Underspecification for incremental generation

Markus Guhe and **Frank Schilder**
Department of Informatics
Hamburg University, Germany
{schilder|guhe}@informatik.uni-hamburg.de

Abstract

State-of-the-art NLG systems have a mostly static perspective on language generation, i.e. they have a fixed knowledge representation from which they generate language. This is fine for offline working systems, where the language is read only after it has been completely generated. However, systems set in a dynamic environment, e.g. those interacting with a human user, need a dynamic way of processing and, therefore, dynamic representations to exhibit appropriate response behaviour. For such settings we suggest to use an incremental mode of processing in conjunction with underspecified semantic representations. We especially focus on the case where the content to be verbalised changes during language generation.

1 Introduction

NLG systems that interact with human users, especially speech-based systems, have to show a huge amount of flexibility as well as support incremental processing (Brennan, 2000). Incremental processing means (1) that the system uses a piecemeal way of generating language and (2) that processing takes place simultaneously on multiple, subsequent levels. In order to accomplish this, *dynamic* representations are needed, i.e. representations that are flexible enough to make possible a integration of changes into an existing representation instead of producing a new representation after each change. This is a strong requirement on the representational formalism, which the formalisms commonly used in NLG do not meet. In this paper we show that underspecified representations are ideal for this purpose.

As an example we use a setting in which the *incremental conceptualiser* (INC) incrementally generates semantic representations for online descriptions of events, in this case taxiing planes on the manoeuvring area of an airport (Guhe et al., 2000; Guhe and Habel, 2001). This setting simplifies matters in that no model of the interaction partner is needed but allows us to focus on the advantages of using an underspecification formalism. In particular, we argue that the extendability of underspecified representations and the dynamics of incremental processing allow for an adaptation of the initially planned utterance to newly obtained input.

NLG systems are usually divided into a *what-to-say* and a *how-to-say* part (De Smedt et al., 1996). In the first component the content to be verbalised is decided upon while in the second component the linguistic encoding takes place. In the terminology of Levelt (1989), which we use here, the first component is called *conceptualiser*, the second *formulator*. We are concerned with the interface between the two, which is constituted by *preverbal messages*, and in particular, how these are generated by INC. Since they are generated incrementally, we also call them *incremental preverbal messages* (Guhe, under review). The conceptualiser has access to conceptual knowledge, the discourse model and possibly a hearer–speaker model. In our generation model, this information is represented by referential nets (Habel, 1982; Habel, 1986). For the production of an utterance, a conceptual structure is chosen by the conceptualiser and translated into an underspecified lambda-structure. We use the underspecification formalism CLLS (Constraint Language for Lambda Structures), cf. Egg et al. (2001). A subsequent formulator can then transform this representation into a linguistic structure for speech or written output. An example regarding the generation of the preverbal message for a VP ellipsis demonstrates the usefulness of combining incrementality and underspecification in more detail. Note that the emphasis of our work here lies on how VP ellipses can be *generated* in order to demonstrate the usefulness of this combination. Our aim is not to give a full account of the phenomenon of VP ellipsis. Along the lines layed out in this paper also

the incremental generation of self-corrections can be handled, cf. Guhe and Schilder (2002).

2 Incrementality and Underspecification

NLG systems can be divided into offline working systems and systems in a dynamically changing environment, e.g. systems that interact with a human user or systems that generate online descriptions of events. Offline working NLG systems can use top-down or even multi-pass techniques to plan a whole discourse or text, which is read by humans after it has been generated completely, for example the well-known case of text generation systems for weather forecasts (Goldberg et al., 1994). However, such systems need a considerable time for the generation of output, which systems working in a dynamic environment usually do not have: they must have a short response time, in order to not produce gaps while planning the next part, and must be flexible enough to respond to changes without starting the planning process all over again. Otherwise their overall quality and usability is severely limited. Such systems must therefore work *incrementally*, which leads to a more fluent and adaptable output, cf. Kempen and Hoenkamp (1987), De Smedt and Kempen (1987), Levelt (1989), Reithinger (1991), Reiter (1994), Finkler (1997). When natural language is generated incrementally, smaller pieces (*increments*) can be processed further by subsequent components before all information that may be relevant for the complete and correct computation of the final result are available. In this way generation can take place simultaneously on all levels.

Like incremental NLG systems, semantic underspecification formalisms have been proving their usefulness over the last years. There are two main reasons for this. The first reason is that underspecification makes it possible to represent ambiguous utterances by only one structure for *all* readings instead of one structure for *each* reading, for instance in the case of scope ambiguities. Additionally, it is an elegant representational method for the semantic description of anaphora, reinterpretation in lexical semantics, and the meaning of elliptical expressions (Egg et al., 2001; Schilder and Guhe, 2002). Although the idea of semantic underspecification is not particular to one direction of language processing – comprehension or generation – it has only been used in comprehension up to now. (There are a few exceptions, though, as we will see soon.) This probably is so, because the mentioned problems are typical problems in the parsing of sentences and discourses.

The second reason is that underspecification can be used for representations that change over time, which we call *dynamic representations* here. An underspecified representation is usually not totally underspecified, but in some parts it is specified. It is therefore possible to distinguish those parts of the representation that cannot be modified (without creating an error or reanalysis) from those parts where elements can be added or inserted. It is this extendability property of underspecification that we exploit for incremental generation.

While we are not the first to think about underspecified structures in NLG, cf. for example Pianta and Tovena (1999), we are the first who suggest to exploit the extendability property of underspecification formalisms. – Extendability of representations is a prerequisite for incrementality after all. – The VERBMOBIL system, a machine translation system (Wahlster, 2000) that uses a semantic transfer approach, comes closest to this idea. It uses underspecification in the semantic representations and incremental parsing and generation components. However, it uses a dialogue-act based translation, which means that each dialogue act is completely parsed into an underspecified semantic representation and then completely translated before generation commences. In this way the 'usual' advantages of underspecification mentioned above – plus, to some extent, the preservation of lexical ambiguities, cf. also Knight and Langkilde (2000) – can be used, but not the fact that an utterance can be generated incrementally while planning is not finished.

As we argued above, incrementality is a means to cope with dynamically changing states of affairs, e.g. when the NLG system is working in a dynamic setting. The dynamics of the environment and incremental processing necessitate changes in the representation of the states of affairs. The main advantage of underspecification is that these representations need not be restructured or constructed from scratch each time. Quite the contrary: new information can simply be added to the already existing underspecified representation at the allowed places. One can imagine these places as 'holes' in the representation where new information can be filled in. However, this is not obligatory; the representation is already complete as it is. This method facilitates NLG systems that need not plan a whole utterance in advance to generate a valid structure but that can

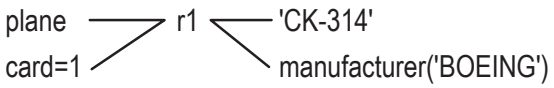
plan an utterance piecemeal while it is already leaving the system.

3 Representational Formalisms

We now give a short outline of the representational formalism we use here and describe how they can be combined in order to produce incremental preverbal messages.

3.1 Referential Nets

Referential nets (refNets) consist of interrelated *referential objects* (refOs), cf. Habel (1982) and Habel (1986). A refO represents the knowledge about an entity in the world. An example for a refO representing a taxiing planes is:



In this notation r1 is a pointer that serves to refer to the refO. The values to the left of r1 are *attributes*, the ones to the right *designations*. Attributes represent conceptual knowledge, e.g. its sort (plane) or cardinality (card=1). Designations represent knowledge about meaning or (here) knowledge that is used to generate semantic representations (preverbal messages) out of the conceptual representation. Designations can be *names*, like the flight number 'CK-314'[1], or functional expressions like manufacturer('AIRBUS'), where 'AIRBUS' is a name, or manufacturer(r20) when r20 represents the plane building company.[2] Since all knowledge about an entity is stored in one spot, refNets contain a considerable amount of redundancy, which makes them costly in terms of storage capacity but highly efficient with respect to access time.

One increment of a preverbal message is modelled by one refO, cf. section 4. However, not all knowledge about an entity is needed for a verbalisation. Think, for example, of a person you know very well. When you speak of her, very often you only need her name but not her favourite colour or artist. Consequently, INC selects an increment of an incremental preverbal message by deciding that a refO will be verbalised. Then, in a separate step it determines what designation from this refO is an adequate verbalisation given the current context (Guhe, under review).

3.2 Constraint Language for Lambda Structures

The Constraint Language for Lambda Structures (CLLS) is a formal framework for the description of underspecified lambda structures which are tree-like structures representing λ-terms. An important feature of CLLS is the introduction of variables denoting tree nodes (e.g. X_0, X_1 in figure 1) and a transitive and reflexive dominance relation ($\lhd^*$) holding between tree nodes. These variables and the dominance relation allow underspecification, the specification of anaphoric references, etc. In addition, the lambda tree structures may contain two partial functions: lam for abstraction and binding variables and ante for modelling anaphoric expressions. Graphically, additional labels and lines indicate application (@), variables (var) as well as the binding relation between the variable and the binder (the dashed arrow).

Formally, a CLLS formula is a partial description of lambda structures. A constraint graph is a graphical representation for a CLLS formula, as in figure 1. In order to satisfy this formula a lambda structure and a variable assignment have to be found such that every literal is satisfied. A CLLS constraint φ is defined as follows: $\varphi ::= X : f(X_1, \ldots, X_n) \mid X \lhd^* Y \mid \lambda(X) = Y \mid \mathrm{ante}(X) = Y \mid X_1/X_2 \sim Y_1/Y_2 \mid \varphi \wedge \varphi'$

We refer the reader to Egg et al. (2001) for a more detailed explanation of the CLLS constraints. However, since we use the *parallelism constraint* in the following, we will describe it in more detail now. The parallelism constraint describes parallel segments: $A \sim B$ is satisfied iff the segment A is parallel to a segment B. Segments are defined as X/Y where X denotes the root of the segment and Y a 'hole' such that $X \lhd^* Y$. That means, the segment covers all nodes that are dominated by the root X with the exception of Y and all nodes dominated by Y. In other words, a segment is a subtree starting with node X, with the exception of a further subtree whose root node is Y. In figure 1 the segment X_1/X_2 has the root node X_1, and includes all nodes dominated by X_1 except node X_2. Sometimes segments are indicated by brackets in the constraint graphs, but usually the constraint is

[1] Since refNets allow changing the representation, names can be attached to the refO temporally. The name 'CK-314' attached to this refO will be replaced by a different name for the return flight.

[2] Designations can also be descriptions of the form: $op\ var\ pred$ with operators $op \in \{\iota, \eta, \mathsf{all\_t}, \mathsf{some\_t}\}$, variables $var \in \{\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots\}$, and $pred$ being a predicate–argument structure. Operators reflect the cardinality and definiteness of the refO (Habel, 1986).

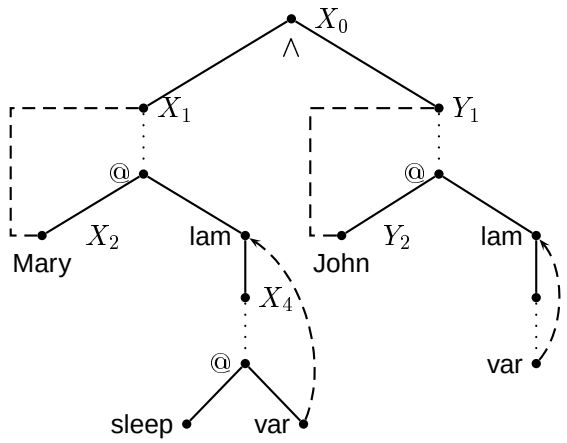


Figure 1: The parallelism constraint for the elliptical sentence in (1)

explicitly added to the constraint graph. The parallelism constraint has proven to be especially useful for the description of VP ellipses, as in the following example sentence:

(1) Mary sleeps and John does, too.

Formally, the parallelism between two segments is captured via a correspondence function, which is defined as a bijective mapping between the segments.[3] The CLLS description for (1) can be found in figure 1. The parallelism constraint $X_1/X_2 \sim Y_1/Y_2$ is indicated by the two dashed brackets denoting the two parallel segments X_1/X_2 and Y_1/Y_2. The brackets determine the part of the source segment that has to be copied into the target segment and the part that has to be left out (Mary). A lambda structure that satisfies this constraint graph is given in figure 2.

3.3 Conceptualiser–Formulator Interface

The preverbal messages generated by INC are refOs, e.g. r1, together with one designation, e.g. 'CK-314'. These refOs are translated into CLLS structures. The translation is carried out via the function $\mathcal{CLLS}$, cf. table 1. $\mathcal{CLLS}$ takes a refO and translates the designator description into CLLS formulae. However, the incrementally working conceptualiser only provides one refO at a time. Therefore, the translation is carried out as far as possible until the translation function comes to a recursive call of $\mathcal{CLLS}(\mathrm{r})$ where r is a refO. This is the point where

[3] See Erk et al. (2001) for a semi-decision procedure and further details on processing complexity.

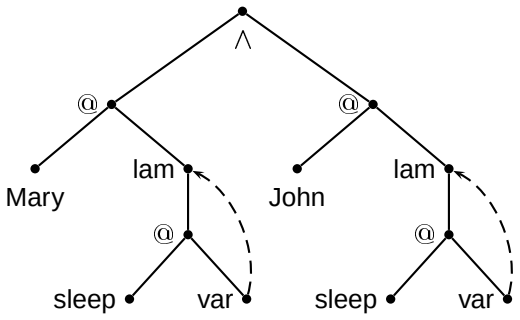


Figure 2: A lambda structure that satisfies the constraint graph in figure 1

underspecification pays off: a new tree node labelled X_n is introduced that is dominated by the current node X_m ($X_n \lhd^* X_m$). As soon as this refO becomes available the translation process continues. Note that an incremental formulator can take each generated CLLS formula as input before the preverbal message is complete.

4 Generation of VP ellipses: An Example

We now put the described mechanism to work in an example. We show how a preverbal message is produced to incrementally generate a VP ellipsis (VPE).

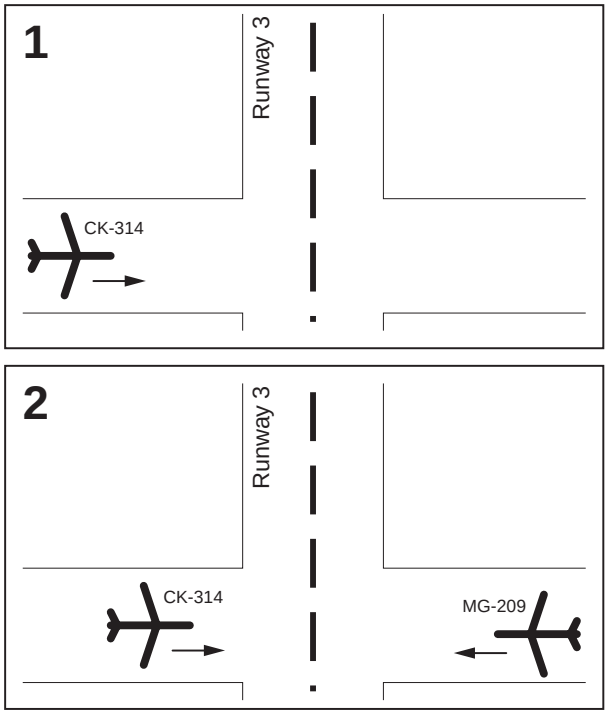


Figure 3: Example scenario that leads to the generation of a VP ellipsis

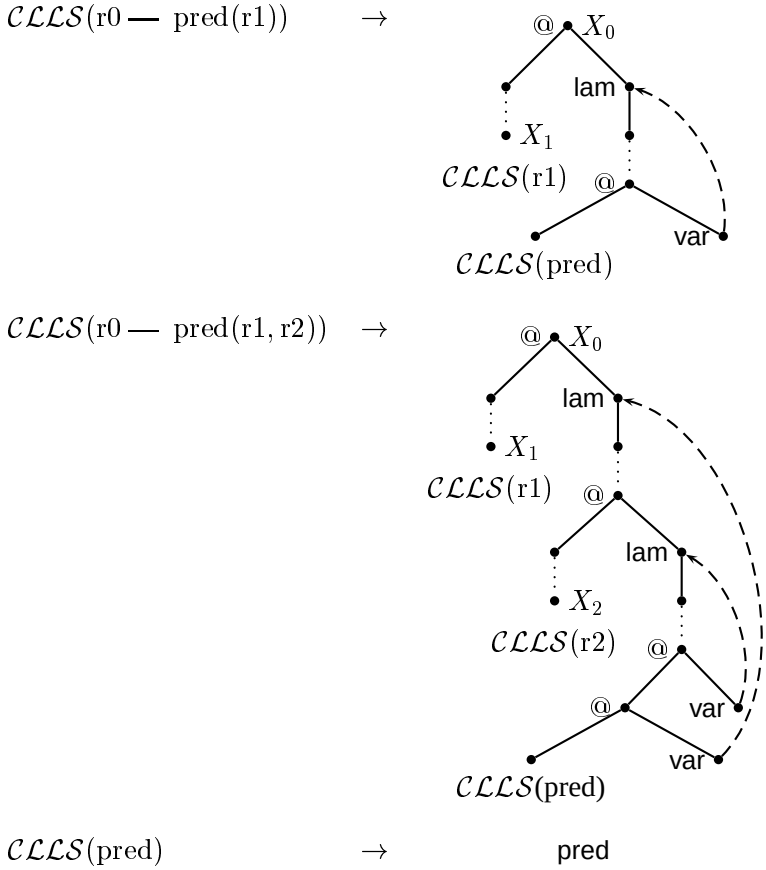


Table 1: The function $\mathcal{CLLS}$ translates refOs into underspecified lambda structures

One reason for choosing VPEs is that despite the extensive body of literature on the analysis of VPE, the automated generation of VPE has only rarely been addressed (Hardt and Rambow, 2001). Here, we focus especially on the case in which a *conceptual change* takes place. In a conceptual change the state of affairs that is verbalised changes during verbalisation, cf. also Guhe and Schilder (2002). Take for example an NLG system that incrementally generates descriptions for airport scenarios in an online fashion. In our example the system describes movements on the manoeuvring area. Figure 3 shows two snapshots of such a scenario. In phase 1 a plane with the flight number CK-314 is heading for Runway 3. An adequate verbalisation is:

(2) "Flight CK-314 is heading for Runway 3."

If a second plane (flight number MG-209) arrives on the scene (phase 2) while (2) is uttered, the system has the possibility to extend (2) via a VP ellipsis:

(3) "Flight CK-314 is heading for Runway 3 and so is Flight MG-209."

The generation of such a VP ellipsis is only possible if the preverbal message has not been 'closed off'. Thus, the semantic representation for (2) must allow the extension to be added 'on the fly', i.e. without recomputations. The preverbal message is built up incrementally as follows. The state of affairs in phase 1 is represented by three refOs, r1, r2, and r3, cf. figure 4.[4] In order to produce the preverbal message, a single designation is selected for each refO and passed on to the formulator. Each of these 'reduced' refOs is an increment of the incremental preverbal message. The generation of the preverbal message that (after formulation) leads to an utterance like (2) starts with the refO representing the event (r3). The generation of the incremental

[4]The conceptual representation usually contains many more details, e.g. the length of the runway or the number of seats, which we left out here.

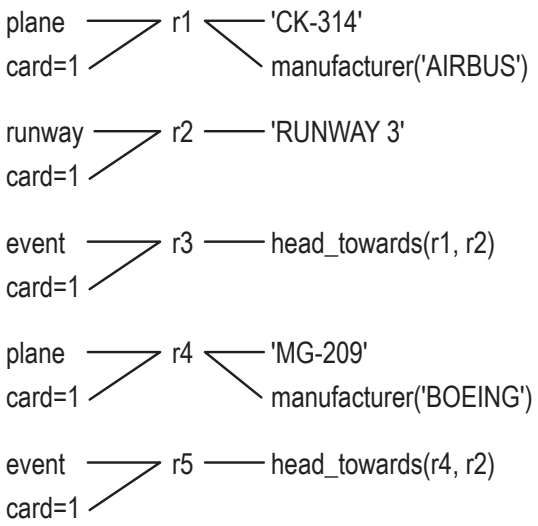


Figure 4: RefNet for the example

preverbal message starts with the event refO, because INC is a system that is specialised on conceptualising events. However, the mechanism we describe here does not require to start with an event refO.

r3 and head_towards(r1, r2) is translated into an underspecified semantic structure via the function $\mathcal{CLLS}$, cf. table 2. The translation process is suspended, because the other refOs mentioned by r3 are not yet available. Note that the formulator can nevertheless start to process the first increment, e.g. by accessing the lexicon to find one or more appropriate lemmas for the first increment. The translation is resumed when r1 is available and continues until a first utterance is completed. Then a discourse planning refO is added that connects the first utterance with subsequent ones.[5]

The preverbal message is completed by the refOs r5, r4 and r2. Since r2 has been used previously, the binding function ante inserts a link between X_5 and X_2. The formulator checks whether a parallel structure is given for the preverbal message currently under construction. If parallel segments are found, these are marked by the parallelism constraint $A \sim B$. The final decision on producing a VPE is postponed until further information becomes available, e.g. about distance between VPs (Hardt and Rambow, 2001). Based on the CLLS representation the following three utterances (among others) can be produced:

(4) "Flight CK-314 is heading for Runway 3 and Flight MG-209 is heading for Runway 3." (no VPE)

(5) "Flight CK-314 is heading for Runway 3 and so is Flight MG-209." (VPE)

(6) "Flight CK-314 is heading for Runway 3 and Flight MG-209 is heading for it, too." (NP ellipsis and anaphora)

Note that the decision on which utterance is actually produced depends on the functioning of the formulator. Apart from its general production capabilities and preferences this decision very much depends on the state the formulator is in at the point of time when the increments from the conceptualiser arrive. A more elaborated discussion of this point can be found in Guhe and Schilder (2002).

5 Conclusion

The static perspective on language generation of most NLG systems runs into difficulties when the system is set in a dynamically changing environment. In such settings incremental processing is required. However, with incremental processing comes the need for a flexible, extendable kind of representation. We showed that the semantic underspecification formalism CLLS can be used to represent preverbal messages (semantic structures) that are incrementally generated from an underlying conceptual structure represented with refNets. The preverbal message changes dynamically. We demonstrated for the case of VP ellipses that it is capable of an adaptation when the original utterance plan is modified due to a change in the conceptual representation.

Acknowledgments

The research reported in this paper was partly conducted in the project ConcEv (Conceptualizing Events), which is supported by the DFG (Deutsche Forschungsgemeinschaft) in the priority program 'Language Production' under grant Ha-1237/10.

References

Susan E. Brennan. 2000. Processes that shape conversation and their implications for computational linguistics. In *Proceedings of the 38th ACL*.

[5] Since the generation of discourse structures is beyond the scope of this paper, we simplify the discourse relation list. For the CLLS translation, we use the logical connector $\wedge$ instead of $\lambda P \lambda Q\ list(P, Q)$.

Koenraad De Smedt and Gerard Kempen. 1987. Incremental sentence production, self-correction and coordination. In Gerard Kempen, editor, *Natural Language Generation*. Martinus Nijhoff Publishers, Boston, Dordrecht.

Koenraad De Smedt, Helmut Horacek, and Michael Zock. 1996. Some problems with current architectures in natural language generation. In Giovanni Adorni and Michael Zock, editors, *Trends in Natural Language Generation*. Springer, New York.

Markus Egg, Alexander Koller, and Joachim Niehren. 2001. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, 10:1–29.

Katrin Erk, Alexander Koller, and Joachim Niehren. 2001. Processing underspecified semantic representations in the constraint language for lambda structures. *Journal of Language and Computation*. To appear.

Wolfgang Finkler. 1997. *Automatische Selbstkorrektur bei der inkrementellen Generierung gesprochener Sprache unter Realzeitbedingungen*. infix, Sankt Augustin.

Eli Goldberg, Norbert Driedger, and Richard I. Kittredge. 1994. Using natural-language processing to produce weather forecasts. *IEEE Expert*, 9(2).

Markus Guhe and Christopher Habel. 2001. The influence of resource parameters on incremental conceptualization. In *Proceedings of the 4th ICCM*.

Markus Guhe and Frank Schilder. 2002. Incremental generation of self-corrections using underspecification. In *Proceedings of Computational Linguistics in the Netherlands 2001*.

Markus Guhe, Christopher Habel, and Heike Tappe. 2000. Incremental event conceptualization and natural language generation in monitoring evironments. In *Proceedings of the 1st INLG*.

Markus Guhe. under review. Incremental preverbal messages.

Christopher Habel. 1982. Referential nets with attributes. In *Proceedings of Coling 82*.

Christopher Habel. 1986. *Prinzipien der Referentialität: Untersuchungen zur propositionalen Repräsentation von Wissen*. Springer, Berlin, Heidelberg.

Daniel Hardt and Owen Rambow. 2001. Generation of VP ellipsis: A corpus-based approach. In *Proceedings of the 39th Annual Meeting of the ACL*.

Gerard Kempen and Edward Hoenkamp. 1987. An incremental procedural grammar for sentence formulation. *Cognitive Science*, 11(2):201–258.

Kevin Knight and Irene Langkilde. 2000. Preserving ambiguities in generation via automata intersection. In *Proceedings of AAAI 2000*.

Willem J.M. Levelt. 1989. *Speaking: From Intention to Articulation*. MIT Press.

Emanuele Pianta and Lucia M. Tovena. 1999. XIG: Generating from interchange format using mixed representations. In *Proceedings of the 6th AIIA Conference*.

Ehud Reiter. 1994. Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the 7th INLGW*.

Norbert Reithinger. 1991. POPEL – a parallel and incremental natural language generation system. In Cécile L. Paris, William R. Swartout, and William C. Mann, editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, Boston.

Frank Schilder and Markus Guhe. 2002. Underspecified parallelism contraint. This volume.

Wolfgang Wahlster, editor. 2000. *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer, Berlin.

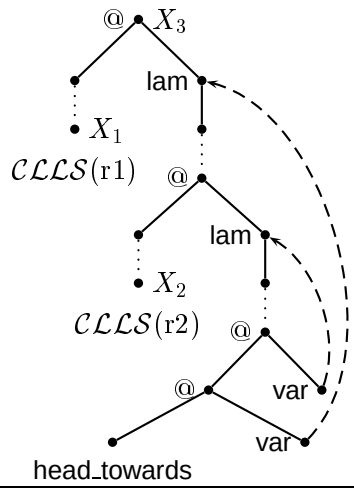
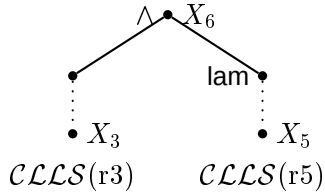
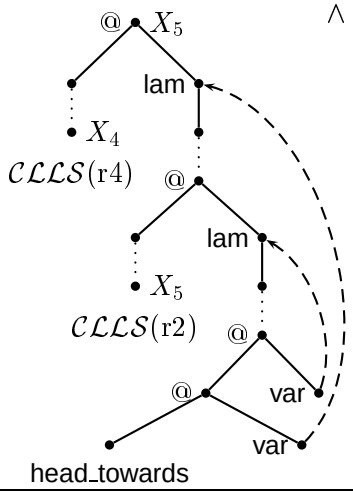
time	refO	→	underspecified λ-structure
t_0	r3 — head_towards(r1,r2)	→	
t_1	r1 — 'CK-314'	→	CK-314
t_2	r2 — 'RUNWAY 3'	→	Runway_3
t_3	r6 — list(r3,r5)	→	
t_4	r5 — head_towards(r4,r2)	→	$\wedge\ \mathsf{ante}(X_5) = X_2$ 
t_5	r4 — 'MG-209'	→	MG-209
t_6	r2 — 'RUNWAY 3'	→	Runway_3

Table 2: The preverbal message for (3)