

## The Influence of Resource Parameters on Incremental Conceptualization

**Markus Guhe (guhe@informatik.uni-hamburg.de)**

Research Group Knowledge and Language Processing (WSV),  
Department for Informatics, University of Hamburg,  
Vogt-Kölln-Straße 30, D-22527 Hamburg, Germany

**Christopher Habel (habel@informatik.uni-hamburg.de)**

Research Group Knowledge and Language Processing (WSV),  
Department for Informatics, University of Hamburg,  
Vogt-Kölln-Straße 30, D-22527 Hamburg, Germany

### Abstract

INC (*incremental conceptualizer*) is a cognitively motivated model of the first component of the language production process, conceptualization. It produces preverbal output for the domain of giving on-line descriptions of events that has the same structure as verbalizations of humans given the same task. The architecture and processing mechanisms of INC are based on simple, cognitively motivated principles. Its behavior is controlled by parameters that are set according to the available resources. Thus, INC requires no explicit and complex instructions about how to perform a given verbalization task, e.g. with respect to the level of detail, because the level of detail is an effect that depends on resources.

### Towards an Incremental Architecture for Conceptualization

An important goal in psycholinguistics is the development of a cognitively plausible architecture of language production. In contrast to language comprehension—cf. Lewis (to appear)—only few proposals for the architecture of the human language production faculty follow the methodological idea of computational architectures (cf. Anderson 1983, Newell 1990), viz. to combine empirical investigation and computational modeling. Levelt (1989) argues convincingly in his psycholinguistic approach that the language production component works incrementally, which means that a sequential information stream is processed in parallel. To achieve this, multiple processes, which are ordered in a fixed sequence, work simultaneously on an information stream, so that the output of one process is the input of the following one. In performing the task of describing of what happens in a permanently changing world, incrementality can improve the overall quality of the model substantially: (1) The dynamic nature of a continuous input stream can be handled faster, as only the most recent change has to be considered—in relation to the current state of affairs. This also optimizes the required storage capacity. (2) Incremental models can account for the fact that humans are capable of producing

fluent speech, i.e. speech without artificial auditory gaps. (3) The parallelism that comes with incrementality optimizes the runtime, because the processes need not wait for each other, or only for short periods.

There are some incremental models for that part of the language production faculty that does the linguistic encoding, i.e. the one that solves the *how to say* problem, cf. Kempen & Hoenkamp (1987), De Smedt, Horacek & Zock (1996). This component, called the *formulator* by Levelt (1989), which contains the stages of *sentence planning* and *surface generation* in Reiter's (1994) analysis, gets the input from the *what-to-say* component—*content planning* according to Reiter, or the *conceptualizer* in Levelt's approach. From a cognitive point of view, an incremental formulator (e.g. Abb, Günther, Herweg, Lebeth, Maienborn & Schopp 1996; De Smedt 1990; Kempen & Hoenkamp 1987) is plausible only in combination with a conceptualizer that produces preverbal messages, which are the interface between conceptualizer and formulator, incrementally.

INC (*incremental conceptualizer*) models this first stage of the language production process. It is based on the considerations described in Habel & Tappe (1999) and Guhe, Habel & Tappe (2000) and is specialized on producing preverbal messages for the on-line description of events. In the present paper we report on simulations performed by the implementation of INC in the domain of on-line descriptions of drawing sketch maps.<sup>1</sup> For that purpose we recorded in a first study the drawing of sketch maps with a drawing tablet. These sequences of space-time coordinates are the fundamental data structures for the two lines of our research. First, the recorded drawing processes—not the resulting sketch maps—are presented to participants of a psycholinguistic verbalization study, who are instructed to 'describe what they see', cf. Tappe & Habel (1998) and

---

<sup>1</sup> INC is restricted to describing events but domain-independent otherwise. Domain specificity is due to the content of the *concept storage*, cf. Figure 3. Beyond the domain of drawing events we have, up to now, used it for the description of motion events, cf. Guhe, Habel & Tappe (2000).

Habel & Tappe (1999). Second, they are the input for the implementation of the computational model INC. The simulations described in the following show a high correspondence between these two lines of research.

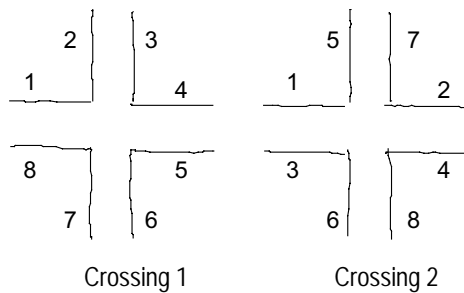


Figure 1: Example sketches

The two drawing processes symbolically depicted in Figure 1 serve as examples in the following. The two processes, i.e. their recordings, differ solely in the order in which their lines are drawn. This is indicated by the annotated numbers, which were not present in the recordings presented to the study participants. The different sequencing leads to different hierarchical event structures built up by the conceptualizer, e.g. the temporal ordering of the elements. The event structure of the genesis of Crossing1 is depicted in Figure 2. (The relation between dr-Parallel4 and dr-H-Line1 is left out for better readability.) There, dr-H-Line stands for ‘drawing a horizontal line’, dr-V-Line for ‘drawing a vertical line’; hence, dr-H-Line2 stands for the event that created the fourth line. The left-to-right ordering in the event structure stands for temporal precedence on each of the three layers. The nodes of the intermediate layer each subsume two nodes of the basic layer; dr-Parallel means ‘two parallel lines were drawn’, dr-Corner ‘two lines that touch each other at their endpoints were drawn’ (regardless of their angle). [NB: events, not their results are the subject of verbalization task and modeling.]

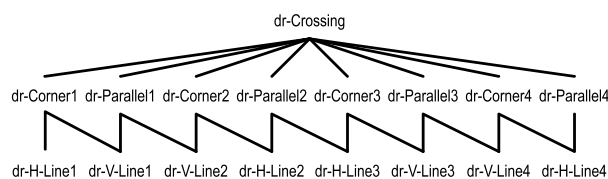


Figure 2: Event structure of the sketch map Crossing 1

## An Overview of INC

### Overall Architecture

INC starts with so-called *basic entities* as input. They are produced by a pre-processing unit that converts the input stream—consisting of coordinates of the drawing pen, which are read in a fixed time interval—into basic events and objects. This conversion is computed ac-

ording to the empirically founded cut-hypothesis of Avrahami & Kareev (1994, p. 239): “A sub-sequence of stimuli is cut out of a sequence to become a cognitive entity if it has been experienced many times in different contexts.” These basic entities are used to build up a hierarchical event structure, from which elements are selected for verbalization. In this way INC forms preverbal messages out of basic entities. The basic events for our examples here are dr-H-Line and dr-V-Line. For each line-drawing one *basic object*, which represents the result (the line), and one *basic event*, containing information like drawing speed and duration, are generated.

The processes of incremental models are often depicted as a cascade of processes (Levelt 1989): like water in a water cascade splashes down from one level to the next, information splashes down from one process to the next. This symbolizes the simultaneous processing of information on subsequent stages. In INC four processes form this cascade: *construction*, *selection*, *linearization*, and *pvm-generation* (preverbal message generation), cf. Figure 3. The processes all operate on the *current conceptual representation* (CCR), which contains a representation of the current state of affairs, e.g. the event structure from Figure 2 enriched by object information. This representation is a network of inter-related concepts, where the concepts are represented as nodes and the relations between them as links. The *construction* process builds up the CCR. *Selection* chooses the to-be-verbalized events from it, i.e. it chooses a path through the network, which we call the *traverse*. *Linearization* brings these events into an appropriate order, and *pvm-generation* generates preverbal messages from the chosen events. Since linearization is not yet implemented, it plays only a minor role here.

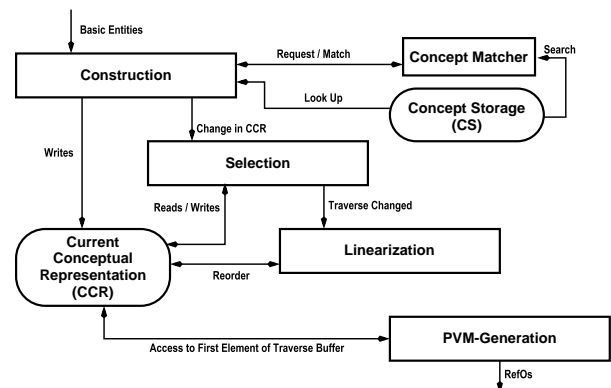


Figure 3: Architecture of INC

### Construction

The construction process reads the basic entities that are provided by the pre-processing unit and builds up the CCR from them. To do so, it uses an additional process: the *concept matcher*: each time a new basic entity or a

modification for an already existing one<sup>2</sup> is read from the pre-processing unit, construction calls the concept matcher to determine the best matching concept from the *concept storage* that subsumes this entity and some ‘surrounding’ ones, which are determined by a heuristic. The concept matcher then determines the *best match* by computing the *degree of agreement* (DoA) between these entities and the concepts stored in the concept storage. The best match is the concept with the highest DoA. A pair consisting of best match and DoA is given back to construction. The best match need not be complete; that is, even if parts of this concept are missing—up to this point—it can nevertheless be introduced into the CCR, with the missing parts marked as ‘expected’. Finally, construction compares the DoA with the *degree of agreement threshold* (DoAT). If DoA is greater than DoAT, construction carries out one of three available operations:

*Generate* introduces a new concept into the CCR. This operation is needed when (1) a new basic entity comes in from the pre-processing unit or (2) DoA > DoAT for the best match, and the best match is not yet contained in the CCR. In the latter case the matched entities are subsumed under the concept.

*Modify* is used when (1) the observed events demand an adaptation of an already existing basic entity or (2) an element of the CCR marked ‘expected’ is actually recognized. Then its status is changed to ‘regular’.

*Discard* is called when expectations in the CCR no longer correspond to the best match. It removes all elements marked ‘expected’. After *discard* is finished, the operation *generate* is usually called to introduce (an expectation for) the new best match into the CCR. Basic entities cannot be discarded.

Varying DoAT influences the behavior of INC. By this principle INC’s behavior is determined: there is a set of parameters, whose values are set when INC is initialized and which are the reason for different outputs.

## Selection

As already mentioned, selection assembles the traverse. To be more precise, it operates on the *traverse buffer*, viz. the last part of the traverse. Its variable length is determined by the parameter *length of traverse buffer* (LoTB). This part of the traverse can be modified, because its elements are not yet handed on to the subsequent unit of the language generation system, the formulator. This buffering is necessary for two main reasons; first, linearization needs some time to take place, and a part of the traverse where it can exchange the order of the elements. Second, selection needs a possibility for revising its choices, especially with the selection method we are using here. It is based on what we call *Extended Wundt’s Principle*. Levelt (1989) formulates

<sup>2</sup> A basic event needs to be modified if, for example, the drawing of a line has been interrupted (the basic event is generated) but then has been continued (modification).

*Wundt’s Principle*: an incremental process starts computing as soon as it obtains some characteristic input. Our extension consists in not only starting to process input as soon as possible but to produce output as quickly as possible, as well. The disadvantage of this method is that quite a lot of choices need to be revised. However, the advantage is that linearization has the maximum amount of time to operate on the elements in the traverse buffer, before they are taken out by pvm-generation. Additionally, this method results in an *any-time* capability of INC: pvm-generation is capable of producing a preverbal message at any point in time.

Selection has available two operations for modifying the traverse buffer: *append* and *replace*. The selection algorithm is applied to each new or modified node and consists of three parts:

1. *Append* the node if 2 and 3 are not the case (default case).
2. If there is already a more complex node in the traverse buffer that contains this one, do *not* select it.
3. If there are nodes in the traverse buffer that are contained by the node *replace* these nodes with it.

In the cases 1 and 3 the time of insertion—a time stamp—is also saved with the element.

## PVM-Generation

Pvm-generation has access to the first element of the traverse buffer. It takes out this first element to produce a preverbal message based thereon if the specified *latency time* (LT), computed as the difference between the current point of time and the time stamp, has passed. The element is then no longer part of the traverse buffer (but, of course, still a part of the traverse).

For the present purpose, viz. producing preverbal structures that can be compared to our recorded human verbalizations, the output of pvm-generation consists simply of the unique number with which this element is internally labeled. This, obviously, is no complete preverbal message that can be used by a formulator to produce an utterance, but it will suffice here.

## Three Resource Parameters that Determine the Behavior of INC

The behavior of INC is highly dependent on resource parameters, of which we discuss here:

1. *degree of agreement threshold* (DoAT)
2. *length of traverse buffer* (LoTB)
3. *latency time* (LT)

In the present section we describe these parameters, their function and their effects in processing in more detail. To test their functioning and effects in detail, we studied INC’s behavior by varying them systematically; these simulations are described in the next section.

### Degree of Agreement Threshold (DoAT)

**Function** DoAT is instantiated with a value between 0 and 1. It determines whether the *degree of agreement*

(DoA) of a *best match* determined by the concept matcher is high enough, i.e. whether the match is good enough, to cause construction to work on the CCR.

**Motivation** DoAT can be regarded as a technical necessity as well as being cognitively adequate. The crucial question is, whether humans do not generate expectations if the DoA is not high enough (in our terms: introduce expected nodes into the CCR) or whether they generate any expectation they are capable of creating but do not select these expectations from the CCR for verbalization. Since generating expectations is costly (in the sense that it requires time and other resources), the solution we have used in INC seems to be the plausible one, cognitively as well as technically.

**Expected Effects** By varying the value for DoAT the amount of expectations the construction process generates is increased or decreased. The effect of DoAT = 0 is that construction uses any best match to work on the CCR. For our two example sketches this means that the drawing of only one line already leads to the expectation that this is the drawing of a rectangle, because this is the simplest concept in the concept storage containing a line. DoAT = 1 means that no expectations are generated at all; only after the drawing is finished, the completely perceived crossing is verbalized. (Basic events can, of course, be verbalized before that.)

**Comments** Both borderline cases are untypical in our recorded verbalizations, because (1) almost no participant keeps silent or talks only of lines until the sketch is complete (DoAT = 1)—it looks like they cannot resist speculating about what the final result will be—, (2) no participant utters expectations permanently (DoAT = 0), and (3) no participant jumps to such ‘unjustified’ conclusions that the drawing of a line will result in the drawing of a rectangle (DoAT = 0). The typical case is that they use lines and their interrelations until they find the DoAT is high enough, whereupon they produce an utterance about what they expect the final result will be.

### Length of Traverse Buffer (LoTB)

**Function** LoTB specifies the number of elements that can be stored in the traverse buffer.

**Motivation** LoTB models the capacity of the working memory of the verbalizer that is available for conceptualization. Thus, the maximum value of LoTB is determined by the individual’s current storage capacities.

**Expected Effects** If the value for LoTB is too small, elements get lost (are forgotten), because pvm-generation does not use them in time to produce a preverbal message. Thus, by increasing its value the chance is decreased that elements get lost and vice versa.

**Comments** LoTB = 0 is not possible: INC would not work, because elements must be put in the traverse

buffer to be accessible for pvm-generation; this would be like a working memory with no storage capacity, thus  $LoTB \geq 1$ . There is no technical reason to limit length of the traverse buffer, because of processing limitations of INC. Thus, this value is purely motivated by cognitive considerations. In Guhe, Habel & Tappe (2000) we showed that there are technical settings in which this capacity limitation is not desirable.

### Latency Time (LT)

**Function** LT is the span of time that an element is kept in the traverse buffer, before it is taken by pvm-generation so as to produce a preverbal message for it.

**Motivation** This value is necessary for selection to be able to change its choices and for linearization to be able to perform its operations.

**Expected Effects** A lower value for LT has the effect that utterances are temporally closer to the genesis of the sketch map. A higher value means that the produced output is better and more reliable, because (1) selection and linearization have had more time to improve their output, and (2) more input from the pre-processing unit is available, which means that the CCR contains more reliable information about what is happening.

**Comments** The latency time models the time pressure on the verbalization task. If the traverse buffer is permanently filled or if elements are even lost, the latency time can be reduced to compensate this—the participant ‘talks faster’. This can be especially important when the value for LoTB is comparatively small.

## Fixing the Resource Parameters

### The experimental design

The two sketches of our simulations are shown in Figure 1. We obtained corresponding human verbalizations by presenting sketches to 12 participants, 10 of which used the following pattern for Crossing 1:

- first segment (dr-H-Line)
- intermediary complexes (dr-Corner<sup>3</sup>, dr-Parallel, dr-Corner)
- crossing expected
- intermediary complexes (dr-Corner, dr-Parallel, dr-Corner)
- crossing complete

One of the other two kept silent until the sketch was complete and only then uttered: ‘A crossing or a cross.’ The twelfth uttered no expectations but only described the emerging lines, before naming the result. The other 10 differed only slightly in how much of the crossing

---

<sup>3</sup> When pvm-generation produces an intermediary complex concept like ‘dr-Parallel’, this stands for utterances like ‘A line parallel to the previous one is drawn.’

was visible when they produced the expectation; but all named it before it was fully visible. The verbalizations of Crossing 2 stick to a similar standard pattern:

- the first two segments (dr-H-Line, dr-H-Line)
- intermediary complexes (dr-Parallel, dr-Parallel, dr-Corner)
- crossing expected
- intermediary complexes (dr-Parallel / dr-Corner<sup>4</sup>, Parallel / dr-Corner)
- crossing complete

### Example Output of a Simulation

The output of the simulation for Crossing 1 with DoA = 0.5, LoTB = 6, and LT = 125ms yields this output:

```
hline1
hline1   vline1
gen(hline1)
corner1
corner1
corner1   vline2
gen(corner1, sum(vline1, hline1))
paralle11
paralle11
paralle11   hline2
paralle11   corner2
gen(paralle11, sum(vline2, vline1))
corner2
corner2   hline3
gen(corner2, sum(hline2, vline2))
paralle12
crossingE
crossingE   vline3
gen(crossingE, sum(vline1, hline1, hline4,
  vline4, vline3, hline3, hline2, vline2))
corner3
corner3
gen(corner3, sum(vline3, hline3))
vline4
paralle13
paralle13
paralle13   hline4
gen(paralle13, sum(vline4, vline3))
corner4
corner4
gen(corner4, sum(hline4, vline4))
paralle14
crossingR
crossingR
gen(crossingR, sum(vline1, hline1, hline4,
  vline4, vline3, hline3, hline2, vline2))
```

Here, we adopt the following notation: names of nodes like `hline1` stand for the node `dr-H-Line1`. `cross-`

<sup>4</sup> Here, `dr-Parallel` and `dr-Corner` are generated at the same time. Since there is no temporal criterion for selecting one of the two, the choice is made randomly.

`ingE` stands for an expected crossing, `crossingR` for a complete one. The lines containing only node names show the content of the traverse buffer at that point of time; lines starting with `gen` (generate) mean that pvm-generation generates a preverbal message for this element. For example, `gen(hline1)` can be understood as the core of a preverbal message that leads to an utterance like ‘A horizontal line has been drawn.’ If a complex node is generated its parts are given as well to verify the internal relations produced during the simulation. A comparison with Figure 2 shows that `INC` produces the desired structures in the CCR, the only difference (due to a peculiarity of the implementation) being that the parts of `crossing` are traced back to the level of basic events. Two consecutive lines with the same content, e.g. `corner3`, are typed out by the system, when something in the CCR is modified, but the traverse buffer does not change.

### Results and Discussion

**Overall Results** With the simulations we come close to the observed verbalizations of humans (Tappe & Habel 1998; Habel & Tappe 1999) for both examples. This means that the sequences of preverbal messages (`gen` lines) in the output correspond structurally to the sequences of utterances of the observed human verbalizations. The values for LT and the runtime of the program in the following are machine dependent and possess only limited cognitive adequacy, because the implementation of `INC` reads prefabricated input files. This is faster than simulations that run in real-time<sup>5</sup> but directly proportional to it. The values for DoAT and LoTB certainly come closer to the goal of being psychologically real, but are merely proportional to the actual measures.

For DoAT we used two values: 0.5 and 0.9, for LoTB values between 2 and 6, and we varied LT between 50ms and 2000ms. The runtime of one execution of the program is split into two phases: 1400ms for the initialization of program and concept storage, and the runtime of the conceptualization. The second value, which is the important one here, is 2700ms to 3000ms for Crossing 1 and 2500ms to 2800ms for Crossing 2. (We used several pre-processing file per sketch, all of which yield equivalent results but take different runtimes.)

The values for both crossings that result in the simulation of the human verbalizations sketched above are: DoAT = 0.5, LoTB > 1, LT = 125ms. When varying these values, which we regard as our standard settings, we obtain the expected effects.

**Degree of Agreement Threshold (DoAT)** The maximum DoA for the crossings before they are complete is 0.81. Therefore, the maximum value of DoAT that has the effect that an expected crossing is introduced into

<sup>5</sup> Real-time means that the input for the pre-processing unit reads coordinates in the time interval with which they were recorded and produces basic entities for `INC` accordingly.

the CCR is 0.8. Thus, using DoAT = 0.9 instead of 0.5 means that no expectation is inserted into the CCR.

**Latency Time (LT)** When increasing the value for LT, more preverbal messages are produced, decreasing LT leads to the production of less. From a  $LT \geq 800$ ms on no simple elements and no intermediary complexes are verbalized but only the expected and the complete crossing. The maximum value that still leads to the production of a preverbal message for the expected crossing is ca. 500ms for Crossing 1 and ca. 1000ms for Crossing 2. (These values should be—more or less—identical. This gap is an artifact that arises, because we use pre-processed files as input.) With  $LT < 125$ ms more basic events and some intermediary complexes—which are otherwise ‘swallowed’ by condition 3 of the selection strategy—are verbalized. With  $LT = 0$ ms each selected node is verbalized.

**Length of Traverse Buffer (LoTB)** We never need a value for  $LoTB > 5$ .  $LoTB = 5$  is required for  $LT \geq 800$ ms, if no elements of the traverse buffer shall get lost.  $LoTB = 3$  suffices for most settings; only with  $LT < 125$ ms a value of 2 suffices. Other settings ( $LoTB < 3$  with  $125$ ms  $< LoTB < 800$ ms or  $LoTB < 5$  for  $LT \geq 800$ ms) have the effect that elements are lost. A value of  $LoTB = 1$  means that elements are lost, and in almost all cases only the complete crossing is verbalized.

### Conclusion and future work

With INC we present a cognitively motivated model of the first part of the language production process, the conceptualizer. By using simple design principles we get a model that exhibits behavior very similar to that observed in verbalizations of humans when they give on-line descriptions of events. By varying three resource parameters we were able to identify those parameter settings that lead to the typical verbalizations. Furthermore, INC produced all types of verbalizations we found in the verbalization corpus, e.g. even the rare case of participants uttering nothing until it is plain that the final result is or will be the drawing of a crossing.

Thus, INC does not need explicit instructions about what degree of detail or what level in the event hierarchy is used for the verbalization. This accounts for that it is very unlikely that humans deliberately choose one level. Instead, we use three parameters, motivated by cognitive considerations (DoAT, LoTB, LT), that determine INC’s resources. By assigning different combinations of values we obtain our results.

A more elaborate model should be capable of adapting the values to the current demands of processing and to the currently available resources. For example, LoTB is a measure of what amount of the working memory is available for utterance planning. If, however, a participant has to perform additional tasks, LoTB needs to be reduced in accordance with the other cognitive subsystems that make use of this resource.

### Acknowledgements

The research reported in this paper was 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 to Christopher Habel.

### References

- Abb, B.; Günther, C.; Herweg, M.; Lebeth, K.; Maienborn, M. & Schopp, A. (1996) Incremental grammatical encoding—an outline of the SYNPHONICS formulator. In G. Adorni & M. Zock, eds., *Trends in natural language generation: An artificial intelligence perspective*, Berlin: Springer.
- Anderson, J. R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard UP.
- Avrahami, J. & Kareev, Y. (1994) The emergence of events. *Cognition*, 53, 239–261.
- De Smedt, K. (1990) IPF: An Incremental Parallel Formulator. In R. Dale, C. Mellish & M. Zock, eds., *Current Research in Natural Language Generation*, London: Academic Press.
- De Smedt, K.; Horacek, H. & Zock, M. (1996) Architectures for natural language generation. In G. Adorni & M. Zock, eds., *Trends in natural language generation*, Berlin: Springer.
- Guhe, M., Habel, C. & Tappe, H. (2000) Incremental Event Conceptualization and Natural Language Generation in Monitoring Environments. *INLG 2000*.
- Habel, C. & Tappe, H. (1999) Processes of segmentation and linearization in describing events. In R. Klambunde & C. von Stutterheim, eds., *Representations and Processes in Language Production*, 117–153, Wiesbaden: Deutscher Universitäts-Verlag.
- Kempen, G. & Hoenkamp, E. (1987) An Incremental Procedural Grammar for Sentence Formulation. *Cognitive Science*, 11:2, 201–258.
- Levelt, W.J.M. (1989) *Speaking: From intention to articulation*. Cambridge, MA: MIT Press.
- Lewis, R.L. (to appear). Specifying architectures of language processing: Process, control, and memory in parsing and interpretation. In M. Crocker, M. Pickering & C. Clifton, eds., *Architectures and mechanisms for language processing*. Cambridge: CUP.
- Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard UP.
- Reiter E. (1994) Has a consensus NL generation architecture appeared, and is it psycholinguistically plausible? *IWNLG-1994*, 163–170, Kennebunkport, ME.
- Tappe, H., Habel, C. (1998) Verbalization of Dynamic Sketch Maps: Layers of Representation and their Interaction. [full version of one page abstract / poster at CogSci 1998.] <http://www.informatik.uni-hamburg.de/WSV/sprachproduktion/CogSci98.ps.gz>