/Ullman 1972), as reflected in the following simplified PROLOG program:

```
parse(RCat, [Word | L1], L) :-
      lex(Word, Cat),
      left_corner(Cat, RCat, L1, L).

left_corner(Cat, Cat, L, L).

left_corner(Cat, RCat, L1, L) :-
      rule(Cat0, [Cat | Cats]),
      sisters(Cats, L1, L2),
      left_corner(Cat0, RCat, L2, L).

sisters([], L, L).

sisters([Cat | Cats], L1, L) :-
      left_corner(Cat, L1, L2),
      sisters(Cats, L2, L).
```

The variables **L**, **L1**, and **L2** designate difference lists for the input string as in Definite Clause Grammars, while **RCat** is the category assigned to the analyzed string. Lexical entries are represented in the form **lex(Word, Cat)** and rules with **rule(LHS, RHS)**, whereby **LHS** is a single category and **RHS** a list of categories. Adaptation of the program to the CCR formalism simply amounts to a replacement of the predicate **rule** as discussed above for the chart parser; the predicates **left_corner** and **sisters** are augmented with an argument for the clause set stating conditions that remain to be fulfilled for the phrase currently being analyzed. Performance is greatly improved by the addition of top-down filtering.

4. References

Aho, Alfred V. / Ullman, Jeffrey D. (1972): *The Theory of Parsing, Translation, and Compiling.* Volume 1: *Parsing.* Englewood Cliffs, N.J.: Prentice-Hall.

Chang, Chin-Liang / Lee, Richard Char-Tung (1973): *Symbolic Logic and Mechanical Theorem Proving*. New York and London: Academic Press.

Earley, Jay (1970): "An efficient context-free parsing algorithm," *Communications of the ACM* 13: 94-102.

Gazdar, Gerald / Klein, Ewan / Pullum, Geoffrey / Sag, Ivan (1985): *Generalized Phrase Structure Grammar.* Oxford: Blackwell.

Kay, Martin (1985): "Parsing in functional unification grammar," in D. R. Dowty et al. (eds), *Natural Language Parsing*, 251-278. Cambridge et al.: Cambridge University Press.

Kilbury, James (1985): "Chart parsing and the Earley algorithm," in U. Klenk (ed.), *Kontextfreie Syntaxen und verwandte Systeme*, 76-89. Tübingen: Niemeyer.

Kilbury, James (1986): "Category cooccurrence restrictions and the elimination of metarules," *Proceedings of COLING 86*, 50-55.

Kilbury, James (1987): "A proposal for modifications in the formalism of GPSG," *Proceedings of the Third Conference of the European Chapter of the ACL*, 156-159.

Seiffert, Roland (1987): "Chart parsing of unification-based grammars with ID/LP-rules," LILOG Report 22, IBM Deutschland GmbH.

Shieber, Stuart M. (1984): "Direct parsing of ID/LP grammars," *Linguistics and Philosophy* 7: 135-154.

Shieber, Stuart M. (1986): *An Introduction to Unification-Based Approaches to Grammar*. Stanford: CLSI.

Toulmin, Stephen (1958): *The Uses of Argument*. Cambridge: Cambridge University Press.

Weisweber, Wilhelm (1988): "Using constraints in a constructive version of GPSG" (in this volume).

Wirén, Mats (1987): "A comparison of rule-invocation strategies in context-free chart parsing," *Proceedings of the Third Conference of the European Chapter of the ACL*, 226-233.

Footnotes

[1] Interpretations in terms of the theory of feature instantiation in GKPS would be 'the root of LT t is an extension of α' and 'some daughter in LT t is an extension of α'.

[2] Compare the distinction between inherited and instantiated features in GPSG.

[3] This applies to the recognizer. The parser has items with a string of corresponding trees as α and may add an additional argument to items for semantic representations.

[4] The proposal to use F^+ in the predictor stems from an unpublished paper by Jochen Dörre and Stefan Momma of the University of Stuttgart. See Wirén (1987) for a discussion of such top-down filtering and of the advantages of a modification of the Earley predictor proposed in Kilbury (1985).

[5] Note that an item is inactive when the clause set of conditions remaining to be fulfilled is empty. An inactive edge may still be extended with optional categories.

[6] As Seiffer (1987) and Weisweber (1988) have pointed out, the treatment of LP restrictions is more difficult with complex categories subject to unification. LP restrictions may be fulfilled by a partially specified category but violated by its extension arising through further instantiation during parsing.