A GENERALIZED RECONSTRUCTION ALGORITHM FOR ELLIPSIS RESOLUTION

Shalom Lappin and Hsuc-Hueh Shih
Department of Linguistics
School of Oriental and African Studies
University of London
Thornhaugh Street, Russell Square
London WC1H OXG, UK
shalom@semantics.soas.ac.uk, hh112@eng.cam.ac.uk

Abstract

We present an algorithm which assigns interpretations to several major types of ellipsis structures through a generalized procedure of syntactic reconstruction. Ellipsis structures are taken to be sequences of lexically realized arguments and/or adjuncts of an empty verbal head. Reconstruction is characterized as the specification of a (partial) correspondence relation between the unrealized head verb of an elided clause and its argument and adjuncts on one hand, and the head of a non-elided antecedent sentence and its arguments and adjuncts on the other. The algorithm generates appropriate interpretations for cases of VP ellipsis, pseudo-gapping, bare ellipsis (stripping), and gapping. It provides a uniform computational approach to a wide range of ellipsis phenomena, and it has significant advantages over several other approaches to ellipsis which have recently been suggested in the computational and linguistic literature.

1 Introduction

Ellipsis structures pose an important problem for NLP systems designed to provide text understanding or to handle dialogue. They contain information which is not overtly expressed, but which must be recovered through the identification of an antecedent. However, unlike pronominal anaphora, which is resolved by matching a pronoun with an antecedent noun phrase, the interpretation of an ellipsis fragment (or sequence of fragments) generally involves mapping it (them) into a sentential structure by association with an antecedent clause. It is possible to distinguish two main approaches to ellipsis resolution. The first seeks to associate an elided construction directly with a semantic representation, while the second mediates semantic interpretation through the reconstruction of the syntactic structure of the antecedent. The algorithm we propose implements the second view of ellipsis, by characterizing ellipsis resolution as the specification of a relation of (possibly partial) correspondence between the lexically unrealized head of an elided clause and its arguments and adjuncts as one term of the relation, and the realized head of the antecedent clause and its arguments and adjuncts as the second term.

The algorithm is a generalized procedure for syntactic reconstruction which provides a unified way of handling a significant variety of ellipsis constructions. It modifies and extends the reconstruction strategy for handling VP ellipsis suggested in Lappin and McCord (1990). The algorithm covers VP ellipsis, illustrated in 1, pseudo-gapping (in 2), bare ellipsis involving sequences of bare arguments, adjuncts or both (in 3), and gapping (in 4).

1. John completed his paper before he expected to.
2. John sent the flowers to Lucy before he did the chocolates.
3. Bill wrote reviews for the journal last year, and articles this year.
4. Sam teaches in London, and Lucy in Boston.

It will be a useful component for source analysis in machine translation, text understanding systems, and discourse interpretation systems.

2 The Reconstruction Algorithm

Let an *ellipsis fragment* be a phrase which (i) occurs outside of a lexically realized sentence, and (ii) is interpreted as an argument or an adjunct of the head verb of a non-elided sentence. Let s = <b_1,...,b_k> (1 ≤ k) be a sequence of ellipsis fragments such that, for each $b_i \in s$, b_{i+1} immediately follows b_i. Take s to be maximal in that there is no ellipsis fragment, b_0 or b_{k+1}, not contained in s, which immediately precedes or immediately follows an element of s.

A. Identify an antecedent sentence S for s.
B. Take the head verb of S, A, as the new interpreted head of the sentence to be constructed from s (we will refer to the new head as A').
C. Consider in sequence each argument slot $Slot_i$ in the SUBCAT list of A.

1. If there is a phrase C' in s which is of the appropriate type for filling $Slot_i$, then fill $Slot_i$ in the SUBCAT list of A' with C' and remove C' from s. Else,

2. If $Slot_i$ is filled by a phrase C, then fill $Slot_i$ in A' with C, and list C as a new argument of A'. Else,

3. If $Slot_i$ is empty in the frame of A, it remains empty in the frame of A'.

4. Construct a list, Arg-List, of the phrases which fill the SUBCAT list slots of A'.

D. Construct a list of adjunct phrases for A' as follows.

1. Construct the list L of adjunct phrases in s.

a. If L ≠ nil, then for each element AdjP' of L, fill an adjunct slot for A' with AdjP'.

2. Consider each adjunct slot of A filled by a phrase AdjP.

a. If there is a phrase AdjP' filling an adjunct slot of the same type in A', then leave AdjP' in this slot and remove AdjP' from s. Else,

b. Fill an adjunct slot for A' with AdjP, and list AdjP as a new adjunct of A'.

3. Construct a list, Adj-List, of all the phrases which fill adjunct slots of A'.

E. Generate a new syntactic structure as follows.

1. Concatenate Arg-List and Adj-List to create a combined list, Ph-List, of the phrasal arguments and adjuncts of A'.

2. Reorder the elements of Ph-List to produce a new list, Ord-Ph-List, in which the sequence of arguments and adjunct phrases corresponds to the order of arguments and adjuncts phrases of A.

3. Construct a new clause headed by A'.

4. Substitute Ord-Ph-List for the list of arguments and adjunct phrases of A' in the new structure.

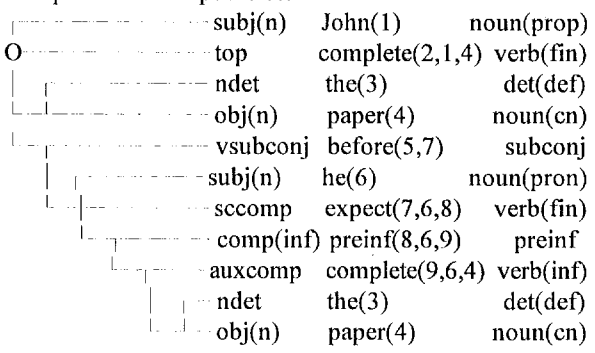
3 Coverage and Implementation of the Algorithm

At this point, the algorithm has been partially implemented in Prolog to apply to the output of McCord's English Slot Grammar ESG parser (which also runs in Prolog) in order to generate reconstructed trees for VP ellipsis and pseudo-gapping constructions (see McCord et al. (1992) for a description of ESG and NLP systems which run on top of it). Examples of the algorithm's output for theses cases are given in 5 and 6.

VP Ellipsis

5. John completed the paper before he expected to.

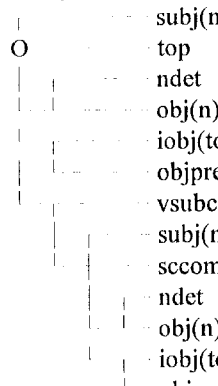
Interpreted VP ellipsis tree.



Pseudo-Gapping

6. John sent the flowers to Mary before he did the chocolates.

Interpreted VP ellipsis tree.



	subj(n)	John(1)	noun(prop)
O	top	send(2,1,4,6)	verb(fin)
	ndet	the(3)	det(def)
	obj(n)	flower(4)	noun(cn)
	iobj(to)	to(5,6)	prep(to)
	objprep	Mary(6)	noun(prop)
	vsubconj	before(7,9)	subconj
	subj(n)	he(8)	noun(pron)
	sccomp	send(9,8,11,6)	verb(fin)
	ndet	the(10)	det(def)
	obj(n)	chocolate(11)	noun(cn)
	iobj(to)	to(5,6)	prep(to)
	objprep	Mary(6)	noun(prop)

The algorithm is currently being re-implemented in Prolog to apply to the output of a modified HPSG (Pollard and Sag (1994)) grammar designed to handle ellipsis. We are developing the grammar within the framework of Erbach's (1995) ProFIT system for augmenting Prolog with typed feature structures. The feature structures which the grammar currently generates for simple bare argument and bare adjunct ellipsis cases are illustrated by the AVM's in 7 and 8, respectivcly (cases of bare adverb ellipsis are discussed in Chao (1988) and Kempson and Gabbay (1993)).

7. John gives Mary flowers, and chocolates too.

```
phon![john, gives, mary, flowers, and, chocolates, too]&
   syn!loc!subcat![]&
   dtrs!head_dtr!phon![and]&
            syn!loc!head!conj!<conj&
                  subcat![_Q1, _O2]&
      comp_dtrs![_Q1&
            phon![john, gives, mary, flowers]&
            syn!loc!head!_J&
                  subcat![]&
            dtrs!head_dtr!phon![gives, mary, flowers]&
                     syn!loc!head!_J&
                           subcat![_S]&
                     dtrs!head_dtr!phon![gives]&
                              syn!loc!head!_J&
                                    vform!<fin&
                                 subcat![_S, _C1, _I1]&
                        comp_dtrs![_C1&
                              phon![mary]&
                              syn!loc!head!case!<acc&
                                    subcat![],
                              _I1&
                              phon![flowers]&
                              syn!loc!head!case!<acc&
                                    subcat![]
                              ]&
                        comp_dtrs![_S&
                              phon![john]&
                              syn!loc!head!case!<nom&
                                    subcat![]          ],
                  _O2&
                  phon![chocolates, too]&
                  syn!loc!head!_T1&
                        subcat![]&
                  dtrs!head_dtr!phon![chocolates]&
                           syn!loc!head!_T1&
                                 subcat![]&
                           dtrs!head_dtr!phon![chocolates]&
                                    syn!loc!head!_T1&
                                          case!<acc&
                                       subcat![]&
                              comp_dtrs![]&
                     comp_dtrs![]&
                     adj_dtrs!phon![too]&
                           syn!loc!head!atype!<too&
                                 subcat![]          ]
```

8. John sings, and beautifully too.

```
phon![john, sings, and, beautifully, too]&
   syn!loc!subcat![]&
   dtrs!head_dtr!phon![and]&
                  syn!loc!head!conj!<conj&
                        subcat![_E1, _W2]&
      comp_dtrs![_E1&
            phon![john, sings]&
            syn!loc!head!_J&
                  subcat![]&
            dtrs!head_dtr!phon![sings]&
                     syn!loc!head!_J&
                           subcat![_S]&
                     dtrs!head_dtr!phon![sings]&
                              syn!loc!head!_J&
                                    vform!<fin&
                                 subcat![_S]&
                        comp_dtrs![]&
            comp_dtrs![_S&
                     phon![john]&
                     syn!loc!head!case!<nom&
                           subcat![]          ],
      _W2&
      phon![beautifully, too]&
      syn!loc!head!_J1&
            subcat![]&
      dtrs!head_dtr!phon![beautifully]&
                     syn!loc!head!_J1&
                           subcat![]&
               dtrs!head_dtr!phon![beautifully]&
                        syn!loc!head!_J1&
                              subcat![]&
                        dtrs!head_dtr!phon![]&
                                 syn!loc!head!_J1&
                                       subcat![]&
                              dtrs!head_dtr!phon![]&
                                 syn!loc!head!_J1&
                                       vform!<elided&
                                             subcat![]&
                                    comp_dtrs![]&
                           comp_dtrs![]&
                     adj_dtrs!phon![beautifully]&
                        syn!loc!head!atype!<others&
                              subcat![]&
                  comp_dtrs![]&
            comp_dtrs![]&
            adj_dtrs!phon![too]&
                  syn!loc!head!atype!<too&
                  subcat![]          ]
```

The bare NP *chocolates* is the head of the

elided clause in the second conjunct of 7. The generalized ellipsis reconstruction algorithm will identify *gives* as the head V of the antecedent clause in the first conjunct, and then will fill one of the positions in its SUBCAT list with the local features of *chocolates*. If it fills the direct object (third complement) position of this list with the bare NP, then it will fill the subject and indirect object positions with the local features of *John* and *Mary*, generating the reconstructed feature structure corresponding to 9.

9. $[_{IP}\,[_{NP}$ John$]\,[_{VP}\,[_{V}$ gives$]\,[_{NP}$ Mary$]$
$[_{NP}$ flowers$]]]$ and
$[_{IP}\,[_{NP}$ *John*$]\,[_{VP}\,[_{VP}\,[_{V}$ *gives*$][_{NP}$ *Mary*$]$
$[_{NP}$ chocolates$]]\,[_{AdvP}$ too$]]]$

By contrast, the bare adverb *beautifully* is an adjunct daughter of a VP headed by an empty verb in 8. This is due to the fact that in our grammar, an adverb is an adjunct which modifies a VP. The algorithm will identify *sings* as the head V of the antecedent clause and substitute it for the empty V in 8. This will yield a feature structure corresponding to 10.

10. $[_{IP}\,[_{NP}$ John$]\,[_{VP}\,[_{V}$ plays$]]]$ and
$[_{IP}[_{NP}$ *John*$]\,[_{VP}\,[\,_{VP}\,[_{VP}\,[_{V}$ *plays*$]]$
$[_{AdvP}$ beautifully$]]$ too$]]$

We employ a rule which permits an unbounded number of adverbs to be generated in successively higher VP's through left recursion on the daughter VP node. The relevant PS rule is of the form VP $\rightarrow$ VP, ADV. We require this rule in order to allow for the fact that there is no apparent upper bound on the number of adverbs in a VP. 11 indicates that it is possible to obtain an unbounded number of bare adverbial adjuncts in an ellipsis site.

11a. John sang, but not in New York.
b. John sang, but not in New York at the concert.
c. John sang, but not in New York at the concert for three hours.
d. John sang, but not in New York at the concert for three hours on Tuesday.
e. John sang, but not in New York at the concert for three hours on Tuesday to impress his music teacher.

4 Comparison with Other Approaches to Ellipsis

Reinhart (1991) suggests a syntactic reconstruction account of bare ellipsis which adjoins an NP in the antecedent clause to an NP fragment by LF movement. The result is a conjoined NP which, taken as a generalized quantifier, applies to the antecedent clause, interpreted as a predicate formed by lambda abstraction. So, for example, adjunction of *flowers* in the antecedent clause of 7 to the NP fragment *chocolates* in the ellipsis site produces the LF structure 12a, which is interpreted as 12b.

12a. $[_{IP'}[_{IP}$ John gives Mary t_1 $]$
$[_{NP}[_{NP}$ flowers$]_1\,[_{NP}$ and $[_{NP}$ chocolates$]_2]_2]]$
b. (flowers and chocolates)(λx[john gives mary x])

Given that Reinhart's analysis relies on LF adjunction of an NP in the antecedent to an NP in the ellipsis site in order to create a generalized quantifier corresponding to a coordinate NP, it is not clear how it can apply to bare ellipsis cases like 3, in which a sequence of arguments and adjuncts appear in the ellipsis site. Moreover, the analysis cannot deal with bare ellipsis cases like 8, where a bare adjunct fragment does not correspond to any constituent in the antecedent clause. Therefore, this account does not cover the full range of bare ellipsis cases. As we have seen, the proposed generalized reconstruction algorithm does handle bare ellipsis structures like 8. In cases like 3 the algorithm will substitute the head V of the antecedent for the empty verb of the elided clause, and the bare PP adverb will modify the VP headed by this verb. The algorithm will fill some of the complement positions in the SUBCAT list of the reconstructed V with the NP arguments in the ellipsis site, and it will fill the remaining positions with arguments inherited from the antecedent head V. This procedure will yield

at least one appropriate reconstruction for the elided clause.

Dalrymple et al. (1991) and Shieber et al. (1995) present a generalized semantic account which employs higher order-unification of property and relation variables to resolve ellipsis. Their general strategy is to specify the interpretation of the antecedent clause as an equation between a propositional variable S and a predicate-argument structure. The arguments of the predicate correspond to the fragments in the ellipsis site, and ellipsis resolution consists in finding an appropriate value for the predicate variable which can apply to both the sequence of arguments in the interpretation of the antecedent clause, and the sequence of arguments in the ellipsis site. Given the equations in 13a-c, higher-order unification correctly generates 13d as the interpretation of 3.

13a. $\langle a_1,a_2\rangle$ = <book reviews,last year> & $\langle b_1,b_2\rangle$ = <articles,this year>
b. S1 = (wrote book reviews for the journal (during) last year)(bill)
c. R = $\lambda x\lambda y$[bill wrote x for the journal (during) y]
d. (book reviews)(λx[(last year)(λy[bill wrote x for the journal (during) y])]) and (articles)(λx[(this year)(λy[bill wrote x for the journal (during) y])])

While the higher-order unification analysis can deal with bare ellipsis cases like 3 (as well as VP ellipsis and pseudo-gapping), it is not clear how to apply it to bare ellipsis examples like 8, where the adjunct in the ellipsis site lacks a corresponding element in the antecedent clause. Lappin (1996) suggests positing a free manner adverbial function variable in the lexical semantic representation of verbs like *sing*. This will permit the specification of the equations in 14a-c for 8. Higher-order unification solves these equations to yield 14d, the desired interpretation of 8.

14a. $a_1 = f_{manner}$ & b_1 = beautifully
b. S_1 = (f_{manner}(plays))(john)
c. P = λf[(f(plays)(john)]
d. λf[(f(plays))(john)](f_{manner}) and λf[(f(plays))(john)](beautifully)

In fact, this solution does not generalize to cases like 11, which indicate that there is no upper bound on the number of antecedentless bare adjuncts which can appear in a bare ellipsis sequence. As it is not possible to posit an unbounded number of free adjunct function variables in the semantic representation of a verb (VP), it seems that the higher-order unification analysis cannot deal with these cases.

The generalized reconstruction algorithm presented here does not require the presence of constituents in the antecedent corresponding to adjunct elements of the fragment sequence. When a bare adjunct phrase AdjP does not correspond to a phrase in the antecedent clause, AdjP is simply added to the list of adjuncts of the new head verb of the reconstructed clause. Therefore, the algorithm produces the correct reconstructed forms for the elided clauses in 11.

Another problem is posed by the fact that, as higher-order unification applies to semantic interpretations of antecedents, it will not have access to syntactic structure. But at least some cases of ellipsis resolution seem to require reference to this structure. Consider the contrast between 15a and 15b.

15a. The students sent invitations to the professors yesterday, and to each other today.
b.??The students said that John sent invitations to the professors yesterday, and to each other today.

The elided conjunct in 15b is ill-formed because the reciprocal NP *each other* in the bare argument is interpreted as illicitly bound from outside of its local syntactic domain. By contrast, the generalized reconstruction algorithm generates the full syntactic structure of the elided clause, and so it provides the representation required to specify the contrast between 15a and 15b.

5 Conclusion

We have proposed a generalized reconstruction algorithm for ellipsis resolution. The algorithm provides a unified computational procedure for assigning interpretations to a significant variety of ellipsis constructions. The basic strategy which the algorithm encodes is to reconstruct an elided clause by (i) taking its head verb V' to be identical to the head verb V of an antecedent clause, (ii) filling the argument positions in the SUBCAT list of V' with the NP's in the ellipsis site, (iii) inheriting NP arguments of V to fill the corresponding argument positions in the SUBCAT list of V' which the NP's in the ellipsis site do not occupy, (iv) applying the adjuncts in the ellipsis site to the (possibly successive) VP('s) headed by V', and (v) inheriting any adjuncts modifying V as modifiers of the VP('s) which V' heads, when these adjuncts do not correspond to adjuncts in the ellipsis site. The algorithm has wider empirical coverage than other current approaches which have been suggested within the computational and linguistic literature. It can be integrated into a more comprehensive NLP system to recover missing information for purposes of text and dialogue understanding.

6 Acknowledgments

We are grateful to Chris Brew, Jo Calder, Claire Grover, and Suresh Manandhar for helpful comments on some of the ideas proposed here and for useful advice on implementational issues. The research described in this paper is supported by grant GR/K59576 from the Engineering and Physical Science Research Council of the UK.

References

Chao, W. (1988), *On Ellipsis*, Garland Publishing Co., New York.

Dalrymple, M., S. Shieber, and F. Pereira (1991), "Ellipsis and Higher-Order Unification", *Linguistics and Philosophy* 14, pp. 399-452.

Erbach, G. (1995), "ProFIT: Prolog with Features, Inheritance and Templates", *Proceedings of the Seventh Conference of the European Association for Computational Linguistics*, pp. 180-187.

Kempson, R. and D. Gabbay, (1993), "How We Understand Sentences. And Fragments Too?" in M. Cobb (ed.), *SOAS Working Papers in Linguistics and Phonetics* 3, pp. 259-336.

Lappin, S. (1996), "The Interpretation of Ellipsis" in S. Lappin (ed.), *The Handbook of Contemporary Semantic Theory*, Blackwell, Oxford, pp. 145-175.

Lappin, S. and M. McCord (1990), "Anaphora Resolution in Slot Grammar", *Computational Linguistics* 16, pp. 197-212.

McCord, M., A. Bernth, S. Lappin, and W. Zadrozny (1992), "Natural Language Processing within a Slot Grammar Framework", *International Journal on Artificial Intelligence Tools* 1, pp. 229-277.

Pollard, C. and I. Sag (1994), *Head-Driven Phrase Structure Grammar*, University of Chicago Press, Chicago, IL.

Reinhart, T. (1991), "Elliptic Conjunctions-Non-Quantificational QR" in A. Kasher (ed.), *The Chomskyan Turn*, Blackwell, Oxford, pp. 360-384.

Shieber, S., F. Pereira, and M. Dalrymple (1995), *Interactions of Scope and Ellipsis*, ms., Harvard University, AT&T Bell Laboratories, and Xerox Parc.