# ELLEIPO: A module that computes coordinative ellipsis for language generators that don't

**Karin Harbusch**
Computer Science Department
University of Koblenz-Landau
PO Box 201602, 56016 Koblenz/DE
harbusch@uni-koblenz.de

**Gerard Kempen**
Max Planck Institute for Psycholinguistics &
Cognitive Psychology Unit, Leiden University
PO Box 310, 6500AH Nijmegen /NL
gerard.kempen@mpi.nl

## Abstract

Many current sentence generators lack the ability to compute elliptical versions of coordinated clauses in accordance with the rules for Gapping, Forward and Backward Conjunction Reduction, and SGF (Subject Gap in clauses with Finite/Fronted verb). We describe a module (implemented in JAVA, with German and Dutch as target languages) that takes non-elliptical coordinated clauses as input and returns all reduced versions licensed by coordinative ellipsis. It is loosely based on a new psycholinguistic theory of coordinative ellipsis proposed by Kempen. In this theory, coordinative ellipsis is not supposed to result from the application of declarative grammar rules for clause formation but from a procedural component that interacts with the sentence generator and may block the overt expression of certain constituents.

## 1 Introduction

Coordination and coordinative ellipsis are essential tools for the sentence aggregation component of any language generator. Very often, when the aggregator chooses to combine several clauses into a single coordinate structure, the need arises to eliminate unnatural reduplications of coreferential constituents.

In the literature, one often distinguishes four major types of clause-level coordinative ellipsis:

- *Gapping* (as in (1)), with a special variant called *Long-Distance Gapping* (LDG). In LDG, the second conjunct consists of constituents stemming from different clauses — in (2), the main clause and the complement.
- *Forward Conjunction Reduction* (FCR; cf. (3) and the relative clause in (4)).
- *SGF* (Subject Gap in clauses with Finite/Fronted verb; as in (5), and
- *Backward Conjunction reduction* (BCR, also termed *Right Node Raising*; see (6)).

(1) Henk lives in Leiden and Chris ~~lives~~$_g$ in Delft
(2) My wife wants to buy a car, my son ~~wants~~$_g$ [~~to buy~~]$_{gl}$ a motorcycle.
(3) My sister lives in Utrecht and [~~my sister~~]$_f$ works in Amsterdam
(4) Amsterdam is the city [S where Jan lives and ~~where~~$_f$ Piet works]
(5) Why did you leave but didn't ~~you~~$_s$ warn me?
(6) Anne arrived before [~~three o'clock~~]$_b$, and Susi left after three o'clock

The subscripts denote the elliptical mechanism at work: *g*=Gapping, *gl*=LDG, *f*=FCR, *s*=SGF, *b*=BCR. We will not deal with VP Ellipsis and VP Anaphora because they generate pro-forms rather than elisions and are not restricted to coordination (cf. the title of the paper).

In current sentence generators, the coordinative ellipsis rules are often inextricably intertwined with the rules for generating non-elliptical coordinate structures, so that they cannot easily be ported to other grammar formalisms — e.g., Sarkar & Joshi (1996) for Tree Adjoining Grammar; Steedman (2000) for Combinatory Categorial Grammar; Bateman, Matthiessen & Zeng (1999) for Functional Grammar. Generators that do include an autonomous component for coordinative ellipsis (Dalianis, 1999; Shaw, 2002; Hielkema, 2005), use incomplete rule sets, thus risking over- or undergeneration, and incorrect or unnatural output.

The module (dubbed ELLEIPO, from Greek Ἐλλείπω 'I leave out') we present here, is less formalism-dependent and, in principle, less liable to over- or undergeneration than its competitors. In Section 2, we sketch the theoretical background. Section 3 and the Appendix describe our implementation, with examples from German. Finally, in Section 4, we discuss the prospects of extending the module to additional constructions.

## 2 Some theoretical background

ELLEIPO is loosely based on Kempen's (subm.) psycholinguistically motivated syntactic theory of clausal coordination and coordinative ellipsis. It departs from the assumption that the generator's strategic (conceptual, pragmatic) component is responsible for selecting the concepts and conceptual structures that enable identification of discourse referents (except in case of syntactically conditioned pronominalization). The strategic component may conjoin two or more clauses into a coordination and deliver as output a *non-reduced sequence* of conjuncts.[1] The concepts in these conjuncts are adorned with *reference tags,* and identical tags express coreferentiality.[2]

Structures of this kind serve as input to the (syn)tactical component of the generator, where they are grammatically encoded (lexicalized and given syntactic form) without any form of coordinative ellipsis. The resulting non-elliptical structures are input to ELLEIPO, which computes and executes options for coordinative ellipsis.

ELLEIPO's functioning is based on the assumption that *coordinative ellipsis does not result from the application of declarative grammar rules for clause formation but from a procedural component that interacts with the sentence generator and may block the overt expression of certain constituents.* Due to this feature, ELLEIPO can be combined, at least in principle, with various grammar formalisms. However, this advantage is not entirely *gratis*: The module needs a formalism-dependent interface that converts generator output to a (simple) *canonical form.*

[1]The strategic component is also supposed to apply rules of logical inference yielding the conceptual structures that underlie "*respectively* coordinations." Hence, the conversion of clausal into NP coordination (such as *Anne likes biking and Susi likes skating* into *Anne and Susi like biking and skating, respectively* is supposed to arise in the strategic, not the (syn)tactical component of the generator. This also applies to simpler cases without *respectively*, such as *John is skating and Peter is skating* versus *John and Peter are skating*. The module presented here does not handle these conversions (see Reiter & Dale (2000, pp. 133-139) for examples and possible solutions.)

[2]Coordinative ellipsis is insensitive to the distinction between "strict" and "sloppy" (token- vs. type-)identity.

## 3 A sketch of the algorithm

This sketch presupposes *and*-coordinations of only $n$=2 conjuncts. Actually, ELLEIPO handles *and*-coordinations with $n \geq 2$ conjuncts if, in every pair of conjuncts, the major constituents embody the same pattern of coreferences and contrasts.

ELLEIPO takes as input a non-elliptical *syntactic* structure that should meet the following four canonical form criteria (see Fig. 1 for the input tree corresponding to example (7).

(7) Susi hörte dass Hans einen Unfall hatte
Susi heard that Hans an accident had
und ~~dass~~$_f$ ~~Hans~~$_f$ sterben könnte
and that Hans die might
'Susi heard that Hans had an accident and might die'

□ Categorial (phrasal and lexical) nodes — bolded in Fig. 1 — carry reference tags (presumably propagated from the generator's strategic component). E.g., the tag "7" is attached to the root and head nodes of both exemplars of NP *Hans* in Fig. 1, indicating their coreferentiality. For the sake of computational uniformity, we also attach reference tags to non-referring lexical elements. In such cases, the tags denote *lexical* instead of referential identity. For instance, the fact that the two tokens of subordinating conjunction *dass* 'that' in Fig. 1 carry the same tag, is interpreted by ELLEIPO as indicating lexical identity. In combination with other properties, this licenses elision of the second *dass* (see (7)).

□ The conjuncts are sister nodes separated by coordinating conjunctions; we call these configurations *coordination domains*. The order of the conjuncts and their constituents is defined.

□ Every categorial node of the input tree is immediately dominated by a functional node.

□ Each clausal conjunct is rooted in an S-node whose daughter nodes (immediate constituents) are grammatical functions. Within a clausal conjunct, all functions are represented at the same hierarchical level. Hence, the trees are "flat," as illustrated in Fig. 1, and similar to the trees in German treebanks (NEGRA-II, TIGER).

ELLEIPO starts by demarcating "superclauses." Kempen (subm.) introduced this notion in his treatment of Gapping and LDG. An S-node dominates a superclause iff it dominates the entire sentence or a clause beginning with a subordinating conjunction (CNJ). In Fig. 1, the strings dominated by $S_1$, $S_5$ and $S_{12}$ are super-

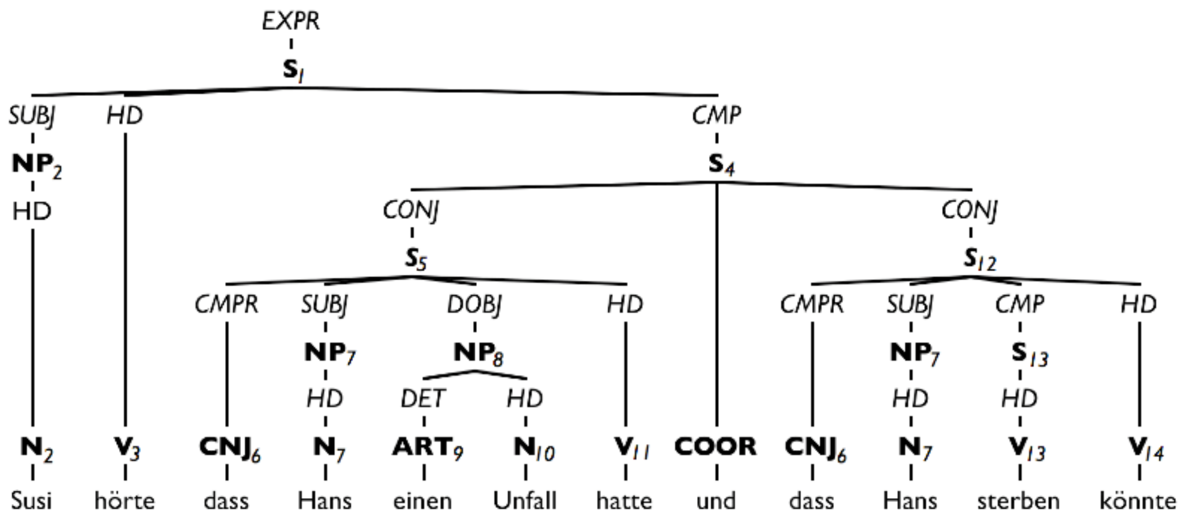

Figure 1. Slightly simplified canonical form of the non-elliptical input tree underlying sentence (7).

clauses. Note that $S_{12}$ includes clause $S_{13}$, which is not a superclause.

Then, ELLEIPO checks all coordination domains for elision options, as follows:

□ *Testing for forward ellipsis:* Gapping (including LDG), FCR, or SGF. This involves inspecting (recursively for every S-node) the set of immediate constituents (grammatical functions) of the two conjuncts, and their reference tags. Complete constituents of the right-hand conjunct may get marked for elision, depending on the specific conditions listed in the Appendix.

□ *Testing for BCR*. ELLEIPO checks — word-by-word, going from right to left — the coreference tags of the conjuncts. As a result, complete or partial constituents in the right-hand periphery of the left conjunct may get marked for elision.

The final step of the module is *ReadOut*. After all coordination domains have been processed, a (possibly empty) subset of the terminal leaves of the input tree has been marked for elision. In the examples below, this is indicated by subscript marks. E.g., the subscript “*g*” attached to *esst* ‘eat’ in (9b) indicates that Gapping is allowed. ReadOut interprets the elision marks and, in ‘standard mode,’ produces the shortest elliptical string(s) as output (e.g. (9c)). In ‘demo mode,’ it shows individual and combined elliptical options on user request. Furthermore, *auch* ‘too’ is added in case of “Stripping,” i.e. when Gapping leaves only one constituent as remnant.

Example (10) illustrates a combination of Gapping and BCR, with the three licensed elliptical output strings shown in (10c). In (11), Gapping combines with BCR in the subordinate clauses. The fact that here, in contrast with (10), the subordinate clauses do not start their own superclauses, now licenses LDG. However, ReadOut prevents LDG to combine with BCR, which would have yielded the unintended string *Anne versucht Bücher und Susi Artikel*.

(9) *a*. Wir essen Äpfel und ihr esst Birnen
‘We eat apples and you(pl.) eat pears’
*b*. Wir essen Äpfel und ihr esst$_g$ Birnen
*c*. Elliptical option:
Wir essen Äpfel und ihr Birnen

(10) *a*. Ich hoffe, dass Hans schläft und du hoffst, dass Peter schläft
‘I hope that Hans sleeps and you hope that Peter sleeps’
*b*. Ich hoffe dass Hans schläft$_b$ und du hoffst$_g$ dass Peter schläft
*c*. Elliptical options:
*Gapping*: Ich hoffe, dass Hans schläft und du, dass Peter schläft
*BCR*: Ich hoffe, dass Hans und du hoffst, dass Peter schläft
*Gapping and BCR*: Ich hoffe, dass Hans und du, dass Peter schläft

(11) *a*. Anne versucht Bücher zu schreiben and Susi versucht Artikel zu schreiben
‘Anne tries to write books and Susi tries to write articles’
*b*. Anne versucht Bücher zu$_b$ schreiben$_b$ und Susi versucht$_g$ Artikel zu$_{gl}$ schreiben$_{gl}$
*c*. Elliptical options:
*Gapping:* Anne versucht Bücher zu schreiben und Susi Artikel zu schreiben
*BCR:* Anne versucht Bücher und Susi versucht Artikel zu schreiben
*Gapping and BCR:* Anne versucht Bücher und Susi Artikel zu schreiben
*LDG:* Anne versucht Bücher zu schreiben und Susi Artikel

## 4 Conclusion

Currently, ELLEIPO can handle all major types of clausal coordinative ellipsis in German and Dutch. However, further finetuning of the rules is needed, e.g., in order to take subtle *semantic* conditions on SGF and Gapping into account. We expect further improvements by allowing for interactions between the ellipsis module and the generator's pronominalization strategy. Work on porting ELLEIPO to related languages, in particular English, and to coordinations of non-clausal constituents (NP, PP, AP) is in progress.

## References

John A. Bateman, Christian M.I.M. Matthiessen & Licheng Zeng (1999). Multilingual natural language generation for multilingual software: a functional linguistic approach. *Applied Artificial Intelligence, 13*, 607–639.

Ehud Reiter & Robert Dale (2000). *Building natural language generation systems.* Cambridge UK: Cambridge University Press.

Hercules Dalianis, (1999). Aggregation in natural language generation. *Computational Intelligence, 15*, 384–414.

Feikje Hielkema (2005). *Performing syntactic aggregation using discourse structures.* Unpublished Master's thesis, Artificial Intelligence Unit, University of Groningen.

Gerard Kempen (subm.). Symmetrical clausal coordination and coordinative ellipsis as incremental updating. Downloadable from: `www.gerardkempen.nl/publicationfiles`

Anoop Sarkar & Aravind Joshi (1996). Coordination in Tree Adjoining Grammars: Formalization and implementation. In: *Procs. of COLING 1996,* Copenhagen, pp. 610–615.

James Shaw (1998). Segregatory coordination and ellipsis in text generation. In: *Procs. of COLING 1998,* Montreal, pp. 1220–1226.

Mark Steedman (2000). *The syntactic process.* Cambridge MA: MIT Press.

## Appendix: A sketch of the algorithm

```
1  proc ELLEIPO(SENT) {
2  mark root nodes of all superclauses in SENT;
3  for all coordinators and their left- and right-
   neighboring clauses (LCONJ, RCONJ) {
4     call GAP(LCONJ, RCONJ, "g"); // string "g"
      gets an "l" attached for any level of LDG; the
      resulting string is attached, in line 9 of GAP, to
      leaves that ReadOut interprets as elidable//
5     FCRcontrol=TRUE; BCRcontrol=TRUE;
      //global variables communicating the end of
      left- or right-peripheral identical strings//
6     call FCR(LCONJ, RCONJ);
7     call SGF(LCONJ, RCONJ);
8     call BCR(LCONJ, RCONJ);};
9  call ReadOut();}
```

```
1  proc GAP(LC, RC, ELLIM) {//ELLIM records
   the 'elliptical mechanism(s)' applied: "g" for
   Gapping; "gl", "gll", etc., for LDG levels//
2  check whether the HEAD verb of LC and the
   HEAD verb of RC have the same reference tag;
3  if not then return; //verbs differ=>no gapping//
4  check whether all other constituents in LC have a
   counterpart in RC with same grammatical function,
   not necessarily at the same left-to-right position;
   modifiers need identical mod-type;
5  if not then return; // no proper set of contrastive
   pairs of immediate constituents found//
6  for all pairs (LSIB, RSIB) resulting from (4) {
7     if (LSIB is an S-node) & (LSIB is not a super-
      clause root) then {//LSIB = "left sibling"//
8        if (LSIB and RSIB are not coreferential)
9           then attach "l" to ELLIM;//LDG variant//
10       call GAP(LSIB, RSIB, ELLIM);}
11    if NOT((LSIB is an S-node) & (LSIB and RSIB
      are coreferential))
12       then mark RSIB for elision, with ELLIM;}}
```

```
1  proc FCR(LC, RC) {
2  while (FCRcontrol) {
3     set LSIB and RSIB to left-most daughter of LC
      and RC, resp.;
4     if (LSIB and RSIB are not coreferential)
5           then {FCRcontrol = FALSE;
6                 return;}
7     if (LSIB is an S-node)
8         then call FCR(LSIB, RSIB);
9     call FCR(right neighbor of LSIB, right neigh-
      bor of RSIB);
10    mark RSIB for elision by adding "f";}}
```

```
1  proc SGF(LC, RC) {
2  if (NOT(SUBJ is 1st daughter of LC)) & (HEAD
   is 2nd daughter of LC) & (SUBJ is 1st or 2nd
   daughter of RC) & (HEAD is 1st or 2nd daughter
   of RC)
3     then mark RC's SUBJ for elision, with "s";}
```

```
1  proc BCR(LC, RC) {
2  while (BCRcontrol) {
3     set LSIB and RSIB to right-most daughter node
      of LC and RC, respectively;
4     if (LSIB and RSIB are not coreferential)
5         then {BCRcontrol = FALSE; return;};
6     call BCR(LSIB, RSIB);
7     call BCR(left neighbor of LSIB, left neighbor
      of RSIB);
8     if (RSIB is a terminal node)
9         then mark LSIB for elision, with "b";}}
```