A Revision-Based Generation Architecture for Reporting Facts in their Historical Context

Jacques Robin

Department of Computer Science

Columbia University

New York, NY 10027

Abstract

Natural language reports generated by existing systems ignore the historical context of the facts and events they relate. In this paper, I argue that going beyond this limitation requires abandoning the pipelined architecture of existing report generators. I propose a new architecture in which a first draft of the report is organized around new information and then incrementally revised to opportunistically add related historical information. This type of information-adding revision allows to elaborate inside a clause or a nominal while taking into account surface structure constraints from any other portion of the report. In addition to providing the additional flexibility required to convey historical information, the proposed architecture constitutes an interesting testbed to investigate a wide range of open questions concerning content planning below the sentence level, generation with revision and generation architecture.

1 Introduction: generating reports in their historical context

As computer access to larger and more diverse sources of quantitative data is becoming commonplace, the automatic production of natural language reports summarizing such data is a generation application of increasing interest. The feasibility of report generation has been established for a fair variety of application domains: the stock market, meteorology, wargames and office automation. The field was pioneered by K.Kukich with her system ANA [Kukich 1983]. ANA summarizes in a few sentences the daily fluctuations of several stock market indexes from half-hourly updates of their values. In a similar vein but involving spatial smoothing as well as temporal smoothing, FoG [Bourbeau *et al.* 1990] produces local marine weather bulletins in both English and French. Differing from these two systems in its non-numerical input, TEXPLAN [Maybury 1990] summarizes battlefield activity from an event network produced by an air-land battle simulator. Finally, the multi-media system SAGE [Roth *et al.* 1988] generates status reports of large projects in both text and graphics.

Although some of these systems, especially ANA, produce reports of impressive quality, comparing their output with their human-generated counterparts reveals one drastic limitation: they do not convey any *historical information*. In report generation, historical information is any event or cumulative data whose lifetime exceeds the time-slice covered by a report, as opposed to *new information* whose lifetime is bounded by that time-slice. So for example when ANA reports about the fluctuations of a financial index for the day (new information) it does not relate it with the fluctuations of that index over the past week or month (historical information). Thus, existing report generators are not able to put the facts and events they relate in their historical perspective. In contrast, humans generating reports systematically present new information in combination with related historical information. My research focuses on building a report generation system that is able to do the same.

As a starting point I analyzed a corpus of basketball game reports taken from the UPI newswire. Figure 1 shows a typical report from this corpus. Figure 2 shows the same report stripped off its historical information. It gives an idea of the best possible output a system designed like existing report generators could produce

for the basketball domain. In Figure 1 **historical information** is emphasized by a boldface font[1].

SACRAMENTO, Calif. – Michael Adams scored **a career-high** 44 points Wednesday night, including seven 3-point baskets, to HELP the **short-handed** Denver Nuggets **end a five-game losing streak** WITH a 128-112 victory over the Sacramento Kings.
Adams, **who was drafted and then discarded by the Kings four seasons ago**, made 17 of 26 field goals, including seven of 11 3-point attempts, and hit three of four free throws **to break his previous career high of 35 points**. ADAMS also dished out a game-high 10 assists and had five steals.
Rookie Chris Jackson added 22 points and center Blair Rasmussen pumped in 21 and grabbed 12 rebounds for Denver, which outscored Sacramento 19-4 during the final 5:01 of the fourth quarter.
The Nuggets, who had only eight players available for the game, improved to 2-12 on the road and 6-20 overall.
Rookie guard Travis Mays, **playing his second game after missing 11 with back spasms**, scored **a season-high** 36 points *for the Kings*, **who lost their fourth straight**. Lionel Simmons added 24 points and 15 rebounds.

Figure 1: A human-generated report conveying historical information.

SACRAMENTO, Calif. – Michael Adams scored 44 points Wednesday night, including seven 3-point baskets, to LEAD the Denver Nuggets TO a 128-112 victory over the Sacramento Kings.
Adams made 17 of 26 field goals, including seven of 11 3-point attempts, and hit three of four free throws. HE also dished out a game-high 10 assists and had five steals.
Chris Jackson added 22 points and center Blair Rasmussen pumped in 21 and grabbed 12 rebounds for Denver, which outscored Sacramento 19-4 during the final 5:01 of the fourth quarter.
Guard Travis Mays scored 36 points and Lionel Simmons added 24 points AND 15 rebounds *for the Kings*.

Figure 2: The report of figure 1 stripped off its historical information.

The difference between the reports of figure 1 and figure 2 illustrates the importance of historical information in report generation. Almost all reports in the corpus I analyzed contain some historical information[2]. These newswire reports are organized following the so-called *inverted pyramid structure* [Fensch 1988] meaning that crucial information is packed in the lead followed by facts of decreasing importance towards the end of the report. Such a structure makes reports instantly editable by cutting their tails. If space in the client newspaper is scarce, only the lead will remain. In my corpus all the leads, often consisting of a single sentence, are of the summary type [Fensch 1988]. They are thus self-contained small reports containing the basic facts. This type of report structure is not particular to sports reporting but is pervasive in newswire articles [Mencher 1984]. It allowed me to focus my corpus analysis on the first sentence of the reports. I looked at over 800 of them. They tend to be quite complex[3] with an average number of words of 31 varying

[1] The other font effects indicate information whose wording or position differs in both versions due to the introduction of historical information.
[2] I also casually looked at stock market reports where historical information is also pervasive.
[3] At least by natural language generation standards.

from a minimum of 21 to a maximum of 46. They all convey at least four propositions: the location, date and score of the game, as well as the most significant statistic of a player from the winning team. The simplest of these sentences convey nothing else, e.g.:

"CHARLOTTE, Va - Patrick Ewing scored 41 points Tuesday night to lead the New York Knicks to a 97 79 victory over the Hornets."

However, in the most complex sentences, up to five additional propositions can be added to the four basic ones for a total of nine[4], e.g.:

"LOS ANGELES, Ca - Karl Malone scored 27 points and John Stockton added **a season high** *27 points and handed out* **a league high** *23 assists Saturday night leading the Utah Jazz to* **it fourth consecutive victory**, *a 105 95 win over the Los Angeles Clippers."*

Note that in this example three of these additional propositions are historical. *52% of all opening sentences in the corpus contained at least one historical proposition.* Making report generators sensitive to the historical context would therefore significantly widen their semantic coverage and make their output much more realistic. It would also allow them to determine in a more principled way what new information to include in the report. The relevance of a new fact is often directly dependent on related historical facts. For example consider the following report excerpt:

"Seattle guard Dale Ellis made his first appearance of the season after being sidelined for 17 games with nerve damage in a foot. *He scored 8 points."*

In the absence of any historical knowledge, an 8 point scoring performance is not worth reporting. However knowledge that the athlete is just back from an injury (as above) or that he is averaging 30 points a game in the current season can make an 8 point statistic very relevant. In short, by carrying out the task of putting reported facts in their historical perspective, a generation system provides its users with additional clues in order to interpret the raw data.

In the next section I examine the demands that providing such additional clues puts on the architecture of

[4] The average number of propositions in an opening sentence is six.

a report generation system and show that they cannot be answered by the traditional pipelined architecture. In section three I propose a new architecture answering these demands. In this architecture, a report draft organized around new information is incrementally revised to opportunistically convey related historical information. In section four I compare the type of revision performed in this architecture to previous proposals for revision in generation. I conclude and discuss future work in section five.

2 Historical context and report generation architecture

In this section I discuss the issue of historical information from a generation architecture standpoint. I first define a vocabulary to clarify architecture issues and then enumerate the properties that a report generation architecture must have in order to flexibly convey historical information. I then show that the architecture of existing report generators lacks these necessary properties.

2.1 The text generation subtasks

One problem in discussing architectural issues in generation is the absence of a precise and standard terminology: the same terms have been used to describe different notions by different authors in the literature. In this section I briefly define a vocabulary that I will then use throughout the paper.

Text generation is traditionally decomposed in three subtasks: content determination, content organization and content realization. Together, the first two subtasks are also generally referred to as content planning or deep-generation. Useful as it may be such a decomposition is too coarse grained. It is necessary to make additional distinctions and to further decompose the text generation process.

Content determination can be decomposed into content *production* and content *selection*. Although in many cases an underlying application provides the generator with all the potential content, in other cases the generator's input is but one part of that content. The rest of the content has to be produced by the

generator itself. This is what Hovy calls interpretation in generation [Hovy 1988]: the generator needs to enrich its input with more content. In the extreme case the input consists only of communicative goals and it is the generator that produces all content from the knowledge sources it has access to. Content *production* subtly differs from content *selection* which consists in deciding which part of the produced content is to be included in the generated text. In general (at least part of) content selection is performed in combination with content organization during content planning. Yet another case requiring content production by the generator is when its input format is totally inadequate to content planning. The generator must first produce content of an acceptable format before being able to start planning.

What each generation subtask recovers highly depends on the level of *microcoding* at which it is performed. This level of microcoding is defined by the minimal linguistic rank of the entries stored in the generator's lexicon. A *macro*coded generator like ANA produces sentences by assembling entire clause and group patterns *stored* as a whole in its phrasal lexicon. In contrast, a *micro*coded generator like EPICURE [Dale 1988] dynamically builds from a word-based lexicon[5] every sentence constituent down to the group rank[6]. The level of microcoding at which a generator operates is a crucial architectural characteristic because, apart from content realization, it also has repercussions on content selection and content organization. Both tasks significantly differ in nature depending on the rank of the linguistic unit being planned: text, paragraph, sentence, clause or group. In particular, macrocoded generators are *not* concerned with content selection and organization at the clause and group ranks. For these systems, everything below the sentence level is a content realization matter. In contrast, for microcoded generators, the distinction between content selection, content organization and content realization is relevant even at the clause and group ranks. Thus, while for ANA generating *"The industrial average"* instead of *"The Dow Jones average of 30 industrials"* is a content realization decision, for EPICURE generating *"the ripe banana"* instead of *"the banana"* results from content selection and content organization choices. Generating *"the large shrimp"* and not *"the large prawn"* would be a genuine content realization choice for EPICURE. In the rest of the paper, I therefore distinguish between *macro-level* content selection and organization at the text, paragraph and sentence ranks and *micro-level*

[5] i.e. a lexicon whose entries are individual words

[6] In that sense, systems like PHRED [Jacobs 1985] and PAULINE [Hovy 1988] whose lexicon contains both phrasal patterns and individual words are thus rather microcoded than macrocoded.

content selection and organization at the clause and group ranks.

Content realization can be decomposed into the four following subtasks:

- Lexicalization: the expression of semantic content by choice of open-class lexical items.
- Semantic grammaticalization: the expression of semantic content by choice of grammatical category (e.g. clause vs. NP) and grammatical features (e.g. tense for clauses, definiteness for NPs).
- Morpho-syntactic grammaticalization: the enforcement of syntax by choice of closed-class lexical items, choice of open-class lexical item inflections, specification of ordering constraints etc.
- Linearization: the spelling out of the inflected lexical items following the ordering constraints.

It is during the first two of these subtasks that the mapping between content units and linguistic units occurs; everything preceding them is purely conceptual manipulation, everything following them purely syntactic manipulation. Because they bridge the gap between conceptual and syntactic processing these two tasks have been considered part of content planning by some authors while part of content realization by others. In this paper I am holding this second view.

2.2 How historical context affects report generation subtasks

When taking historical context into account, content production no longer consists of just formatting a statistical input for text planning purposes. It becomes also necessary to (1) use this input information to update an historical knowledge base and (2) enrich this input information with related facts retrieved from this historical knowledge base.

Moreover, taking into account a larger temporal context multiplies the dimensions along which to evaluate the significance of an individual statistic. Consider for example, the scoring of a player during a basketball game. Ignoring the historical context there are basically only two ways in which this information can be significant: when compared to the scoring of all the other players in the game (game-high) and when

compared to the scoring of his teammates only (team-high). However such a statistic can be *historically* significant in a combinatorially explosive number of ways: when compared to his own scoring average (or high or low) over the season (or over his entire career or since he joined his current team etc), when compared to the highest scoring performance of any player on his team (or on any team in a given division, conference, etc) this season (or in the last 10 games, or ever) while playing at home (or on the road) against a particular opponent (or against any opponent) etc. Because of the sheer number of historical facts, any such fact is likely to be of similar relevance as many others. Therefore, the output of content production becomes a much larger set of content units with many more shades of importance.

This makes content selection much harder. General rules based on purely encyclopedic grounds (e.g., a if a player scores more than 20 points or if he scores more than anybody in his team, then include his scoring in the report) no longer suffice. Other types of constraints need to be considered to decide which content units to include in the report and which to leave out. These constraints include, how well a content unit fits in stereotypical content organization patterns and how its possible realizations can be combined or juxtaposed with those of other content units in cohesive and stylistically appropriate surface forms. In other words, *final content selection must be mutually constrained by content organization and content realization*[7].

As far as content organization is concerned, comparing the report of figure 1 with the report of figure 2 shows that adding historical information has little effect on the macro-level organization of a report. Only in the rare corpus examples where most facts reported are historical is macro-level organization affected. At the micro-level, however, content organization is highly affected by the addition of historical information. This last point is further illustrated in figure 7, where each of sentences 1-6 conveys, in a different way, the same historical information, namely that the losing team is having a bad streak. Two important observations to make about these paraphrases are: (1) they differ in grammatical category (clause vs. NP), and (2) they cut across linguistic rank (from clause-complex to nominal group). Therefore, flexibly expressing historical information requires the ability to make *cross-ranking content organization decisions*[8]. Note also that in most

[7] This has been also observed in other domains (cf. [Danlos 1986] among others).

[8] This is a particular instance of the general problem of non-isomorphism between semantic structure and syntactic structure which has been pointed out in other domains by [Talmy 1985] and [Zock 1988]. The effects of this non-isomorphism on controlling the generation process are also discussed in [Elhadad and Robin 1992].

of these paraphrases the historical facts are conveyed inside an NP, as a modifier of a head referring to a non-historical fact. Therefore, incorporating historical informations requires to *microcode content realization down to the individual word level.*

Not only can a given type of historical fact be conveyed at various linguistic rank, but it can also be conveyed in various *locations* in the report. Consider, for example, where streak information about each team is conveyed in the report of figure 1. Denver's streak information is hooked-up (together with another historical fact) to game result information in the first sentence of the report. In contrast, Sacramento's streak information is hooked-up (together with three other historical facts) to the scoring statistic of its highest scoring player. Note that Sacramento is also referred to in the first sentence and thus its streak information could have been hooked-up there as well. Another possibility would have been to group both pieces of streak information in an additional sentence inserted right after the opening sentence[9]. Both these options are used in some other reports of the corpus. This non-locality of historical information combined with its predominant expression by low rank linguistic constituents requires *part of content selection and organization to be performed down to the micro-level and yet non-locally.*

In contrast to historical information, new information generally does not display such flexibility in terms of where it appears in a report. For example, the game's final score and the winning team's leading scorer statistic systematically appear in the first sentence whereas a losing team's statistic never appears in the first sentence. This contrast between the way new and historical information are respectively conveyed suggests to *prescriptively* perform selection, organization and realization of non-historical content, while *opportunistically* performing selection, organization and realization of historical content.

The observations I made in this section on the way historical information is conveyed in human-generated reports suggests that handling historical information requires a report generation architecture to meet the four following criteria:

1. Part of content selection, content organization and content realization must be interdependent.

[9] Occurrences of more than two consecutive sentences expressing only historical facts are very rare in the corpus.

2. Part of micro-level content selection and organization must be carried out in a non-local opportunistic fashion.

3. Cross-ranking content organization choices down to the micro-level must be handled.

4. Content realization must be microcoded using a word-based lexicon.

Existing report generators are all based on the traditional pipelined architecture. In this type of architecture *all* content realization decisions are made after *all* content selection and organization choices have been finalized. Moreover all content selection and organization operations are performed on a clause-by-clause basis precluding non-local micro-level decisions. Their pipelined architecture thus prevent existing report generators from meeting the first two criteria. In addition, ANA does not meet the last two criteria because it is macrocoded.

Therefore, making report generators sensitive to the historical context requires not only access to a historical knowledge base but also re-thinking of the architecture. In the next section I propose a new architecture devised with the above four criteria in mind.

3 A new architecture for report generation

In this section I propose a new architecture for report generation that allows historical information to be conveyed. This architecture is presented in figures 3 and 4. The graphical conventions in these two figures are the following: processing components are in rectangles and the internal representation they produce or manipulate are in ovals. A bi-directional arc between two components means that they can exchange information and an one-directional arc between a component and an internal representation signals that this representation is either the input or the output of the component. The white rectangles correspond to components lying outside the focus of my research. The light grey rectangles correspond to the components that I designed and partially implemented in the generation system STREAK (Surface Text Reviser to Express

Additional Knowledge). These components are all rule-based systems using declarative knowledge sources[10].

In this new architecture, generation proceeds in two main steps: draft, then revision. The processing components and internal representations involved in the construction of the draft are shown in figure 3. Those involved in the revision of this draft and the final natural language output are shown in figure 4[11].

The surface sentence generator has been designed and implemented by M.Elhadad. It is composed of a functional unifier FUF [Elhadad 1990] [Elhadad 1991] and a grammar of English[12]. The functional unifier FUF also drives all the rule-based components of the system.

Although STREAK does not currently have a knowledge base management system, it does have a small basketball knowledge base. The latter is implemented in CLASSIC [Resnick *et al.* 1990] and it can be queried by the other components.

In the following sections I first sketch a walk through the system. I then provide more detailed information on each level of representation and each processing component. I then explain where and when the subtasks defined in section 2.1 are carried out. Finally, I discuss the applicability of this architecture for text generation in general.

3.1 Data flow

Generation proceeds in two steps. During the first step a complete specification of a report draft is built. This draft contains only the basic facts and is structured around new information. In the second step, this draft is incrementally revised in order to include secondary information such as historical facts. The final specification is handed to the surface sentence generator which produces the final version of the text.

The first pass starts upon reception of new statistical data (e.g. the box-score of basketball game or an

[10] In order to avoid overloading the figures I did not include the knowledge sources. I discuss them in more detail later on in this section. The implementation is still at an early stage: the knowledge sources currently used contain just enough information for running a few example sentences.

[11] Note how several processing components and internal representations play a role in both steps.

[12] A somewhat outdated presentation of this grammar can be found in [McKeown *et al.* 1990]. It has been extended and revised several times since then, most notably to incorporate some of the ideas developed by Fawcett [Fawcett 1987].

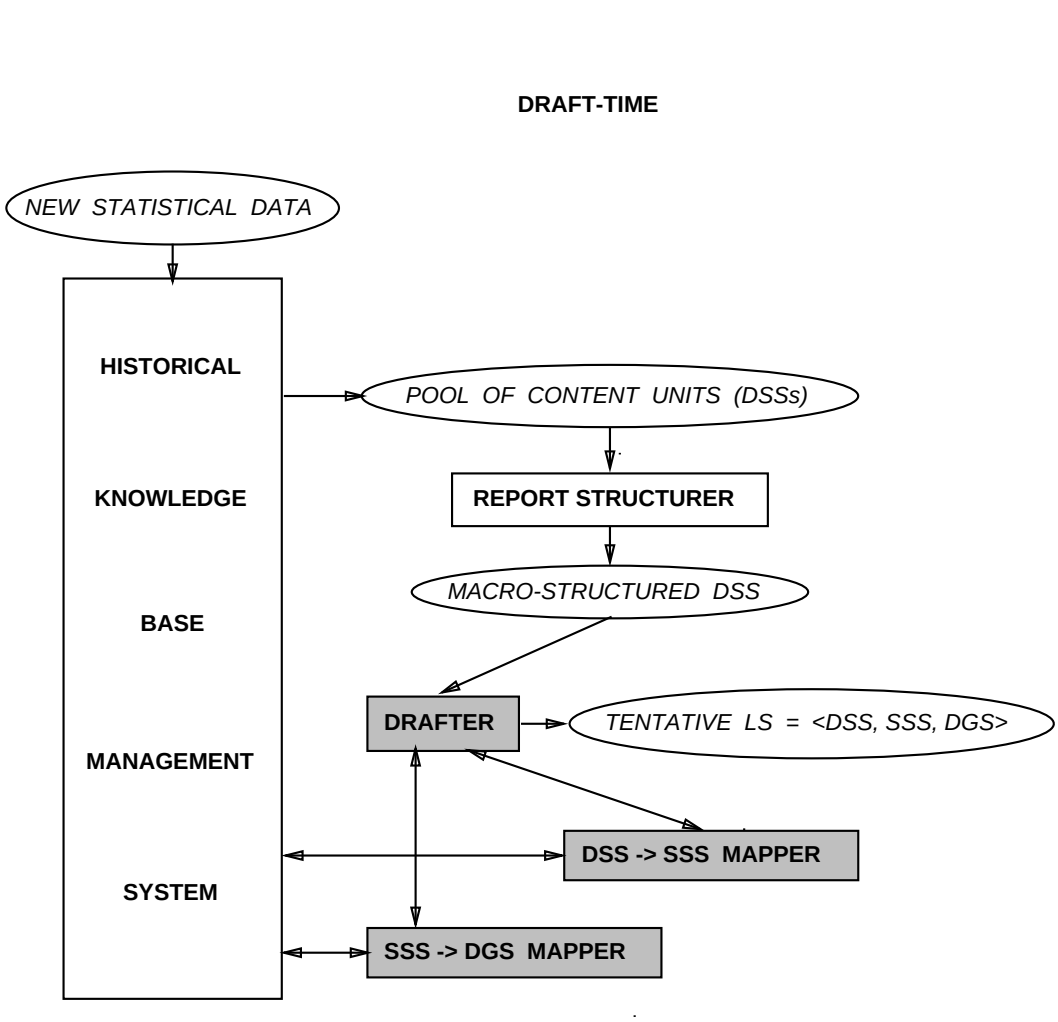

Figure 3: A draft and revision report generation architecture: draft-time

update on financial indexes). This statistical data is formatted as new individuals in a frame-based historical knowledge base. The arrival of these new individuals activates demons performing an incremental update of the knowledge base. Remarkable slots of both new and updated individuals are selected to form a pool of content units, candidates to be conveyed in the report[13].

Each element in this pool is a partial description of a knowledge base individual. Following stereotypical organization patterns, the report structurer picks some of these partial descriptions as it groups and orders them in agreement with known rhetorical patterns[14]. The result of this structuring process is a first draft

[13] This selection process is performed by the historical knowledge base management component. Since this component is out of the scope of my research, I will not address the issues raised by this selection process. I simply assume that it can be carried out, if only on largely domain-specific grounds.

[14] The report structuring component is also out of the scope of my research. I will thus not address the issues raised by the identification and operational use of these rhetorical patterns. I simply assume that it can be carried out using known techniques such as schematas [McKeown 1985] or RST planning [Hovy 1991].

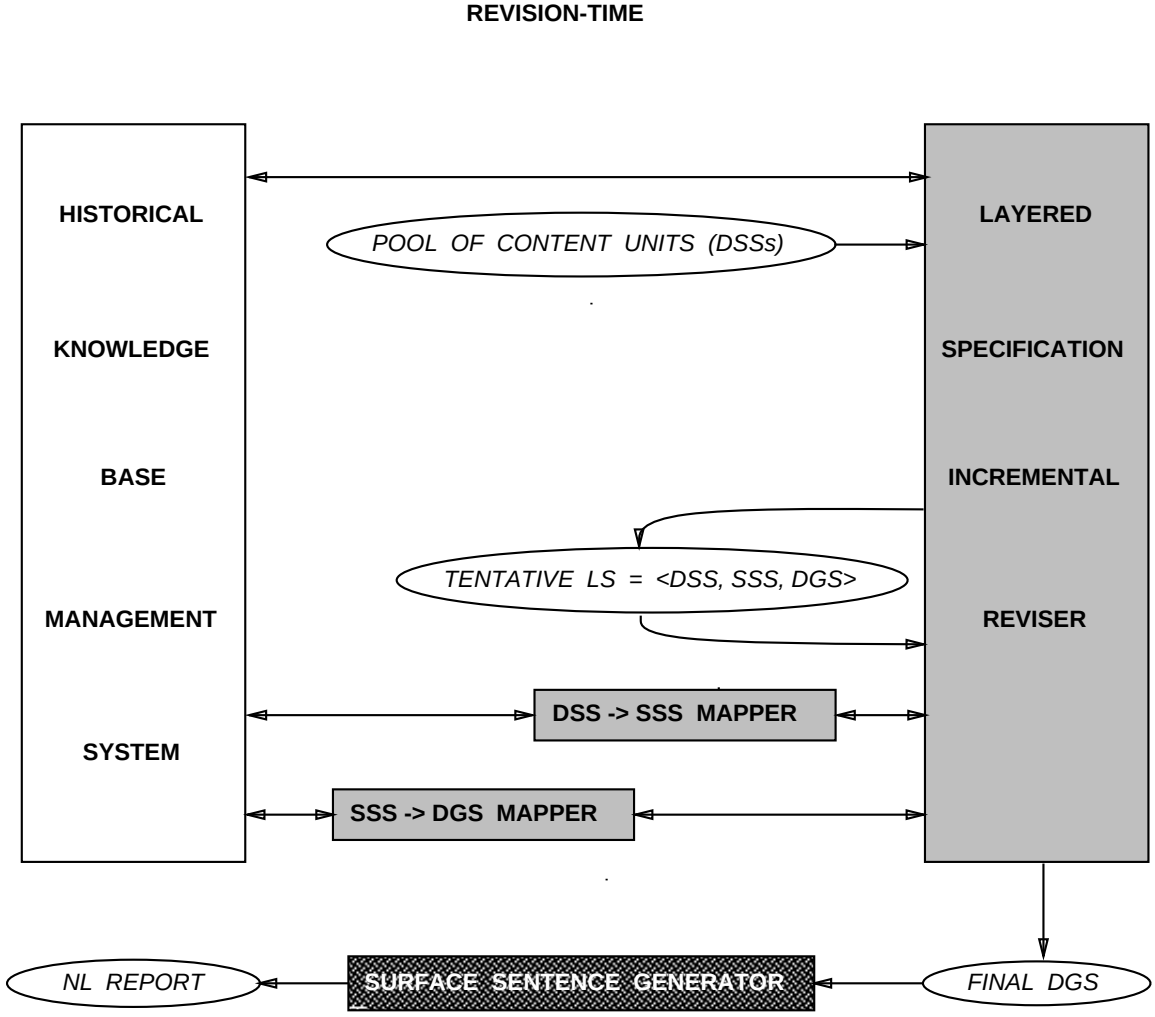

Figure 4: A draft and revision report generation architecture: revision-time

of the Deep Semantic Specification (DSS)[15].

The drafter dispatches this DSS to the DSS↦SSS mapper which returns a corresponding Surface Semantic Specification (SSS). The drafter then dispatches this SSS to the SSS↦DGS mapper which returns a corresponding Deep Grammatical Specification (DGS). This DGS is the starting point for surface sentence generation. Once it has all three levels of representation at its disposal, the drafter assembles them together in a Layered Specification (LS). This LS includes the DSS, SSS and DGS of the draft as well as the explicit correspondence between their respective constituents.

[15] The DSS as well as the other levels of representations referred to in this section are defined more precisely in the next section.

During the second pass, the reviser pulls out Additional DSSs (ADSSs) from the pool of elementary content units and includes them in the report draft by revising the LS. It attempts to integrate as many ADSSs as it can while taking into account surface constraints (such as number of words, depth of embedding, etc) in order to avoid unreadable or stylistically poor text. In order to realize the additional content units (in the context of the tentative LS) the reviser locally calls the mappers. When revision is completed, the reviser extracts the DGS part from the final LS and hands it to the surface sentence generator.

3.2 Internal representations

In this section we present in more detail the intermediate levels of utterance representation used in STREAK. All levels share the same syntax: sets of feature-value pairs. Given the fact that all components of STREAK are interpreters implemented on top of FUF, all internal representations are FUF feature structures. In FUF, the value of a feature can be: (1) an atom, (2) an embedded feature structure (recursion), (3) a path to the value of some other feature or (4) one of FUF special features. These special features include disjunctions, procedural attachments and declarative control annotations. The representations differ by the semantics of their respective features. In the following paragraphs I discuss the semantics of the different levels of representation.

3.2.1 The Deep Semantic Specification (DSS)

A Deep Semantic Specification is either a partial description of a knowledge base individual or a rhetorical structure composed of such partial descriptions. In the first case the semantics of its feature is purely encyclopedic (i.e. it captures a piece of the application domain's common-sense knowledge). This is the case with the elementary DSSs that populate STREAK's content unit pool. In the second case, in addition to encyclopedic features, the DSS also has rhetorical features recursively filled in a subDSS. This is the case with the macro-structured DSS initially produced by the report structurer. Examples of DSSs can be found under the feature DSS of the Layered Specifications of figures 5 and 6. In a DSS, pointers to knowledge

base individuals are indicated by the prefix "i-". Note that the DSS in figure 5 does not represent only the clause *"John Stockton scored 27 points"* but a whole set of synonymous expressions including *"John Stockton pumped in 27 points"*, *"John Stockton added 27"*, *"John Stockton's 27 points"*, *"a 27 points performance by John Stockton"* etc.

Layered Specification for the clause: "*John Stockton scored 27 points*"

```
'((dss ((deepsemcat game-stat-ref) (referent i-stockton-pts) (realz {^2 sss})
       (attr-props ((actor ((deepsemcat player-ref) (referent i-stockton)
                            (realz {^4 sss partic agent})
                            (id-props ((first-name "John") (last-name "Stockton")))))
                    (value ((deepsemcat simple-game-stat-val-ref)
                            (referent i-stockton-pts-val)
                            (realz {^4 sss partic created})
                            (id-props ((value 27) (unit i-pts)))))))))
  (sss ((surfsemcat evt-spec) (realz {^2 dgs}) (concept game-stat)
        (proc ((type material) (effect-type creative)))
        (partic ((agent ((surfsemcat ent-spec) (concept player)
                         (realz {^4 dgs partic agent})
                         (props ((first-name "John") (last-name "Stockton")))))
                 (created ((surfsemcat simple-quant-spec) (concept simple-game-stat-val)
                           (realz {^4 dgs partic created})
                           (props ((value 27) (unit pts)))))))))
  (dgs ((syntcat clause)
        (proc ((type material) (effect-type creative) (lex "score")))
        (partic ((agent ((syntcat person-name) (first-name ((lex "John")))
                         (last-name ((lex "Stockton")))))
                 (created ((syntcat common) (definite no) (cardinal ((value "27")))
                           (head ((lex "point")))))))))
```

Figure 5: An example of Layered Specification

3.2.2 The Surface Semantic Specification (SSS)

The features of a Surface Semantic Specification, just like those of a DSS, are encyclopedic and rhetorical. The feature AGENT and CREATED under the path {SSS PARTIC} in figure 6 are examples of encyclopedic features. The feature ELABORATION and ELABORATED under the path {SSS PARTIC CREATED} are examples of rhetorical features[16].

[16] The distinction between encyclopedic and rhetorical features made here is similar to the distinction made by Meteer [Meteer 1990] between application-program objects on one hand and both domain objects and text-structure objects on the other.

The essential difference between DSS features and SSS features is that the former are domain-dependent and conceptually motivated whereas the latter are domain-independent and linguistically motivated. Although still purely semantic the SSS is already a proto-linguistic representation. Its goal is to abstract from the *domain*. In contrast, the goal of the DSS is to abstract from *linguistic form*. One of the main reasons for such a dual semantic representation is that both these goals cannot be satisfactorily fulfilled by a single representation. This is due to the fact that, in general, a given domain concept can be expressed by completely different linguistic categories. Modelling this flexible mapping from domain concepts to linguistic categories while using linguistically motivated subcategorizations of domain concepts like PENMAN's Upper-Model [Bateman *et al.* 1990] is thus problematic.

Consider, for example, the DSS of category GAME-STAT-REF (standing for "game statistic reference") in figure 5. It is mapped onto a SSS of category EVT-SPEC (standing for "event specification"). Moreover, its ACTOR and VALUE slots have been respectively mapped onto the AGENT and CREATED roles of the SSS. This SSS will ultimately be expressed by a clause such as *"John Stockton scored 27 points"* or *"John Stockton pumped in 27 points"*. However, this DSS of category GAME-STAT-REF could also have been mapped onto an SSS of category ENT-SPEC (standing for "entity specification") with its ACTOR and VALUE slots respectively mapped onto POSSESSOR and POSSESSED roles. In this case it would have been ultimately realized by an NP such as *"John Stockton's 27 points"* or *"a 27 points performance by John Stockton"*. Therefore, this dual semantic representation allows a choice between different *surface perspectives* to refer to a domain entity. That kind of choice highlights the inherent conflicts between the tasks of (1) abstracting from the application domain and (2) abstracting from linguistic form, which the authors of the Upper-Model approach attempt to reconcile by using a single representation[17].

The other reason for having a dual semantic representation is to distinguish between implicit and explicit content. The former is inferrable from the context and represented only at the DSS level, whereas the latter is realized by some linguistic constituent and also represented at the SSS level. For similar reasons, Dale also adopted a dual semantic representation in EPICURE [Dale 1988]. This raises among other things problems

[17] Many features of the SSS are, like the Upper-Model, inspired by systemic linguistics. They are used however as a sort of "*Surface-Perspective*-Model" rather than as an "*Upper*-Model".

such as user models and hearer's knowledge, issues which I will not address here.

3.2.3 The Deep Grammatical Specification (DGS)

The Deep Grammatical Specification is the input formalism of the surface sentence generator FUF. It specifies grammatical constituency, grammatical features and open-class lexical items. It is "deep", however, in the sense that at the clause level, constituency is still expressed in terms of thematic roles such as agent, affected, attribute, location etc. It is during unification with the grammar that these thematic roles are mapped onto surface syntactic roles such as subject, object, complement, adjunct etc. When defining the set of thematic roles to be used we drew on work done by various linguists [Quirk *et al.* 1972] [Winograd 1983] [Halliday 1985] [Fawcett 1987]. Because FUF completes partially specified inputs with defaults, a DGS can be both concise and in one-to-one correspondence with an actual linguistic unit. Examples of DGS can be found under the feature DGS in the Layered Specifications of figure 5 and 6.

3.2.4 The Layered Specification (LS)

A Layered Specification is simply a triple <DSS,SSS,DGS> cumulating the three layers of representations of an utterance in STREAK. Each constituent of a given layer contains an additional feature REALZ whose value is a path leading to the constituent realizing it in the next lower layer. A FUF path is surrounded by curly brackets and the ^ notation indicates a relative path; ^ N means "go up N levels from the current level in the feature structure before following down the rest of the path". Examples of LS are given in figure 5 and 6.

3.3 Processing components

This section describes in more detail the four components of figures 3 and 4 which are the focus of my research.

Layered Specification for the clause: *"John Stockton scored A SEASON-HIGH 27 points"*

```
'((dss ((deepsemcat game-stat-ref) (referent i-stockton-pts) (realz {^2 sss})
       (attr-props
        ((actor ((deepsemcat player-ref) (referent i-stockton) (realz {^4 sss partic agent})
                 (id-props ((first-name "John") (last-name "Stockton")))))
         (value ((deepsemcat simple-game-stat-val-ref) (referent i-stockton-pts-val)
                 (realz {^4 sss partic created ELABORATED})
                 (id-props ((value 27) (unit i-pts)))))
         (CONSEQ
          ((DEEPSEMCAT HISTO-STAT-UPDATE-REF) (REFERENT I-STOCKTON-EXTR1-UPDATE)
           (ATTR-PROPS
            ((AFTER ((DEEPSEMCAT HISTO-STAT-REF) (REFERENT I-STOCKTON2-EXTR1)
                     (ATTR-PROPS ((INSTANT-GAME-STAT I-STOCKTON-PTS)
                                  (CARRIER ((DEEPSEMCAT PLAYER-TOKEN-REF)
                                            (REFERENT I-STOCKTON2)
                                            (ATTR-PROPS ((TOKEN-OF I-STOCKTON)))))
                                  (HISTO-SIGNIF
                                   ((DIRECTION I-HIGH) (MEMBERSHIP-SCOPE I-PERSONAL)
                                    (TIME-SCOPE I-SEASON)
                                    (REALZ
                                     {^8 SSS PARTIC CREATED ELABORATION}))))))))))))
  (sss ((surfsemcat evt-spec) (concept game-stat) (realz {^2 dgs})
        (proc ((type material) (effect-type creative)))
        (partic ((agent ((surfsemcat ent-spec) (concept player) (realz {^4 dgs partic agent})
                         (props ((first-name "John") (last-name "Stockton")))))
                 (created ((SURFSEMCAT PROPOSITION-CLUSTER)
                           (RHETOR-RELT ((SEMANTICS ELABORATION) (SURFSTRUCT HYPOTAXIS)))
                           (MAIN {^ ELABORATED}) (DEPENDT {^ ELABORATION})
                           (ELABORATED ((surfsemcat simple-quant-spec)
                                        (concept simple-game-stat-val)
                                        (realz {^5 dgs partic created HEAD})
                                        (props ((value 27) (unit pts)))))
                           (ELABORATION ((SURFSEMCAT COMPOUND-QUAL-SPEC)
                                         (CONCEPT GAME-STAT-HISTO-SIGNIF)
                                         (REALZ {^5 DGS PARTIC CREATED CLASSIFIER})
                                         (PROPS ((DIRECTION HIGH)
                                                 (TIME-SCOPE SEASON)))))))))
  (dgs ((syntcat clause) (proc ((type material) (effect-type creative) (lex "score")))
        (partic ((agent ((syntcat person-name) (first-name ((lex "John")))
                         (last-name ((lex  "Stockton")))))
                 (created ((SYNTCAT COMMON) (CLASSIFIER ((SYNTCAT COMMON) (DEFINITE NO)
                                                         (CLASSIFIER ((lex "SEASON")))
                                                         (HEAD ((lex "HIGH")))))
                           (HEAD ((syntcat common) (definite no) (cardinal ((lex "27")))
                                  (head ((lex "point")))))))))))
```

Figure 6: An example of revised Layered Specification

3.3.1 The Drafter

In input, the drafter receives a macro-structured DSS from the report structurer. It first calls the DSS↦SSS mapper on this macro-structured DSS and obtains a corresponding SSS. It then calls the SSS↦DGS mapper on this SSS and obtains a corresponding DGS. Finally, it assembles these three levels of representations (DSS, SSS and DGS) into an initial Layered Specification (LS).

3.3.2 The Mappers

Taking in input a feature structure at a given representation layer, a mapper builds a corresponding feature structure at the next representation layer. This mapping is carried out by local application of rewrite rules. The Left Hand Side (LHS) is a feature structure which partially specifies a constituent of the input layer. The Right Hand Side (RHS) is a feature structure which partially specifies a constituent of the output layer.

The input feature structure is first traversed recursively top-to-bottom in order to identify the constituents. Each one of these constituents is rewritten into a corresponding output constituent. The output constituents are then assembled bottom-up. The rewriting of a constituent is carried out in two steps. The first step consists of unifying the constituent with the LHSs of the rewriting rule base. The RHSs of all the matching rules are then unified together to form the output constituent.

The advantage of this approach is that it allows the specification of very modular rules applicable across constituency ranks. The drawback of this bottom-up only approach is that it does not account for top-down constraints in an elegant way. In order to prevent this problem, I intend to use FUF's choice delaying mechanism. A FUF feature structure can include explicit control annotations expressing the fact that a given alternative can not be considered before another problem is solved (ordering constraints). Upon encountering such an annotation, FUF will delay the processing of the alternative until the needed information has become available. Using these annotations top-down and bottom-up processing can be combined.

Another special feature of FUF, procedural attachment, is used in the mapping rules to give the mappers

access to the historical knowledge base[18] during the rewriting process. If the value of some feature is essential to decide how to map a particular kind of knowledge base individual, but is not needed by most other types of individual, it will not be included in the generic input description of an individual but instead be given a procedural attachment value.

When such a value is encountered, unification is suspended, the procedural attachment gathers the necessary information from the knowledge base and then unification resumes. This mechanism allows the use of internal representations restricted to *standard* features of *general* necessity. It thus relieves the server components from the burden having complete knowledge of each one of their client component needs. Such a facility is essential for circumscribing knowledge of a given kind inside a single component and in order to achieve genuine modularization.

To illustrate this point, consider an example from the basketball domain. Statistics of two players are often aggregated in conjunctive constructs. In the general case, two such statistics differ in either value or unit and this difference is used by the mapper to decide which one to topicalize. However, in the rare case when both statistics have the same value and the same unit, another criterion needs to be used. I observed in the corpus that, in this situation, it is usually the statistic of the player with the highest season point average that is topicalized. So, for instance, if Karl Malone's season average is 29.5 points while John Stockton's is just 18.1, the sentence *"Karl Malone scored 27 points and John Stockton added 27"* will be preferred over *"John Stockton scored 27 points and Karl Malone added 27"*. The important point here is that this season average information is *neither* to be conveyed in the sentence *nor* needed in general for topicalization or any other generation decision. Therefore it should not be part of the generic player description. However, in this particular situation, it is needed to choose the sentence topic.

The use of FUF's choice delaying mechanism and procedural attachments is discussed in detail in [Elhadad and Robin 1992].

[18] Procedural attachment is also used in reviser rules to give the reviser access to the historical knowledge base.

3.3.3 The Reviser

In its full complexity, incrementally revising the tentative LS for an entire report to incorporate Additional DSSs (ADSSs) from a pool of content units raises the following issues:

- Revision scheduling: Choosing in what order to consider the ADSSs in the pool for revision.
- Hook localization: For a given ADSS, exploring the tentative LS to identify the various locations where it can be hooked.
- Hook choice: If several such locations exist, choosing one.
- Tool choice: For a given ADSS and a given base DSS as the chosen hook, choosing among the various ways to revise the base so that it incorporates the ADSS.

To illustrate these issues consider the text of figure 2 as the report draft and the content units realized by boldfaced constituents in the text of figure 1 as the elaboration pool. Revision scheduling involves choosing to consider for inclusion Denver's streak information before Sacramento's. Hook localization involves identifying the first and last sentences as candidate hooks for Sacramento's streak information. Hook choice involves choosing the last sentence over the first one. Tool choice involves choosing to postmodify the reference *"the Kings"* by the relative clause *"who lost their fourth straight"* over the nominal apposition *"losers of four in a row"*.

In order to focus on the issue of tool choice, STREAK's reviser design is based on two simplifying assumptions concerning its input: (1) the ADSSs candidate for inclusion are ordered in a priority stack and (2) the draft represented by the tentative LS consists of a single sentence[19].

The reviser consists of an interpreter of revision rules. The LHS of a revision rule has two fields: an LS pattern and an ADSS pattern. The RHS is a list of revision operations. Each operation is an elementary structural manipulation on the LS. During one revision increment, the ADSS on top of the stack and the

[19] Note that since such a sentence can be quite complex, the issues of hook localization and hook choice are not totally obviated by these assumptions. However, at the present stage I have only considered sentences containing a single candidate hook for a given ADSS.

current LS are grouped into a feature structure. That feature structure is unified with the LHSs of the revision rule base. The revision actions contained in the RHS of the matching rule[20] are executed in order.

The kind of manipulation performed by revision actions during a revision increment is illustrated in figure 5 and figure 6. Figure 5 contains the LS for the hook sentence: *"John Stockton scored 27 points"*. This sentence is revised in order to convey also the historical fact that John Stockton had never scored that many points before. Figure 6 contains the LS for the revised sentence: *"John Stockton scored* **a career high** *27 points"*.

In that figure, the new features resulting from the revision are written in uppercase. The uppercased portion of the DSS is the input representation of the additional historical content unit. This representation and the LS of figure 5 are the input of the revision increment. The output of that increment is the LS of figure 6. By comparing figure 5 with figure 6 one can see that three types of revision actions occur during a revision increment: (1) displacement of base constituents, (2) creation and insertion of new constituents and (3) update of the REALZ features that keeps track of the interlayer constituent correspondence.

The new constituent at the DSS layer is realized by new constituent(s) at the SSS layer in turn realized by new constituent(s) at the DGS layer. These realizations are carried out by the mappers called from the reviser. After the call, the reviser inserts the created constituent at the appropriate location in the LS. So for example in figure 6 the constituent under the path {DGS PARTIC CREATED CLASSIFIER} was obtained by the reviser by calling the SSS$\mapsto$DGS mapper on the constituent under the path {SSS PARTIC CREATED ELABORATION}.

A revision increment can be characterized by the structural manipulations that the LS has gone through. A class of such structural manipulations can be viewed as a *revision tool* available in a given language to hook-up a given type of additional information on a given type of linguistic construct. The following parameters play an important role in the structural characterization of a revision tool: (1) the grammatical category of the hook constituent, (2) the grammatical category of the constituent realizing the additional

[20] The issue of multiple rule matches has not yet been addressed. The current rule LHSs are probing both the ADSS and the LS in a way that is specific enough so that only one rule can exactly match a given <ADSS,LS> pair. If no rule matches, the LS is left untouched and the next candidate ADSS is popped from the stack.

content unit and (3) the structural relation between the hook constituent and the additional constituent in the revised constituent. Moreover, a given tool can be used at various depths of embedding in the base sentence.

In the revised example given here above the grammatical category of the hook constituent - *"27 points"* - is an NP, the grammatical category of the added constituent - **"a career high"** is also an NP and the structural relation between the two is hypotaxis with the hook as head. The revision tool used is thus nominal-level adjunction of an NP. It is used at the first level of embedding with regards to the base sentence.

Sentences (1)-(6) in figure 7 illustrate the structural diversity of the revision tools available to hook-up the same piece of information on the same base linguistic construct. Each one of them corresponds to a different revision of sentence (0) incorporating the same additional streak information about the losing team. In these sentences the hook constituent is written in uppercase and the added constituent is written in bold uppercase.

Sentences (4) and (5) show that the same revision tool can be used with hooks embedded at different levels. Nominal-apposition is applied both on the hook constituent - "*127 111 victory over the Denver Nuggets*" - in sentence (5) and on the more embedded hook constituent - "*the Denver Nuggets*" - in sentence (6). This is also the case for nominal-level adjunction of relative clause in sentences 3 and 4 respectively.

Sentences (1)-(3) and (5) show that the incorporation of the added constituent affects the surface form of the hook. This is why these transformations are complete *revisions* of the base sentence and not simple elaborating insertions. In these sentences the reference *"the Denver Nuggets"* is reduced to just *"Denver"* due to the nearby presence of another reference to that team in the added constituent.

In all the examples of figure 7 the structural relation between the hook and the added constituent is either parataxis (sentences (1), (5) and (6)) or hypotaxis with the hook as the head (sentences (2)-(4)). However, some revision tools involve hypotaxis *with the added constituent as head.* Although this type of tool cannot be applied on base sentence (0) of figure 7, it *can* be applied to the synonymous, yet structurally different sentence:

Revision of the base sentence:
(0) *"David Robinson scored 32 points lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets."*
to add the historical fact that the Nuggets had also lost their six previous games using different tools:

- Clause-level coordination applied at depth one:
  (1) *"David Robinson scored 32 points Friday night LIFTING THE SAN ANTONIO SPURS TO A 127 111 VICTORY OVER DENVER* **AND HANDING THE NUGGETS THEIR SEVENTH STRAIGHT LOSS"**.
- Nominal-level adjunction of non-finite clause applied at depth two:
  (2) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to A 127 111 VICTORY OVER DENVER* **SENDING THE NUGGETS TO THEIR SEVENTH STRAIGHT LOSS**."
- Nominal-level adjunction of relative clause applied at depth two:
  (3) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to A 127 111 VICTORY OVER DENVER* **THAT EXTENDED THE NUGGETS' LOSING STREAK TO SEVEN GAMES**."
- Nominal-level adjunction of relative clause applied at depth three:
  (4) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over THE DENVER NUGGETS* **WHO LOST FOR THE SEVENTH CONSECUTIVE TIME**".
- Nominal-level apposition applied at depth two:
  (5) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to A 127 111 VICTORY OVER DENVER,* **THE NUGGETS' SEVENTH STRAIGHT DEFEAT**".
- Nominal-level apposition applied at depth three:
  (6) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over THE DENVER NUGGETS,* **LOSERS OF SEVEN IN A ROW**".

Figure 7: The diversity of revision tools

(0') *"David Robinson scored 32 points and the San Antonio Spurs rolled to a 127 111 victory over the Denver Nuggets"*,

yielding the revised sentence:

(1') *"David Robinson scored 32 points and the San Antonio Spurs* **EXTENDED DENVER'S LOSING STREAK TO SEVEN WITH** *A 127 111 VICTORY OVER THE NUGGETS"*.

In this revised sentence the NP of the base sentence expressing the game result - *"a 127 111 victory over the Nuggets"* - has been *absorbed* by the added constituent expressing the historical consequence of this result.

A detailed structural classification of such information-adding revision tools as well as some constraints on their respective usage in the corpus is presented elsewhere [Robin 1992]. This classification and these

constraints form the basis of the reviser rule base.

3.4 Architectural issues revisited

In this section I review the proposed architecture in terms of the vocabulary defined in section 2.1. I show that it has the necessary properties for report generation sensitive to the historical context defined in section 2.2. I then suggests its applicability to a wider range of natural language generation applications.

The text generation subtasks defined in section 2.1 are distributed among the various components of the architecture presented in the three previous sections as follows:

- *Content production* is carried out by the Historical Knowledge Base Management System (HKBMS)
- *Content selection* is carried out in three rounds. In the first round encyclopedic constraints are taken into account (this is done by the HKBMS). In the next round macro-level organization constraints are taken into account (this is done by the report structurer). In the last round, both micro-level organization and surface realization constraints are taken into account (this is done by the reviser).
- *Macro-level content organization* is carried out by the report structurer.
- *Micro-level content organization* is carried out by the DSS$\mapsto$SSS mapper. It thus occurs in two rounds: a first round when this mapper is called from the drafter and a second round when its is called from the reviser.
- *Semantic grammaticalization* is carried out in part by the DSS$\mapsto$SSS mapper and in part by the SSS$\mapsto$DGS mapper. It also occurs in two rounds, one at draft-time and one at revision-time.
- *Lexicalization* is carried out by SSS$\mapsto$DGS mapper and also occurs in two rounds, one at draft-time and one at revision-time.
- *Morpho-syntactic grammaticalization and linearization* is carried out *after* revision is completed by the surface sentence generator FUF.

Now recall the four criteria laid down in section 2.2. for report generation architecture sensitive to the historical context. The first one was that content selection, organization and realization should be, at least in part, mutually constrained. It is met because the third round of content selection is performed in conjunction with the second rounds of micro-level content organization and of content realization during revision. The second was that part of micro-level content selection and organization should be carried out non-locally. It is met because the second rounds of micro-level content organization and realization is performed by the reviser *which has access to the global representation of the draft.* The third was that cross-ranking content organization choices should be handled down to the micro-level. It is met because (1) the dual level of semantic representation handles realization perspectives and (2) rhetorical aggregation below the clause level is represented in the SSS[21]. Finally, the fourth was that content realization should be microcoded using a word-based lexicon. It is met because the SSS↦DGS mapper recursively goes down to the individual word level if necessary. Because it is based on unification, this SSS↦DGS mapping allows for flexibly combining the use of phrasal and individual word entries as shown by Smadja and McKeown [Smadja and McKeown 1991].

Although the proposed architecture was conceived to answer the specific needs of historical context sensitive report generation, its scope of applicability is wider. Basically, it is well-suited for any application with the following properties:

- The content to convey[22] can be divided into a core content predominantly influencing macro-level organization and a more peripheral content predominantly influencing micro-level organization.
- The peripheral content units are numerous and there is no sharp distinction among their respective encyclopedic relevance.
- The peripheral content units can be alternatively realized by micro-level constituents which are much less constrained in terms of their global rhetorical position in the text than in terms of their local realization environment.

[21] Such a micro-level aggregation, can also be represented by Meteer's Text-Structure [Meteer 1990], but not by the RST [Mann and Thompson 1987], [Hovy 1990] in its present form.

[22] Whether provided by the underlying application program or produced by the generator.

It is also a cognitively plausible model for the generation of complex and information-dense sentences. An interesting experiment carried out by Pavard [Pavard 1985] suggests that humans also need revision in order to generate such sentences. It also suggests that they can handle *core* information in one-shot and need revision specifically to incorporate *peripheral* information. In this experiment, a group of people was asked to paraphrase by a single sentence the content expressed in three sentences. The group was subdivided into subgroups, each having at its disposal a different type of medium. The group using a medium not allowing for any revision (e.g. microphone) failed to pack into their single sentence an average 34% of the content conveyed by circumstantials (i.e. peripheral content) in the original three sentences. In contrast, an average of 4% of such circumstantial omission was scored by the group using a medium allowing many revisions (e.g. text editor). The difference in average omissions for content conveyed by participants (i.e. core content) in the original three sentences was minimal: 4% without revision and 0% with it. Together, the three sentences used in that experiment expressed eight propositions in 42 words[23]. The people who took part in the experiment were thus asked to generate a single sentence of complexity comparable to opening sentences in my corpus.

Finally, the proposed architecture also has a unique *combination* of properties whose desirability has been *independently* supported in previous work:

- It interleaves planning and realization [Appelt 1985] [Hovy 1988]
- It is incremental [De Smedt 1990]
- It is stratificational and modular [Polguere 1990]
- It uses declarative knowledge sources [Nogier 1990] [Polguere 1990]
- It uses a uniform underlying formalism (FUF's feature structures) to represent utterances at all levels of abstraction and all linguistic ranks [Simonin 1985] [Dale 1988]
- It distinguishes between foreground content to convey obligatorily and background content to convey opportunistically [Rubinoff 1990]

[23] In French which is more verbose than English.

- It lexicalizes reference to a domain entity with access to both (a) a set of features to convey about the entity in order to satisfy the current communicative goals, and (b) the complete description of this entity in the domain knowledge base [Reiter 1991].
- It works in a two-pass draft and revison mode.

I focus on this last characteristic in the next section.

4 Revision

The idea of revision in generation is not really new. Actually, it has been proposed in different flavors by Mann [Mann 1983], Meteer [Vaughan and McDonald 1986] [Meteer 1990], Yazdani [Yazdani 1987], Gabriel [Gabriel 1988], Wong and Simmons [Wong and Simmons 1988] Cline and Nutter [Cline and Nutter 1991] and very recently [Inui *et al.* 1992]. However, to the best of my knowledge, only two implemented generation system emerged from these proposals: Gabriel's YH and Inui *et. al*'s WEIVER.

The types of revision proposed by these authors differ from that carried out in STREAK in terms of both: (1) the type of representation that is revised (at what level to revise?) and (2) the goals of the revision (why revise?). I compare revision in STREAK with previous proposals with respect to each of these two terms in the two next sections.

4.1 At what level to perform revision?

Both Yazdani and Meteer (at least in her initial paper [Vaughan and McDonald 1986]) proposed performing revision from an actual natural language draft. At that level, revision is a three-step process: (1) rediscovering the generator's output anew by interpreting it using a text-understanding system, (2) evaluating this output by matching its interpretation with the communicative goals that underlied its generation and (3) regeneration. The problem with this proposal is that the development of a text understanding system

remains in itself such a tremendous endeavor that it cannot realistically be contemplated as a substep in the development of a revision component for generation, at least for generation-only applications like report generation[24]. This is why the other authors proposed, like I do, to perform revision directly on some representation of the generator's internal draft. This is also perhaps why in her thesis [Meteer 1990], after having dismissed revision of an internal representation as mere "optimization" and opposed it to "true revision" requiring analyzing an actual natural language output, Meteer then proposes to perform revision on the Text-Structure, a level of representation internal to her SPOKESMAN generation system[25].

However, Meteer's distinction between "optimization" and "true revision" is pertinent because unless an internal representation has some special characteristics, revising it may end up being closer to backtracking[26] and plan-elaboration during single-pass generation than it is to the draft reviewing and editing phase in two-pass generation. In my view, revising an internal representation is clearly distinct from backtracking and/or plan elaboration if that representation has the following characteristics: (1) it includes a layer "surfacey" enough to univocally determine a natural language utterance and (2) it includes all unretracted intermediate decisions that ultimately led to that surface layer.

The first characteristic distinguishes draft revision from plan-elaboration because the latter is performed on a purely conceptual representation in one-to-many mapping with various natural language utterances. The second characteristic distinguishes draft revision from local backtracking because *the whole spectrum* of generation decision is affected at once by a revision[27]. The Layered Specification (LS) in the architecture proposed in section three has both these characteristics. Its surface element, the Deep Grammatical Specification (DGS) univocally determines a natural language output[28]. Moreover all its unretracted intermediate decisions are compiled in the Deep Semantic Specification (DSS) and the Surface Semantic Specification (SSS) layers.

[24] Using an existing natural language *understanding* system is also not an option because each of such systems works only for its very specific domain, sublanguage and application. Although some cross-domain robustness has been achieved by some *parsing* systems, their output is merely a partial parse tree annotated with some standard semantic features, which is far from the kind of encyclopedic and discursive *interpretation* needed to guide revision.

[25] Note that this is a proposal for future work. SPOKESMAN does *not* perform revision.

[26] Of course, every form of revision *is indeed* a form of *global* backtracking while searching the N-dimensional space of all generation alternatives. It is important though to distinguish draft revision from *local* forms of backtracking restricted to a given class of generation choices.

[27] In that sense the hill-climbing phase of KDS [Mann and Moore 1981] is not revision but local backtracking.

[28] thanks to the defaults provided by the FUF's English grammar for unspecified features.

Apart from distinguishing information-adding revision from plan-elaboration, the presence of a surface layer in the representation on which revision is performed is needed for another reason. By nature, report generation is a summarization task. It thus involves trade-offs between inherently conflicting goals[29]: (1) informativity, (2) conciseness and (3) readability. But factors like conciseness and readability directly depend on surface form and monitoring them cannot be done by reasoning only at higher layers. In that sense Meteer's Text-Structure remains too abstract. Although grammatical constituency is already decided at that level, many grammatical features and open-class lexical items with different stylistic impacts are not yet specified. The level of representation that univocally determines a natural language utterance in SPOKESMAN is the Linguistic Specification input to MUMBLE-86 [Meteer *et al.* 1987]. Using SPOKESMAN levels of representation, revising to improve the draft trade-off between informativity, conciseness and readability, would probably require acting upon both the Text-Structure level and the Linguistic Specification level.

Without being specific on the characteristics of the representation to revise, Cline and Nutter also stressed the need that such a representation must reflect both content planning and surface realization decisions. Wong and Simmons advocate performing revision using a blackboard. Although they are not specific about what type of information is to be posted on the blackboard during revision, the inherent flexibility of such mechanism would certainly allow manipulations simultaneously affecting several layers of decisions.

4.2 Revising to satisfy what goals?

The revision goals considered by Meteer, Gabriel, Wong and Simmons and Inui *et al.* are purely stylistic. They are *information-preserving* revisions to make the draft more concise, clearer or more coherent. In contrast, STREAK performs *information-adding* revisions to make the draft more informative. Cline and Nutter propose performing both varieties of revisions. However, the type of information-adding revision they propose - expanding during the revision some unexpanded nodes of the textual schemas used for content

[29] By multiplying the number of candidate facts to convey, dealing with historical information only exacerbates the tension between these conflicting goals.

planning during the first pass[30] - operates at the *macro*-level (addition of a sentence or a paragraph) whereas those performed by STREAK operates at the *micro*-level.

We have already seen how *information-adding* revision in STREAK differs from plan-elaboration. The incremental *revision* in STREAK also differs from incremental *first-pass generation*, taking place for example in Gabriel's YH. The former operates on a pre-existing macro-structure prescriptively built by the report structurer during the first pass while the latter starts from scratch and its macro-structure progressively emerges from locally planned and realized additions. Although providing for maximum flexibility, this type of incremental addition from scratch is underconstrained and may lead to the exploration of a very large search space[31].

5 Conclusion and future work

By analyzing a corpus of human-generated reports I have shown that making report generators sensitive to the historical context would broaden their semantic coverage and make their output look much more natural. I have also shown that such sensitivity to the historical context cannot be achieved using the pipelined architecture of existing report generators. That is why I suggested a new architecture based on revision. In this architecture a draft specification which organizes and realizes mostly new information is incrementally revised to incorporate, among other things, historical information. When implementing STREAK using this architecture I have made three simplifying assumptions concerning the reviser's task:

1. The only type of revision considered is *information-adding* revision.
2. The order in which to attempt hooking-up additional candidate content units onto the draft specification is provided as input to the reviser.
3. The draft specification organizes and realizes only the first sentence of the report.

[30] Or in Hovy's term, expanding *growth-points* [Hovy 1990].
[31] Gabriel avoided this problem by using knowledge extremely specific to the tiny domain of the Dutch National Flag game.

These assumptions allowed me to focus on the issue of revision tool identification. That is, determining for a given type of additional content unit and a given type of hook sentence, what are the various ways to merge them into a revised sentence. I am currently switching the focus to tool choice. That is, determining in what contextual circumstances to prefer one tool rather than another. Such preferences can be incorporated as constraints in the Left Hand Side of the revision rules. A possible agenda for future work consists of removing the simplifying assumptions one by one. Each removal brings about interesting issues to investigate.

Removing the single sentence assumption brings about the issues of hook localization and hook choice. If the draft specification is multi-sentential, localizing a hook requires carrying out some form of structure traversal of the draft specification. This draft specification being a tri-layered structure with non-isomorphic layers in the general case, such a traversal is a non-trivial task. Moreover if several hooks are identified during the traversal the reviser has to choose one. Finding criteria on which to base such a choice is another interesting issue.

Note that the issues of hook localization and hook choice in the information-adding revision paradigm proposed here closely correspond to the issue of suggested growth-point consideration in the hybrid[32] RST planning paradigm [Hovy 1991]. In both cases, a prescriptively built macro-structure is incrementally elaborated. The main difference between the two paradigms is that information-adding revision can be constrained upon surface realization factors. In that sense, it is probably more flexible and less underconstrained than hybrid RST planning[33]. Another interesting aspect of the information-adding revision paradigm proposed here is that it addresses *micro-level* planning. By considering relations between *clauses* as its basis, RST has limited itself to *macro-level* planning up to now. What is instead needed is to consider relations between *propositions* independently of the linguistic rank at which they will eventually be realized (word, group, clause, sentence etc). In particular, identifying a set of general semantic and rhetoric relations between subconstituents inside the NP and comparing them with RST relations is a much needed step towards a harmonious integration of micro-level and macro-level planning.

[32] i.e. combining top-down planning and open-ended planning.

[33] Both paradigms can be combined. The report structurer in figure 3 can be based on prescriptive top-down RST planning and RST relations can be used as rhetorical features in the Surface Semantic Specification.

Removing the preordered revision assumption raises the issue of revision scheduling. That is, in what order to consider additional content units in the pool of candidate information? And also what to do with content units which were scheduled at some point during the incremental revision process but did not fit anywhere in the draft at that moment? Should they be rescheduled later on and if so when? Note that this issue also has its counterpart in hybrid RST planning: suggested growth points scheduling. Again the advantage of handling it at revision time is that surface form constraints can then be called upon. Another issue related to revision scheduling is when to stop revising? In the single sentence case, simple corpus-based stylistic constraints limiting sentence complexity can be used as criteria to stop revision. In the multi-sentential case, other factors like the need to alternate between simple and complex sentences would have to be taken into account as well.

By removing the single revision goal assumption, the reviser in the proposed architecture could pursue other goals aside from enhancing informativity. One such goal could be enhancing conciseness and style by using the information-conserving revisions inventoried by Meteer [Vaughan and McDonald 1986], [Meteer 1990] and/or by postponing aggregations [Hovy 1990] factorizing common semantic elements until revision-time, like YH. Another such goal could be enhancing textual cohesion [Halliday 1985]. This goal could be pursued by choosing abridged forms of subsequent references [Granville 1984] [Dale 1988] at revision-time and/or by revising lexical choice [Robin 1990] for more numerous cohesive lexical affinities.

Pursuing such a variety of goals during revision would enhance the overall quality of the generated report. However, some of these goals being conflicting (e.g. informativity vs. conciseness), it would also make the issues of scheduling revisions and deciding when to halt much harder. Criteria to insure that the revision process converges would need to be defined.

Acknowledgments

Many thanks to Michael Elhadad, David Kurlander, Kathy McKeown, Frank Smadja and Michael Zock for their feedback on early drafts of this paper. The reported research was partially supported by ONR grant N00014-89-J-1782, DARPA grant N00039-84-C-0165, NSF grant IRT-84-51438, CAT grant 4-62397 and a

grant from General Electric.

References

[Appelt 1985] D. Appelt. *Planning Natural Language Utterances.* Studies in Natural Language Processing. Cambridge University Press, 1985.

[Bateman *et al.* 1990] J.A. Bateman, R.T. Kasper, J.D. Moore, and R.A. Whitney. A general organization of knowledge for natural language processing: the PENMAN Upper-Model. Technical report, ISI, Marina del Rey, CA, 1990.

[Bourbeau *et al.* 1990] L. Bourbeau, D. Carcagno, E. Goldberg, R. Kittredge, and A. Polguere. Bilingual generation of weather forecasts in an operations environment. In *Proceedings of the 13th International Conference on Computational Linguistics.* COLING, 1990.

[Cline and Nutter 1991] B.E. Cline and J.T. Nutter. Conceptual revision for natural language generation, 1991. Student session of the 29th Annual Meeting of the ACL.

[Dale 1988] R. Dale. *Generating referring expressions in a domain of objects and processes.* PhD thesis, University of Edinburgh, Scotland, 1988.

[Danlos 1986] L. Danlos. *The linguistic basis of text generation.* Studies in Natural Language Processing. Cambridge University Press, 1986.

[De Smedt 1990] K.J.M.J. De Smedt. *Incremental sentence generation: a computer model of grammatical encoding.* Samson-Sijthoff, The Netherlands, 1990.

[Elhadad and Robin 1992] M. Elhadad and J. Robin. Controlling Content Realization with Functional Unification Grammars. In R. Dale, H. Hovy, D. Roesner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, pages 89–104. Springer Verlag, 1992.

[Elhadad 1990] M. Elhadad. Types in Functional Unification Grammars. In *Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics*, Detroit, MI, 1990. ACL.

[Elhadad 1991] M. Elhadad. FUF: The universal unifier - user manual, version 5.0. Technical Report CUCS-038-91, Columbia University, 1991.

[Fawcett 1987] R.P. Fawcett. The semantics of clause and verb for relational processes in English. In M.A.K. Halliday and R.P. Fawcett, editors, *New developments in systemic linguistics*. Frances Pinter, London and New York, 1987.

[Fensch 1988] T. Fensch. *The sports writing handbook*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1988.

[Gabriel 1988] R. Gabriel. Deliberate writing. In D.D. McDonald and Bolc L., editors, *Natural Language Generation Systems*. Springer-Verlag, New York, NY, 1988.

[Granville 1984] R. Granville. Controlling lexical substitution in computer text generation. In *Proceedings of the 10th International Conference on Computational Linguistics*. COLING, 1984.

[Halliday 1985] M.A.K. Halliday. *An introduction to functional grammar*. Edward Arnold, London, 1985.

[Hovy 1988] E. Hovy. *Generating natural language under pragmatic constraints*. L. Erlbaum Associates, Hillsdale, N.J., 1988.

[Hovy 1990] E. Hovy. Unresolved issues in paragraph planning. In R. Dale, C.S. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*. Academic Press, 1990.

[Hovy 1991] E. Hovy. Approaches to the planning of coherent text. In C. Paris, W. Swartout, and Mann. W.C., editors, *Natutal Language Generation in Artificial Intelligence and Computational Linguistics*. Kluwer Academic Publishers, 1991.

[Inui *et al.* 1992] K. Inui, T. Tokunaga, and H. Tanaka. Text revision: a model and its implementation. In R. Dale, E. Hovy, D. Roesner, and O. Stock, editors, *Aspects of Automated Natural Language Generation*, pages 215–230. Springler-Verlag, 1992.

[Jacobs 1985] P. Jacobs. PHRED: a generator for natural language interfaces. *Computational Linguistics*, 11(4):219–242, 1985.

[Kukich 1983] K. Kukich. *Knowledge-based report generation: a knowledge engineering approach to natural language report generation.* PhD thesis, University of Pittsburgh, 1983.

[Mann and Moore 1981] W.C. Mann and J.A. Moore. Computer Generation of Multiparagraph English Text. *Computational Linguistics*, 7(1):17–29, 1981.

[Mann and Thompson 1987] W.C. Mann and S. Thompson. Rhetorical Structure Theory: description and constructions of text structures. In Gerard Kempen, editor, *Natural Language Generation: New Results in Artificial Intellligence, Psychology and Linguistics*, pages 85–96. Martinus Ninjhoff Publishers, 1987.

[Mann 1983] W.C. Mann. An overview of the PENMAN text generation system. Technical Report ISI/RR-83-114, ISI, Marina del Rey, CA, 1983.

[Maybury 1990] M.T. Maybury. Using discourse focus, temporal focus and spatial focus to generate multisentential text. In *Proceedings of the 5th International Workshop on Natural Language Generation*, Pittsburgh, PA, 1990.

[McKeown *et al.* 1990] K. R. McKeown, M. Elhadad, Y. Fukumoto, J.G. Lim, C. Lombardi, J. Robin, and F.A. Smadja. Text generation in COMET. In R. Dale, C.S. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation.* Academic Press, 1990.

[McKeown 1985] K. R. McKeown. *Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Studies in Natural Language Processing. Cambridge University Press, 1985.

[Mencher 1984] M. Mencher. *News reporting and writing.* Wm. C. Brown Publishers, Dubuque, Iowa, 1984.

[Meteer *et al.* 1987] M.W. Meteer, D.D. McDonald, S.D. Anderson, D. Forster, L.S. Gay, A.K. Huettner, and P. Sibun. Mumble-86: Design and implementation. Technical Report COINS 87-87, University of Massachussets at Amherst, Ahmerst, Ma., 1987.

[Meteer 1990] M.W. Meteer. *The generation gap: the problem of expressibility in text planning.* PhD thesis, University of Massachussets at Ahmerst, 1990. Also available as BBN technical report No. 7347.

[Nogier 1990] J.F. Nogier. *Un systeme de production de language fonde sur le modele des graphes conceptuels.* PhD thesis, Universite de Paris VII, 1990.

[Pavard 1985] B. Pavard. La conception de systemes de traitement de texte. *Intellectica*, 1(1):37–67, 1985.

[Polguere 1990] A. Polguere. *Structuration et mise en jeu procedurale d'un modele linguistique declaratif dans un cadre de generation de texte.* PhD thesis, Universite de Montreal, Quebec, Canada, 1990.

[Quirk *et al.* 1972] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A grammar of contemporary English.* Longman, 1972.

[Reiter 1991] E.B. Reiter. A New Model for Lexical Choice for Open-Class Words. *Computational Intelligence*, (7), December 1991.

[Resnick *et al.* 1990] L.A. Resnick, A. Borgida, R.J. Brachman, D.L. McGuiness, and Patel-Schneider R.F. *CLASSIC: description and reference manual for the COMMON LISP implementation version 1.02*, 1990.

[Robin 1990] J. Robin. Lexical Choice in Natural Language Generation. Technical Report CUCS-040-90, Columbia University, 1990.

[Robin 1992] J. Robin. Generating Newswire Report Leads with Historical Information: a Draft and Revision Approach, 1992. PhD. Thesis Proposal, Computer Science Department, Columbia University.

[Roth *et al.* 1988] S. Roth, J. Mattis, and X. Mesnard. Graphics and natural language as component of automatic explanation. In *Proceedings of the ACM SIGCHI Workshop on Architectures for Intelligent Interfaces*, pages 109–128, Monterey, 1988.

[Rubinoff 1990] R. Rubinoff. Natural Language Generation as an Intelligent Activity, 1990. PhD. Thesis Proposal, Computer Science Department, University of Pennsylvania.

[Simonin 1985] N. Simonin. Essai de modelisation de l'expertise en redaction de textes. In *Proceedings of COGNITIVA 85.* COGNITIVA, 1985.

[Smadja and McKeown 1991] F.A. Smadja and K.R. McKeown. Using collocations for language generation. *Computational Intelligence*, (7):229–239, December 1991.

[Talmy 1985] L. Talmy. Lexicalization patterns: semantic structure in lexical form. In T. Shopen, editor, *Grammatical categories and the lexicon*, volume 3 of *Language typology and syntactic description.* Cambridge University Press, 1985.

[Vaughan and McDonald 1986] M. Vaughan and D.D. McDonald. A Model of Revision in Natural Language Generation. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, Columbia University, New York, 1986. ACL.

[Winograd 1983] T. Winograd. *Language as a cognitive process.* Addison-Wesley, 1983.

[Wong and Simmons 1988] W.K.C. Wong and R.F Simmons. A blackboard model for text production with revision. In *Proceedings of the AAAI workshop on text-planning and realization*, St-Paul, MN, 1988. AAAI.

[Yazdani 1987] M. Yazdani. Reviewing as a component of the text generation process. In Gerard Kempen, editor, *Natural Language Generation: New Results in Artificial Intellligence, Psychology and Linguistics.* Martinus Ninjhoff Publishers, 1987.

[Zock 1988] M. Zock. Natural languages are flexible tools, that's what makes them hard to explain, to learn and to use. In M. Zock and G. Sabah, editors, *Advances in Natural Language Generation: an Interdisciplinary Perspective.* Pinter and Ablex, 1988.

Contents