A compositional account of VP ellipsis

Markus Egg[1] and Katrin Erk[2]
[1]Department of Computational Linguistics, Universität des Saarlandes
[2]Programming Systems Lab, Universität des Saarlandes

Abstract

We present an approach to VP ellipsis that allows the direct derivation of source and target sentences (the former need not be unique) during semantic construction. Specific syntactic constituent structures are associated with ellipsis potential, which can then be discharged by pro-verbs like *did (too)*. The determination of source and target sentence, which is done with semantic features in an HPSG framework, is coupled with a comprehensive analysis of ellipsis, which also handles its interaction with scope and anaphora.

1 Introduction

VP ellipses like (1) consist of two sentences, the *source sentence* (SoS) and the *target sentence* (TaS). In the TaS a VP is left out and replaced by a pro-form, here *does too*. The SoS and the TaS have the same meaning, except for the semantic contributions of the respective subject NPs, in this case *John* in the SoS and *Bill* in the TaS.

(1) John wants to read *Jane Eyre*, and Bill does too.

Sentence (1) is not ambiguous w.r.t. potential SoSs. Thus, there is only one feasible way of understanding the TaS: 'Bill wants to read *Jane Eyre*'. But this is not always so. E.g., the TaS of (2) may be understood as 'Bill reads' or 'Bill wants Max to read'. Such sentences, where the TaS is part of a relative clause within an object NP, are called 'antecedent-contained deletions' (ACD).

(2) John wants Max to read everything that Bill does.

The question we are addressing in this paper is: given an elliptical target sentence, where do we find a suitable source sentence? There are few previous approaches to an automatic determination of source and target sentences in VP ellipsis. Hardt

Proceedings of the 8th International HPSG Conference, Norwegian University of Science and Technology (3-5 August 2001)
Frank van Eynde, Lars Hellan and Dorothee Beermann (editors)
2002.CSLI Publications. http://csli-publications.stanford.edu/

(1997) considers all VPs within the previous three sentences as potential SoS-VPs and ranks them by several heuristics. We feel, however, that the set of possible SoSs can be narrowed down much more, using hard syntactic constraints.

The main idea of our approach is that there is always a kind of *joint* between SoS and TaS. Sentence conjunction can function as such a joint, as can structures that underlie ACD cases (in particular, VPs that consist of a transitive verb and an NP). These joints can be identified as constituent structures in the syntactic analysis. Given an elliptical target sentence, there must be such a joint above it, and for each joint we can specify where the source sentence candidates it licenses are. So we associate certain constituent structures with *ellipsis potential*, that is, they collect source sentence candidates. (If the ellipsis is ambiguous with respect to the possible source sentence, then more than one candidate is gathered.) This potential can only be *discharged* by a pro-form like *do (too)*. Discharging the potential means determining the meaning of the target sentence by relating it to the meaning of one of the source sentence candidates. For ellipses like (2) with more than one feasible SoS, the ellipsis potential will comprise more than one candidate. We model this underdetermination by *semantic underspecification.*

There are two questions connected with ellipsis: on the one hand finding a suitable SoS candidate, on the other hand reconstructing the missing material. We couple our approach to finding SoS candidates with a reconstruction using the Constraint Language for Lambda Structures (CLLS) (Egg, Niehren, Ruhrberg, and Xu 1998; Egg, Koller, and Niehren 2001). It reconstructs the missing material within the semantic structure, in a similar way as Dalrymple, Shieber, and Pereira (1991).

Plan of the paper. In Section 2 we give a brief introduction into the Constraint Language for Lambda Structures and show how it can be used to model ellipsis. The central section is Section 3, where we present our approach to finding ellipsis antecedents. Section 4 compares it to the other approaches mentioned above. Section 5 concludes and delineates further research.

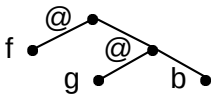
2 CLLS

In this section, we give a brief overview of the Constraint Language for Lambda Structures, CLLS. An extensive introduction (including all the necessary definitions) are given in the paper by Egg, Koller, and Niehren (2001). Generally speaking, CLLS is a language for the partial description of λ-terms that can be used in an underspecified analysis of scope, ellipsis, and anaphora.

2.1 Introduction to CLLS

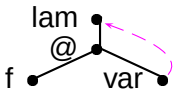
A constraint in the Constraint Language for Lambda Structures describes a λ-term as a *lambda structure*, which is a node-labelled tree enriched by additional edges to model λ-binding. Let us explain this in two steps. First, terms can be represented as trees. For example, the term $f(g(b))$ can be drawn as the tree (3). Functional application is represented by a node with the binary label '@' for *apply*. Its first child is the function being applied, the second child is the argument:

(3)



Second, λ-terms can be represented as trees decorated with λ-links. For example, the λ-term $\lambda x.f(x)$ can be drawn as the lambda structure (4). Abstraction and bound variables are represented using the labels lam and var, and the λ-link indicates which variable is bound by which binder:

(4)



Anaphoric binding in lambda structures is handled in a similar way as λ-binding: an anaphor results in a node labelled ANA, and an anaphoric binding edge (an ante-link from an anaphor node to the node for its antecedent) indicates where it is bound. Note that while the mechanism is similar to λ-binding, anaphoric binding edges and λ-binding edges are distinct.

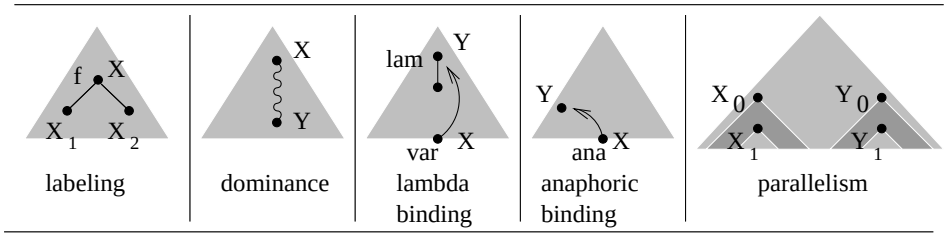


Figure 1: The literals of CLLS

The language CLLS consists of predicates for describing lambda structures. Its syntax is defined as follows. X, Y, etc. are variables which denote nodes in a lambda structure.

$$\begin{array}{rcl} \varphi & ::= & X{:}f(X_1, \ldots, X_n) \mid X \lhd^* Y \\ & \mid & \lambda(X) = Y \mid \mathsf{ante}(X) = Y \\ & \mid & X_0/X_1 \sim Y_0/Y_1 \mid \varphi \wedge \varphi' \end{array}$$

A CLLS formula or *constraint* is a conjunction of atomic literals; it is satisfied by a lambda structure and a variable assignment iff all its literals are satisfied in the following sense. A *labelling literal* $X{:}f(X_1, \ldots, X_n)$ is satisfied iff X denotes a node with label f and children that are denoted by $X_1, \ldots, X_n$. A *dominance literal* $X \triangleleft^* Y$ is satisfied iff X denotes a (reflexive, transitive) ancestor of Y in the lambda structure. The *binding literals* $\lambda(X) = Y$ and $\mathsf{ante}(X) = Y$ are satisfied iff the lambda structure contains corresponding lambda or anaphoric binding edges. The most complex literals are the *parallelism literals*; we return to them in a minute. Figure 1 illustrates the literals of CLLS.

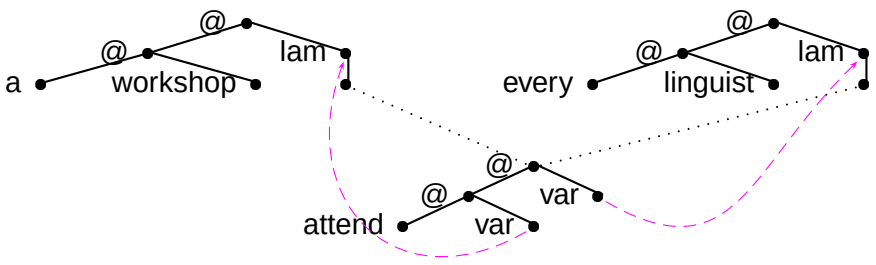
By a partial description of a lambda structure, we can model semantic underdetermination, in particular *scope ambiguity*. For instance, the sentence (5) has two closely related readings (either quantifier may outscope the other one) as represented in (6):

(5) Every linguist attends a workshop.

(6) (a)
```
(a workshop)(λx
    (every linguist)(λy
        (attend x) y))
```
(b)
```
(every linguist)(λy
    (a workshop)(λx
        (attend x) y))
```

Two λ-structures correspond to these λ-terms. We describe them both in a single CLLS constraint, which we draw as a *constraint graph*, shown in (7). The nodes in this graph stand for variables in a constraint and the edges and labels represent different types of literals. A node connected to its children by solid lines represents a labelling literal, dotted lines stand for dominance literals, and a dashed arrow stands for λ-binding.

(7)



The constraint graph in (7) is satisfied by all λ-structures into which the graph can be embedded in such a way that no labelled nodes of the graph overlap. The description leaves the ordering between the quantifier fragments unspecified. But since both fragments dominate the nuclear scope and, like trees, λ-structures cannot branch upwards, one of the quantifier fragments must dominate the other.

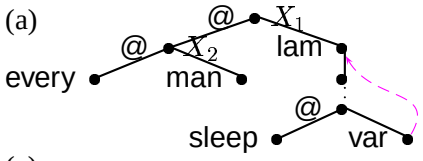
2.2 Representing ellipsis in CLLS

*Parallelism literal*s state that two pieces of a λ-structure have the same structure. A parallelism literal $X_0/X_1 \sim Y_0/Y_1$ says that 'X_0 upto X_1' looks the same as 'Y_0 upto Y_1'. Consider the last picture in Fig. 1. X_0 denotes a node that is above the node for X_1, and together they designate a tree with a hole: the tree starting at the node for X_0, from which another tree has been cut out, namely the tree starting at the node for X_1. We call such a tree with a hole a *segment*. In the same way, Y_0/Y_1 designates a segment, and the parallelism literal states that these two segments have the same structure.
Parallelism literals can be used to model ellipsis. Consider e.g. the elliptical sentence (8):

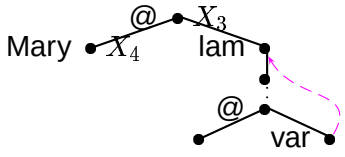
(8) Every man slept, and Mary did, too

The meaning of its TaS *so does Mary* is equal to the one of its SoS *every man slept*, except that the semantic contribution of the source parallel element *every man* is replaced by the one of the target parallel element *Mary*. The constraints for SoS and TaS are (9a) and (9b). The first part of the constraint for sentence (8) conjoins the SoS and TaS structures, as expressed in (9c). Two occurrences of the same variable name, like X_1 in (9a) and (9c), describe the same variable, so the two constraints (9a) and (9c) are to be joined at X_1.

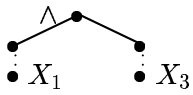
(9) (a)



(b)



(c)

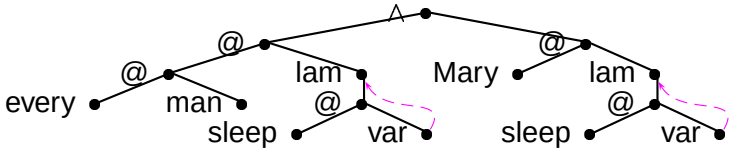


The second part of the constraint is the parallelism literal (10), which states that the tree which is the semantics of the SoS has the same structure as the tree that is the semantics of the TaS, except for the two subtrees which are the semantics of the SoS subject and the TaS subject.[1] The λ-structure (11), the intended semantic representation of (8), satisfies all constraints in (9a), (9b), (9c), and (10).

(10) $X_1/X_2 \sim X_3/X_4$

[1] Since proper names and quantifier NPs can be parallel elements in an ellipsis (as in (8)), proper names are type-raised; the constant Mary in (9b) corresponds to a λ-term of type $\langle\langle e,t\rangle,t\rangle$.

(11)



CLLS can also handle the interaction of ellipsis and scope, as in (12) and the interaction of ellipsis and anaphora that arise from 'strict' and 'sloppy' reconstructions of pronouns. We refer to Egg et al. (2001) for analyses of these issues in CLLS.

(12) Every linguist attends a workshop. Every computer scientist does, too.

3 The analysis

Our syntax-semantics interface assigns each syntactic constituent a CLLS constraint. If a constituent X has daughters $\mathtt{X_1}, \ldots, \mathtt{X_n}$, then the constraint for X has the form

$$\varphi_{\text{rule}} \wedge \bigwedge_{i=1}^{n} \text{constraint for } \mathtt{X_i},$$

where φ_{rule} is a constraint associated with the phrase structure rule $\mathtt{X_1}, \ldots, \mathtt{X_n} \rightarrow \mathtt{X}$. In prose: the constraint for a complex constituent is the conjunction of the constraints of its immediate constituents (ICs), plus the constraint that stems from the relevant phrase structure rule.

For the construction of complex constraints, each constituent makes accessible two distinguished CLLS node variables of its constraint by auxiliary features, a technique reminiscent of the semantic construction in MRS (Copestake, Flickinger, and Sag 1997). The features are called the SCOPE and the ROOT feature of the constraint. The value of SCOPE models the local scope domain; usually this is the uppermost variable in the constraint for the lowest sentence that comprises the constituent in the syntactic structure. This information is needed to handle scope islands like e.g. relative clauses. The variable that is the value of the ROOT feature is the uppermost variable in the constraint for the constituent. (Sometimes we will talk about SCOPE and ROOT of a constituent C in order to abbreviate the clumsier 'value of the SCOPE or ROOT feature of C'.) Finally, we encode the constraint associated with a constituent in a feature CON.

This information is collected in a semantic feature SEM, which looks like this:

(13)
$$\text{SEM}\begin{bmatrix} \text{SCOPE} & \text{the scope variable} \\ \text{ROOT} & \text{the root variable} \\ \text{CON} & \text{the constraint for this constituent} \end{bmatrix}$$

To construct the constraint for a complex constituent, the constraint φ_{rule} that comes with a phrase structure rule refers to the SCOPE and ROOT of the constituent's ICs. This constraint φ_{rule} 'glues together' the constraints of the ICs using these SCOPE and ROOT variables. In this way, we ensure that the constraints for the individual constituents form a coherent whole. Moreover, the constraint for the complex constituent introduces SCOPE and ROOT features of its own.

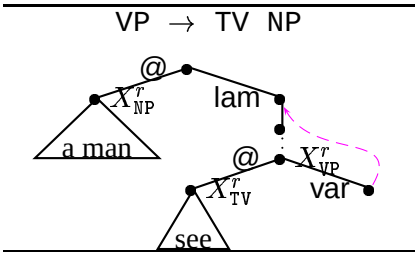


Figure 2: A phrase structure rule and the associated constraint

An example is shown in Fig. 2: it is the constraint associated with the phrase structure rule VP → TV NP, plus two triangles that sketch where the constraints for the two children are attached. The variable X^r_{NP} is the ROOT variable of the NP constituent, and analogously for X^r_{TV} and X^r_{VP}. Note the introduction of a ROOT node for the VP constraint: it is the node immediately below the dominance relation that is labelled by '@'.[2]

To handle VP ellipsis in this framework, we add two more features to SEM. The exception in a VP ellipsis is always the subject of the sentence. So we use a feature SEM|SUBJECT to store the variable that is the ROOT of the subject of the sentence. And we collect the list of SoS candidates in a feature SEM|SOS. Each candidate is represented by a pair of variables: the ROOT variable for the subject of the sentence, and the the SCOPE variable for that sentence.

3.1 A Case of Conjunction Ellipsis

We first show what our analysis looks like for sentence (8). In this case, source and target sentence are connected by a conjunction. The constituent structure for this sentence is given in Fig. 3 (where we have omitted the irrelevant details of the NP *every man*). It is annotated with the SEM feature values. The constraint that the syntax/semantics interface constructs for this sentence is shown in (9).

The relevant parts of the interface rules are shown in Fig. 4. They are simplified in

[2]The SCOPE nodes are neglected here; the rule would merely identify the SCOPE nodes of NP, VP, and TV.

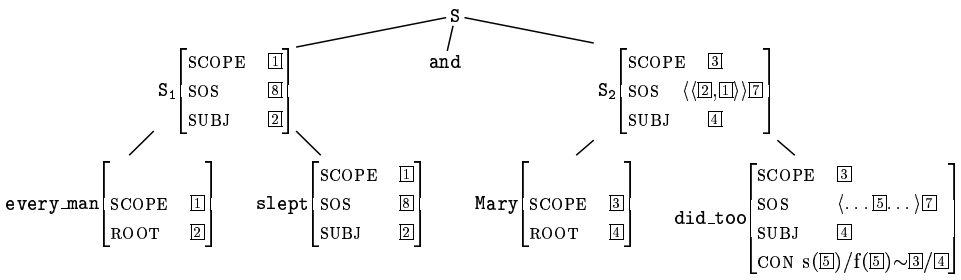


Figure 3: Constituent structure for sentence (8)

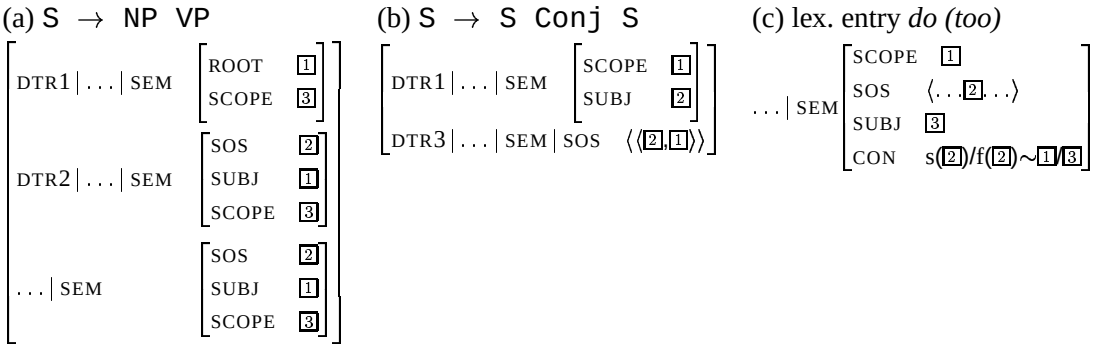


Figure 4: Interface rules used in Fig. 3

that the respective immediate constituents appear as DTR1 ... DTR*n*; dots in paths abbreviate SYNSEM|LOC. The rule in (a) collects information on SUBJECT and SCOPE of a sentence. This information is turned into ellipsis potential in (b). The conjunction is the joint between SoS and TaS, as it puts the SoS to the SOS list of the TaS. The lexical entry (c) then discharges the ellipsis potential.

Now we take a closer look at the interaction of the rules, starting at S_1. Here Fig. 4 (a) identifies the SUBJ value of the sentence with the NP's ROOT value, in this case the root 2 of every_man. Now at S, the interface rule (b) builds up ellipsis potential for the SoS candidate S_1. It identifies the SOS value of its DTR3 (which will later be determined to be S_2) with a singleton list.[3] Its member is a pair consisting of the SUBJ and SCOPE values of S_1. By rule (a) of Fig. 4, the SOS value 7 and the SCOPE value 3 of S_2 are handed down to did_too, and the SUBJ of did_too is identified with the ROOT value 4 of Mary.

[3]This is a simplification in that for other conjunctions (e.g., *as*) there may be more than one SoS candidate. We envisage that a future analysis will admit for this possibility, and that the lexical entries of conjunctions will carry information to determine the range of SoS candidates and, eventually, also preferences between these candidates.

Now in `did_too`, the built-up ellipsis potential is discharged according to the lexical entry (c): its constraint in CON is a parallelism literal. It says that the semantics of the SoS up to the root of the SoS subject has the same form as the semantics of the TaS up to the TaS subject. ('f' and 's' are functions that map a pair to its first and second member, respectively.) The SoS values are a tuple from the SOS list, which in this case possesses only one entry anyway. The TaS local scope and subject variables are the SCOPE and SUBJ values of `did_too`. For our example (8) this means that the meaning of $\mathtt{s}_2$ with the exception of the semantic contribution of *every man* is the same as the one of $\mathtt{s}_1$ except for the semantics of *Mary*.

3.2 Two Cases of Antecedent-Contained Deletion

In this paragraph we show what our analysis looks like for antecedent-contained deletion cases. We examine two sentences, first (14), which has just one reading, and then (2), repeated here as (15), which is ambiguous.

(14) John greeted everyone that Bill did.

(15) John wants Max to read everything that Bill does.

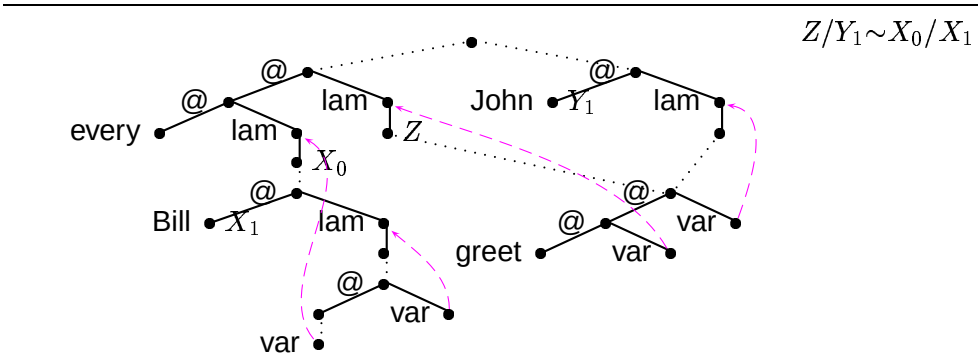


Figure 5: Constraint for sentence (14)

The constraint for sentence (14) is shown in Fig. 5. Its constituent structure is given in Fig. 6 (with the syntactic structure of the elliptical NP simplified), annotated with the SEM feature values.

In this case, the joint that builds up the ellipsis potential is a VP that consists of a transitive verb and an NP. We need an additional interface rule that achieves this, plus an interface rule that percolates SOS information down into a relative clause.

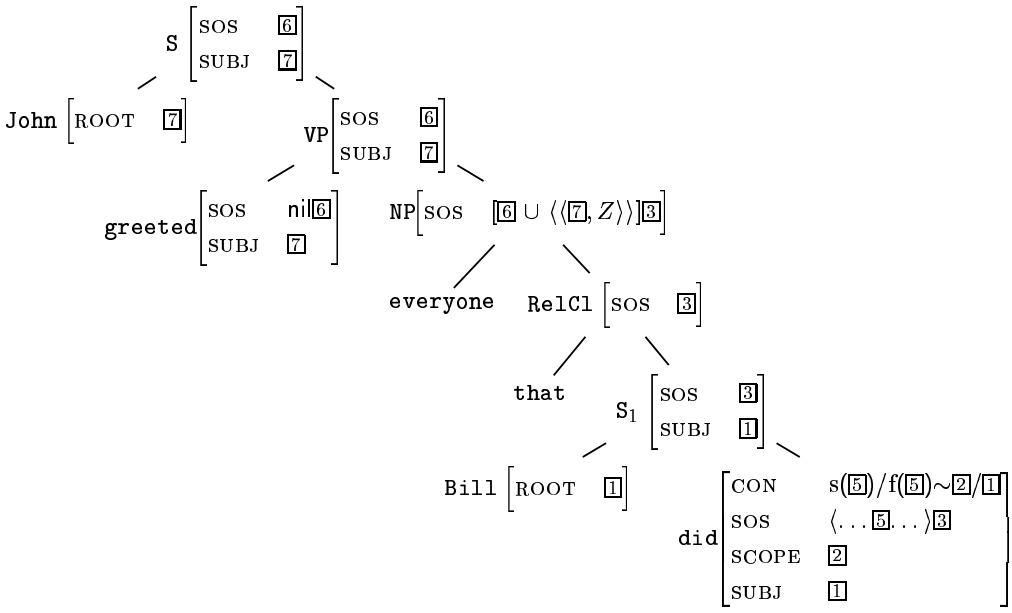


Figure 6: Constituent structure for sentence (14)

These two additional interface rules are given in Fig. 7. The rule in (a) adds an SoS candidate to the SOS list. While the variable for the sentence subject is located as before, the uppermost variable of the sentence (the relative clause, in this case) is part of the constraint for the VP, which is shown in Fig. 7 (c). It reappears in Fig. 5.

Rule (a) of Fig. 7 is applied at VP in Fig. 6. It states that the SOS and SUBJ values of VP are the same as those of its head daughter greeted. Since we assume that the derivation of the SOS value of a sentence is completed once we have reached the finite verb, we assign the SOS feature $\boxed{6}$ of the finite verb greeted the empty set as a value. In this way, we express the fact that there is no further augmentation of the SOS value of the sentence. Furthermore, the rule adds a new element to the NP's set of potential source sentences. The first element of this pair is the SUBJ of VP, which is the ROOT of John. The second element of the pair, the uppermost variable Z of the relative clause, can best be seen in Fig. 5.

The rule (b) of Fig. 7 is applied at NP and RelCl. It hands down the SOS value from the constituent to its semantic head. Finally, the pro-form again discharges the built-up ellipsis potential by Fig. 4 (a). It picks the only element $\boxed{5}$ of the SOS list $\boxed{3}$ to determine the SoS and uses it to construct the parallelism literal $Z/Y_1 \sim X_0/X_1$.

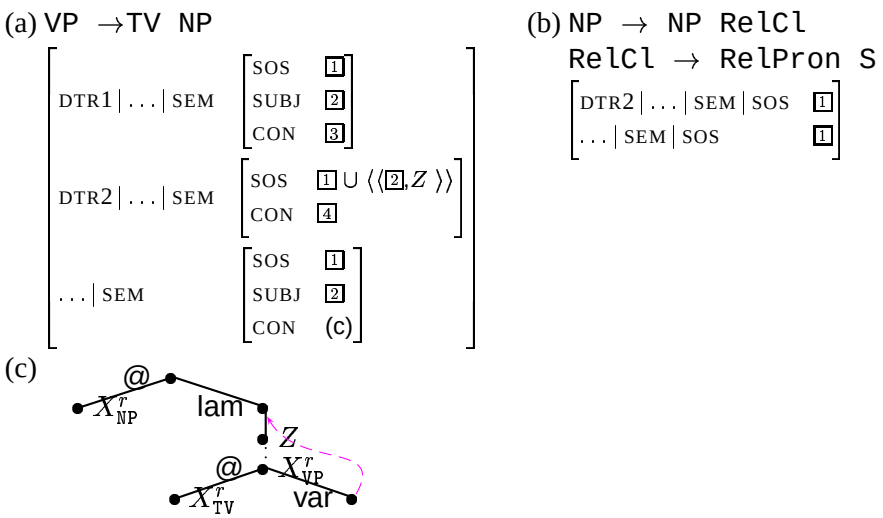


Figure 7: Additional interface rules for handling ACD

Now we turn to the more complicated ACD sentence (15). It has three readings. Either the target sentence means ... *that Bill reads*, which leads to a de dicto/de re ambiguity: either there is a concrete list of books that John wants Max to work through; or John thinks that Max should read anything that Bill reads, whatever it may be. The constraint in Fig. 8 anticipates this ambiguity by leaving the scope between *wants* and *everything* open. Or the target sentence means ... *that Bill wants Max to read*. Here there is no de dicto/de re ambiguity, a point noted in Sag (1976).

The analysis of this sentence is similar to that of the more simple ACD example (14). The main difference to the simpler example is that we now make use of the list value of the SCOPE feature to encode the fact that there are several available TaS.

Two factors are crucial for the derivation of an SOS value with more than one element: first, the interface rule for control verbs adds a tuple to the argument VP's SOS list – see Fig. 9 (a). In this case, this tuple consists of the ROOT of `Max` and the variable Z_1, a local scope dominated by the semantics of `wants`. Second, `to read` does not set its SOS value to `nil`, in contrast to `greeted` above. This encodes the observation that the SOS list of a VP whose head is infinite can be further augmented if the VP is the argument of another verb, e.g. a control or raising verb.

Fig. 10 demonstrates the derivation of the SOS list for (15). The interface rule Fig. 9 (a) applies at the top `VP` node and adds an element to the SOS list of *wanted*

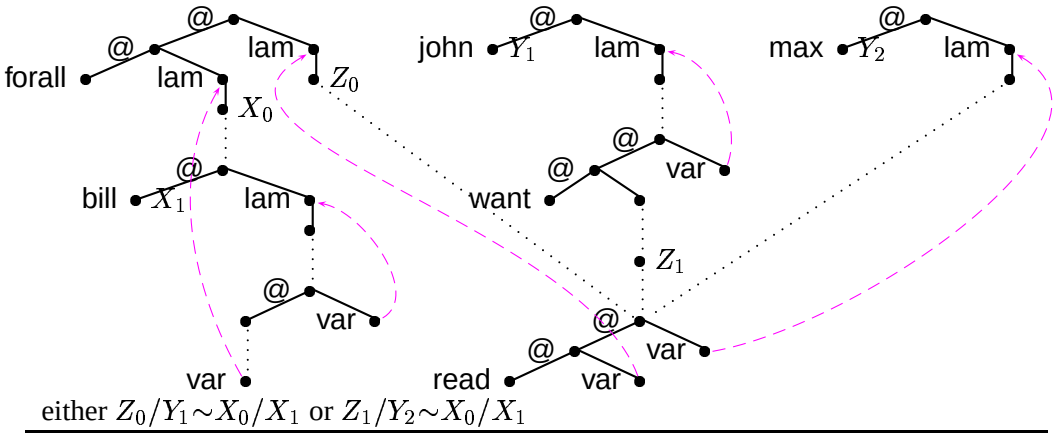


Figure 8: Constraint for sentence (15)

(i.e., to the empty set since this is the transitive verb). The list with one element is then the value of the lower VP's SOS feature. The SOS list of the NP argument of this VP is then augmented by the interface rule 7(a) in the same fashion as for sentence (14). So in this case, the SOS list from which the SoS of `Bill does` is determined has two elements, corresponding to the two potential SoSs in sentence (15).

Correspondingly, we get to two different feasible parallelism constraints. The TaS segment of the constraint is always the same, it is X_0/X_1. If the SoS segment is Z_0/Y_1, then *John*, and with it *wants*, are below Z_0, and *John* is the source parallel element. Hence, the TaS meaning is *. . . that Bill wants Max to read*, and must be de re, since *every* outscopes *wants*. This rules out a de dicto reading for this constraint alternative, as desired.

If the SoS segment is Z_1/Y_2, then Z_1 must be above *Max*, and the TaS meaning is *. . . that Bill reads*. Now *wants* can be either above or below Z_0: if it is above Z_0, *wants* outscopes *every*, so this is the de dicto reading. If, however, *wants* is below Z_0, then we obtain the de re reading. The resulting representations for the three readings are sketched in Fig. 11.

4 Related work

The issue of how to determine the number of available readings of an elliptical sentence is widely discussed in the literature. However, attention focuses on the problem of how to *constrain* the number of available readings for an already iden-

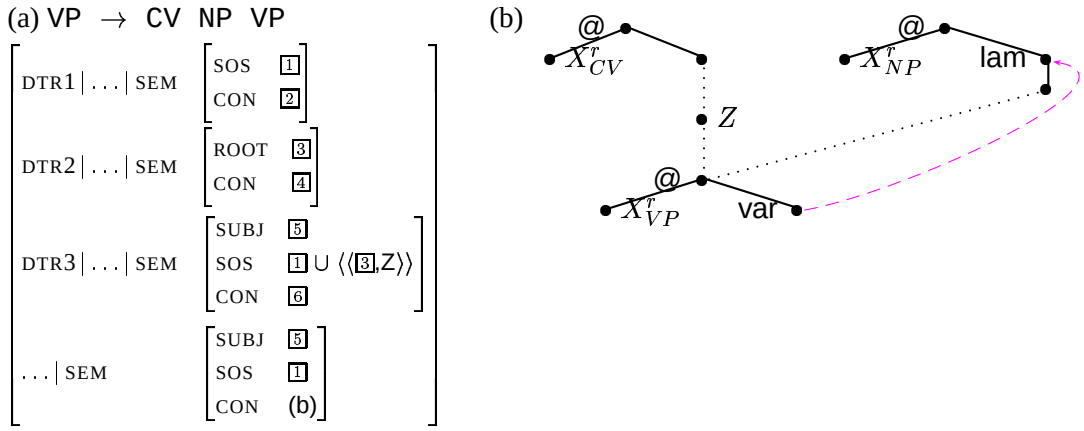


Figure 9: Interface rule for control verbs

tified set of antecedent candidates for a given ellipsis (see e.g. Sag 1976 and Fiengo and May 1994). In contrast, there are few previous approaches to an automatic *detection* of antecedent candidates for an elliptical sentence (Hardt 1997; Gregory and Lappin 1998; Lappin 1999; Ginzburg and Cooper 2001).

The approach that comes closest to what we are doing is the one by Hardt (1997). He presents a corpus-based determination of antecedents of VP ellipses (source sentence VPs in our terminology). His basic strategy is to use *heuristics* to choose an antecedent VP from a set of potential antecedent VP candidates. This set comprises almost all VPs within the last three sentences.[4] He then ranks the elements of this set by several heuristics. For instance, there is a preference for more recent antecedents. Another important group of heuristics compares the *parallel* material in ellipsis and potential antecedents, i.e., the overt material in the ellipsis and the matching material in the potential antecedent. Preference is given to potential antecedents with similar parallel material.

An example for such a check is the comparison of auxiliaries (Hardt's '*aux-match*' and '*parallel-match*'), i.e., auxiliaries of ellipsis and potential antecent should have the same base form. This may overrule the recency factor, e.g., in (16). Here *be ideal* is chosen as antecedent over the more recent *advises*, because it involves the same auxiliary (*would*) as the ellipsis:

(16) Someone with a master's degree would be ideal, litigation sciences advises.

[4]The only VPs that are ruled out are those that contain the VP ellipsis in a sentential complement as e.g. in (i). Here the VP *said that she would not* is not available as an antecedent to the VP ellipsis in the embedded sentence:

(i) She said she would not

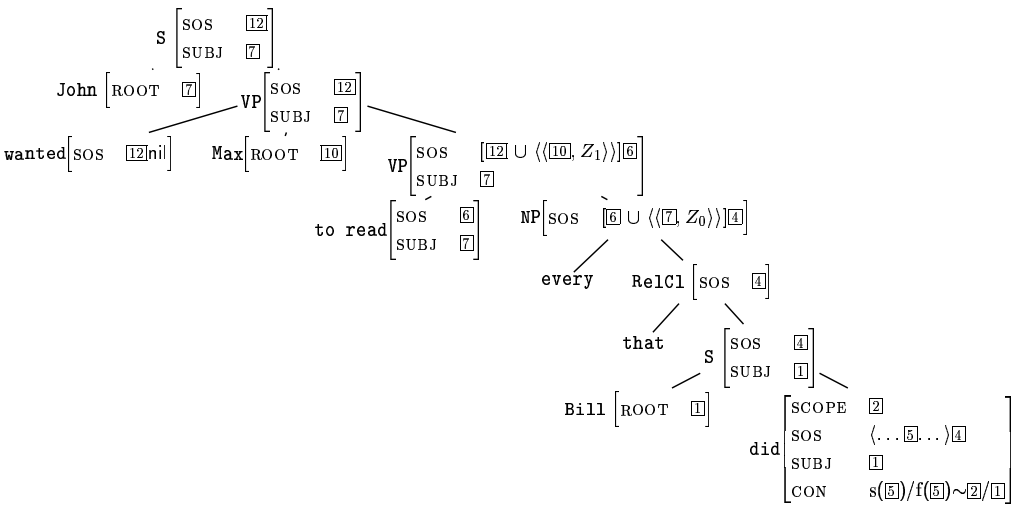


Figure 10: Constituent structure for sentence (15)

So would be someone recently divorced or widowed.

In cases with more than one possible antecedent, like the example (15) that we have studied in the previous section, it is necessary to use heuristics to determine the source sentence. However, we feel that Hardt's approach uses heuristics even for cases when syntactic structure seems to make a SoS candidate not only dispreferred but actually impossible. (17) may illustrate this point:

(17) Amélie came to the studio. She entered, and Ariadne did, too

Our analysis would require the first element of the conjunction to be the SoS. I.e., what Ariadne is reported to have done is entering, and not coming to the studio. Hardt would obtain the very same result by his recency heuristics. However, we believe that our analysis captures a general fact about VP ellipsis in the case of sentence conjunction with *and*, viz., that a TaS in the second conjunct must always pick the first conjunct as its SoS. This is also responsible for the interpretation of sentences like (18), where the recency heuristics would have to be overridden by another heuristics in a purely heuristic analysis in order to avoid the incorrect selection of *hated rap songs* as the antecedent of the ellipsis. The interpretation of (18) is that Ariadne expressed her dislike of rap songs:

(18) Amélie said that she hated rap songs, and Ariadne did, too

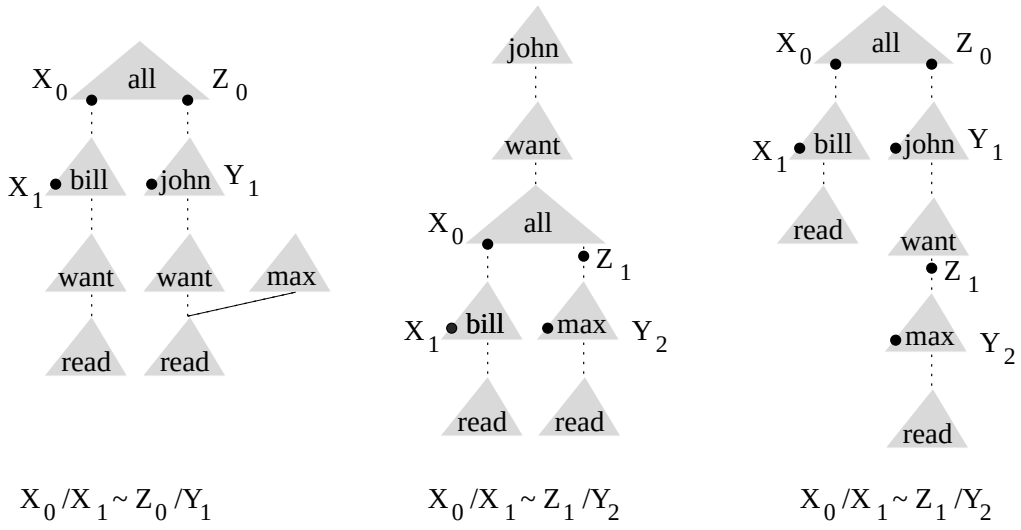


Figure 11: Sketch of the the three readings of sentence (15)

But our approach and a heuristics-based one like Hardt's are by no means mutually exclusive. Rather, we believe that they should be integrated, since they are well suited to complement one another in the task of ellipsis resolution. Syntactic information can be used as hard constraints on the set of potential antecedents (or source sentences). This syntactic information will (and should) fail to choose one single potential antecedent in cases like (15), which have several potential antecedents. Here preferences like the ones discussed by Hardt can select the most preferred antecedent candidate. See section 5 for a sketch of what such an integration might look like.

Gregory and Lappin (1998) and Lappin (1999) focus on ACD cases. Here naive approaches to VP ellipsis resolution (which simply copy an available VP and substitute it for the pro-form) run into infinite recursion, since the pro-form is part of the VP. For these sentences, they automatically detect the antecedent, then reconstruct the ellided part within the syntactic surface structure. (The reconstruction mechanism is the one by Lappin and Shih 1996.) Their approach crucially uses the HPSG (Pollard and Sag 1994) treatment of unbounded dependencies and relative clauses. Roughly, they define a means of identifying the VP of the SoS starting from an elliptical pro-form like *does.* This identification propagates along structure-shared information in the syntax tree. Once the VP is identified, it is copied into the syntax tree for the TaS. Recursion is avoided by substituting a trace for the NP of which the pro-form is a part. E.g., in (14) they set out from *does* and identify the the VP as *read everything that Bill does.* They then copy *read* (with a trace in object

position) into the target sentence.
However, we envisage problems with this strategy for ACD cases with several potential SoSs like (15): here *wants Max to read everything that Bill does* is also available as a potential SoS-VP. Even if the analysis could be generalized to derive this additional possibility, it could not capture the resulting ambiguity of (15) directly in a semantic representation, since it operates destructively on syntactic tree structures. But such a representation would be important for an integration of preferences: it would serve as the input to a module of language processing that resolves ambiguities like in (15) on the basis of all kinds of preferences. We expect preferences to arise not only within syntax; other kinds are bound to emerge in the domains of semantics or world and context knowledge.
Ginzburg and Cooper (2001) deal with cases of clarification ellipsis, i.e. dialogues like *Did Bo leave? – Bo?* This problem is quite different from the ellipsis cases we are considering in that the elided material ellipsis has to be reconstructed not as a copy (on some level) of SoS material but as one of several fixed possible readings of a clarification ellipsis. The analysis is relying critically on information structure for determining the most likely utterance to which the clarification question is referring.

5 Further research

We envisage extending our approach to one that

- uses preferences as well as certain knowledge to determine the most suitable SoS candidate,
- integrates different sources of information for determining candidates, and
- can handle different kinds of ellipsis.

The approach that we have presented is constructed in such a way that it is extensible. Preferences are especially easy to integrate: the SOS feature is list-valued, i.e., it can contain more than one candidate (we have seen that feature in use with the third example, sentence (15)). The elements of this list can be annotated with preferences to mark the most suitable candidate.
For example, consider sentences (19) and (20). In both cases, both the *when*- and the main clause are potential SoSs, and in both cases the 'IBM'-sentence is the preferred candidate, i.e., the elliptical sentences mean *now they don't have to go to IBM*. This shows that no hard syntactic constraint like e.g. 'Consider only the main clause as SoS candidate' can be used to get the right readings in both cases.

(19) In the past, customers had to go to IBM when they outgrew the Vax. Now they don't have to.Hardt's (1997)

(20) Customers had usually outgrown the Vax, when they had to go to IBM. Now they don't have to.

To handle these sentences, we would first have to extend our analysis such that conjunctions can build up ellipsis potential with more than one potential SoS. To decide between the two potential SoSs, one could apply Hardt's *aux-match* preference as illustrated in Sec. 4 on sentence (16). In either case, this heuristics would select the 'IBM'-sentence.

An interesting question related to conjunctions that collect more than one antecedent candidate is: do different conjunctions build up different ellipsis potential?

Second, to integrate semantic information in the antecedent candidate selection, we could make use of the fact that CLLS constraints are constructed incrementally. In a situation where the ellipsis antecedent has not yet been determined, the constraint for the elliptical sentence *except for the parallelism literal* can already be constructed – and evaluated to extract information on possible antecedents.[5]

(21) John likes golf, and Mary too.

Finally, there are many different kinds of elliptical constructions. The next phenomenon that we intend to cover with our approach is *bare ellipsis* as in sentence (21) (the verb as well as a complement is missing in the TaS). Here the source parallel element need not be the subject of the SoS, which for (21) yields two interpretations of the TaS, *John likes Mary* and *Mary likes golf*.

References

Copestake, A., D. Flickinger, and I. Sag (1997). Minimal Recursion Semantics. An introduction. Available from `ftp://ftp-csli.stanford.edu/linguistics/sag/mrs.ps.gz`.

Dalrymple, M., S. Shieber, and F. Pereira (1991). Ellipsis and higher-order unification. *Linguistics & Philosophy 14*, 399–452.

Egg, M., A. Koller, and J. Niehren (2001). The constraint language over lambda-structures. *Journal of Logic, Language, and Information 10*, 1–29.

[5] This possibility was pointed out to us by Hans Kamp.

Egg, M., J. Niehren, P. Ruhrberg, and F. Xu (1998). Constraints over lambda-structures in semantic underspecification. In *Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'98)*, Montreal, Canada, pp. 353–359.

Fiengo, R. and R. May (1994). *Indices and Identity*. Cambridge: MIT Press.

Ginzburg, J. and R. Cooper (2001). Resolving ellipsis in clarification. In *Proceedings of the 39^{th} Annual Meeting of the Association for Computational Linguistics (ACL)*, Toulouse, France, pp. 236–243.

Gregory, H. and S. Lappin (1998). Antecentent contained ellipsis in HPSG. In G. Webelhuth, J. Koening, and A. Kathol (Eds.), *Lexical and constructional aspects of linguistic explanation.* CSLI.

Hardt, D. (1997). An empirical approach to VP ellipsis. *Computational Linguistics 23*, 525–541.

Lappin, S. (1999). An HPSG account of antecentent contained ellipsis. In S. Lappin and H. Gregory (Eds.), *Fragments: studies in ellipsis and gapping*. Oxford University Press.

Lappin, S. and H.-H. Shih (1996). A generalized reconstruction algorithm for ellipsis resolution. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING'96)*, Copenhagen, Denmark.

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

Sag, I. (1976). *Deletion and logical form*. Ph. D. thesis, MIT, Cambridge.