Generating Newswire Report Leads with Historical Information: a Draft and Revision Approach

Jacques Robin

(PhD. Thesis Proposal)

Technical Report
CUCS-042-92

Department of Computer Science
Columbia University
New York, NY 10027

September 9, 1992

Abstract

In this paper I investigate the issue of providing historical background in computer-generated reports. I first observe that ignoring this issue is the most drastic limitation of existing report generation systems. I then present an empirical corpus analysis of basketball summaries aimed at discovering the specific means by which historical information is conveyed in human-generated reports. This analysis resulted in a set of data that forms the basis for the implementation of STREAK, a system generating basketball report leads with historical information. This data shows that building such a system cannot be based on the single-pass pipelined architecture of previous systems. Instead, I propose an entirely new architecture in which generation proceeds in two passes. The first pass builds a report draft containing only the basic facts. The second pass incrementally revises this draft to include additional facts providing the historical background. Independently of the issue of historical information, I show that this architecture also allows the generation of more complex and syntactically diverse sentences than previous generation systems. I also explain how the information-adding revision framework defined by this architecture differs from previous work in generation with revision. I then delimit the additional research I propose to pursue within this framework. I conclude by showing how the proposed work will make several significant contributions to four natural language generation topics: report generation, complex sentence generation, generation architecture and generation with revision.

1 Introduction

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. Two existing systems have established the feasibility of report generation from statistics[1]: ANA [Kukich 1983] and FoG [Bourbeau *et al.* 1990]. ANA summarizes the daily fluctuations of several stock market indexes from half-hourly updates of their values. A report generated by ANA is given in Fig. 1. In a similar vein, but averaging data along spatial dimensions in addition to time, FoG produces daily local marine weather bulletins from meteorological measurements. A report generated by FoG is given in Fig. 2.

Comparing the reports generated by these two systems with reports produced by human writers reveals that their most drastic limitation is that 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. In contrast, the lifetime of *new information* 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)[2]. Thus, existing report generators are not able to put the facts and events they relate into historical perspective.

In contrast, human writers systematically present new information in combination with related historical information. The basketball game report presented in Fig. 3 illustrates this point[3]. **Historical information** is shown in this report with a boldface font. Note the overall proportion of historical information and its presence in every paragraph. Historical information provides the reader with clues to assess the significance of the reported facts. Consider, for example, the opening sentence. It contains two pieces of historical information. The first, expressed by the adjective *"career-high"*, stresses the remarkable scoring performance of Adams by noting that it is the best of his entire career. The second, expressed by the clause *"end a five-game losing streak"*, puts the game outcome into perspective by recalling that Denver had lost its five preceding games.

The general importance of historical information is emphasized in Fig. 4 where the report of Fig. 3 has been stripped off its historical information[4]. The drastic qualitative difference between both reports shows that historical information is needed for good semantic coverage and realistic output. It is also needed for pertinently selecting what new information to include in the report. The relevance of a new fact is in general largely determined by its historical significance. The following excerpt from another basketball game report illustrates this point:
"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 an 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.

The overall goal of the research proposed in this paper is to build a report generation system providing the historical background for the facts and events it relates. In order to discover the specific means by which historical information is conveyed, I carried out an empirical analysis of a corpus of basketball game reports

[1] Danlos' generator [Danlos 1986], PAULINE [Hovy 1988] and TEXPLAN [Maybury 1990] are other generation systems producing "reports". However, they work in non-quantitative domains and take as input frame-like structures instead of statistics. The system SAGE suggests the great potential for office-automation applications of quantitative report generation. However, focusing on the issue of coordinating textual and graphical media, this system relies on ad-hoc techniques for text generation (cf. [Roth *et al.* 1991], p.216).

[2] The mention of the big board volume of the preceding day in the report of Fig. 1 is *not* historical information. It refers to the *initial* volume of the current day which coincides to the *final* volume of the preceding business day. The scope of this report is indeed limited to the current day. In all fairness, ANA does not *totally* ignore historical information: ANA can generate sentences noting an all-time extremum (i.e. an all-time high or an all-time low) for a financial index. ANA detects such extrema by having the all-time high and low values of each index encoded in its content determination rules. This treatment of historical information is *ad-hoc*: everytime a record is broken, these values need to be updated by hand in the rules in which they are used. Moreover, all-time highs and lows are but a very restricted type of historical information.

[3] This report was taken from the UPI newswire.

[4] In the report of Fig. 4, the new facts whose wording had to be changed due to the suppression of nearby historical facts are showed in a smallcap fonts. Similarly, the new facts which have been displaced are in a slanted font.

Thursday June 24, 1982
wall street's securities markets meandered upward through most of the morning, before being pushed downhill late in the day yesterday. the stock market closed out the day with a small loss and turned in a mixed showing in moderate trading.
the Dow Jones average of 30 industrials declined slightly, finishing the day at 810.41, off 2.76 points. the transportation and utility indicators edged higher.
volume on the big board was 558,600,000 shares compared with 627,100,000 shares on Wednesday. advances were ahead by about 8 to 7 at the final bell.

Figure 1: An example report generated by Ana

Frobisher-bay
Gale warning issued...
Winds light becoming southeasterly 15 early Tuesday morning then backing and strengthening to easterly 30 Tuesday afternoon then strengthening to northeasterly gales 35 Tuesday evening. Mostly cloudy with snow, fog and mist patches. Visibility fair in snow, fair in mist and poor in fog. Outlook for Wednesday ... gale force northeasterlies becoming gale force northerlies.

Figure 2: An example report generated by FoG

Sacramento, Ca. – 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 3: A human-generated report conveying historical information.

Sacramento, Ca. – Michael Adams scored 44 points Wednesday night, including seven 3-point baskets, to help 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 4: The report of Fig.3 stripped off its historical information.

similar to that of Fig. 3. This data forms the basis for the on-going implementation of a prototype system, called STREAK (Surface Text Revision Expressing Additional Knowledge), to generate basketball report leads with historical facts. Both aspects of this research, corpora-analysis and prototype implementation, support and complete each other. The implementation of STREAK provides an operational testbed for the hypothesis of the corpus analysis while the corpora analysis results provide criteria for the architecture of STREAK and data for its knowledge sources.

The corpus analysis and its results are presented in detail in section 2 of this paper. Two general observations about the way historical information is conveyed in the corpus provide the initial motivation for the other aspects of the proposed research:

- Historical information is associated with new information in very complex sentences.
- Historical information is conveyed by a wide variety of syntactic constructs.

The sentence complexity associated with historical information is highlighted in the table below. This table compares the complexity of the corpus sentences with historical information with the maximum sentence complexity generated by existing systems.

	Previous systems				Corpus sentences with historical information		
	FoG	Ana	Danlos	PAULINE			
	max	max	max	max	min	avg	max
Proposition number per sentences	4	4	7	7	5	6.5	9
Word number per sentences	10	28	33	34	23	34	46

The syntactic diversity associated with historical information is illustrated by the sentences of Fig. 5. Sentences (1) is an original corpus sentence including an historical fact emphasized by a boldface font. Sentences (2)-(6) are paraphrases of (1) in which different syntactic constructs are used to convey this historical fact. Each of these constructs was used in other corpus sentences to convey this type of fact.

The ability to flexibly convey historical background thus entails the ability to generate a wide variety of very complex sentence structures. This double demand of syntactic complexity and diversity has profound consequences on the type of generation architecture needed for conveying historical information. Previous work in generation has focused on either one of these demands while ignoring the other.

Generators focusing on syntactic complexity, (e.g., ANA, PAULINE and Danlos' system) all resort to a *macro*coding approach to sentence generation. With this approach, sentences are produced by assembling entire phrasal patterns pre-defined in a phrasal lexicon. Macrocoding thus obviates the need for planning any syntactic structure below the sentence rank. Its drawback is that it does not scale-up in the face of great syntactic diversity. Far too many phrasal patterns need to be hand-coded in the lexicon to achieve such diversity (cf. [Smadja and McKeown 1991] for a detailed discussion of this problem).

In contrast, this diversity can be economically achieved by a *micro*coding approach to sentence generation, where syntactic structures at all ranks are constructed from individual words[5]. Generators focusing on syntactic diversity (e.g., the system of [Boyer and Lapalme 1985] and COOK [Smadja 1991]) perform such micro-level planning. They have, however, limited themselves to the generation of simple sentences.

Simultaneously achieving surface form complexity and surface form diversity requires microcoding complex sentences down to the individual words when necessary. This task involves making a much larger number of decisions than any existing systems make for the generation of a single sentence. I this paper, I propose a *draft and revision approach* to report generation which decomposes this overall task into modular substeps, each involving a more manageable number of decisions. This approach consists of organizing a report draft around the new information and then incrementally revising this draft to add related historical information.

Consider again the example sentences of Fig. 5. The draft and revision approach I propose would decompose their generation into three steps. The first step would consist of producing a draft sentence

[5] Because of non-compositional phenomena like collocations and idioms, some phrasal entries remain necessary, even with a microcoding approach.

Variety of syntactic constructs to convey the same additional historical fact:

- Clause-complex coordinative conjoin:
  (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**".

- Adjoin of non-finite clause in top-level nominal:
  (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**".

- Adjoin of relative clause in top-level nominal:
  (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**".

- Adjoin of relative clause in embedded nominal:
  (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**".

- Top-level nominal appositive conjoin:
  (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**".

- Embedded nominal appositive conjoin:
  (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 5: Syntactic diversity of historical information

(temporarily) ignoring the historical fact. The sentences of Fig. 5, differing only in the way the historical fact is conveyed, would all be generated from the same draft sentence:
(0) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets"*.
The second step would consist of choosing a constituent in this draft sentence onto which to attach the historical fact. This constituent is uppercased in each of the example sentences of Fig. 5. Finally, the third step would consist of choosing both a syntactic category for the historical fact and a syntactic relation attaching it to the chosen constituent[6]. This decomposition in three steps is possible because the decisions taken in each of these steps are largely independent. Note, for example, that while in both (3) and (4) the historical fact is realized by way of a relative clause, this clause is attached to a draft sentence NP which is located at a different embedding level in (3) than it is in (4). Similarly, attachment by an apposition occurs at different embedding levels in (5) and (6).

The fact that it takes advantage of the compositionality of natural language to decompose the microcoding of a complex sentence into simpler subtasks is only one of several motivations for the draft and revision approach advocated in this paper. Other observations of the way historical information in the corpus further motivate this approach. These observations are discussed in detail in section 3.2 of this paper. One of them is that *historical information is predominantly conveyed by small syntactic constituents scattered throughout the report*. This observation is illustrated by the report of Fig. 3. All the syntactic constructs conveying historical information in this report (i.e. those emphasized with a boldface font) are embedded constituents, ranging from the clause rank down to the individual word rank. The historical facts thus appear to have

[6] These last two choices are grouped in a single step because they are mutually constrained by syntax.

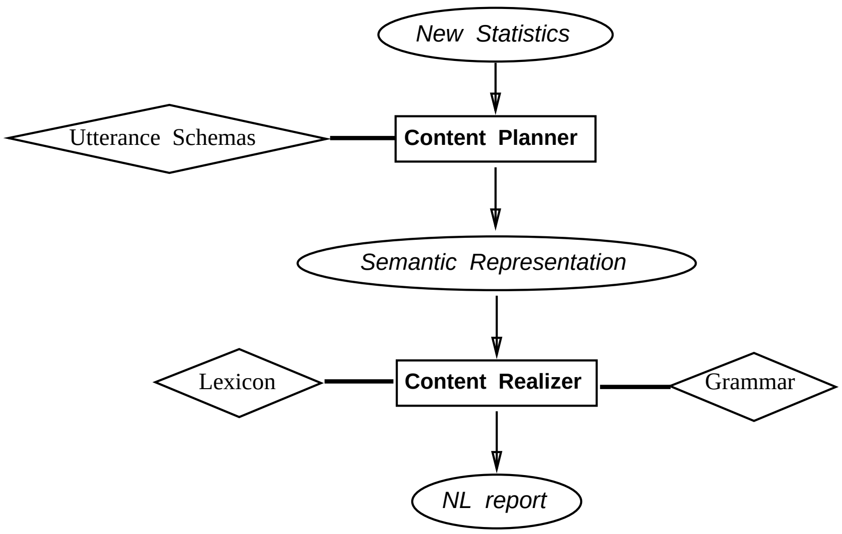

Figure 6: Classic one-pass pipeline generation architecture

been *opportunistically added* as modifiers of pre-existing syntactic structures conveying related new facts.

To be able to convey historical information in such an opportunistic way, a generation system needs a new architecture. Existing report generators are all based on the classic pipeline architecture shown in Fig. 6. With this classic architecture the content of each sentence is structured as a whole all at once. Fig. 7 gives a first approximation of the new draft and revision architecture I propose. At the heart of this new architecture lays a dedicated revision component. This component allows the *incremental* planning of complex sentence structures by incorporating additional propositions on a basic draft structure in successive steps. In section 3.3 of this paper, I present in detail a more sophisticated revision-based architecture, in which not only content planning but also content realization is incrementally performed by successive draft revisions. It is on this more elaborate architecture that the prototype system STREAK is based. I discuss the on-going implementation of STREAK in section 4 of this paper.

I carried out the last part of the corpus analysis with the draft and revision approach underlying the implementation of STREAK in mind. I systematically paired syntactic structures whose respective content differ by only one proposition and viewed the simplest structure in the pair as draft material for the more complex and informative one. This differential analysis allowed me to identify a set of *revision tools* to incrementally generate sentences of increasing complexity and informativity. These tools are characterized by the structural transformations involved in going from the simple structure to the more complex one. Each sentence of Fig. 5 corresponds to the application of a particular revision tool on sentence (0) above. A classification of these tools is presented in section 2.5.2 along with some constraints on their respective applicability.

These tools perform information-*adding* revisions improving the draft's informativity. This type of revision, which brings together generation with revision and incremental generation, is entirely new. Previous work in generation with revision considered only information-*preserving* revisions improving the draft's conciseness and style. This previous work is presented and compared in more detail with the revisions performed by STREAK in section 5 of this paper. In section 6, I delimit the dissertation work by describing the additional work needed to complete the proposed research.

In this introduction I have first noted that a crucial step towards reducing the current qualitative gap between computer-generated reports and their human-generated counterpart is to provide report generators with the ability to convey historical information. By analyzing a corpus of human-generated reports, I then observed that flexibly conveying historical information entails generating a wide variety of very complex sentences, something that has not been attempted by previous work in generation. In order to breakdown the overall complexity of this task, I proposed a two-pass approach to generation that first structures a

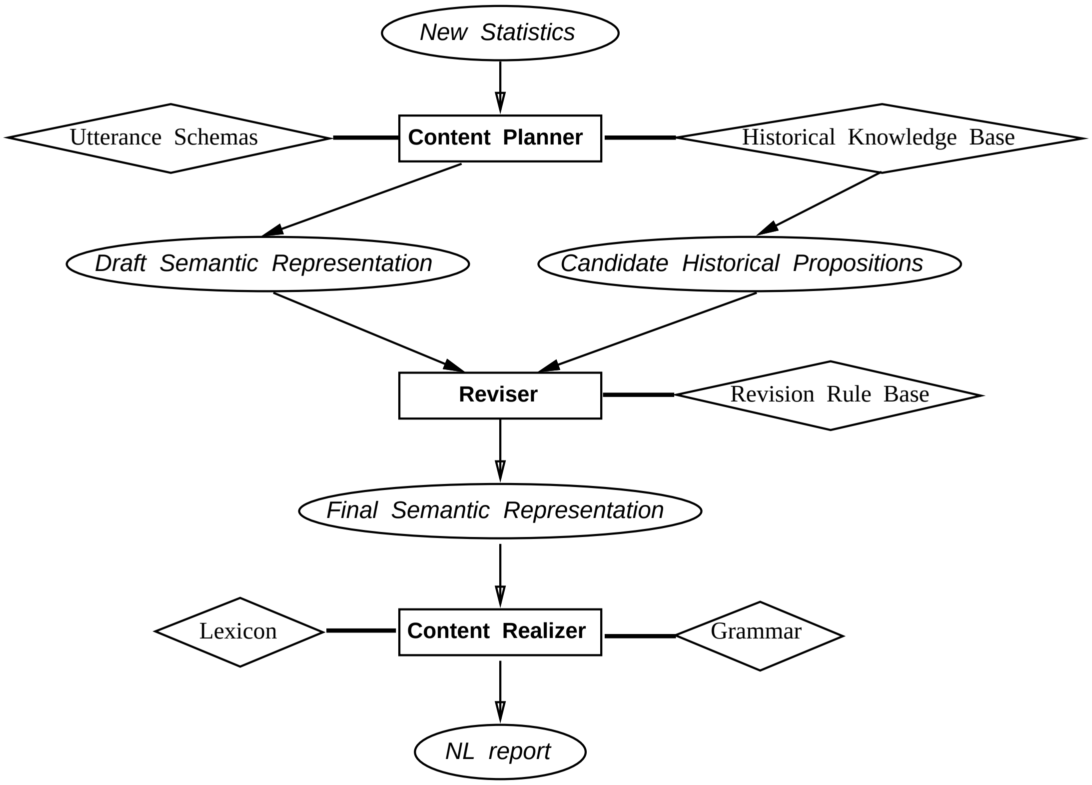

Figure 7: First approximation of the proposed draft and revision architecture

draft around basic facts, and then incrementally revises this draft to incorporate related historical facts. In addition to addressing three original issues - historical background, sentence complexity and syntactic diversity below the clause rank - this approach is also innovative in two other ways. First, it defines a new generation system architecture that fully exploits the compositionality of natural language. Second, it defines a new application for revision: incremental generation. The research proposed in this paper will thus make contributions in four distinct subfields of generation research: report generation, complex sentence generation, generation architecture and generation with revision. These contributions are discussed in detail in section 7 of this paper.

2 The data: a corpus analysis of basketball game summaries

2.1 Goals and methodology

In this section I present a corpus analysis of newswire reports summarizing basketball games[7]. An example report from this corpus was given in Fig. 3 of section 1. Another example is given in Fig. 8 in section 2.2. The corpus consisted of 833 reports covering an entire professional basketball season. Almost all of them contained some historical information.

The overall goal of this corpus analysis was threefold:

- Identify the set of linguistic resources available to convey specifically historical information.
- Provide criteria for a generation architecture able to flexibly convey historical information.
- Provide data for the knowledge sources of a prototype system based on such an architecture.

The core of this analysis involved distinguishing the different types of information expressed in the corpus, and for each of these types, the linguistic forms used for its expression. Such a *semantic* analysis can be carried out only empirically by hand[8]. It was decomposed in five successive steps, each with its particular goal.

The first step consisted of focusing on a restricted category of corpus sentences. This restricted category defined both a subcorpus for systematic and in-depth analysis and a realistic target output for the prototype system STREAK. Sorting the corpus sentences pertaining to the subcorpus was carried out semi-automatically using free-text information retrieval tools described in [Smadja 1991].

The second step consisted of defining the ontology of the subcorpus. It involved identifying the different types of entities mentioned in the corpus[9] as well as the different types of propositions expressed about these entities. The resulting ontology served as the basis for both the implementation of STREAK's encyclopedic knowledge base and the third step of the analysis.

This third step consisted of identifying the semantic constraints on the combination of different ontological types of propositions inside the corpus sentence constituents. These constraints can be viewed as schemata for sentence planning. They will be integrated with other constraints in the revision rules of STREAK.

The fourth step consisted of identifying the various syntactic forms used in the corpus to express each combination of proposition types. It resulted in a set of mappings from semantic to syntactic structures. These mappings, which I call *realization patterns*, define an upper-bound for the paraphrasing power and grammatical coverage of STREAK's realization component.

Finally, the fifth step consisted of identifying a set of revision tools to attach additional information to basic sentence structures. It was carried out through a differential analysis of the realization patterns identified in step four. For this differential analysis, the realization patterns of each proposition combination was compared with those of its semantic increments[10]. These revision tools together with constraints on their respective applicability will be integrated in the revision rules of STREAK.

These five steps are presented in order in the following subsections.

---

[7] These reports have been taken from the UPI newswire.

[8] I use corpus analysis as a methodology for semantic and syntactic knowledge acquisition. My work thus pertains to the *knowledge-based* tradition of NLP research and *not* to the probabilisitic tradition of NLP research that has become glossed as "corpus-based NLP" in its most recent revival. Probabilistic NLP attempts to build systems without any knowledge but which instead rely on correlation between their input and occurrence probabilities computed on huge textual corpora.

[9] Where there is no risk of ambiguity, I will refer to the *sub*corpus simply as "the corpus".

[10] A semantic increment of a proposition combination C is any corpus-occurring proposition combination containing C plus one additional proposition.

2.2 Focusing on a subcorpus

In order to fulfill its goals, the corpus analysis needed to be simultaneously semantic, syntactic, very fine-grained[11] and systematic. Such an ambitious analysis could not have been carried out on the total corpus, *i.e.,* on each and every sentence of 833 reports. It was necessary to focus on a restricted category of corpus sentence. I defined this restricted category in two steps. I first made observations about the structure of the corpus reports to choose a *discursive* focus. I then distinguished between different types of information conveyed in the corpus to choose a *semantic* focus.

Report structure and discursive focus Journalism textbooks define a variety of standard report structures. The corpus reports are organized following the so-called *inverted pyramid structure with summary lead* [Fensch 1988], meaning that crucial information is packed in the lead followed by facts of decreasing importance towards the end of the report. Summary type leads, often consisting in a single sentence in the corpus reports, are thus self-contained mini-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 is preferred because newswire reports are essentially used as draft material to be edited by client newspapers under heavy time-pressure and stringent space constraints. An inverted pyramid structure with a summary lead makes a report instantly editable by cutting its tail. When space is really scarce only the lead remains. This particular structure of the corpus reports allowed me to narrow the discursive scope of the analysis to a single sentence per report: the opening sentence. This opening sentence contained historical information in 52% of the 833 corpus reports. That historical information was part of the essential facts in a majority of reports confirms the importance of historical information.

Proposition classes and semantic focus In order to define a semantic focus for the fine-grained analysis, I first had to carry out a high-level semantic analysis of *all* 833 opening sentences in the corpus. This high-level analysis consisted of distinguishing different types of information conveyed in these sentences. Among these different types I singled out one particular type of new information, *box-score statistic* and two particular types of historical information, *extrema* and *streaks*.

A box-score is a table containing a standard set of statistics for one game. In daily newspapers, there is one box-score accompanying each game report. An example box-score with the corresponding report is given in Fig. 8. In this report, fine-grained statistics *not* available in the box-score are emphasized by an italic font (historical information is, as usual, emphasized by a boldface font). Such finer grained statistics come from complete game charts [Anderson 1985]. Box-scores are available on-line through newswire and sport statistic computer services, whereas game charts are not[12]. Therefore, only box-scores constitute a realistic statistical input for a report generation system. This is the first reason to choose box-score statistics as the semantic focus for new information. There are two other reasons:

- Box-score statistics are the most common type of new information. It was the sole type of new information in 50.2% of the corpus sentences.[13].
- Box-score statistics are the type of new information most often associated with historical information. 65% of the corpus sentences in which box-score statistics was the sole type of new information also contained historical information. Only 39% of the first sentences containing new information *other* than box-score statistics also contained historical information.

An extremum is the property of a statistic to constitute a maximum or a minimum in a set of similar statistics over some period of time, e.g., *"Michael Adams scored* **a career-high** *44 points"*

A streak is the property of the game result to extend or interrupt a series of similar results for a given team, e.g., *"the Denver Nuggets* **ended a five-game losing streak with** *a 128-112 victory over the Sacramento Kings"*

[11] i.e., down to the individual word whenever necessary

[12] To the best of my knowledge.

[13] Note also in Fig. 8 that most of the information conveyed in the report comes directly from the box-score.

ORLANDO MAGIC					24	24 (**48**)		21 (69)		25 (**94**)	
		fg	**3pt-fg**	**ft**	**rb**						
Players	**mn**	**m-a**	**m-a**	**m-a**	**o-t**	**bl**	**st**	**as**	**to**	**pf**	**tp**
Catledge	44	8-15	0-0	0-0	4-11	0	3	3	2	3	16
Reynolds	27	3-9	1-1	2-2	0-2	1	1	1	1	3	9
Roberts	16	4-7	0-0	1-2	2-6	2	0	1	1	5	9
Vincent	22	4-12	0-0	0-0	3-6	0	1	3	4	0	8
Anderson	30	6-10	0-0	2-4	1-2	0	1	1	0	2	14
Kite	22	2-4	0-0	0-0	3-4	0	0	0	0	2	4
O.Smith	26	4-11	0-0	1-2	0-1	0	2	1	0	2	9
Skiles	26	8-15	0-3	5-5	1-2	0	**4**	**7**	1	3	**21**
Scott	14	1-7	0-1	2-2	0-0	0	0	0	1	1	4
Je. Turner	4	0-1	0-0	0-0	0-1	0	0	1	0	3	0
Williams	9	0-1	0-0	0-0	0-1	0	0	0	0	0	0
Team totals	240	40-92	1-5	13-17	14-**36**	2	12	18	12	24	**94**
Team %		**.435**	.200	.765							

HOUSTON ROCKETS					20	25 (**45**)		31 (76)		23 (**99**)	
		fg	**3pt-fg**	**ft**	**rb**						
Players	**mn**	**m-a**	**m-a**	**m-a**	**o-t**	**bl**	**st**	**as**	**to**	**pf**	**tp**
Johnson	38	4-10	0-1	2-4	2-8	1	2	4	1	3	10
Thorpe	40	5-11	0-0	4-4	5-**10**	0	0	2	1	3	**14**
Olajuwon	33	6-16	0-0	5-6	3-**14**	**8**	2	4	3	**6**	**17**
Maxwell	42	**4-14**	3-9	3-4	0-3	0	1	5	3	3	**14**
K.Smith	38	**12-15**	1-1	3-3	1-4	0	2	**9**	6	0	**28**
Rollins	14	0-0	0-0	0-0	1-2	1	0	0	0	2	0
Floyd	19	2-5	1-2	7-8	0-3	0	0	4	4	2	12
Herrera	8	1-3	0-0	2-4	0-1	0	0	0	1	2	4
Jo. Turner	5	0-0	0-0	0-0	0-1	0	0	0	1	2	0
Bullard	3	0-0	0-0	0-0	0-0	0	0	0	0	0	0
Team totals	240	34-74	5-13	26-33	12-**46**	10	7	28	20	21	**99**
Team %		.549	.385	.788							

ORLANDO, Fla. (UPI) – Kenny Smith scored 28 points Sunday night to pace the Houston Rockets to a 99-94 victory over Orlando, **giving the Magic their league-high 10th straight loss**.
Hakeem Olajuwon contributed 17 points, 14 rebounds and eight blocked shots before fouling out *with 3:15 remaining in the game*.
The Magic led 48-45 at halftime, *but Houston outscored Orlando 23-12 in the first eight minutes of the third quarter to take the lead for good*.
Smith converted 12 of 15 shots from the field and dished out nine assists **to give the Rockets their sixth win in eight meetings with Orlando**.
Scott Skiles provided spark of the bench with 21 points, seven assists and four steals for the Magic, **which lost for the 14th time in 15 games**.
Otis Thorpe added 14 points and 10 rebounds for the Rockets. Vernon Maxwell scored 14 points despite 4-for-14 shooting from the field.
Orlando was outrebounded 46-36 and shot just 43.5-percent from the field. **The Magic have dropped their last six home games**.

Font conventions in the report: *italics = information not coming from the box-score*, **boldface = historical information**.
Font conventions in the box-score: **boldface = statistic included in the report**.
Abbreviations in the box-score : mn = MiNutes-played, fg = Field-Goals, ft = Free-Throws, rb = ReBounds, as = ASsists, st = STeals, bs = Blocked Shots, to = TurnOver, pf = Personal Foul, tp = Total Points, 3pt = three-PoinT, m = Made, a = Attempted, o = Offensive, t = Total..

Figure 8: Standard box-score and corresponding natural language report

There are two reasons for choosing extrema and streaks as the semantic focus for historical information:

- They can be easily maintained from box-scores (they are also available on-line through sports statistic computer services).
- They are the most common type of historical information (together, extrema and streaks were the sole type of historical information in 62% of the first sentences).

Focusing on box-score statistics, extrema and streaks defines a subcorpus of 293 opening sentences for fine-grained semantic and syntactic analysis. It also defines the subdomain for the prototype system STREAK. The ontology of this subdomain is presented in section 2.3.

2.3 Domain ontology

The semantic subdomain defined in the previous subsection can be modeled by one hierarchy of argument classes and one hierarchy of predicate classes. The argument classes represent the different types of entities mentioned in the corpus while the predicate classes represent the different types of propositions conveyed about these entities. The argument and predicate hierarchies are presented in detail in appendix A. Together they decompose the domain propositions into 42 classes. The present subsection just intuitively defines the minimum subset of predicate classes necessary to understand the corpus sentence semantic analysis presented in the next subsections. A hierarchy of this subset of classes is shown in Fig. 9.

Recall from section 2.2 that the subdomain includes only one type of new information: box-score propositions. These propositions can concern either individual players, entire teams, or the game as a whole.

Propositions about players specialize as:

- Statistics of a winning player[14], **st(WP)**.
- Statistics of a winning player group, **st(WPG)**.
- Statistics of a losing player, **st(LP)**.
- Attribute of a winning Player, **attr(WP)**, such as the fact that he is a reserve.

The opening sentences contained only one type of propositions concerning an entire team, statistics of the winning team, **st(WT)**, and four types of propositions concerning the game as a whole:

- Result of the game, **res(G)**, specifying who were the teams participating, who won and on what score.
- Location of the game, **loc(G)**.
- Day of the game, **day(G)**.
- Other attributes of the game, **attr(G)**, such as the fact that it went overtime.

The subdomain includes two main types of historical information, extrema and streaks. Extrema specialize into:

- Extrema of winning player statistic, **ex(st(WP))**.
- Extrema of game result, **ex(res(G))**[15].
- Extrema of streak, **ex(sk)**.

Streaks of game results were the only type of streaks **sk** of the subdomain. Streaks of player or team statistics were never conveyed in opening sentence of the corpus reports.

[14] i.e. a player from the team that won the game.
[15] More precisely, of the game score.

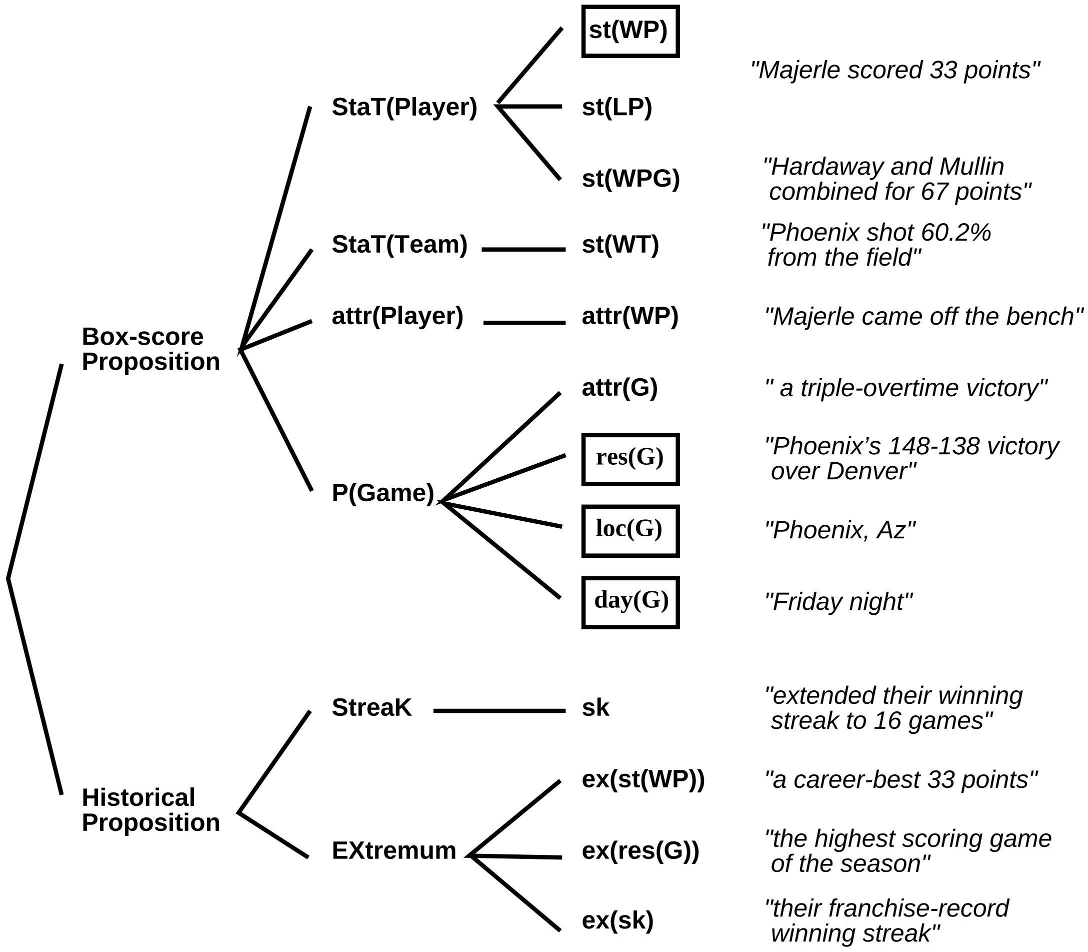

Figure 9: Domain ontology

2.4 Corpus sentence structures

2.4.1 Top-level structure: proposition clustering

The semantic structure of corpus sentences is defined by constraints on proposition co-occurrence, *i.e.,* what proposition combinations appear inside particular sentence constituents. Their surface structure is defined by constraints on syntactic dependency between the constituents realizing the various elements of each appearing proposition combination.

I made two main observations on proposition co-occurrence inside whole corpus sentences. The first observation is that the number of propositions in these sentences vary from a minimum of four to a maximum of nine. The second observation is that each sentence contains one instance of the four following proposition classes: game result **res(G)**, game location **loc(G)**, game day **day(G)** and winning player statistic **st(WP)**. A game has only one result, location and day. In contrast, most corpus sentences contain several propositions of the class winning player statistic. However, only one of these winning player statistics is the focus of the sentence. I call this particular statistic the *prominent statistic* of the proposition combination.

The conclusion of these two observations is that any proposition combination appearing in a corpus sentence is made of:

- The *basic* proposition combination: <prominent statistic, game result, game location, game day>.
- From zero to five *additional* propositions from the ten remaining propositions classes in the domain

ontology presented in section 2.3

At the abstract proposition class level, all corpus sentences are thus semantic supersets of the basic, four proposition sentences.

To discover the constraints on syntactic constituent dependencies inside the corpus sentences, I first looked at these basic sentences. I observed that they all follow one of only two high-level syntactic structures. These structures are illustrated on two example sentences in Fig. 10[16]. In both of them the structural status of the syntactic constituents that respectively express the location and the day of the game is identical. The location is expressed by a constituent prefixed to the rest of the sentence as a parenthetical, while the day is expressed by a floating adjunct that is attached to different constituents in different corpus sentences. What sets these two structures apart is the relation between the constituents that respectively express the prominent statistic and the game result. In the first structure they are in *parataxis* while in the second they are in *hypotaxis*, with the prominent statistic as head. In this paper, I use the notions of parataxis and hypotaxis defined in [Halliday 1985], because they are general relations between syntactic constituents occurring at *all* linguistic ranks (sentence, clause, group). Two constituents are in parataxis if they are both at the same structural level. Parataxis is thus a general symmetric relation covering both coordination and apposition. In contrast, two constituents are in hypotaxis if it is possible to distinguish one as a main element and the other as a dependent element. Hypotaxis is an thus a general asymmetric relation covering both head-argument dependency and head-adjunct dependency. I take clause subordination to be a special case of head-adjunct dependency where both the head and the adjunct are clauses.

When I looked at the complex corpus sentences, I observed that they have the same structural characteristics as the basic sentences. In these complex sentences, the location is still expressed as a parenthetical prefix, the day is still expressed by a floating adjunct, there are still two top-level constituents - one including the prominent statistic and one including the game result - and these two constituents are still either in parataxis or in hypotaxis with the constituent including the prominent-statistic as head. The only difference from the basic sentence structures is that additional propositions are now grouped with the prominent statistic and/or with the game result in these two top-level constituents.

An example of a hypotactically structured complex sentence is shown in Fig. 11. This sentence contains nine propositions: the four basic propositions[17], three historical additional propositions[18] and two non-historical additional propositions. Compare this sentence with the basic sentence at the bottom of Fig. 10. They both share the same high-level structure: hypotaxis with the constituent including the prominent statistic as head and the constituent including the game result as dependent. In the complex sentence, however, the head constituent contains four additional propositions and the dependent constituent contains one additional proposition. The corpus also contained paratactically structured complex sentences with additional propositions grouped with the prominent statistic and/or with the game result.

In short, the top-level structure of any corpus sentence is one of the two basic structures illustrated in Fig. 10, with additional propositions *clustered* either around the prominent statistic or around the game result. A systematic analysis of the resulting clusters revealed that whether a proposition is part of the prominent statistic cluster or the game result cluster, is directly dependent on its ontological class. Among the ten classes of additional propositions from the domain ontology defined in section 2.3 only one was part of the prominent statistic cluster in some corpus sentences and part of the game result cluster in some others. All the other classes consistently appeared in the same cluster. This proposition clustering phenomenon is illustrated in Fig. 12. It shows where additional propositions from each ontological class gets attached. These semantic constraints on proposition clustering can be viewed as schemata[McKeown 1985] for sentence-rank planning.

Fig. 12 also shows that the overall structure of any corpus sentence depends on only three factors:

- The top-level relation between the two clusters, either parataxis or hypotaxis with the prominent statistic cluster as head.

[16] In this figure sentence constituents are represented by boxes with their proposition class at the top and their text at the bottom. These boxes are linked by lines showing the dependency relations between the constituents.
[17] with their ontological class in boldface font.
[18] in dashed boxes.

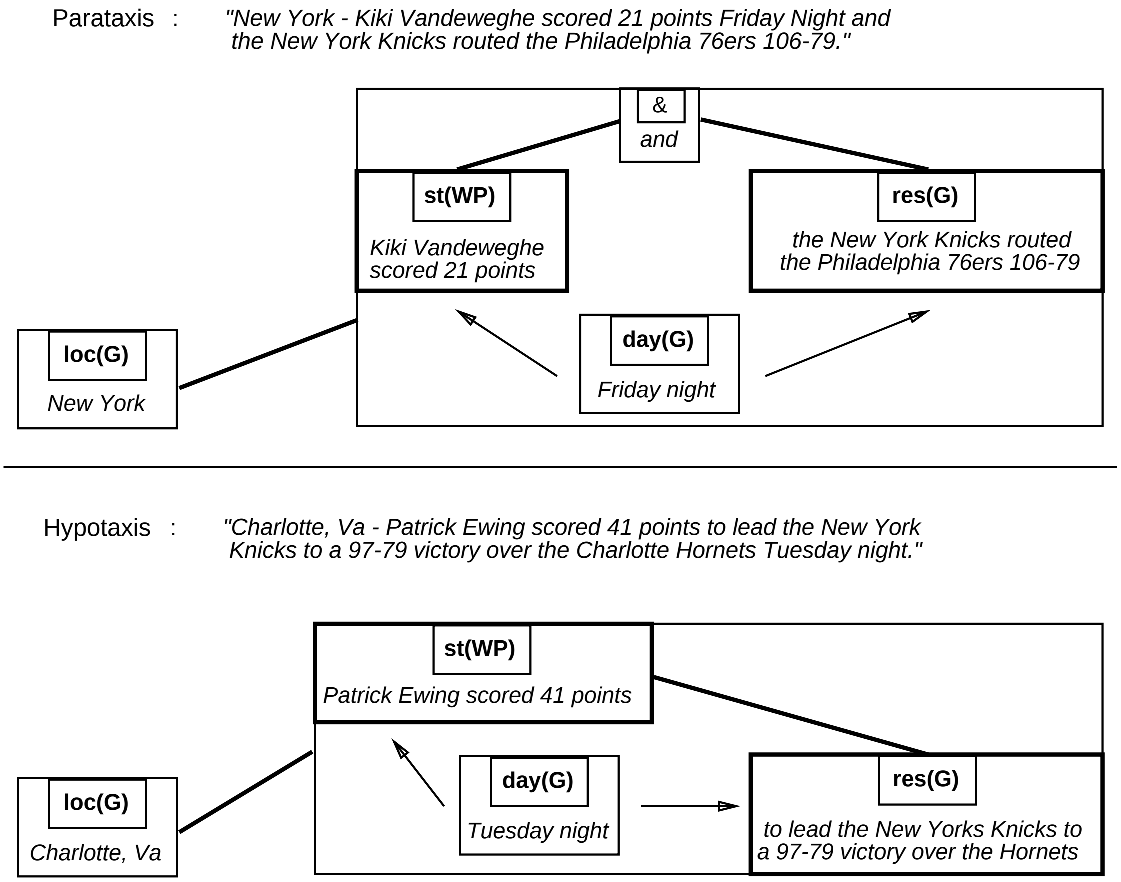

Figure 10: Top-level structures of basic sentences

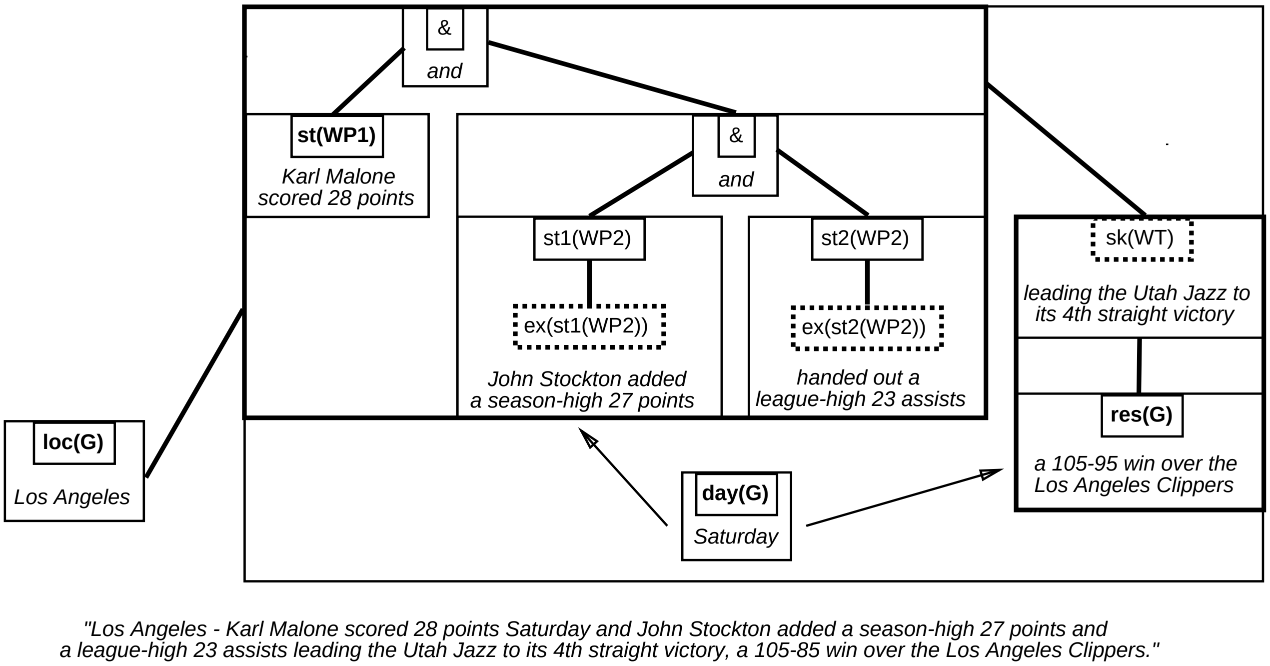

Figure 11: Complex sentence with top-level hypotactic structure

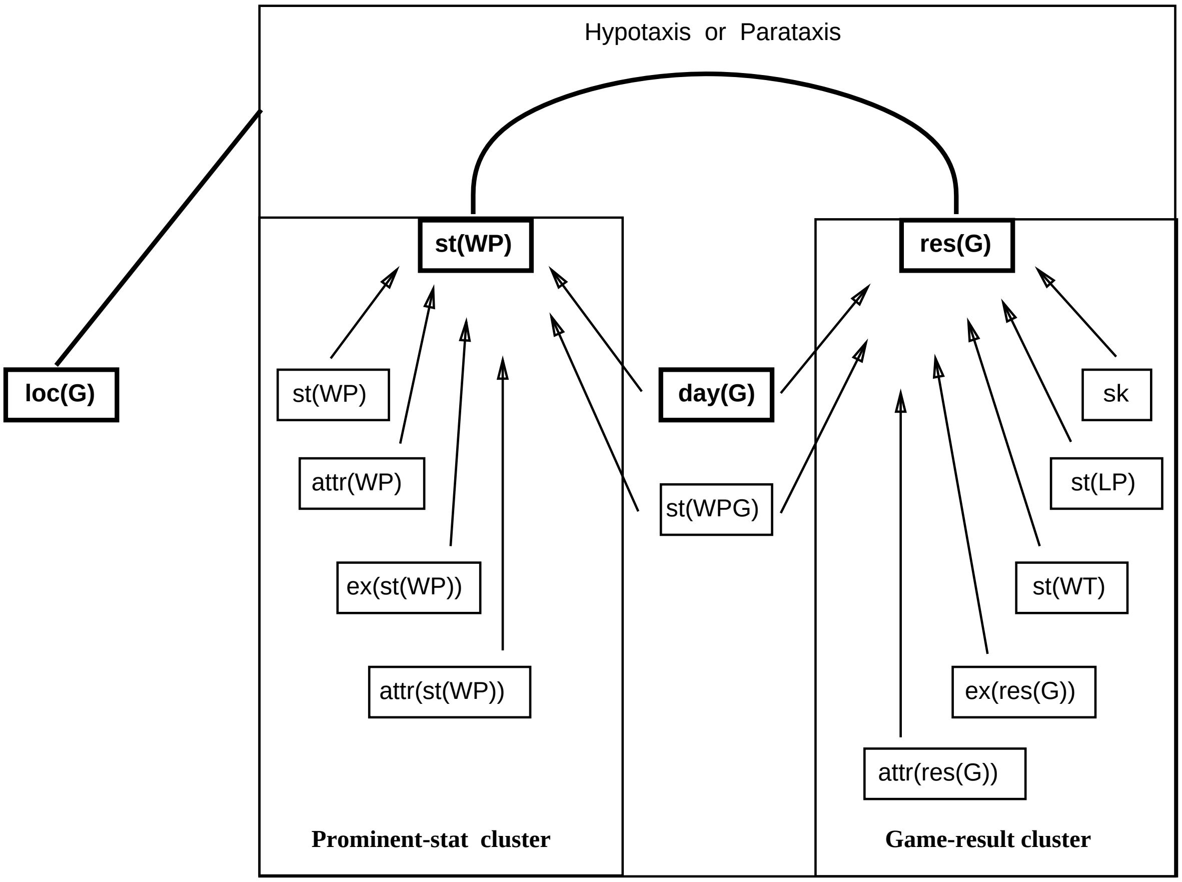

Figure 12: Proposition class clusters

- The internal structure of its prominent statistic cluster
- The internal structure of its game result cluster

I thus pursued the analysis of the corpus sentence by looking at the internal structure of each cluster. This analysis is discussed in section 2.4.2.

2.4.2 Internal cluster structure: realization patterns

The analysis of the top-level corpus sentence structure presented in the previous subsection, resulted in a set of *semantic* constraints to guide sentence *planning*. These semantic constraints define the set of proposition class combinations observed in specific corpus sentence constituents that I called clusters. In order to identify *linguistic* constraints on sentence *realization*, I then looked, for each clustered proposition combination, at the set of syntactic structures realizing it in the corpus. I call a mapping from a proposition class combination onto a syntactic structure a *realization pattern*. Fig. 13 gives an example of two alternative realization patterns for a two proposition combination. In this paper, a realization pattern is represented as an array, with each column corresponding to a syntactic constituent and each row corresponding to some level of information about this constituent[19]. The first row indicates the semantic element that the constituent realizes; the second row its grammatical function; the third row its structural status (i.e. head, **arg**ument,

[19] Some rows are double. A single line separates the two halves of a double row, whereas a double line separates two different rows.

Game-result proposition cluster **Cr**:
<game-result(winner,loser,score), streak(winner,aspect,result-type,length)>

Realization pattern **Rr1**:

winner	game-result	loser	score			length	streak+aspect	result-type
agent	process	affected	manner	purpose				
arg	head	arg	adjunct	adjunct				
proper	verb	proper	number	PP				
				prep	[det	ordinal	adj	noun]
Chicago	*defeated*	*Phoenix*	*99-91*	*for*	*its*	*third*	*straight*	*win*

Realization pattern **Rr2**:

winner	aspect		result-type	streak	length			score	game-result	loser
agent	process	affected/located			location	instrument				
arg	head	arg			adjunct	adjunct				
proper	verb	NP			PP	PP				
		det	participle	noun		prep	[det	number	noun	PP]
Utah	*extended*	*its*	*winning*	*streak*	*to six games*	*with*	*a*	*118-94*	*victory*	*over Denver*

Figure 13: Examples of proposition combination realization patterns

adjunct or **conj**unct) and the fourth its syntactic category[20]. The last row contains a cluster example from the corpus illustrating the pattern.

These patterns abstract from lexical material and syntactic details[21] to focus on representing the different mappings from semantic structure to syntactic structure. Note for example how the semantic elements `streak` and `aspect`[22] are mapped onto syntactic constituents in drastically different ways in the two realization patterns of Fig. 13. In pattern **Rr1**, they are both expressed by the adjective *"straight"* inside a purpose adjunct PP. In pattern **Rr2**, `aspect` is expressed by the head verb *"to extend"* and `streak` by the head noun *"streak"* of the object NP.

I identified the realization patterns of each clustered proposition class combinations, finding a total of 160 patterns for 79 combinations. Each of these realizations patterns can be obtained by applying a set of proposition-adding revisions on one of the three realization patterns of the basic four proposition combination. In this paper, instead of going through the list of the patterns resulting from these revisions, I focus on the more concise presentation of the revision operations themselves. They are listed in appendix B.

The realization patterns capture the *structural* paraphrasing power of the corpus sublanguage[23]. Concerning this paraphrasing, I discovered a significant discrepancy between the prominent statistic cluster and the game result cluster: the corpus contained an average 1.29 and a maximum 4 realization patterns per prominent statistic cluster proposition combination against an average 3.83 and a maximum 7 realization patterns per game result cluster proposition combination.

[20] The particular grammatical functions, structural relations and syntactic categories used in this paper are inspired from: [Quirk *et al.* 1972], [Winograd 1983], [Halliday 1985] and [Fawcett 1987].

[21] e.g. connectives, mood etc.

[22] The "aspect" of a streak event specifies whether the streak has been extended or interrupted by the reported game result. It is an entirely different notion from the grammatical aspect of a clause.

[23] But abstracts from the more productive grammatical and lexical paraphrasing.

2.5 Revision tools

2.5.1 Differential analysis of realization patterns: identifying revision tools

In the introduction of this paper, I proposed an incremental approach to generate complex sentence structures by incorporating additional propositions on a basic draft structure in successive steps. This approach takes advantage of the compositionality of natural language to simultaneously achieve variety and complexity of the generated structures from a small number of simple lexical entries.

The observations made so far confirmed the high degree of compositionality of the corpus sublanguage. In particular, the observation that complex sentences share the same top-level structure with basic sentences, shows that incorporating an additional proposition can be done *locally* at the cluster level, regardless of the top-level sentence structure. For example the incorporation of the streak proposition in the sentence of Fig. 11 in section 2.4.1 is local to the game result cluster and independent of the larger hypotactic structure of the sentence. It could have equally been performed on the game result cluster of a paratactically structured sentence.

In order to maximally exploit this compositionality, the best approach consists of starting from a basic sentence structure with only one proposition per cluster and then incorporating additional propositions into each cluster[24] one by one. A generator based on this type of incremental revision would generate the example sentence of Fig. 11 in section 2.4.1 in six steps, with the following sentences as successive drafts[25]:

1. "Los Angeles – Karl Malone scored 28 points Saturday, leading the Utah Jazz to a 105-95 win over the Los Angeles Clippers."
2. "Los Angeles – Karl Malone scored 28 points **and John Stockton added 27** Saturday, leading the Utah Jazz to a 105-95 win over the Los Angeles Clippers."
3. "Los Angeles – Karl Malone scored 28 points and John Stockton added 27 **points and handed out 23 assists** Saturday, leading the Utah Jazz to a 105-95 win over the Los Angeles Clippers."
4. "Los Angeles – Karl Malone scored 28 points and John Stockton added **a season-high** 27 points and handed out 23 assists Saturday, leading the Utah Jazz to a 105-95 win over the Los Angeles Clippers."
5. "Los Angeles – Karl Malone scored 28 points and John Stockton added a season-high 27 points and handed out **a league-high** 23 assists Saturday, leading the Utah Jazz to a 105-95 win over the Los Angeles Clippers."
6. "Los Angeles – Karl Malone scored 28 points and John Stockton added a season-high 27 points and handed out a league-high 23 assists Saturday, leading the Utah Jazz to **its fourth straight victory**, a 105-95 win over the Los Angeles Clippers."

The issue of scheduling such a set of additional propositions for incorporation in the draft is touched upon first in section 4.1 and elaborated p.59 of section 6.1.1.

The realization patterns discussed in section 2.4.2 specify syntactic structures for *entire* proposition clusters. The final sentence of the above example contains two such patterns: one pattern for the five proposition prominent statistic cluster (*"Karl Malone scored 28 points and John Stockton added a season-high 27 points and handed out a league-high 23 assists"*) and another pattern for the two proposition game result cluster (*"leading the Utah Jazz to its fourth straight victory, a 105-95 win over the Los Angeles Clippers"*).

These realization patterns are thus too coarse grained to be usable by a revision-based generator exploiting compositionality inside clusters. The linguistic data needed by such a system is a set of *sub*patterns specifying the realization of a *single* additional proposition class in the context of a pre-existing draft cluster structure. For example, one such subpattern would specify that in the context of a draft game result cluster like *"leading*

[24] Recall from section 2.4.2 that because different classes of propositions fit in different clusters, the respective incremental revisions of each cluster are independents.

[25] In each of these drafts, the most recent addition is boldfaced.

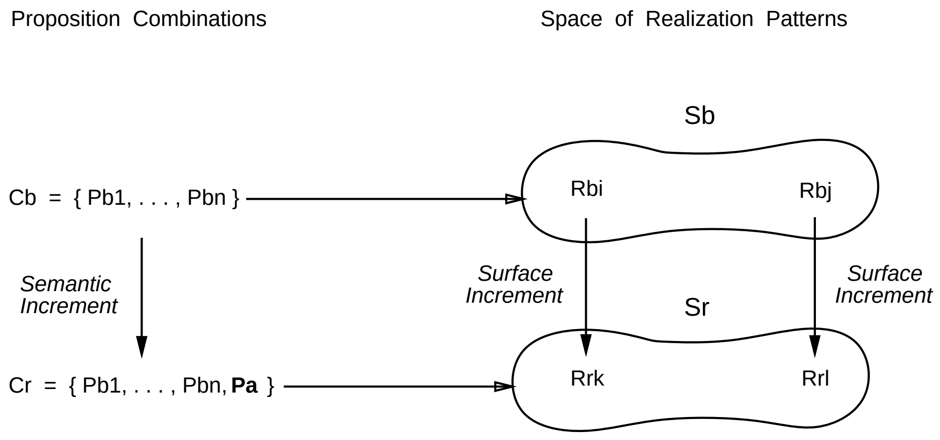

Figure 14: Differential analysis of realization patterns

the Utah Jazz to a 105-95 win over the Los Angeles Clippers" an additional streak proposition can be realized by an apposition like *"its fourth straight victory"*.

In order to identify these contextual subpatterns I carried out a differential analysis of the cluster realization patterns. A pictorial description of this differential analysis is given in Fig. 14. It shows how the notion of surface increment can be defined from the notions of semantic increment and of realization pattern.

A proposition combination Cr appearing in a corpus sentence cluster is a *semantic increment* of another proposition combination Cb appearing in some other corpus sentence cluster, if Cr contains Cb plus one additional proposition Pa. If in addition, Sr is the set of Cr's realization patterns and Sb the set of Cb's realization patterns, then an element Rr of Sr is a *surface increment* of an element Rb in Sb if Rr is structurally closer[26] to Rb than to any other element in Sb.

As a first example of surface increment pair, consider <Rb1,Rr1> in Fig. 15. Rr1 is a realization pattern for the two proposition class combination: <game result, winning team streak>. It is thus the surface increment of some realization pattern for the basic game result cluster that contains only the game result. Among these basic realization patterns, Rb1 is the only one to appear *in its entirety* inside Rr1, as Rr1's head subpattern. Rr1 is thus a surface increment of Rb1. Recall that the last row of a table representing a realization pattern is filled by an example sentence from the corpus illustrating the pattern. The pattern itself is described by the first four rows. The surface increment relation holds between *patterns* and abstracts from the details of distinct corpus sentences following the same pattern. In the tables of Fig. 15, the fact that Rb1 is a subpattern of Rr1 is indicated by the identity between the first four rows of the Rb1 table and the first four rows of the first four columns of the Rr1 table.

The structural relation between a base pattern and its surface increment is sometimes more subtle than in the case of Rb1 and Rr1. The pair <Rb2,Rr2> in Fig. 16, illustrates this point. Rr2 is an alternative to Rr1 for the realization of the two proposition class combination: <game result, winning team streak>. It is thus also a surface increment of some realization pattern for the basic game result cluster. Rb2 and Rb1 are two members of this set of basic game result realization patterns. However, no element in this set appears in its entirety as a subpattern of Rr2 the way Rb1 appears in Rr1. A weaker structural similarity is thus needed to determine which among these basic patterns is structurally the closest to Rr1. In this case, similarity can be based on the respective realizations of the basic cluster semantic head: the game-result. Compare the way the game result is realized in the syntactic structure of Rb1, Rb2 and Rr2. In Rb1 it is realized by the verbal head of the syntactic structure: "*to defeat*". In contrast, in both Rb2 and Rr2 the game result is realized by the head noun "*victory*" of a nominal appearing as an embedded constituent in that structure. Rb2 is therefore structurally closer to Rr2 than Rb1 is. Rr2 is thus a surface increment of

[26] I did not use a formal definition of structural distance between two realization patterns. I instead identified surface increment empirically by inspection. This inspection process is intuitively presented on two examples in the remaining of this section.

Cr, semantic increment of **Cb**: <game-result(winner,loser,score),**streak(winner,aspect,result-type,length)**>

Rr1, realization pattern of **Cr** and surface increment of **Rb1**:

winner	game-result	loser	score			**length**	**streak+aspect**	**result-type**
agent	process	affected	manner	purpose				
arg	head	arg	adjunct	adjunct				
proper	verb	proper	number	PP				
				prep	[det	ordinal	adj	noun]
Chicago	*defeated*	*Phoenix*	*99-91*	**for**	**its**	**third**	**straight**	**win**

Base game result proposition cluster **Cb**: <game-result(winner,loser,score)>

Base realization pattern **Rb1**:

winner	game-result	loser	score
agent	process	affected	manner
arg	head	arg	adjunct
proper	verb	proper	number
Seattle	*defeated*	*Sacramento*	*121-93*

Figure 15: Simple surface increment example

Cr, semantic increment of **Cb**: <game-result(winner,loser,score),**streak(winner,aspect,result-type,length)**>

Rr2, realization pattern of **Cr** and surface increment of **Rb2**:

winner	**aspect**		**result-type**	**streak**	**length**			score	game-result	loser
agent	process	affected/located			location	instrument				
arg	head	arg			adjunct	adjunct				
proper	verb	NP			PP	PP				
		det	participle	noun		prep	[det	number	noun	PP]
Utah	**extended**	**its**	**winning**	**streak**	**to six games**	**with**	*a*	*118-94*	*victory*	*over Denver*

Base game-result proposition cluster **Cb**: <game-result(winner,loser,score)>

Base realization pattern **Rb2**:

winner			score	game-result	loser
agent	process	range			
arg	head	arg			
proper	support-verb	NP			
		[det	number	noun	PP]
Chicago	*claimed*	*a*	*128-94*	*victory*	*over New Jersey*

Figure 16: Complex surface increment example

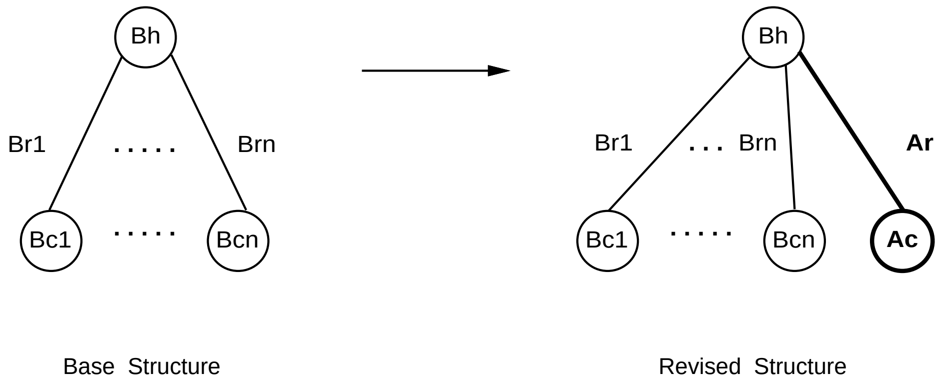

Figure 17: adjoin schema

Rb2 and not of Rb1. The structural relation between Rr2 and Rb2, however, is more subtle than the simple inclusion relation between Rr1 and Rb1. This is indicated by the fact that only the first and the last four columns (of the first four rows) of Rb2 also appear in Rr2.

By systematically listing the surface increments (and conversely the surface decrements) of each of the 160 realization patterns resulting from the previous step of the analysis, I identified 270 surface increment pairs. I then considered each of these 270 pairs as an instance of revision. The simplest pattern in the pair is the *base* pattern of the revision and the most complex pattern in the pair the *revised* pattern. For example, pattern Cb1 in Fig. 15 and pattern Cb2 in Fig. 16 are base patterns whereas pattern Cr1 in Fig. 15 and pattern Cr2 in Fig. 16 are revised patterns.

For each revision instance, there is a set of structural transformations to change the base pattern into the revised pattern. Each such set of structural transformations defines a revision tool[27]. In the last step of this corpus analysis I classified revision tools in terms of their structural characteristics. This last step is presented in section 2.5.2 and appendix B.

2.5.2 Classifying revision tools

Simple revision Procedurally, a revision tool is defined by a set of transformations to change the base pattern into the revised pattern. The first element of any such set is an introductory transformation, whose goal is to attach a new constituent onto the base pattern. This new constituent realizes the additional proposition introduced by the revision.

For simple revision tools, the set of transformations is reduced to a singleton consisting of this introductory transformation that does not affect the existing realizations of the base propositions.

I identified four basic types of simple revisions: *adjoin, append, conjoin* and *absorb*. They differ from each other in terms of either the type of the base structure on which they can be applied or the type of revised structure they produce.

For example, adjoin applies only to hypotactic base patterns. It consists in the attachment of an adjunct Ac conveying the additional proposition. Ac is thus *adjoined* to the base structure under the base head Bh. The general structural schema of an adjoin is shown in Fig. 17[28].

[27] The transformations involved in the application of a revision tool are not related to the notion of transformation as defined by transformational grammarians. In transformational grammars, a transformation relates surface forms that are grammatical paraphrases of *a common* deep semantic form. In contrast, a transformation involved in the application of a revision tool maps a surface form S_1 realizing a deep semantic form D_1 onto a surface form S_2 realizing *a distinct* deep semantic form D_2, which contains D_1 *plus an additional proposition*.

[28] The pictorial conventions used in this figure, and in Fig. 18 are the following: constituents are circles or ovals, structural relationships are lines, the lines corresponding to role relations (either argument or adjunct) are labeled. The elements **added** by the revision, either constituents or relations, are boldfaced.

In the corpus, adjoin is used at both the clause and nominal ranks. The surface increment pair <Rb1,Rr1> of Fig. 15 in section 2.5.1 provide an example of clause rank adjoin. In this example, the purpose PP - **"for its third straight win"** - realizing an additional streak proposition, is adjoined to the clausal base structure - "Chicago defeated Phoenix 99-91" - realizing a basic game result cluster. The resulting revised cluster is: "Chicago defeated Phoenix 99-91 **for its third straight win**".

The three other types of simple revisions are presented in appendix B.

Complex revisions In complex revisions a single transformation cannot account for the structural differences between the base pattern and the revised pattern; the introductory transformation attaching the additional proposition onto the base pattern is followed by restructuring transformations. The goal of these restructuring transformations is to maintain grammaticality despite the structural shake up brought about by the introductory transformation.

I identified six basic types of complex revisions: *recast, argument demotion, nominalization, adjunctization, constituent promotion* and *coordination promotion*. Each type is characterized by a different set of restructuring transformations which involve displacing base constituents, altering the base argument structure or changing the base lexical head. These types of transformations also distinguish complex revisions (as a whole) from simple revisions. Simple revisions conserve both the argument structure and the lexical head of the base and do not involve moving base constituents around.

Certain complex revision tools can only be applied to specific types of base structure. For example, adjunctization applies only to clausal base patterns headed by a support verb Sv. Following Gross [Gross 1984], I call *support verb* any verb that does not realize any semantic element. Appearing only because each clause syntactically requires a verb in English, its sole function is to support one of its meaning-bearing arguments. Support verbs are opposed to full verbs, which do convey a semantic element by themselves. A support verb can have an argument set of various length. The supported argument can fill various semantic roles but it distinguishes itself from the other arguments in that it must be realized by an NP whose head collocates with the support verb. In the corpus sentence instance of the pattern Rb2 given in Fig. 16 for example, this collocation is the verb-object collocation <"*to claim*", "*victory*">. The head verb "*to claim*" does not realize any semantic element but only syntactically supports its range role, the NP "`a 128-94 victory over the New Jersey`" whose head noun "*victory*" realize the semantic head of the game result proposition.

The general structural schema of an adjunctization is given in Fig. 18. The introductory transformation consists of realizing the additional proposition by a combination of a new head verb V and one additional constituent Ac. This change of main verb leaves the constituent Bc2, which was filling the supported argument slot in the base pattern, without a support verb; and since the argument structure of the new head verb is already saturated, Bc2 has to migrate to an adjunct position. This restructuring transformation, thus "adjunctizes" Bc2.

The surface increment pair <Rb2,Rr2> of Fig. 16 in section 2.5.1 is an example of adjunctization. In this example, the base pattern "Utah claimed *a 188-94 victory over Denver*" realizes a basic game result cluster. In the revised pattern, "Utah **extended its winning streak to six with** `a 188-94 victory over Denver`" an additional streak proposition is realized by the combination of the new head verb "to extend", the new argument "its winning streak" and a location adjunct "to six"[29]. Deprived of its support verb "to claim", the base pattern argument "a 188-94 victory over Denver" is adjunctized as a instrument PP in the revised pattern.

The five other types of complex revisions are presented in appendix B.

Side transformations The previous subsection presented the *core* transformations involved in the corpus revisions:

- The introductory transformation whose goal is to attach the additional proposition the base pattern.

[29] When appearing as argument of the verb "to extend", cardinal such as "six" are used as metaphorical locations on a numerical axis.

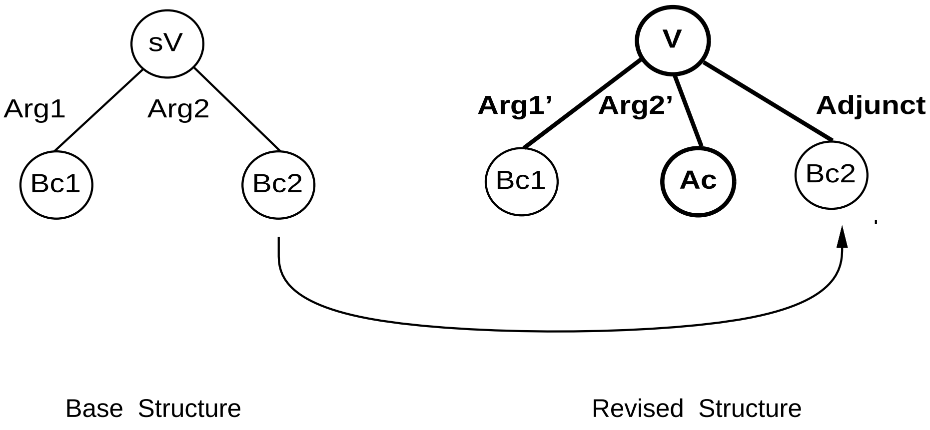

Figure 18: adjunctization schema

- The restructuring transformations whose goal is to maintain grammaticality.

Some revisions, however also involve *side* transformations that satisfy other goals such as avoiding repetitions, avoiding ambiguities and enforcing collocation constraints. I identified five basic types of side transformations: *reference abridging, ellipsis, argument control, scope marking, lexical adjustment* and *ordering adjustment.*

The most common of these transformations is reference abridging. Its goal is to suppress repetitions introduced by the attachment of the additional proposition. In the corpus, initial references use a set of default properties associated with the class of the referred entity in the domain ontology. For example, initial references to players use their first and last name, initial references to teams use their name and location, etc. However, when a revision introduces a second reference, this set of properties is *distributed* between these two references. The base reference is thus *abridged* by the revision[30].

The surface increment pair <Rb3,Rr3> below illustrates this phenomenon:

(Rb3) "to direct the Los Angeles Lakers past [*the* Washington *Bullets*] 87 72"

(Rr3) "to direct the Los Angeles Lakers past [Washington] 87 72 **handing** *the Bullets* **their ninth straight defeat**"

In the base pattern Rb3, both the name, "*the Bullets*" and the location, "Washington" of the losing team are used in the single reference to that team. When this cluster is revised, the added constituent contains a second reference to this team. In this second reference the team name alone is used. As a side transformation, the first reference is abridged using the team location alone. Without such a side transformation, the team name would be repeated just three words after being introduced:

(Rr3') "to direct the Los Angeles Lakers past <u>the</u> Washington <u>Bullets</u> 87 72 **handing <u>the Bullets</u> their ninth straight defeat**"

The other types of side transformations are presented in appendix B.

Classifying criteria Table 1 summarizes the essential characteristics of the revision tools I identified in the corpus. The columns of this table provide five essential criteria to classify these tools:

[30]Note that the use of this transformation is based on the assumption that the report targeted audience knows about every property in the default set, since otherwise it could not establish the co-reference link in the revised pattern

	base structure	revised structure	subconstituent displacement	argument structure	head
adjoin	hypotaxis	hypotaxis head = base const	none	untouched	untouched
append	parataxis	parataxis	none	untouched	untouched
conjoin	any	parataxis	none	demoted	demoted
absorb	any	hypotaxis head = added const	none	demoted	demoted
recast	hypotaxis	hypotaxis head = base	argument $\mapsto$ adjunct	untouched	untouched
argument demotion	hypotaxis	hypotaxis head = base const	top-level arg $\mapsto$ embedded const	shrunk	untouched
nominalization	hypotaxis	hypotaxis head = added const	none	expanded	fV $\mapsto$ sV
adjunctization	hypotaxis	hypotaxis head = added const	argument $\mapsto$ adjunct	changed	changed
constituent promotion	hypotaxis	hypotaxis head = added const	embedded const $\mapsto$ top-level arg	expanded	sV $\mapsto$ fV
coordination promotion	hypotaxis w/ paratactic arg	parataxis	multiple	demoted	demoted

Table 1: Comparative structural characterization of the revision tools

- The type of base pattern structures on which they can be applied: whether a tool is applicable only to hypotactic bases, only to paratactic bases or equally well to both.
- The type of revised pattern structures resulting from their application: either paratactic or hypotactic and in this latter case whether the revised pattern head is the base pattern head or the new constituent introduced by the revision.
- How base pattern constituents are displaced by the revision.
- How the argument structure of the base pattern is affected by the revision, whether it is expanded, shrunk, etc.
- How the lexical head of the base pattern is affected by the revision and in particular whether the revision involves replacing a support verb by a full verb or vice-versa.

Each distinct combination of these five factors defines a basic type of revision. Each instance of revision can be further characterized by taking into account the following factors:

- The linguistic rank of the revision base pattern[31]: clause, nominal etc.
- The syntactic category of the constituent added by the revision.
- The grammatical function filled by this added constituent in the revised cluster.

The eight criteria above classify revisions in terms of core transformations. For revisions that also involve side transformation(s), the various types of side transformations constitute an additional classifying criterion.

Constraints on revision tools application Table 2 summarizes five different constraints on the application of each of the ten basic revision tools. The first column specifies the types of side transformations that may accompany their application and the second column the linguistic ranks of the base patterns onto which they are applied. The final three columns specify the type of propositions they can add: streak, extrema and/or box-score statistic (i.e. non-historical information). Table 2 also contains occurrences of

[31] The constraints on this rank for each basic tool type are given in table 2

		side transformations		scope ranks		streak info	extrema info	*non-*histo
adjoin	88	reference adjust	15	clause	16	8	1	7
		ordering adjust	2	nominal	72	23	46	3
append	10	ellipsis	1	clause	8	-	-	8
				nominal	2	-	-	2
conjoin	127	reference adjust	3	clause	91	5	-	86
		scope marking	12	NP coordination	28	-	-	28
		lexical adjust	3	NP apposition	8	8	-	–
		ellipsis	86				-	–
absorb	11	argument control	10	clause	10	3	-	7
				nominal	1	-	1	–
recast	4	none		clause	3	2	1	–
				nominal	1	1	-	–
argument demotion	7	none		clause	7	7	-	–
nominalization	1	none		clause	1	1	-	–
adjunctization	20	reference adjust	6	clause	20	8	8	4
constituent promotion	1	none		clause	1	1	-	–
coordination promotion	1	none		clause	1	-	1	–
	270		138		270	67	58	145

Table 2: Constraints on revision tools usage

each of these constraints in the corpus of 270 surface increment pairs. For example, concerning the simple revision tool *adjoin* presented the first paragraph of section 2.5.2, table 2 indicates that they were 88 total occurrences of its usage in the corpus, breaking down into 72 occurrences at the nominal-rank and 16 at the clause-rank. It also notes that among the 16 clause-rank occurrences, eight were used to convey a streak, one to convey an extremum and the seven remaining to convey a non-historical statistic. Finally, it indicates that 17 occurrences of adjoin involved a side transformation, 15 reference adjustments and two ordering adjustments.

2.6 Summary

The corpus analysis described in this section has fulfilled five functions:

- Identify the ontology of the basketball report domain.
- Identify the sentence-rank schemata used in the report leads.
- Identify realization patterns determining which syntactic structures can be used to express each combination of semantic propositions allowed by these schemata.
- Identify revision tools to incrementally build complex realization patterns from basic ones.
- Identify constraints on the applicability of these revision tools.

The corpus analysis has thus provided us with data for all the knowledge sources of the prototype generator STREAK. It has also defined a target output which is useful to evaluate the success of STREAK. Finally, it has confirmed the plausibility of the revision approach, by showing the highly compositional and regular structure of the complex sentences found in the corpus.

3 A key issue: generation architecture

I initially linked the issues of historical background and generation architecture in the introduction. I observed that in order to flexibly convey historical information, one must generate a wide variety of very complex sentence structures. Because existing generation architectures are inadequate for this task, I outlined a new architecture based on two design principles: revision and microcoding.

In this section, I elaborate on the relation between historical information and generation architecture. I first define a vocabulary to discuss generation architecture, a necessary step given the lack of standard terminology in the literature concerning this issue. I then motivate the need for a new architecture. This motivation starts by a set of observations about historical information from the corpus analysis presented in section 2. From these observations, I deduce a set of abilities that a report generator must have in order to flexibly convey historical information. I also establish that previous report generation architectures lack most of them. This set of abilities further motivates the revision and microcoding principles and defines two additional principles concerning the type of draft representation on which to perform revision. Having defined and motivated these four principles, I then present in detail a new architecture based on them. This presentation starts by a description of the draft representation, followed by a description of the different processing components building and revising this representation. In conclusion, I briefly discuss the wider applicability of this new architecture to generation in general.

3.1 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 subsection 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's task to produce all content from the knowledge sources it can access. 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. Content *production* subtly differs from content *selection* which consists of 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.

What each generation subtask recovers highly depends on the level of *microcoding* at which it is performed. Recall from section 1 that the 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 clauses 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[32] every sentence constituent down to the group rank. 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 realization matter. In contrast, for microcoded generators, the distinction between

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

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 realization decision, for EPICURE generating *"the ripe banana"* instead of *"the banana"* results from content selection and organization choices. Generating *"the large shrimp"* and not *"the large prawn"* would be a genuine 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* content selection and organization at the clause and group ranks. One of the most innovative characteristics of the generation model proposed in this paper is that it allows micro-level content selection and organization to be performed under realization constraints.

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. I hold this second view. The research proposed in this paper focuses on the following generation subtasks: micro-level content planning (both selection and organization) and all four surface realization subtasks.

3.2 Motivation: the need for a new report generation architecture

In this subsection, I propose four design principles for designing a report generation system with the ability to flexibly convey historical information. These principles are motivated by a set of corpus observations on the way historical information is conveyed in human-generated reports. From these observations I deduce a set of abilities that a report generation system must have in order to flexibly convey historical information. I then show how these abilities can be provided by four design principles.

3.2.1 From corpus observations to needed abilities

The corpus observations which motivate the need for a new generation architecture are the following:

1. Historical information is combined with new information in sentences of a greater complexity than those produced by existing generation systems.
2. Historical information of the same type is conveyed by a wide range of different syntactic constructs.
3. Historical information of the same type is conveyed at different linguistic ranks.
4. Historical information of the same type is scattered in distant locations inside a report.
5. Taking into account historical information triggers a combinatorial explosion of the number of relevant propositions to consider for inclusion in the report.
6. Assuming a draft and revision approach, the applicability of revision tools for adding an historical proposition on a given draft sentence is constrained by the surface form of that sentence.

Variety of syntactic constructs to convey the same additional historical fact:

- Clause-complex coordinative conjoin:
(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**".
- Adjoin of non-finite clause in top-level nominal:
(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**".
- Adjoin of relative clause in top-level nominal:
(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**".
- Adjoin of relative clause in embedded nominal:
(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**".
- Top-level nominal appositive conjoin:
(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**".
- Embedded nominal appositive conjoin:
(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 19: Syntactic diversity of historical information

In the following paragraphs I review in turn each of these observations and their consequences in term of desired abilities for a report generator conveying historical information.

The first three of these observations have already been made in the introduction of this paper. The first observation, the syntactic complexity associated with historical information, was summarized in the table on p. 4 comparing corpus sentences containing historical information with sentences produced by existing generators in terms of number of words and propositions. This table showed that the ability to convey historical information entails *the ability to generate very complex sentences.*

The second observation, the syntactic variety associated with historical information, is illustrated in Fig. 19, first presented in the introduction on p.5 and duplicated here. This figure contains six corpus paraphrases, containing an historical streak proposition emphasized by a boldface font. In each of these paraphrases, the same historical proposition is attached to the same basic sentence by a different syntactic construct. The ability to flexibly convey historical information thus entails *the ability to generate a wide variety of syntactic constructs below the sentence rank.*

The third and fourth observations are based on the notion of "floating semantic element". A fact to convey in a report is a "floating semantic element" if it can alternatively be realized at various levels inside the report structure. *Floating* semantic elements are opposed to *structural* semantic elements which are always realized at one fixed level. In section 2.4.1, I noted that in the corpus reports, the prominent statistic and the game result were systematically realized by the two top-level clauses of the first sentence. These two propositions are thus examples of structural semantic elements. There are two aspects to the phenomenon of floating semantic element: an *intra*-sentential aspect and a *inter*-sentential aspect. Intra-sentential floating refers to the property of a proposition to be alternatively realizable at different linguistic ranks inside a given report sentence. Inter-sentential floating refers to the property of a proposition to be alternatively realizable in different and perhaps distant report sentences. In the corpus, historical information displayed both types of floating[33].

The six paraphrases of Fig. 19 illustrates intra-sentential floating of an historical fact. In these paraphrases, the same historical proposition is realized at different linguistic ranks: clause-complex in (1), clause in (2), (3) and (4), nominal in (5) and (6). This example shows that the ability to flexibly convey historical information entails *the ability to realize floating semantic elements at different linguistic ranks.*

The corpus report of Fig. 20, shows how an historical fact can float across the entire report structure. Consider where the streak propositions (emphasized by a boldface font) about each team is conveyed. Denver's streak is attached to the game result in the *first sentence* of the report. In contrast, Sacramento's streak is attached to the scoring statistic of one of its players in the *last paragraph*. It is essential to note that the distance separating the two streak propositions in this report, cannot be explained on semantic or syntactic grounds. The only semantic constraint on the attachment of a team's streak to a draft sentence is that this sentence contains a reference to that team. The first two sentences of the report contain a reference to Sacramento (emphasized by a smallcap font). Semantically, they are thus as valid a location for Sacramento's streak as the last paragraph's first sentence. Syntactically, they are also valid locations since the relative clause which realizes Sacramento's streak in the last paragraph could also have been added as modifier of either reference, as shown by sentences S_1 and S_2 below.

S_1 "Sacramento, Ca. – 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, **who lost their fourth straight**".

S_2 "Adams, who was drafted and then discarded by THE KINGS, **who lost their fourth straight**, 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."

The preference of the first sentence of the last paragraph over S_1 can be explained on *discursive* grounds. Recall from section 2.2 that because the corpus reports follow an inverted pyramid structure with summary lead, only crucial facts go in the first sentence. Sacramento's streak information was probably not important enough to have been incorporated in the first sentence. As for the preference of sentence of the last paragraph over S_2, it can be explained on *stylistic* grounds. The embedding of the relative clause realizing the streak inside another relative clause, makes S_2 stylistically awkward. The example report of Fig. 20 thus shows that choosing between alternative report structure locations for realizing floating historical propositions requires taking into account not only semantic and syntactic factors but also discursive and stylistic ones. The ability to flexibly convey historical information therefore entails *the ability to realize floating semantic elements under a combination of semantic, syntactic, stylistic and discursive constraints.*

The fifth observation concerning historical information in report generation is that it triggers a combinatorial explosion of the number of relevant propositions. This explosion is due to the fact that the historical context multiplies the dimensions along which to evaluate the significance of each input statistic. Consider for example, a player's scoring statistic in a basketball game. Ignoring the historical context there are basically only two ways in which this statistic can be significant: when compared to the scoring statistics of all the other players in the game (game-high) and when compared to the scoring statistics of his teammates only (team-high). However such a statistic can be *historically* significant in a combinatorially explosive number of ways: when compared to that player's 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 over the last 10 games, or ever) while playing at home (or on the road) against a particular opponent (or against any opponent) etc.

[33] The fact that historical semantic elements of given type are coerced to appear in a specific cluster of the corpus first sentences does not invalidate the floating nature of these elements. The clusters' internal complexity allows these elements to appear at different linguistic ranks inside a given cluster.

Sacramento, Ca. – 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 20: Streak information in a corpus report

Because of the sheer number of historical facts, any such fact is likely to be of similar relevance as many others. Therefore, content production results in a much larger set of candidate propositions with many more shades of relative importance. This makes content selection much harder. General rules based on purely encyclopedic grounds (*e.g.,* 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 to decide which propositions to select. Other factors need to be considered. One such factor is whether the realization patterns of the candidate propositions can be combined in cohesive and stylistically felicitous surface forms. The ability to convey historical information thus entails the ability *to perform final content selection under surface form constraints.*

The sixth and last corpus observation concerning historical information is that which type of revision can be applied to a report draft to add a given historical fact is constrained by the surface form of that draft. This observation has already been made in section 2.5.2. The first column of table 1 in section 2.5.2 indicates that the applicability of most revision tools is restricted to base clusters with specific syntactic structures. Consider again the basic corpus sentence example given in the introduction:
(0) *"David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets".*

The different revisions of this base sentence to add the same additional streak proposition shown in Fig. 19 all result from using two revision tools: conjoin and adjoin[34]. Other revision tools, cannot be used with (0) due to its surface form. For example adjunctization[35], which requires the game cluster of the base sentence to be headed by a support verb, cannot be used with (0) which is headed by the full verb "to lift". Adjunctization, can however be used to add the same streak information on the base sentence (0') synonymous with (0), yielding (1') synonymous with the sentences (1-6) of Fig. 19:
(0') "*David Robinson scored 32 points Friday night and the San Antonio Spurs rolled to* `a 127 111 victory over the Denver Nuggets`"
(1') "*David Robinson scored 32 points Friday night and the San Antonio Spurs* **extended Denver's losing streak to seven games with** `a 127 111 victory over the Denver Nuggets`".

This example shows that, in some cases, the base sentence syntactic structure can be the only discriminatory factor justifying the choice of one revision tool over another.

The conclusion drawn from the corpus observations listed in the beginning of this subsection is that the ability to convey historical information entails the following abilities:

1. Generating very complex sentences.

[34] The variety comes from different application of these tools (i.e. at different linguistic ranks and in different syntactic forms.) in each paraphrase.
[35] Presented in section 2.5.2.

2. Generating a wide variety of syntactic constructs below the sentence rank.
3. Performing final content selection under surface form constraints.
4. Realizing floating semantic elements at different linguistic ranks.
5. Realizing floating semantic elements under a combination of semantic, syntactic, discursive and stylistic constraints.

This list of abilities motivates both the need for a new architecture and the design principles of this new architecture. I discuss each of these questions in the following two subsections.

3.2.2 The need for a new architecture

The need for a new generation architecture arises because the above abilities are precluded by the architecture of existing report generators. In the introduction of this paper, I noted that both ANA [Kukich 1983] and FoG [Bourbeau *et al.* 1990] are based on the classic single-pass pipeline architecture[36]. I also explained that a single pass architecture is impractical for simultaneously achieving syntactic complexity and variety. This combination of abilities requires to microcode complex sentences, a task which involves too many decisions for being carried out in a single pass. In addition, by finalizing *all* content selection decisions before *any* content realization occurs, the pipeline architecture precludes the ability of performing content selection under surface form constraints.

For different reasons, the respective architectures of both FoG and ANA also preclude the abilities to realize floating semantic elements at different linguistic ranks and under a combination of semantic, syntactic, discursive and stylistic constraints. FoG has no use for these abilities, since, as pointed out by Polguere in his thesis (cf. [Polguere 1990], p.14-16), there is only one realization pattern per proposition combination in FoG's meteorological target sublanguage. All semantic elements of this domain are thus structural. This peculiar property was exploited to simplify the architecture of FoG. In particular, FoG maps domain propositions onto a lexicalized syntactic structures in a single step. Polguere notes that for domains with floating semantic elements, this mapping would need to be decomposed in several steps. ANA's target stock market sublanguage contains examples of floating semantic elements (cf. [Kukich 1983], p.74-75). However, although ANA does produce paraphrases where the same floating element is realized at different ranks, ANA does *not* explicitly choose at what rank to realize floating elements. With ANA's macrocoded architecture, no distinction is made between structural and floating elements. They are both integrated in proposition combinations that are realized *all at once* by phrasal patterns hard-wired in the lexicon. This macrocoded approach thus circumvents the decisions involved in the realization of floating semantic elements through hand-coding of numerous and partially redundant phrasal lexicon entries. In particular, a macrocoded generator cannot have the ability to choose between expressing a floating semantic element above the clause rank and below it.

In conclusion, previous report generation architectures preclude the combination of abilities needed for flexibly conveying historical information.

3.2.3 From needed abilities to architecture design principles

What architecture design principles can we deduce from the agenda of providing a report generator with the five abilities needed for flexibly convey historical information?

In the introduction of this paper I have already showed that generating a wide variety of syntactic constructs below the sentence rank requires microcoding sentences, if needed down to individual words. From an engineering perspective, I have also suggested that the need to produce very complex sentences through microcoding calls for a two-pass draft and revision generation model.

[36] This architecture was outlined in Fig. 6 on p.6.

From a cognitive perspective, the need to generate very complex sentences in itself supports this revision approach. [Pavard 1985] describes an experiment providing psychological evidence supporting the revision model for the generation of very complex sentences. In this experiment, human subjects were asked to write a single sentence paraphrasing a text of three sentences which together conveyed eight propositions in 42 words. They were thus asked to generate a sentence of complexity similar to the newswire report leads analyzed in section 2 of this paper. The subjects were divided into two groups. In one group the subjects performed the exercise using a microphone, i.e. a medium that precludes revision, while in the other they performed the exercise using a text editor, i.e. a medium that facilitates revision. For each group two measurements where made:

- Average percentage of semantic elements omitted in the single-sentence paraphrases, among those realized by head and argument constituents in the original text.
- Average percentage of semantic elements omitted in the single-sentence paraphrases, among those realized by adjunct constituents in the original text.

The results of this experiment are summarized in the table below:

media	omitted heads and arguments	omitted adjuncts
microphone	4%	34%
text editor	0%	4%

These results strongly suggest that even humans have trouble generating sentences of such complexity in one shot. They also show that the trouble does not so much arise for structural semantic elements (which essentially correspond to the elements realized by the head and argument constituents in the original text), but specifically for floating semantic elements (which essentially correspond to the elements realized by adjunct constituents in the original text). They thus support the cognitive plausibility of the generation model proposed here where floating elements are added by revising a draft built around structural elements.

I have so far showed that the need for both syntactic variety below the sentence rank and syntactic complexity suggests two principles for a new report generation architecture: incremental information-adding revision and microcoding from a word-based lexicon. Within this framework, the three remaining abilities identified in the previous subsection (i.e. constraining final content selection by surface form factors, realizing floating semantic elements at different linguistic ranks and under a combination of semantic, syntactic, stylistic and discursive constraints) supports two additional principles concerning the draft representation on which to perform revision: (1) that it includes a surface layer reflecting realization choices and (2) that it includes a purely conceptual layer independent of linguistic form.

Recall from Fig. 7 of section 1 that in the draft and revision generation model I propose, final content selection occurs during revision. In particular, it is at that stage that the final decision to include in the report a particular historical fact is made. Within this framework, the ability to constrain final content selection upon surface form factors requires the use of incremental revision to perform not only sentence planning but also lexicalization and semantic grammaticalization. The draft representation must therefore include a layer specifying the realization (i.e. the open-class lexical items and syntactic structure) of each proposition. This need is also supported by the corpus observation that revision tool applicability is constrained by the surface form of the draft.

The need to handle inter-sentential floating of semantic elements provides an additional justification for both: (1) the draft and revision approach sketched in section 1 of this paper and elaborated in the next section, and (2) the representation of realization choices in the draft. At revision time, a generator has access to the draft of the *entire* report. If surface form is represented in the draft, the generator can then choose where to realize a floating semantic element under a combination of semantic, syntactic, stylistic and discursive constraints. In particular, it can choose between two candidate locations for realizing a floating semantic element by comparing the respective stylistic impacts of attaching the floating element to either of them, even if these locations are distant in the text structure. This would be impossible for a one-pass architecture where realization is performed linearly on a sentence-per-sentence basis. With such an architecture, when the current sentence is one of the potential locations for realizing a floating semantic element, there is no way to measure choosing this sentence versus other potential locations that may lie ahead in the text structure but for which no surface form is yet available.

In addition, the need to handle intra-sentential floating of semantic elements requires the draft representation to also include a purely conceptual layer that totally abstracts from linguistic form. The first constraint on the attachment of a floating semantic element onto a draft sentence constituent is semantic: the attachment must be warranted by some semantic relationship holding between the floating element and some base semantic element realized by the draft constituent. The same base element can be realized at different linguistic ranks in different sentences. For example, the base semantic element realized at the clause rank in phrase X below, is realized at the nominal rank in phrase Y:
(X) "*John Stockton scored 27 points*"
(Y) "*John Stockton's 27 points*"

Despite their syntactic differences, the same revision can be applied on both these phrases to incorporate a floating semantic element of type extrema, as shown by X' and Y' below:
(X') "*John Stockton scored* **a season-high** *27 points*"
(Y') "*John Stockton's* **season-high** *27 points*"

The revision component can determine that this revision is applicable to both these phrases, only if the draft representation captures the meaning identity between (X) and (Y). This identity can be captured only by a purely conceptual representation that totally abstracts from linguistic form. A linguistically motivated semantic representation such as PENMAN's Upper-Model [Bateman 1989], would view (X) as a material action and (Y) as a possessed object, two radically different semantic categories. It would thus fail to capture their identity of meaning. This contrast between conceptual and linguistic semantic representations is further discussed in section 3.3.1.

In this subsection I have showed that providing a report generator with the five abilities it needs to flexibly convey historical information suggest that its architectural design must reflect the four following principles:

1. Incremental information-adding revision
2. Microcoding from a word-based lexicon
3. Presence in the draft representation of a purely conceptual layer independent of linguistic form.
4. Presence in the draft representation of a surface layer reflecting realization choices.

In the next subsection, I present in detail a report generation architecture based on these four principles.

3.3 A new report generation architecture

In this subsection, I propose a new generation architecture based on the four principles defined in section 3.2.3. This new architecture is a general design for any report generator conveying historical information in a quantitative domain and encompass the whole range of report generation subtasks[37]. The research proposed in this paper focuses on three of these subtasks: sentence level content selection, sentence organization and surface realization. Details concerning these three tasks are discussed in section 4 presenting the implementation of the STREAK prototype. Issues related to generation subtasks that lay beyond the scope of the proposed research, namely content production and text level content selection and organization are discussed here at a superficial level.

I start by describing the various levels of representation of an utterance in the architecture. I then describe the various processing components of the architecture and how they interact to build a first draft of the report and then revise it. After this description, I make clear where each of the generation subtasks defined in section 3.1 is carried out in this architecture. I conclude by briefly defining its wider scope of applicability for generation in general.

[37]These subtasks were defined in section 3.1.

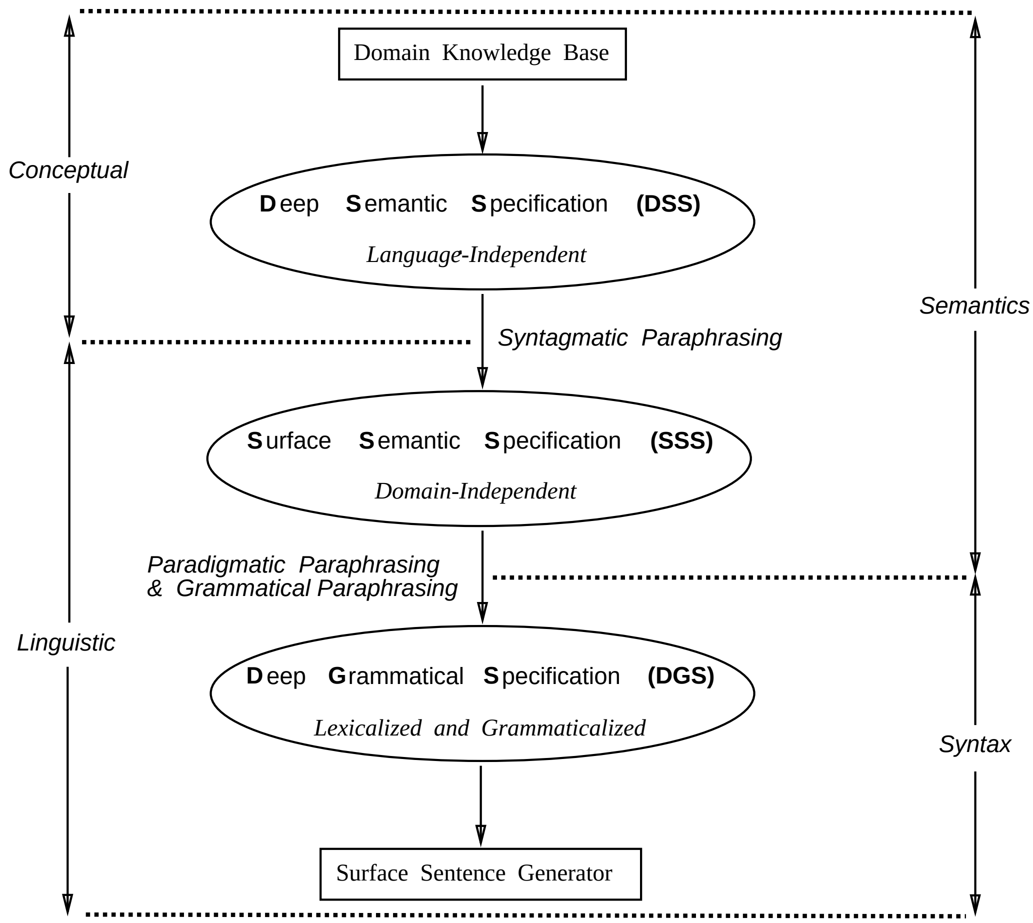

Figure 21: Internal Representation Layers

3.3.1 Representation layers

In the architecture I propose, an utterance is represented at three different levels of abstraction:

- the Deep Semantic Specification (DSS),
- the Surface Semantic Specification (SSS),
- the Deep Grammatical Specification (DGS).

The architecture thus belongs to the stratificational tradition of computational linguistics (cf. [Dale 1988] and [Polguere 1990] for other generation systems using a stratificational utterance representation scheme), with multiple representation layers, each capturing a specific set of regularities. The general stratificational scheme I propose is sketched in Fig. 21.

The three layers define a pipeline of internal representations mapping from the representation internally used by the domain knowledge base for domain-dependent conceptual processing onto the representation internally used by the surface sentence generator for domain-independent syntactic processing. The three intermediate representations have been conceived with the following biases concerning these two end representations: the domain knowledge base uses a frame-like language and the surface sentence generator uses unification with a functional grammar. In the STREAK prototype, the domain knowledge base is implemented in CLASSIC [Resnick *et al.* 1990] and the surface sentence generator is SURGE [Elhadad 1992], an realization grammar of English working with the functional unifier FUF [Elhadad 1990] [Elhadad 1991].

Because of these biases and in order to ensure maximum uniformity the three layers are syntactically identical: they are all FUF feature structures. A FUF feature structure is essentially a set of attribute-value pairs. The value of a FUF attribute can be either: (1) an atom, (2) an embedded feature structure (recursion) or (3) a *path* to the value of some other attribute in the feature structure. Paths encode identity constraints on the value of several attributes. Path values are list of attributes surrounded by curly braces. They can be absolute or relative to their position in the overall feature structure. 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". So for example, the feature structure below contains an absolute path equating the values of attributes `a` and `e` and a relative path equating the values of attributes `f` and `c`.

```
((a 1)
 (b ((c 2)))
 (d ((e {a})
     (f {^2 b c}))))
```

The three layers differ by the semantics of their respective features. In the following paragraphs I describe and exemplify this semantics for each layer in turn. I then discuss the advantages of this three layer representation scheme.

The Deep Semantic Specification (DSS) A Deep Semantic Specification is either a partial description of a knowledge base individual or a rhetorical aggregation of such partial descriptions. In the first case, the semantics of all its features is purely encyclopedic (i.e., it captures a piece of the application domain's common-sense knowledge). I call a DSS of this type an *elementary* DSS. In the second case, in addition to encyclopedic features, it also has domain-dependent rhetorical features recursively filled by subDSSs. I call a DSS of this type a *structured* DSS. Elementary DSSs are directly built through knowledge base queries, whereas structured DSSs result from a content planning process. Fig. 22 gives two examples of elementary DSSs. The first represents a single proposition conveyed by the clause *"John Stockton scored 27 points"*. The second represents a semantic increment of the first, conveyed by the clause *"John Stockton scored A SEASON-HIGH 27 points"*. In a DSS, pointers to knowledge base individuals are indicated by the prefix "`i-`". The DSS is a purely conceptual representation that totally abstracts from linguistic form. All the synonymous expressions of a given sublanguage thus share a common DSS. For example, the DSS_1 in Fig. 22 does not only represent the clause *"John Stockton scored 27 points"* but also a whole set of synonymous phrases including *"John Stockton pumped in 27 points"*, *"John Stockton added 27"*, *"John Stockton's 27 points"*, *"a 27 points performance by John Stockton"* etc.

The Surface Semantic Specification (SSS) The Surface Semantic Specification is an intermediate layer that bridges the gap between the conceptual and domain-dependent DSS and the syntactic and domain-independent DGS. It is composed of domain-independent semantic and rhetorical features abstracted from observed linguistic regularities. Fig. 23 gives three example SSSs. Consider SSS_2 at the bottom of that figure. Its features `agent` and `created` are examples of general semantic features. They are part of a set of thematic roles proposed by systemic linguists [Fawcett 1987]. Similarly, its features `elaboration` and `elaborated` are examples of general rhetorical features. They are part of a set of rhetorical relations proposed by researchers in Rhetorical Structure Theory (RST) [Mann and Thompson 1987][38]. Although linguistically motivated, the SSS is still a semantic representation abstracting from lexical and syntactic details. Thus, SSS_{1a} corresponds not only to the clause "*John Stockton scored 27 points*" but to a whole set of synonymous clauses including *"John Stockton pumped in 27 points"* and *"John Stockton added 27"*. However, at the SSS layer, the perspective taken to view the domain proposition is fixed. All the synonymous clauses represented by SSS_{1a} in Fig. 23 view Stockton's scoring statistic as an event. NPs such as "*John Stockton's 27 points*", which view that statistic as an object are represented by SSS_{1b}.

[38] Note that while, the original RST relations were originally defined as holding between clauses, the example of elaboration relation given here links the head and the modifier of an NP.

DSS_1, corresponding to the clause: "*John Stockton scored 27 points*"

```
((deepsemcat game-stat-ref)
 (referent i-stockton-pts)
 (actor ((deepsemcat player-ref)
         (referent i-stockton)
         (first-name "John")
         (last-name "Stockton")))
 (value ((deepsemcat simple-game-stat-val-ref)
         (referent i-stockton-pts-val)
         (value 27)
         (unit i-pts))))
```

DSS_2, corresponding to the clause: "*John Stockton A SEASON-HIGH scored 27 points*"

```
((deepsemcat game-stat-ref)
 (referent i-stockton-pts)
 (actor ((deepsemcat player-ref)
         (referent i-stockton)
         (first-name "John")
         (last-name "Stockton")))
 (value ((deepsemcat simple-game-stat-val-ref)
         (referent i-stockton-pts-val)
         (value 27)
         (unit i-pts)))
 (CONSEQ ((DEEPSEMCAT HISTO-STAT-UPDATE-REF)
          (REFERENT I-STOCKTON-EXTR1-UPDATE)
          (AFTER ((DEEPSEMCAT HISTO-STAT-REF)
                  (REFERENT I-STOCKTON2-EXTR1)
                  (INSTANT-GAME-STAT I-STOCKTON-PTS)
                  (CARRIER ((DEEPSEMCAT PLAYER-TOKEN-REF)
                            (REFERENT I-STOCKTON2)
                            (TOKEN-OF I-STOCKTON)))
                  (HISTO-SIGNIF (DIRECTION I-HIGH)
                                (MEMBERSHIP-SCOPE I-PERSONAL)
                                (TIME-SCOPE I-SEASON)))))))
```

Figure 22: Two examples of DSSs

SSS_{1a} corresponding to the clause: "*John Stockton scored 27 points*"

```
((surfsemcat evt-spec)
 (concept game-stat)
 (proc ((type material) (effect-type creative)))
 (partic ((agent ((surfsemcat ent-spec)
                  (concept player)
                  (props ((first-name "John")
                          (last-name "Stockton")))))
          (created ((surfsemcat simple-quant-spec)
                    (concept simple-game-stat-val)
                    (props ((value 27) (unit pts))))))))
```

SSS_{1b} corresponding to the NP: "*John Stockton's 27 points*"

```
((surfsemcat ent-spec)
 (concept game-stat)
 (props ((possessor ((surfsemcat ent-spec)
                     (concept player)
                     (props ((first-name "John")
                             (last-name "Stockton")))))
         (possessed ((surfsemcat simple-quant-spec)
                     (concept simple-game-stat-val)
                     (props ((value 27) (unit pts))))))))
```

SSS_2 corresponding to the clause: "*John Stockton scored A SEASON-HIGH 27 points*"

```
((surfsemcat evt-spec)
 (concept game-stat)
 (proc ((type material) (effect-type creative)))
 (partic ((agent ((surfsemcat ent-spec)
                  (concept player)
                  (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)
                                 (props ((value 27) (unit pts)))))
                    (ELABORATION ((SURFSEMCAT COMPOUND-QUAL-SPEC)
                                  (CONCEPT GAME-STAT-HISTO-SIGNIF)
                                  (PROPS ((DIRECTION HIGH)
                                          (TIME-SCOPE SEASON))))))))))
```

Figure 23: Three examples of SSS

The Deep Grammatical Specification (DGS) The Deep Grammatical Specification is the input formalism of the surface sentence generator SURGE. Its features specify syntactic properties, syntactic constituency and open-class lexical items. These lexico-grammatical features express the semantic and rhetorical features of the SSS. The DGS is a *deep* grammatical specification, 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. The representation obtained after this mapping is what constitutes a *surface* grammatical specification. The thematic and syntactic roles used in SURGE are inspired from a variety of sources, most notably [Quirk *et al.* 1972] [Winograd 1983] [Halliday 1985] and [Fawcett 1987]. One great quality of SURGE is its ability to complete partially specified inputs with default features. These defaults concern syntactic roles, syntactic properties and closed-class lexical items. They allow a DGS to be concise and at the same time to unequivocally determine an actual natural language utterance. Fig. 24 gives four examples of DGSs.

Advantages of the three layer scheme In a generation system, utterances must be represented by at least two layers: a semantic layer used for planning purposes and a syntactic layer used for syntactic purposes. In the representation scheme proposed here, there are two semantic layers: the DSS used for text planning purposes[39] and the SSS which bridges the gap between the DSS and the DGS. The need for a representation that bridges the gap between conceptual and syntactic processing has been advocated in detail by [Meteer 1990]. For alternative schemes using a single semantic representation, there are two possibilities: either their single semantic layer is like a DSS or it is like an SSS.

If it is like an SSS, the domain knowledge of the generator must be encoded in terms of linguistically motivated categories. This is the approach advocated by PENMAN's Upper-Model [Bateman 1989], where the propositions and entities of the domain knowledge base are viewed as instances of domain-independent categories defined in terms of linguistic ranks and thematic roles. The drawback of this approach is that it makes the representation of floating propositions realizable at a variety of linguistic ranks and by a variety of thematic role structures quite problematic.

Consider again the two paraphrases below:
(X) "*John Stockton scored 27 points*"
(Y) "*John Stockton's 27 points*"

In the Upper-Model the meaning of these two sentences would have to be represented as two distinct propositions, instances of two different domain-independent categories: material action and possessed object. With such an approach, the only way to account for paraphrasing induced by floating propositions is to artificially introduce in the domain knowledge base, proposition duplicates corresponding to the different realization perspectives. In contrast, with the three layer scheme proposed here, the common meaning of such sentences is captured by a single DSS which is in then mapped onto different SSSs each representing a different realization perspective. X and Y thus share the same DSS, DSS_1 of Fig. 22, but have different SSSs: SSS_{1a} for X and SSS_{1b} for Y, both shown in Fig. 23. The key advantage of having two semantic representations is that each one assumes a single task: the DSS abstracts from linguistic form whereas the SSS abstracts from the domain. Floating propositions highlight the inherent conflicts between these two tasks which the Upper-Model approach attempts to reconcile within a single representation.

Note that the very idea of an Upper-Model is not questioned by the remark I have just made. Having a set of general concepts under which to attach the particular concepts of a given domain, is extremely helpful for developing a conceptual domain hierarchy. The problem arises with overusing linguistic criteria - and in particular linguistic rank and thematic roles - for defining these general concepts. The temptation for such overuse is great because there are not too many non-linguistic criteria to fall back on for defining an Upper-Model (see [Lenat and Guha 1989] for an attempt to build a non-linguistically motivated general ontology). However, semantic linguistic categories are best viewed as constituting a "surface perspective" model rather than an "upper" model.

Without a DSS, a system can thus hardly handle paraphrases that cut across linguistic ranks. Systems

[39] Sentence planning occurs during the mapping from the DSS to the SSS.

DGS_{1a} corresponding to the clause: "*John Stockton scored 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "score")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John")))
                         (last-name ((lex  "Stockton")))))))
          (created ((syntcat measure)
                    (quantity ((value 27)))
                    (unit ((lex "point"))))))))
```

$DGS_{1'}$ corresponding to the clause: "*John Stockton pumped in 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "pump in")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John")))
                         (last-name ((lex  "Stockton")))))))
          (created ((syntcat measure)
                    (quantity ((value 27)))
                    (unit ((lex "point"))))))))
```

DGS_{1b} corresponding to the NP: "*John Stockton's 27 points*"

```
((syntcat common)
 (possessor ((syntcat compound-proper)
             (head ((syntcat person-name)
                    (first-name ((lex "John")))
                    (last-name ((lex "Stockton")))))))
 (head  ((syntcat measure) (quantity ((value 27))) (unit ((lex "point"))))))
```

DGS_2 corresponding to the clause: "*John Stockton scored A SEASON-HIGH 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "score")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John")))
                         (last-name ((lex  "Stockton")))))))
          (created ((SYNTCAT COMMON)
                    (DEFINITE NO)
                    (CLASSIFIER ((SYNTCAT NOUN-COMPOUND)
                                 (CLASSIFIER ((lex "SEASON")))
                                 (HEAD ((lex "HIGH")))))
                    (HEAD ((syntcat measure)
                           (quantity ((lex "27")))
                           (unit ((lex "point")))))))))
```

Figure 24: Four examples of DGS

with a single semantic representation that is like a DSS avoid this problem. However, such systems must perform a direct mapping from the DSS onto the DGS. This mapping is a complex task for the three following reasons:

- It involves several generation subtasks: sentence level content planning, lexicalization and semantic grammaticalization.
- It is one to many.
- It is non-isomorphic.

The entire paraphrasing power of a sublanguage is captured by the mapping from a single DSS onto a whole set of synonymous DGSs. There are, however, three distinct aspects to that paraphrasing power:

- Syntagmatic paraphrasing, which involves choosing between synonymous word combinations that are associated with different syntactic categories and/or argument structures (*e.g., "John Stockton scored 27 points" vs. "John Stockton's 27 points"*).
- Paradigmatic paraphrasing, which involves choosing between synonymous word combinations for a fixed syntactic category and argument structure (*e.g., "John Stockton scored 27 points" vs. "John Stockton pumped in 27 points"*).
- Grammatical paraphrasing, which involves choosing between synonymous surface forms with the same open-class words but different syntactic properties (*e.g., "to extend their winning streak to seven games" vs. "extending their winning streak to seven games"*).

In addition, the mapping from DSS to DGS is in general **non-isomorphic**: several semantic elements can be conflated into a single syntactic element, some syntactic elements may be present for purely grammatical reasons without corresponding to any of the semantic elements, the syntactic element realizing the semantic head can be deeply embedded in the syntactic structure while the syntactic head may realize a semantic element embedded in the semantic structure, etc. This observation has been made in many domains (cf. [Talmy 1975], [Talmy 1985], [Zock 1988]). To illustrate this point in the basketball domain consider the expression of an `extremum` predicate. This historical semantic predicate takes five arguments:

- `carrier`: the statistic that constitutes the extremum.
- `aspect`: a boolean distinguishing between record-equalling and record-beating extrema.
- `direction`: a boolean distinguishing between maxima and minima.
- `membership-scope`: the reference set of authors whose statistics are compared to the `carrier` (possible values include `personal, team` and `league`).
- `time-scope`: the period during which statistics of the members of the `membership-scope` are compared to the `carrier` (possible values include `season`, `ever`).

In the cluster "*John Stockton dished out a career high 28 assists*" both the `membership-scope (personal)` and the `time-scope (ever)` arguments in the conceptual structure are mapped onto a single word in the syntactic structure: "career". On the other hand, the argument `aspect (new)` is not explicitly conveyed by any element in the syntactic structure. It is recoverable because of stylistic conciseness conventions[40] stipulating that extrema are - unless otherwise indicated - of the record-beating type

The main advantage of having an SSS layer, is that it breaks down the overall complexity of mapping a DSS onto a DGS into two stages, each involving a smaller, more manageable number of decisions. In particular:

- Different generation subtasks are handled at different stages: sentence level planning during DSS$\mapsto$SSS mapping and lexicalization and semantic grammaticalization during SSS$\mapsto$DGS mapping.

[40] Verifiable in the corpus.

- Different aspects of paraphrasing are captured by different stages: syntagmatic paraphrasing by the DSS$\mapsto$SSS mapping and paradigmatic and grammatical paraphrasing by the SSS$\mapsto$DGS mapping.
- Different sources of non-isomorphism are handled at different stages.

Another use for a double semantic representation is to distinguish between implicit and explicit content. Implicit content, such as `aspect` in the example above, is inferable from the context and represented only at the DSS level. In contrast, explicit content, such as `time-scope` in the example above, is necessarily realized by some linguistic element and also represented at the SSS level. [Dale 1988] adopted a double semantic representation in EPICURE specifically for such purpose. The issue of implicit content is linked to problems such as user models and hearer's knowledge, that lay beyond the scope of the research proposed here.

Draft representation All three representation layers are needed in the draft representation. Two of the four architectural principles defined in section 3.2.3, concerned the draft representation. One required the presence of a purely conceptual layer independent of linguistic form. This is the DSS. The other required the presence of a surface layer reflecting realization choices. This is the DGS. Finally, as the intermediary layer that bridges the gap between these two end-layers, the SSS is also needed.

The draft representation is thus a triple <DSS,SSS,DGS>. The DGS appears at the top-level of the LS. The SSS is embedded in the DGS below a feature `sss`. The DSS is in turn embedded in the SSS below a feature `dss`. Each element in this triple is a complex structure with several constituents linked by various dependency relations. Since the mapping between any two of these layers is both one-to-many and non-isomorphic, the draft representation must also include explicit markers of the correspondence between the constituents of a given layer and the constituents of the next layer. Revising a syntactic structure without explicit knowledge of what semantic element each of its constituent realizes could have undesirable side-effects such as deletion of part of the original semantic message, introduction of ambiguities etc.

In the current formalism, this correspondence is marked by adding to each DGS subconstituent a feature `sss` whose value is a FUF path leading to the corresponding SSS subconstituent in the overall LS. Similarly, each SSS subconstituent contains a feature `dss` whose value is a FUF path leading to the corresponding DSS subconstituent. An example of draft representation, with these embedded `sss` and `dss` paths uppercased is given in Fig. 25.

It remains unclear whether this method for marking realization correspondence can handle cases of non-isomorphic mappings. The implementation is still at an early stage and does not currently tackle such cases. Clarifying this question is thus on the agenda of remaining work presented in section 6.

3.3.2 Data Flow

This subsection describes the processing components of the new report generation architecture and how they interact during the generation process. Generation proceeds in two passes. During the first pass a complete specification of a report draft is built. This draft contains only the basic facts and is structured around new information. During the second pass, 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 processing components and internal representations involved in the draft construction are shown in figure 26. Those involved in the draft revision and the final natural language output are shown in figure 27[41].

The first pass starts upon reception of new statistical data (e.g., the box-score of a basketball game or an update on financial indexes). This statistical data is formatted and passed to a frame-based Historical Knowledge Base Management System (HKBMS) in the form of new individuals. The arrival of these new individuals activates demons performing an incremental update of the knowledge base. During this updating process, the HKBMS simultaneously evaluates the historical significance of the newly arrived statistics and retrieves historical statistics related to them. The result is a set of partial descriptions of both new and

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

Draft representation for the clause: "*John Stockton scored 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "score")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John")))
                         (last-name ((lex "Stockton")))))
                  (SSS {^3 SSS PARTIC AGENT})))
          (created ((syntcat measure)
                    (quantity ((value 27)))
                    (unit ((lex "point")))
                    (SSS {^3 SSS PARTIC CREATED})))))
 (sss ((surfsemcat evt-spec)
       (concept game-stat)
       (proc ((type material) (effect-type creative)))
       (partic ((agent ((surfsemcat ent-spec)
                        (concept player)
                        (props ((first-name "John") (last-name "Stockton")))
                        (DSS {^3 DSS ACTOR})))
                (created ((surfsemcat simple-quant-spec)
                          (concept simple-game-stat-val)
                          (props ((value 27) (unit point)))
                          (DSS {^3 DSS VALUE})))))
       (dss ((deepsemcat game-stat-ref)
             (referent i-stockton-pts)
             (actor ((deepsemcat player-ref)
                     (referent i-stockton)
                     (first-name "John")
                     (last-name "Stockton")))
             (value ((deepsemcat simple-game-stat-val-ref)
                     (referent i-stockton-pts-val)
                     (value 27)
                     (unit i-point)))))))
```

Figure 25: A example of draft representation

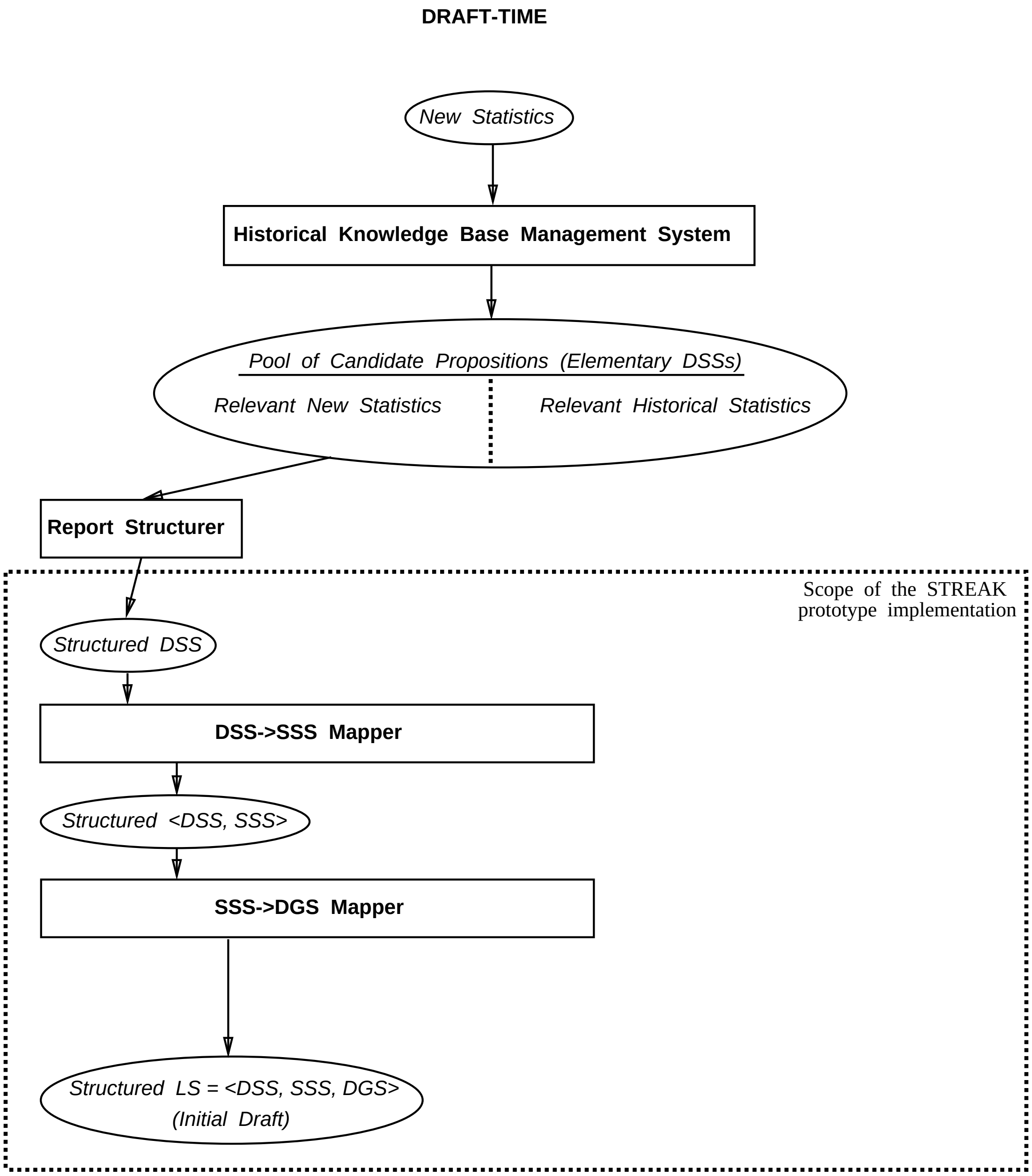

Figure 26: A draft and revision report generation architecture: draft-time

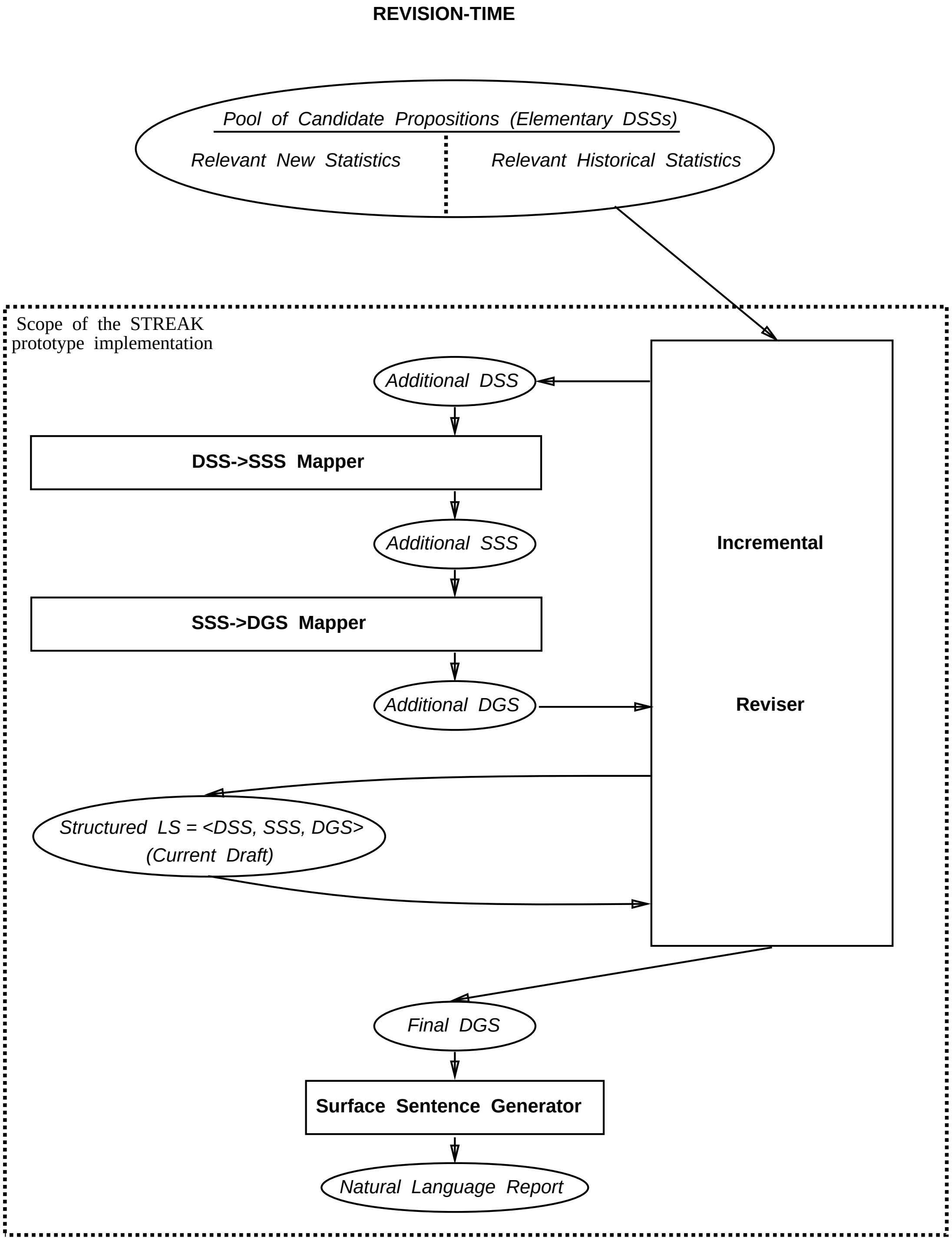

Figure 27: A draft and revision report generation architecture: revision-time

historical individuals which form a pool of candidate propositions to convey in the report. The HKBMS component is out of the scope of the research proposed here. I will thus not address the issues related to the process of updating the historical knowledge base and extracting from it a pool of relevant propositions. I simply assume that it can be carried out, if only on largely domain-specific grounds.

Each element in the pool of relevant proposition is an elementary DSS. Following stereotypical organization patterns, the report structurer picks some of these elementary DSSs and it groups and orders them in agreement with known rhetorical patterns. The report structuring component is also out of the scope of the research proposed here. 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 schemata [McKeown 1985] or RST planning [Hovy 1991]. The result of this structuring process is a structured DSS that constitutes the first layer of the initial draft representation.

This DSS is sent to the DSS$\mapsto$SSS mapper which returns a corresponding SSS. This SSS is in turn sent to the SSS$\mapsto$DGS mapper which returns a corresponding DGS. Each mapping stage conserves its input representation and enrich it with its output representation. The final output of these two mapping stages is a Layered Specification (LS) cumulating the DSS, SSS and DGS of the initial draft. This LS is the final result of the first pass of the overall draft and revision generation process and the starting point of the second pass.

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 current draft LS) the reviser locally calls the two mapper pipeline.

When revision is completed, the reviser extracts the DGS part from the final LS and hands it to the surface sentence generator. The surface generator takes each sentence specification in the DGS and produces a corresponding report sentence for each of them.

Although it is not shown in Figs. 26 and 27[42], both mappers and the reviser can directly query the knowledge base when they need additional information about some knowledge base concept or individual whose partial description in the representation that is being mapped or revised is not complete enough. This querying facility allows the elementary DSSs to consist of standard set of feature of default necessity.

3.3.3 Generation subtasks

In this subsection I explain how the generation subtasks defined in section 3.1 are distributed among the various processing components of the architecture proposed in section 3.3.2.

- *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 on the initial DSS 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.

[42] To avoid clutering them.

- *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.

The focus of the research proposed in this paper is on the final round of content selection and on both rounds of micro-level content organization, semantic grammaticalization and lexicalization. In the STREAK implementation, content production and the two first rounds of content selection will be simulated by hand. The surface sentence generator in charge of morpho-syntactic grammaticalization and linearization will be an extension of the system SURGE [Elhadad 1992].

3.3.4 Wider applicability of the proposed architecture

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

- Semantically dense and syntactically complex sentences are common.
- Many proposition combinations have a large set of paraphrases.
- Most proposition classes are exhibiting intra-sentential and/or inter-sentential floating.
- There is no sharp distinction between the respective relevance of most floating propositions.

3.4 Summary

In this section I have presented a new report generation architecture with the ability to convey historical information. With this architecture, generation proceeds in two passes. The first pass builds an initial report draft organizing and realizing structural semantic elements. The second pass incrementally revises this draft to opportunistically incorporate floating semantic elements, in particular historical facts. Internally, the draft is represented at three levels of abstraction. The most abstract level is purely conceptual. At the next level domain concepts are mapped to general semantic linguistic categories. These abstract linguistic resources are then mapped to specific lexical items and syntactic forms at the next level. The natural language report is produced by unifying this last level of the final draft with a functional grammar.

This new architecture is motivated by the corpus observation that flexibly conveying historical information requires choosing between syntactic constructs both at various linguistic ranks inside complex sentences and in various sentences inside a report. With previously proposed architectures, such alternatives could not be considered. The prototype system STREAK implements part of this complete generation architecture. This implementation is presented in the next section.

4 Implementation: the STREAK prototype system

In section 3 I proposed a general architecture for reporting facts in their historical context. In this section I present STREAK, a prototype system based on this architecture. The goal of this system is to provide an operational testbed for both the most innovative aspects of this general architecture and the linguistic data collected during the corpus analysis. Achieving this goal while ensuring that the implementation remains an endeavor of reasonable size requires restricting its scope in the three following ways:

- Aiming at generating only sentences similar to those of the subcorpus on which the systematic and in-depth analysis was performed[43]
- Focusing on three generation subtasks: final content selection, sentence organization and surface realization.
- Assuming that the pool of additional propositions to attach to the draft during revision is given in input to the reviser as a priority stack with the most relevant propositions on top.

In the following subsections, I first define the target input and output of STREAK resulting from these restrictions. I then describe the internal design of its processing components. Finally, I present the current state of the implementation.

4.1 Target Input/Output

The target output of STREAK is defined by the subcorpus of basketball report sentences which constituted the focus of the corpus analysis. Recall from section 2.2, that this subcorpus of sentences, object of systematic and in-depth analysis, was discursively restricted to the reports' first sentences and semantically restricted to the sentences conveying box-score statistics, streaks and extrema.

The discursive restriction limits the target output of STREAK to a single sentence. It allows STREAK to focus on only three generation subtasks: final content selection, sentence organization and surface realization. The scope of the implementation is therefore limited to the processing components responsible for these two tasks, *i.e.*

- the DSS$\mapsto$SSS mapper
- the SSS$\mapsto$DGS mapper
- the Reviser
- the Surface Sentence Generator

The input of STREAK consists of the internal representations that the above set of components need in the context of the architecture presented in section 3, namely:

- A structured DSS representing the structural semantic elements of the sentence to generate. In the complete architecture, this structured DSS is the output of the report structurer. This structured DSS contains only new propositions.
- A set of elementary DSSs representing floating semantic elements that are candidates for inclusion in this sentence. In a complete architecture, these elementary DSSs are the leftovers not selected by the report structurer in the pool of relevant propositions. These propositions are mostly historical.

In addition, the set of elementary DSSs are assumed to be sorted in order of decreasing relevance. This assumption simplifies the control of the incremental revision process.

[43] This analysis is presented in section 2.4.

4.2 Processing components

STREAK is made of four processing components: the DSS↦SSS mapper, the SSS↦DGS mapper, the reviser and the surface sentence generator. I describe the implementation of each of them in turn in the following subsections.

4.2.1 The mappers

Each mapper takes as input a feature structure at a given layer in the draft representation and produces a corresponding feature structure at a less abstract and more linguistic layer. Both mappers share the same mapping rule interpreter, each with its own rule base. The Left-Hand Side (LHS) of a mapping rule consists of a configuration of features from the input layer and the Right-Hand Side (RHS) consists of a corresponding set of features that rewrites the input configuration at the output layer.

The mapping rule interpreter is implemented on top of FUF and based on top-down recursive functional unification. During the top-down traversal of the input structure each constituent is unified with a FUF grammar encoding the mapping rule base. This encoding takes advantage of the bi-directionality of functional unification. Each rule is represented by a feature structure where the LHS features are grouped under the label of the input layer and the RHS features are grouped under the label of the output layer. Unification of this grammar with an input layer constituent simultaneously fires the appropriate mapping rule and builds the corresponding output layer constituent through enrichment. The details of top-down recursive functional unification process are presented in [Elhadad 1992], which also discusses related control issues.

The DSS↦SSS rule base maps the purely conceptual features of the DSS onto the semantic linguistic features (thematic roles and general rhetorical relations) of the SSS. It thus chooses a surface realization perspective for each semantic constituent. The SSS↦DGS rule base maps the semantic linguistic features of the SSS onto the lexico-grammatical features of the DGS. It thus determines syntactic constituency and chooses open-class lexical items. An example of DSS↦SSS rule and an example of DSS↦SSS rule are given in Fig. 28.

The DSS↦SSS rule of this figure specifies that one possible surface realization perspective for referring to a game statistic is to present it as a creative material action[44]. It also specifies that, in this case, the actor and value slots of the game statistic should be mapped onto the agent and created roles of the material action respectively. This rule is used for game statistic realization of the form "X scores N points", "X grabbed N rebounds" etc. In particular, it is used at the DSS↦SSS mapping stage during the building of the LS of Fig. 29. This rule uses the special FUF value `given`. A feature of the form `(f given)` encodes the constraint that the feature `f` must be present in the feature structure but that its particular value is irrelevant.

The SSS↦DGS rule of Fig. 28 specifies that the verb *"to score"* can be used as the lexical head of a clause realizing a game statistic concept and whose created role is a simple quantity measured in points. This rule is used at the SSS↦DGS mapping stage during the building of the LS of Fig. 29.

4.2.2 The reviser

The input to the reviser is twofold:

- A Layered Specification (LS) incorporating the DSS, SSS and DGS of the initial draft. This LS represents the structural semantic elements of the sentence to generate and their current realizations.
- A stack of Additional DSSs (ADSSs) in order of relevance. This stack represents the floating semantic elements candidate for attachment in the sentence to generate.

The reviser incrementally incorporates the floating elements into the draft. It consists of a revision rule interpreter and a revision rule base. The revision rule interpreter is implemented on top of FUF and uses functional unification for rule matching. The LHS of a revision rule is made of two fields: a base LS pattern

[44] In the sense of [Fawcett 1987].

Example of DSS$\mapsto$SSS mapping rule:

```
((dss ((deepsemcat game-stat-ref)
       (actor GIVEN)
       (value GIVEN)))
 (sss ((surfsemcat evt-spec)
       (concept game-stat)
       (proc ((type material)
              (effect-type creative)))
       (partic ((agent ((dss {^4 dss actor})))
                (created ((dss {^ 4 dss value}))))))))
```

Example of SSS$\mapsto$DGS mapping rule:

```
((sss ((surfsemcat evt-spec)
       (concept game-stat)
       (partic ((agent GIVEN)
                (created ((surfsemcat simple-quant-spec)
                          (concept simple-game-stat-val)
                          (props ((unit #(under point))))))))))
 (dgs ((proc ((lex "score"))))))
```

Figure 28: Example of DSS$\mapsto$SSS rule and SSS$\mapsto$DGS rule

and an ADSS pattern. The RHS of a revision rule is a list of revision operations to perform on the base LS. At each revision increment, the ADSS on top of the stack and the LS of the current draft are grouped into a feature structure. This feature structure is then unified with a FUF grammar, a disjunction of feature structures with each structure encoding the LHS of one revision rule. The rule[45] whose LHS is matched by this unification process is selected and the revision actions of its RHS are executed in order. There are five types of revision operations:

- Insertion of a new constituent in some layer of the LS (`insert-fd`).
- Displacement of base constituent in some layer of the LS (`copy`).
- Deletion of base constituent in some layer of the LS (`del-fd`).
- Update of a marker keeping track of the interlayer constituent correspondence (`del-path` and `add-path`).
- Call to a mapper for building a new constituent at some layer of the LS from a constituent at the higher layer (`map-fd`).

The first four types of operations are elementary structural manipulations of the LS. The fifth type realizes the ADSS at the two lower layers of the LS. These various types of revision actions are illustrated in Fig. 29 and 30. Fig. 29 (duplicated here, from section 3.3.1) contains the LS representing the base sentence *"John Stockton scored 27 points"* and the ADSS representing the historical fact that this scoring performance by Stockton is his best for the current season. Fig. 30 contains the LS for the revised sentence *"John Stockton scored* **a season high** *27 points"*, which incorporates this additional historical fact to the content of the base sentence. In that LS the new features resulting from the revision are written in uppercase.

The uppercased constituent at the DSS layer results from an insertion operation of the ADSS in the DSS of the base LS. The uppercased constituent at the SSS results from two actions: (1) a call to the DSS$\mapsto$SSS mapper onto the constituent newly inserted at the DSS layer and (2) an insertion operation of the resulting

[45] The issue of multiple rule matches has not yet been addressed.

LS of the draft sentence: "*John Stockton scored 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "score")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John"))) (last-name ((lex "Stockton")))))
                  (sss {^3 sss partic agent})))
          (created ((syntcat measure) (quantity ((value 27))) (unit ((lex "point")))
                    (sss {^3 sss partic created})))))
 (sss ((surfsemcat evt-spec)
       (concept game-stat)
       (proc ((type material) (effect-type creative)))
       (partic ((agent ((surfsemcat ent-spec)
                        (concept player)
                        (props ((first-name "John") (last-name "Stockton")))
                        (dss {^3 dss actor})))
                (created ((surfsemcat simple-quant-spec)
                          (concept simple-game-stat-val)
                          (props ((value 27) (unit point)))
                          (dss {^3 dss value})))))
       (dss ((deepsemcat game-stat-ref)
             (referent i-stockton-pts)
             (actor ((deepsemcat player-ref)
                     (referent i-stockton)
                     (first-name "John") (last-name "Stockton")))
             (value ((deepsemcat simple-game-stat-val-ref)
                     (referent i-stockton-pts-val)
                     (value 27)
                     (unit i-point))))))))
```

ADSS encoding the historical proposition that Stockton's 27 points is his best scoring performance of the season:

```
((deepsemcat game-stat-ref)
 (referent i-stockton-pts)
 (conseq ((deepsemcat histo-stat-update-ref)
          (referent i-stockton-extr1-update)
          (after ((deepsemcat histo-stat-ref)
                  (referent i-stockton2-extr1)
                  (instant-game-stat i-stockton-pts)
                  (carrier ((deepsemcat player-token-ref)
                            (referent i-stockton2)
                            (token-of i-stockton)))
                  (histo-signif ((direction i-high)
                                 (membership-scope i-personal)
                                 (time-scope i-season))))))))
```

Figure 29: A example of reviser input

LS of the revised draft: "*John Stockton scored A SEASON-HIGH 27 points*"

```
((syntcat clause)
 (proc ((type material) (effect-type creative) (lex "score")))
 (partic ((agent ((syntcat compound-proper)
                  (head ((syntcat person-name)
                         (first-name ((lex "John")))
                         (last-name ((lex "Stockton")))))
                      (sss {^3 sss partic agent})))
          (created ((SYNTCAT COMMON) (DEFINITE NO)
                    (CLASSIFIER ((SYNTCAT NOUN-COMPOUND)
                                 (CLASSIFIER ((LEX "SEASON")))
                                 (HEAD ((LEX "HIGH")))
                                 (SSS {^4 SSS PARTIC CREATED ELABORATION})))
                    (head ((syntcat measure)
                           (quantity ((value 27))) (unit ((lex "point")))
                           (SSS {^4 SSS PARTIC CREATED ELABORATED})))
                    (sss {^3 sss partic created})))))
 (sss ((surfsemcat evt-spec) (concept game-stat)
       (proc ((type material) (effect-type creative)))
       (partic ((agent ((surfsemcat ent-spec) (concept player)
                        (props ((first-name "John") (last-name "Stockton")))
                        (dss {^3 dss aps actor})))
                (created ((SURFSEMCAT PROPOSITION-CLUSTER)
                          (RHETOR-RELT ((SEMANTICS ELABORATION) (SURFSTRUCT HYPOTAXIS)))
                          (MAIN {^ ELABORATED}) (DEPENDT {^ ELABORATION})
                          (ELABORATED ((surfsemcat simple-quant-spec)
                                       (concept simple-game-stat-val)
                                       (props ((value 27) (unit point)))
                                       (DSS {^4 DSS VALUE})))
                          (ELABORATION ((SURFSEMCAT COMPOUND-QUAL-SPEC)
                                        (CONCEPT GAME-STAT-HISTO-SIGNIF)
                                        (PROPS ((DIRECTION HIGH)
                                                (TIME-SCOPE SEASON)))
                                        (DSS {^4 DSS CONSEQ AFTER HISTO-SIGNIF})))))))
       (dss ((deepsemcat game-stat-ref) (referent i-stockton-pts)
             (actor ((deepsemcat player-ref) (referent i-stockton)
                     (first-name "John") (last-name "Stockton")))
             (value ((deepsemcat simple-game-stat-val-ref)
                     (referent i-stockton-pts-val) (value 27) (unit i-point)))
             (CONSEQ ((DEEPSEMCAT HISTO-STAT-UPDATE-REF)
                      (REFERENT I-STOCKTON-EXTR1-UPDATE)
                      (AFTER ((DEEPSEMCAT HISTO-STAT-REF)
                              (REFERENT I-STOCKTON2-EXTR1)
                              (INSTANT-GAME-STAT I-STOCKTON-PTS)
                              (CARRIER ((DEEPSEMCAT PLAYER-TOKEN-REF)
                                        (REFERENT I-STOCKTON2)
                                        (TOKEN-OF I-STOCKTON)))
                              (HISTO-SIGNIF (DIRECTION I-HIGH)
                                            (MEMBERSHIP-SCOPE I-PERSONAL)
                                            (TIME-SCOPE I-SEASON))))))))))
```

Figure 30: An example of revised draft

new SSS constituent. Similarly, the uppercased constituent at the DGS layer, results from a call to the SSS$\mapsto$DGS mapper followed by an insertion operation. As an example of displacement operation, consider the constituent located under the path {`partic created elaborated`} at the SSS layer of the revised LS. This constituent was originally located under the path {`partic created`} at the SSS layer. The extension of the `sss` path marker in the DGS constituent realizing this displaced SSS constituent realizes is an example of path update operation. This example revision being of the type "adjoin", it does not involve deletion of any base LS constituent. However other types of revisions observed in the corpus would involve such deletion. An example of such revisions was presented in section 2.5.2.

For the reader interested in more details about the revision operations performed in this example, the revision rule used to transform the draft LS of Fig. 29 onto the revised LS of Fig. 30 is given in Fig. 31[46]. This rule is applicable to any pair: <additional-DSS,base-LS>
where the additional DSS represents a game statistic triggering an update of one of its author historical statistics and where the base LS represents the same game statistic realized as a creative material clause with a common NP filling the created role.

4.2.3 The historical knowledge base

The historical knowledge management system of the complete report generation architecture presented in section 3.3.2 is beyond the scope of the research proposed here. There is therefore no such component in STREAK. However, STREAK does include an historical knowledge base implemented in CLASSIC [Resnick *et al.* 1990]. Recall from section 3.3.2, that in the proposed architecture the mappers and the reviser must have the ability to query the historical knowledge base. Implementing the historical knowledge base is thus necessary to test the querying capabilities of these three modules which constitute the heart of STREAK. It is also helpful for defining the features of the DSS formalism and guiding the hand-coding of the DSSs STREAK takes as input. Knowledge base tokens are complete descriptions of the box-score statistics and related historical facts that these hand-coded input DSSs partially describe. These tokens are the leaves of a conceptual hierarchy representing the ontology of the corpus sentences. Remember from section 3.3.1 that the DSS contains features whose value is a token name (prefixed by "`i-`"). These token names serve as handles for accessing the knowledge base from a mapper or from the reviser.

Access to the knowledge base from the mapper or the reviser is needed to limit the DSS formalism to a set of standard features of general necessity. If the value of some feature is needed to decide how to map a particular kind of knowledge base token, but is missing from the DSS describing it because it is not part of the standard feature set, it can be added on-demand by querying the knowledge base. This querying facility also addresses the issue of dual input to the lexical choice component in a generation system. This issue was raised by [Reiter 1991] who argued that in order to generate referring expressions appropriately, the lexical chooser needs two inputs: (1) a set of features describing the referent in way that satisfies the current communicative goals of the system and (2) the complete description of the referent in the domain knowledge base. In STREAK, the first input correspond to the default DSS of the referent passed to the DSS$\mapsto$SSS mapper. This mapper, which is responsible for lexical in STREAK, can then access the second input by querying the knowledge base.

This type of query can be implemented in the context of mapping or revision rules using the special procedural attachment feature of FUF `external`. When an `external` feature is encountered, unification is suspended, the procedural attachment gathers the necessary information from the knowledge base and then unification resumes. This mechanism is a general facility that allows dynamic enrichment of a FUF grammar with information from any outside knowledge source. It is presented in detail in [Elhadad and Robin 1992]. It is not yet used in the current implementation of STREAK.

[46] In a revision rule, the feature `bls` stands for Base LS and the feature `rls` stands for Revised LS.

```
((lhs ((adss ((deepsemcat game-stat-ref)
              (referent GIVEN)
              (actor {^ conseq after carrier token-of})
              (conseq ((deepsemcat histo-stat-update-ref)
                       (after ((deepsemcat histo-stat-ref)
                               (instant-game-stat {^3 referent})
                               (carrier ((deepsemcat player-token-ref)
                                         (token-of GIVEN)))
                               (histo-signif ((direction i-high)
                                              (membership-scope i-personal)
                                              (time-scope i-season)))))))))
       (bls ((sss ((dss ((deepsemcat game-stat-ref)
                         (referent {^4 adss referent})
                         (aps ((actor GIVEN)
                               (value GIVEN)))))))
             (syntcat clause)
             (partic ((created ((cat measure)
                                (sss ((dss {^4 sss dss value})))))))))))
 (rhs ((copy {adss conseq} {sss dss conseq})
       (del-fd {sss partic created})
       (add-fd ((surfsemcat proposition-cluster)
                (rhetor-relt ((semantics elaboration)
                              (surfstruct hypotaxis)))
                (main {^ elaborated})
                (dependt {^ elaboration})
                (elaboration ((surfsemcat compound-qual-spec)
                              (concept game-stat-histo-signif))))
               {sss partic created})
       (map-fd {adss conseq after histo-signif}
               {sss partic created elaboration props}
               dss2sss)
       (add-path {^4 dss conseq after histo-signif} {sss partic created elaboration dss})
       (copy {bls sss partic created} {sss partic created elaborated})
       (del-path {sss partic created elaborated dss})
       (add-path {^4 dss value} {sss partic created elaborated dss})
       (del-fd {partic created})
       (add-fd ((cat common) (definite no)) {partic created})
       (map-fd {rls sss partic created elaboration}
               {partic created classifier}
               sss2dgs)
       (add-path {^4 sss partic created elaboration} {partic created classifier sss})
       (copy {bls partic created} {partic created head})
       (add-path {^4 sss partic created elaborated} {partic created head sss})))))
```

Figure 31: An example of revision rule

4.2.4 The surface sentence generator

To produce actual English sentences STREAK will rely on the general purpose surface sentence generator developed by [Elhadad 1992]. This system consists of the functional unifier FUF [Elhadad 1991] underlying the implementation of the other components of STREAK and of the English grammar SURGE (Semantic Unification Realization Grammar of English). FUF produces an English sentence by unifying SURGE with an input Functional Description (FD). In STREAK, this input FD is the DGS of the final draft.

4.3 Summary

The prototype system STREAK implements the most innovative aspects of the general report generation architecture proposed in section 3. It takes as input the structured DSS of a basic corpus sentence and a stack of elementary DSSs representing additional propositions candidate for attachment to that basic sentence. As output, it produces a complex corpus sentence incorporating the basic propositions and as many additional propositions as could fit.

STREAK is made of three interpreters of declarative knowledge, all implemented using FUF: the mapping rule interpreter, the revision rule interpreter and FUF itself as a surface sentence generator. STREAK relies on five knowledge sources: the domain knowledge base, the DSS$\mapsto$SSS mapping rule base, the SSS$\mapsto$DGS mapping rule base, the revision rule base and the English grammar SURGE.

The implementation of STREAK is in an early stage. While a first version of each of its processing components have been implemented, the current knowledge sources can only handle the type of simple example presented in this section to describe each component. This first scaled-down version, however, shows that the draft and revision approach to generation proposed in this paper is sound and promising. The limitations of the current implementation are discussed in detail in section 6.1.

5 Previous work in generation with revision

In itself, the idea of revision in generation is not really new. 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: (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 along these two dimensions.

5.1 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. Using an existing natural language *understanding* system is not an option either 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. 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[47].

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 and plan-elaboration during single-pass generation than it is to the draft reviewing and editing phase in two-pass generation. 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. 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 decisions is affected at once by a single revision[48]. The Layered Specification (LS) in the architecture proposed in section 3 has both these characteristics. Its surface element, the Deep Grammatical Specification (DGS) unequivocally determines a natural language output via the defaults provided by SURGE for unspecified input features. Moreover all its unretracted intermediate decisions are compiled in the Deep Semantic Specification (DSS) and the Surface Semantic Specification (SSS) layers.

Apart from distinguishing information-adding revision from plan-elaboration, the presence of a surface

[47] Note that Meteer's proposal is for future work. SPOKESMAN does *not* perform revision.

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

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: (1) informativity, (2) conciseness and (3) readability. By multiplying the number of candidate facts to convey, dealing with historical information only exacerbates the tension between these conflicting goals.

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 unequivocally 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.

5.2 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 of some unexpanded nodes of the textual schemata used for content planning during the first pass[49] - operates at the *macro*-level (addition of a sentence or a paragraph) whereas those performed by STREAK operates at the *micro*-level (addition of a clause, group or even a single word).

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. Revision in STREAK operates on a pre-existing structured representation, the initial Layered Specification, built during the drafting phase of the generation process. In contrast, YH starts from scratch and the utterance 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. This expensive search is avoided in YH by relying on knowledge extremely specific to the tiny domain of the Dutch National Flag game. It could not be avoided in a more complex domain.

[49] Or in Hovy's term, expanding *growth-points* [Hovy 1990].

6 Current status and remaining work

In this section I review, for each aspect of the research proposed in this paper, what has already been completed and what part of the remaining work I intend to complete before the thesis defense. I thus attempt to delimit the scope of the dissertation work. Since predicting the difficulty and time requirement of future research is somewhat of a gamble, I leave room for adjustments in the face of unforeseen difficulties or delays. I thus divide the agenda of remaining work between "will do" items that I will complete in all scenarios and "may do" items that I will complete only in the best scenarios. These labels define a priority among the different items on the agenda.

The research proposed in this paper has two different aspects that support and complete each other:

- Implementation of the report lead generator prototype STREAK.
- Corpora analysis of human-generated reports conveying historical information.

I review the current status and define the remaining work for these two aspects in turn in what follows. For each aspect, some items on the agenda of remaining work expand the completed work while some others evaluate it.

6.1 Implementation of STREAK

Most of the prototype implementation aspect of the proposed research remains to be done. The target goal of the STREAK prototype is to draft and revise a selected set of sentences from the basketball corpus to demonstrate both usage of a fair variety of revision tools and cross-ranking paraphrasing power. We saw in section 4.2 that STREAK relies on a set of declarative knowledge sources. Implementing such a system involves the four following tasks:

- Defining the architecture.
- Defining the internal representation formalisms.
- Coding the processing components.
- Coding the knowledge sources.

The current status and remaining work with respect to each of these four tasks is reviewed in turn in the following subsections. Evaluation and testing of the completed version of the implementation are then discussed.

6.1.1 Completing the implementation

Architecture design The architecture design has been entirely completed. The general generation model on which STREAK is based was presented in section 3 and the particularities of STREAK's architecture within this model were presented in section 4.

Internal representation formalisms We saw in section 3.3.1 that there are four internal representations of an utterance in STREAK:

- The Deep Semantic Specification (DSS).
- The Surface Semantic Specification (SSS).
- The Deep Grammatical Specification (DGS).
- The Layered Specification (LS).

Concerning the first three representations, little uncertainty remains. Although the set of features at each layer will have to be extended to parallel the extension of the knowledge sources, the role of each of these formalisms is defined well enough to consider the current features as a stable core. The last representation currently consists of a triple <DSS,SSS,DGS> with additional features at the DGS and SSS layers keeping track of the correspondence between the respective subconstituents of each layer. As pointed out in section 3.3.1, it remains unclear whether, in cases of non-isomorphic mapping between two layers, these additional features can be easily filled via standard enrichment during the recursive top-down unification scheme controlling each mapping.

I thus need to investigate what representation of the LS inter-layer constituent correspondence can be maintained the most elegantly and economically through unification. *This is a "will do" item on the agenda of remaining work.*

Coding the processing components We saw in section 4.2 that STREAK is made of three processing components:

- The mapping rule interpreter
- The revision rule interpreter
- The surface sentence generator

I completed the coding of the mapping rule interpreter. As surface sentence generator STREAK uses SURGE. This system, implemented by M. Elhadad and described in detail in [Elhadad 1992], has been extensively tested in a variety of generation applications developed at several research centers.

The remaining work on processing components thus concerns the revision rule interpreter. The first pilot implementation of this interpreter remains incomplete as it does not address the crucial issue of *seed localization*. To describe this issue, I first need to distinguish three sentence subconstituents that play a special role in an information-adding revision: *the cluster*, *the seed* and *the scope*.

Clusters were defined in section 2.4.2 as the two top-level constituents of a corpus sentence. Recall from that section that the information-adding revisions modeled by STREAK are local to a single cluster. Recall also that such a revision must be warranted by a semantic relation holding between the additional proposition and some semantic element of the base cluster. I call "seed of the revision" the minimal subconstituent of the base cluster that realizes the semantic element related to the additional proposition. Recall also that a revision consists of a structural transformation of the base cluster into a revised cluster. This transformation is sometimes local to a strict subconstituent of the base cluster and does not affect the other constituents. I call "scope of the revision" the maximal subconstituent of the base cluster affected by the structural transformation. Distinguishing between cluster, seed and scope is necessary because they do not always coincide. The revision examples of Fig. 32 illustrates this phenomenon.

In the first example, the revision adds an extrema proposition concerning the scoring statistic of John Stockton. The seed of the revision is thus the clause "John Stockton added 27 points" realizing this statistic. This clause is a strict subconstituent of the prominent-statistic cluster. In addition, the structural transformation of the revision does not affect the seed as a whole, but is local to its object NP "27 points". This example is thus a case where the scope is a strict subconstituent of the seed, itself a strict subconstituent of the cluster.

In the second example, the revision adds a streak proposition concerning the game result. This game result is realized by the NP "a 105-95 victory over the Los Angeles Clippers". This NP is thus the seed of the revision. However, the streak proposition is introduced by way of a transformation affecting the entire game-result cluster of the sentence. This is therefore a case where the seed is a strict subconstituent of the cluster, but where scope and cluster coincide. In these first two examples, the respective seed-scope inclusion relation are thus converse.

In the third example, cluster, seed and scope all coincide. It is the entire game cluster "the Seattle Supersonics defeated the Sacramento Kings 106-91" which realizes the game result and it is entirely affected by the revision.

Example of revision where cluster > seed > scope:

S_{b1}: "Los Angeles – <Karl Malone scored 28 points and *John Stockton added 27 POINTS* and handed out 23 assists> Saturday <as the Utah Jazz claimed a 105-95 victory over the Los Angeles Clippers."

S_{r11}: "Los Angeles – <Karl Malone scored 28 points and John Stockton added **A SEASON HIGH** 27 POINTS and handed out 23 assists> Saturday as the Utah Jazz claimed a 105-95 victory over the Los Angeles Clippers."

Example of revision where cluster = scope > seed:

S_{b1}: "Los Angeles – Karl Malone score 28 points and John Stockton added 27 points and handed out 23 assists Saturday < AS THE UTAH JAZZ CLAIMED *A 105-95 VICTORY OVER THE LOS ANGELES CLIPPERS*>."

S_{r12}: "Los Angeles – Karl Malone score 28 points and John Stockton added 27 points and handed out 23 assists Saturday <AS THE UTAH JAZZ **EXTENDED THEIR WINNING STREAK TO FOUR GAMES WITH** A 105-95 VICTORY OVER THE LOS ANGELES CLIPPERS>."

S_{b2}: "Seattle – Shawn Kemp totaled 20 points and 14 rebounds Wednesday night <*AS THE SEATTLE SUPERSONICS DEFEATED* `THE SACRAMENTO KINGS` *106-91*>".

S_{r21}: "Seattle – Shawn Kemp totaled 20 points and 14 rebounds Wednesday night <AS THE SEATTLE SUPERSONICS **HANDED** `THE SACRAMENTO KINGS` **THEIR 33RD STRAIGHT ROAD LOSS 106-91**>".

Font conventions: the <cluster> is bracketed, the *seed* italicized, the SCOPE uppercased, the **added constituent** boldfaced and the `displaced constituents` in a typewriter font.

Figure 32: Cluster, seed and scope

I observed six different patterns of mutual inclusions for cluster, scope and seed in the corpus[50]:
cluster = seed = scope
cluster > scope > seed
cluster > seed > scope
cluster = seed > scope
cluster > seed = scope
cluster = scope > seed

Recall from section 4.2.2, that STREAK implements revision by applying revision rules. The LHS of a revision rule specifies the base realization pattern on which it can be applied. The RHS of a revision rule specifies the structural transformation this base pattern undergoes during the revision. The non-coincidence of cluster and seed implies that the same seed pattern can appear in embedding clusters of various top-level structures. Similarly, the non-coincidence of cluster and scope implies that the same scope pattern can appear in embedding clusters of various top-level structures. Writing revision rules whose base pattern matches entire clusters would therefore lead to a large set of partially redundant and overly specific rules. Instead, the revision rule base should be made of general and modular rules whose base pattern match only the larger of the seed and scope subconstituents. The revision rule given in Fig. 31 of section 4.2.2 is an example of such general modular rules. It could be used for example for the two following revisions:
S_{b1} "John Stockton scored 27 points"
S_{r1} "John Stockton scored **a season-high** 27 points"

S_{b2} "Karl Malone scored 28 points and John Stockton added 27 points and handed out a league-high 23 assists"
S_{r2} "Karl Malone scored 28 points and John Stockton added **a season-high** 27 points and handed out a

[50]X>Y means Y is a strict subconstituent of Y. X=Y means X and Y coincide.

league-high 23 assists"

In the second example, however, the seed of the revision, the embedded clause, "John Stockton added 27 points" must be localized inside the entire cluster. Seed localization thus consists of searching the seed down the complex structure of the base sentence. The current version of the reviser does not perform such a search. As a consequence, the current version of STREAK can revise only individual clusters (as opposed to whole sentences) and only those clusters that coincide with both seed and scope.

Upgrading the revision rule interpreter so that it performs seed localization is thus required before STREAK can generate whole corpus sentences through draft and revision. *This upgrade is the "will do" item of highest priority on the agenda of remaining work.*

In the current version of STREAK the revision process is controlled using a very simple strategy: Additional DSSs (ADSSs) are popped from the input stack of floating propositions, one after another, until the stack is empty. Moreover, if for the popped ADSS, there is no revision rule for incorporating it to the current draft, it is discarded and the next ADDS is popped. Another upgrade of the revision rule interpreter may be needed for implementing two facilities making the control of the revision process more flexible:

- Halting the revision process before the stack of ADSSs is empty, upon detection that the draft sentence has reached the threshold of maximal complexity[51] observed in the corpus.
- When no revision rule applies to the current pair <draft-LS,ADSS>, backtrack to the previous revision increment(s).

It remains unclear whether granting the reviser the ability to detect that some complexity threshold has been reached and halt the revision process can be simply implemented by a set of revision rules or whether it requires upgrading the revision rule interpreter and/or the architecture of the system. *Investigating this issue is a "will do" item on the agenda of remaining work.*

Concerning the issue of backtracking, consider for example a case where there are two revision rules R_{1a} and R_{1b} to incorporate to the draft the first element P_1 in the stack of additional propositions and that the revision rule interpreter randomly chooses R_{1a}. Now suppose that there is no revision rule for incorporating the next additional proposition P_2 to the revised draft resulting from the application of R_{1a}, but that one revision rule could incorporate P_2 to the alternative draft that the application of R_{1b} would have produced. In such a case, the revision rule interpreter should be able to backtrack to the previous revision increment, restore the original draft and apply R_{1b} to it.

I have not yet determined whether the corpus contains sentences that could not be generated without such a backtracking facility. Backtracking is expensive and should thus be used only if it is really needed. *I therefore propose to put the implementation of a backtracking facility in the revision rule interpreter on the agenda of remaining work as a "may do" item.*

Coding the knowledge sources We saw in section 4.2 that STREAK relies on five knowledge sources:

- The historical knowledge base
- The DSS$\mapsto$SSS mapping rule base.
- The SSS$\mapsto$DGS mapping rule base.
- The revision rule base.
- The English grammar SURGE.

I review the current status of each knowledge source in turn in the following paragraphs. The data collected during the corpus analysis provides a sound foundation for the implementation of these knowledge sources. However, most of this implementation remains to be done.

[51] Whatever measure is used to define this threshold *e.g.,* or(proposition-number = 9, word-number = 46).

Historical knowledge base Implemented in CLASSIC, the historical knowledge base consists of (1) a conceptual hierarchy representing the domain ontology of the corpus sentences and (2) a set of tokens representing the box-score statistics and related historical facts conveyed in a first set of test sentences selected from the corpus. As the coverage of the corpus sentence ontology improves, both more concepts and more tokens will need to be defined. However, the higher portion of the current conceptual hierarchy should constitute a stable core.

Revision rule base and mapping rule bases The coding of all three rule bases (the two mapping rule bases and the revision rule base) remains to be done almost entirely. In its current version, the mapping rule bases of STREAK allow the generation of only four of the corpus' realization patterns. Similarly the current version of the revision rule base encodes only one instance of revision tool: nominal-rank adjoin of adjectival classifier. However, for these realization patterns and this revision tool, example runs involving the coordinated action of all the processing components in the architecture have been successfully tested. These examples show the validity of the approach. Populating the two mapping rule base and the revision rule bases will probably constitute the bulk of the remaining work. Although this is clearly a "will do" item, more precisely defining how much should be done raises the issue of target coverage. I address this issue in section 6.1.2, where I propose a set of criteria for evaluating the completed implementation.

Independently of this issue of coverage, the mapping rule base should contain rules demonstrating two facilities of the generation architecture proposed in section 3: handling of non-isomorphic mappings and query of the historical knowledge base from the realization modules. Since I have not yet determined, how pervasive is the need for these facilities among the corpus sentences[52], *I propose to make the writing of rules implementing these two facilities a "may do" item on the agenda of remaining work.*

The English grammar SURGE In its current version, the portable English grammar SURGE that STREAK will use for surface sentence generation does not cover some syntactic constructs pervasively used in report generation, including:

- To-infinitive, present participial and past participial adjunct clauses.
- Parenthetical adjuncts and numbers.
- Complex NPs with (possibly compound) proper names as head.

Even the most basic sentences of the basketball corpus contain such constructs, as illustrated by the example below[53]:
DALLAS, TX - Otis Thorpe scored 28 points and grabbed 16 rebounds and Akeem Olajuwon registered 24 points and 15 rebounds SATURDAY NIGHT, leading the Houston Rockets to a 113 107 victory over *the slumping Dallas Mavericks*, losers of five in a row.

I must therefore extend the coverage of SURGE to these constructs. *This is a "will do" item on the agenda of remaining work.* Since SURGE must remain domain-independent[54], this extension entails defining the functional roles of these additional adjuncts in a principled way compatible with the current set of circumstancial roles used by SURGE.

6.1.2 Evaluating and testing the implementation

The last "will do" item on the agenda of remaining work is to test and evaluate the completed implementation. STREAK is not an application oriented system, but an experimental research prototype. It should thus be

[52] This question can be precisely answered only by experimenting with a variety of sentence types in the course of the implementation.

[53] In this example the participial adjunct clause is underlined, the parenthetical adjuncts and numbers are slanted and the complex NP headed by a compound proper name italicized.

[54] Note that without these constructs specifically needed for report generation, SURGE is already one of the most extensive generation grammars available (cf. [Elhadad 1992]).

judged only as a testbed for the innovative solutions proposed in this paper. The two main innovative aspects of STREAK are:

- Its usage of the revision tools identified by the corpus analysis to convey historical information.
- Its very compositional architecture achieving semantic density, surface form complexity and surface form variety in an extensible way.

Evaluating STREAK should thus consist of determining how well it demonstrates:

- The operational usage of the revision tools presented in section 2.5.2 and appendix B.
- The gains of its revision-based microcoded architecture over the classic single-pass macrocoded architecture in terms of extendibility, semantic density, surface form complexity and surface form variety.

In the following two paragraphs, I propose a set of criteria for carrying out each of these two aspects of the evaluation

Evaluating STREAK's revision tool usage In order to demonstrate the operationality of the revision tools presented in section 2.5.2 and appendix B, STREAK must generate a test set of corpus sentences that covers a sufficient variety of revision tool usages. Variety of usage can be measured using the dimensions along which revision tools were classified in this paper[55]. The test set will cover a sufficient variety of tool usages when it will contain usages of:

- At least one simple tool and one complex tool.
- At least one tool with side transformation and one tool without side transformation.
- At least one tool applicable to a prominent-statistic cluster and tool applicable to a game-result cluster.
- At least one tool to add streak information, one tool to add extrema information and one tool to add *non*-historical information.
- At least one tool resulting in a paratactic structure, one tool resulting in an hypotactic structure with the seed as head and one tool resulting in an hypotactic structure with the additional constituent as head.
- At least one nominal-rank tool and one clause-rank tool.
- At least one tool for which the seed strictly contains the scope, one tool for which the scope strictly contains the seed and one tool for which seed and scope coincide.
- A subset of tools applicable on the *same* base cluster to add the *same* additional proposition (in order to demonstrate revision-based paraphrasing power)
- A subset of tools to add the *same* additional proposition such that each one is applicable only to one of several synonymous base clusters differing in surface form (in order to demonstrate surface form constraints on revision tool applicability)
- An ordered subset of tools such that each tool can be applied to the revised cluster produced by the preceding tool (in order to demonstrate revision chaining).

Such a test set defines a target coverage for STREAK knowledge sources: the historical knowledge base, the two mapping rule base and the revision rule base. These knowledge sources need to contain only the rules necessary to generate a test set satisfying the above list of criteria. The fact that these criteria exhaustively cover the dimensions along which the revision tool differ from one another insures that further knowledge acquisition would become a mechanical task.

[55] Most of these dimensions were defined in section 2.5.2.

Evaluating STREAK's architecture Evaluating STREAK's architecture involves measuring the benefits of the revision-based microcoded approach over the classic single-pass macrocoded approach in terms of semantic density, surface form complexity surface form variety and extensibility. In what follows, I define a set of parameters to quantitatively measure each of these factors. For each parameter, both the maximum value and the average value bring useful insights.

Semantic density can be measured by the number of propositions in the generated sentences and surface form complexity by the number of open-class words.

For a report generator, the only relevant measure of surface form variety is the paraphrasing power[56], i.e. the number of synonymous sentences generable per conceptual input. In the case of STREAK, this conceptual input is the pair <structured DSS, stack of additional DSS>.

What makes a revision-based microcoded generator more easily extendible than a single-pass macrocoded generator is that it is more compositional. The degree of compositionality of a generator can be measured by the three following parameters:

- *Microcoding factor*, the number of lexicon entries dynamically assembled per generated sentence.
- *Modularity factor*, the number of declarative rules used per generated sentence.
- *Utility factor*, the number of sentences generable involving the same declarative rule.

6.2 Corpora analysis

Most of the corpora-analysis aspect of the proposed research has already been completed. The systematic and quantitative analysis of the basketball summary corpus presented in section 2 and in the appendices produced the following results:

- An ontology of the basketball report domain. In STREAK, this ontology is represented in the historical knowledge base conceptual hierarchy.
- A set of sentence-rank schemata defining how propositions of various ontological classes can be combined inside complex sentences. In STREAK, these schemata will be encoded in part inside DSS$\mapsto$SSS mapping rules and in part inside revision rules.
- A set of realization patterns determining which syntactic structures can be used to express each legit proposition combination. In STREAK these realization patterns will be encoded inside the SSS$\mapsto$DGS mapping rules.
- A set of revision tools to incrementally build complex realization patterns from basic ones. These revision tools will be encoded inside the RHS of the revision rules.
- A set of semantic and syntactic constraints on the applicability of these revision tools. In STREAK these constraints will be encoded inside LHS of the revision rules.

Quantitative data about these results are provided in the appendices. Corpora-analysis can be pursued in two directions: completion and evaluation. For completion, the basketball corpus can be searched for another type of constraint on revision tools. For evaluation, the domain-specificity of the set of revision tools identified in the basketball corpus can be estimated by looking for their usage in a corpus of reports from another domain. I discuss each direction in the following two paragraphs.

6.2.1 Completing the corpus analysis

The corpus analysis of the basketball game summaries carried out so far provided three types of constraints on the applicability of a given revision tool to add a proposition to a base cluster inside a draft sentence:

[56] The overall variety of generable syntactic constructs, a good measure of syntactic for a general purpose surface sentence generator, is immaterial for a report generator since it is inherently limited to the sublanguage of its application domain.

- The semantic content of the additional proposition.
- The semantic content of the base cluster.
- The syntactic form of the base cluster.

The fact noted in section 2.4.2 that the two clusters of a corpus sentence are independent both in content and form, indicates that additional constraints should be searched only locally, at the cluster level. Since the corpus analysis has established that both the semantic content and the syntactic form of the base cluster constrain revision tool applicability, the natural follow-up question is to ask whether the lexical material of the base cluster is also a factor.

Answering this question requires introducing the notion of *lexical increment*. This notion is a specialization of the notion of surface increment defined in section 2.5.1. A corpus cluster C_r is a lexical increment of another corpus cluster C_b, if the realization pattern of C_r is a surface increment of the realization pattern of C_b and if C_r lexicalizes the semantic elements it shares with C_b by the same words. So for example, C_r below is a lexical increment of C_{b0}, but not of C_{b1}, although it is a surface increment of both:
C_r "lifting the San Antonio Spurs to a 127-111 victory over the Denver Nuggets, **and handing the Nuggets their seventh straight loss**"
C_{b0} "to lift the Utah Jazz to a 105-95 victory over the Los Angeles Clippers"
C_{b1} "leading the Chicago Bulls to a 128-94 win over the New Jersey Nets"

Lexical constraints on revision tools can be identified by carrying out the following operations for the prominent-statistic cluster and/or the game-result cluster:

1. Group the realization patterns for the concerned cluster as triples $< R_b, R_{r1}, R_{r2} >$ such that both R_{r1} and R_{r2} are surface increments of R_b via two different revision tools T_1 and T_2.
2. For each such triple, put the corpus sentences using each of the three patterns R_b, R_{r1} and R_{r2} in three corresponding sets S_b, S_{r1} and S_{r2}.
3. Identify the *lexical* increment relations between the elements of S_b and S_{r1} on the one hand and the elements of S_b and S_{r2} on the other.

A priori, each element in S_b may have several lexical increments in S_{r1} and/or S_{r2}. If the elements of S_b sharing a given word w all have their lexical increments in S_{r1}, for example, it suggests that the presence of w in a base cluster requires to choose T_1 over T_2 to revise this cluster. This would indicate a lexical constraint on revision tool usage. However, having all lexical increments relations cutting across surface increment relations would indicate that there is no such constraint.

Intuitively, I expect this second outcome. That is I expect the analysis outlined above to bring a negative result. Confirming this hypothesis by a systematic analysis would require an exhaustive search using the complex procedure described above. Priority should be given to items for which positive results are anticipated since they can be obtained without such an exhaustive search. *I thus propose to put this analysis as a low-priority "may do" item on the agenda of remaining work*[57].

Investigating the impact of other potential constraints on revision tool usage such as interpersonal or discursive factors is definitely beyond the scope of the research proposed here.

6.2.2 Evaluating the corpus analysis

I plan to evaluate the corpus analysis results along three dimensions: operationality, portability and coverage. I propose evaluating methods for each of these three dimensions in the following paragraphs.

[57] Another reasons to give a low priority to this item, is that, even if it turns out that there are a few lexical constraints on revision tools, as lexical data, they are likely to be very domain-specific (much more specific than, for example, the syntactic constraints already identified).

Operationality In terms of operationality, it is the STREAK implementation itself which evaluates the data collected from the corpus analysis. Since this data underlies the development of STREAK's knowledge sources, the ability of STREAK to generate corpus sentences will demonstrate their operationality.

Portability Among the various data resulting from the corpus analysis, the ontology, sentence-rank schemata and realization patterns are largely domain-specific. However, I expect most of the revision tools identified in the basketball domain to be portable to other application domains of report generation. The best way to verify this hypothesis, is to search for the usage of these tools in a corpus of human-generated reports in another domain. As a test corpus on which to perform this search, I could use, for example, the corpus of stock market newswire reports that was used at Columbia for building the generation system COOK [Smadja and McKeown 1991].

This second corpus analysis would *not* consist of *discovering* new tools from scratch but merely in *verifying* the usage of the revision tools from the basketball corpus. Focusing on revision tools to convey historical information, such verification can simply consist of checking:

- That the syntactic patterns used in the basketball corpus to convey historical information occur in the stock market corpus.
- That in some of their occurrences, these syntactic patterns convey historical information of a similar type than in the basketball domain[58].

I hope to find in the stock market corpus, syntactic patterns corresponding to the usage of a majority of the revision tools identified in the basketball corpus. *Searching a stock market corpus for these syntactic patterns is a "will do" item on the agenda of remaining work.*

Coverage In order to evaluate the coverage of the corpus analysis results, I propose to adapt to this specific task a general methodology based on the use of two distinct corpora, one for training purposes and the other for test purposes. This methodology consists in four steps:

- Define a training corpus.
- Define a test corpus.
- Define a quantitative parameter in terms of which to measure coverage.
- Evaluate the coverage of the results obtained on the training corpus by measuring this parameter on the test corpus.

This methodology has often been applied for evaluating the coverage of parsing systems. In that case, the training corpus is the set of sentences used as the target during the development of the parser. The test corpus is a distinct and randomly chosen set of sentences. The coverage is measured in term of the proportion of successful parses of the test corpus sentences.

To adapt this methodology to the evaluation of the corpus analysis presented in this paper, I consider the already analyzed corpus as the training corpus. Recall from section 2.2 that this corpus consists in the subset of the sentences conveying only box-score statistics, extrema and/or streaks from the set of opening sentences of the game reports for the 90-91 NBA season. As a test corpus, I compiled the corresponding opening sentence subset[59] but for the game reports of the *following* NBA season (91-92).

Several measures could be considered to evaluate the coverage of the corpus analysis. The particular measure I propose to use is the number of revision tools identified for conveying historical information. This choice is motivated by two properties of these revision tools: (1) they are the most original aspect of the research proposed in this paper and (2) they are the final and most abstract results of the corpus

[58] The second part of this verification is made possible by the fact that, in the basketball corpus analysis, I have focused on two semantic types of historical information, extrema and streak, which are pervasive in quantitative domains.

[59] i.e. those opening sentence conveying only box-score statistics, extrema and/or streaks.

analysis. I therefore propose to look at each sentence of the test corpus and determine, for each of its historical proposition, whether this proposition could have been attached to that sentence by way of one of the revision tools identified on the training corpus. I will then compute the coverage of the corpus analysis as the number of historical propositions in the test corpus attachable by way of the revision tools identified in the training corpus divided by the total number of historical propositions in the test corpus. *This is a "will do" item onthe agenda of remaining work.*

6.3 Summary

In order to complete the research proposed here I will do the following:

- Finalize the draft formalism.
- Upgrade the revision rule interpreter so that it performs seed localization.
- Extend SURGE so that it covers the syntactic constructs observed in the corpus.
- Populate the historical knowledge base, DSS$\mapsto$SSS mapping rule base, SSS$\mapsto$DGS mapping rule base and revision rule base to cover a test set of corpus sentences requiring revision tool usages which: (1) cover all the dimensions along which they differ from one another, (2) illustrates surface form constraint on revision, (3) demonstrates revision-based paraphrasing and (4) demonstrates revision chaining.
- Upgrade the implementation to allow the reviser to detect threshold of maximum sentence complexity and halt the revision process.
- Quantitatively evaluate the completed implementation using the following parameters: semantic density, surface form complexity, paraphrasing power, microcoding factor, modularity factor and utility factor.
- Quantitatively evaluate the coverage of the corpus analysis by measuring the proportion of historical facts in a test corpus that are expressed using one of the revision tools identified in the training corpus.
- Search a stock market corpus for evaluating the domain-specificity of the revision tools I identified in the basketball domain.

In addition, I may also:

- Further populate the knowledge sources to cover cases of non-isomorphic mappings.
- Further populate the knowledge sources to cover cases where a mapper or the reviser needs to access the historical knowledge base.
- Upgrade the revision rule interpreter to allow for backtracking and populate the knowledge sources to cover cases where backtracking is needed.
- Search the basketball corpus for lexical constraints on revision tool usage.

7 Conclusion: contributions of the proposed work

The research proposed in this paper will make contributions to four subfields of natural language generation: report generation, generation architecture, complex sentence generation and generation with revision. I review the contributions to each of this subfield in turn in the following subsections.

7.1 Contributions to report generation

The major contribution of the proposed research to report generation is that STREAK will be the first system to provide the historical background of the new facts it reports. As we have seen in section 1, this improvement over previous report generators constitutes a significant step towards bridging the gap between computer-generated reports and their human-generated counterparts. In particular, conveying historical information allows for:

- A much broader semantic coverage (recall from section 2.2 that a system ignoring historical information would at best cover 35% of the corpus sentences subject of the in-depth analysis).
- A smarter choice of the new facts to report (since the relevance of a new fact is largely dependent on its historical significance).
- A more realistic output.

Another contribution of the proposed research is the corpus-based approach on which it is based. This step-by-step methodology can be followed for the development of report generators in any application domain. The advantage of this corpus-based methodology is that it facilitates the four most delicate phases in the implementation of a generation system:

- Architecture design
- Knowledge acquisition
- Output testing and evaluation
- Definition of stylistic preferences

The benefits of the corpus-based approach concerning architecture design were discussed in section 3.2, those concerning knowledge acquisition were discussed in section 6.2 and those concerning testing and evaluation were discussed in section 6.1.2.

Defining stylistic preferences to choose among the different valid expressions of each proposition combination in a given application domain is one of the most elusive generation problem. Without the guidance of corpus data, all grammatically correct forms need to be considered and solid evidence supporting any preference criteria is generally lacking. The corpus-based methodology offers a very practical answer to this problem: consider only corpus occurring forms and choose between them in a semi-random fashion mimicking their respective frequency of occurrence in the corpus.

7.2 Contributions to generation architecture

In section 3 I proposed a new generation architecture. The major contribution of this new architecture is to be the first to provide a generator with the following two abilities:

- Opportunistic performance of content selection at and below the sentence rank under surface form constraints.
- Realization of floating semantic elements under a combination of semantic, syntactic, stylistic and discursive constraints.

In addition to these two individually unique abilities, another contribution of the proposed architecture is to be the first to *cumulate* a set of abilities which have been *separately* provided by previously proposed architectures:

- Distinction between foreground content to convey obligatorily and background content to convey opportunistically [Rubinoff 1990]
- Lexicalization of reference to a domain entity by considering 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].
- Realization of floating semantic elements at various linguistic ranks [Elhadad 1992].
- Conflation of several semantic elements inside a single syntactic constituent [Elhadad 1992]

The proposed architecture is also interesting in that it combines a unique set of properties whose desirability has been independently advocated in previous work:

- It works in a two-pass draft and revision mode [Mann 1983] [Vaughan and McDonald 1986] [Yazdani 1987] [Wong and Simmons 1988] [Cline and Nutter 1991] [Inui *et al.* 1992].
- It interleaves planning and realization [Appelt 1985] [Hovy 1988].
- It is incremental [De Smedt 1990].
- It uses declarative knowledge sources [Nogier 1990] [Polguere 1990].
- It is stratificational and modular [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] [Elhadad 1992].

It is this combination of properties that allows the proposed architecture to provide for more expressive flexibility than any previously proposed architecture.

7.3 Contributions to complex sentence generation

Independently of the issues of report generation and generation architecture, another original contribution of the research proposed in this paper is that it specifically tackles the issue of complex sentence generation. This issue has been neglected by previous research, which focused on building complex paragraphs and texts by assembling *simple* sentences. STREAK is the first generator[60] that deliberately attempts to pack as many propositions as possible in a single sentence. It will generate sentences significantly more complex than any existing generator has ever produced. The table below compares the maximum number of propositions and words in the sentences produced by existing generators with the corresponding numbers for the corpus sentences which constitutes STREAK's target output. I use the number of propositions as a simple measure of the semantic density and the number of words as a simple measure of syntactic complexity[61].

	Previous systems				Target output of STREAK		
	μcoded	Macrocoded					
	FoG	Ana	Danlos' system	PAULINE	min	avg	max
Proposition number per sentences	4	4	7	7	4	6	9
Word number per sentences	10	28	33	34	21	32	46

In this table it is important to note that the output of FoG [Bourbeau *et al.* 1990] is much simpler than the output of Ana [Kukich 1983], PAULINE [Hovy 1988] and Danlos' system [Danlos 1986]. This difference

[60] To the best of my knowledge.

[61] More sophisticated measures could be devised, for example counting only open-class words or taking into account maximal depth of constituent embedding. I have not yet looked more deeply into this issue.

is not fortuitous but rooted in the fact that ANA, PAULINE and Danlos' system *macro*code their output from entire phrasal patterns hard-wired in their lexicon, whereas FoG *micro*codes every constituent of its output from individual words. The output complexity of FoG is typical of what most existing microcoded generators produce. Existing systems are thus either microcoding simple sentences or macrocoding complex sentences. STREAK will be the first generator to microcode complex sentences.

Microcoding complex sentences is challenging since it involves addressing a much broader range of sentence planning issues than in previous work. However, it is the only way to combine output complexity with paraphrasing power and extensibility. Recall from section 1 that with a macrocoding approach paraphrasing power can only be achieved by hand-coding a great number of partially redundant phrasal patterns in the lexicon of the generator[62].

Generating sentences semantically denser and syntactically more complex than macrocoded systems with the added flexibility of the microcoding approach will be the major contribution of STREAK to complex sentence generation. A lesser contribution will be to generate these sentences in a cognitively plausible way. The experiment of [Pavard 1985] discussed in section 3.2.3 suggests that, for sentences of a complexity comparable to the target output of STREAK, the draft and revision model on which it is based is more cognitively plausible than the single-pass model of previous systems.

7.4 Contributions to generation with revision

The research proposed in this paper will make two important contributions to generation with revision. The first contribution is that STREAK will be the first implementation of a generation system using revision to incrementally make its output more informative. Recall from section 5 that, although the literature contains many proposal for generation with revision, only one such proposal resulted in an implementation, the system WEIVER [Inui *et al.* 1992]. But WEIVER performs only information-*preserving* revisions that makes its output more stylistically elegant. In addition of being entirely new, the information-*adding* type of revision performed by STREAK is also interesting because it brings together generation with revision and incremental generation.

The second contribution of the research presented in this paper to generation with revision, is the extensive list of revision tools to incorporate additional information[63] into draft sentences. This list, exhaustively presented in appendix B, is the end result of the corpus analysis of section 2. One reason that makes these revision tools significant it that they convey three types of information - final cumulative statistics, extrema and streaks - which are pervasive in quantitative domains. I expect to find examples of their usage in stock market reports as well, which would show that their applicability extend to report generation from statistics in general. Another reason that makes these revision tools significant is that they constitute the first set of corpus-identified linguistic resources specifically geared to the task of performing elaborations under surface form constraints. I expect that the growing interest concerning both corpus-based generation and incremental generation will emphasize the importance of this type of linguistic data in the future. In addition, this type of data can have interesting applications beyond pure generation. For example, in the context of a computer assisted document production system the various revisions available to elaborate a document sentence could be presented as menu options.

[62] See also [Smadja and McKeown 1991] for a detailed discussion of this issue.

[63] Including information about the historical background.

A Domain ontology of the corpus sentences

In section 2.2, the basketball reports corpus analysis underlying the research presented in this paper was focused on the report first sentences which contained only three types of information: box-score statistics, extrema and streaks. In section 2.3, I briefly described the ontology of this subcorpus. In this appendix, I provide more details on this ontology which will be encoded in STREAK's historical knowledge base as a CLASSIC concept hierarchy.

This ontology is defined by a hierarchy of argument classes and a hierarchy of predicate classes. The argument classes represent the different types of entities mentioned in the corpus while the predicate classes represent the different types of propositions conveyed about these entities. In this appendix, I present the predicate hierarchy. The highest level distinctions in this hierarchy are based on the predicate-argument structure. The lower level distinctions are based on the ontological type of the argument fillers.

There are eight different predicate-argument structures. Six of these structures correspond to the six main types of box-score statistics (i.e. non-historical propositions). They are the following:

- `simple-statistic(author,value,unit)` as in (1) *"John Stockton scored 27 points"* where *"John Stockton"* is the `author`, *"27"* is the `value` and *"point"* is the `unit`. This predicate class specializes in function of the value of its `unit` and the ontological type of its `author`. In addition to a player, `author` can be filled by a team as in (2) *"the Indiana Pacers sank 60.5 percent of their field-goal attempts"* or a group of players of the same team as in collective coordinations like (3) *"Dominique Wilkins, Spud Webb and Glenn Rivers combined for 69 points"*[64].

- `abstract-statistic(author,type)`. These statistics abstract several simple-statistics by just specifying constraints on their values. These values are abstracted in different ways depending on the `type` of the abstract-statistic. The `type` filler also constrains the ontological category of the `author` filler. The type `double-figure` abstracts scoring statistics by several players of the same team by specifying that the range of their respective values is lower-bounded by 10 points as in (4) *"four other Phoenix players hit in double figures"*. The type `triple-double` abstracts three different statistics of the same player by specifying that the range of their respective values is lower-bounded by 10 as in (5) *"Akeem Olajuwon registered a triple-double"*. The type `balanced-attack` abstracts the scoring statistics of all the players of a team by specifying their even value distribution as in (6) *"Michael Jordan led [a balanced scoring attack] with 24 points"* where it is realized by a nominal in a sentence also conveying a simple-statistic.

- `attributive(carrier,attribute)` as in (7) *"[reserve guard Sleepy Floyd] added 24 points"* where the `carrier` is *"Sleepy Floyd"* and the `attribute` *"reserve guard"*. In addition to a player, the `carrier` can be a simple-statistic as in (8) *"Patrick Ewing scored [a team-high 24 points] "* or a game-result as in (8') *"to give the Philadelphia 76ers a [112-100 overtime victory] over the New Jersey Nets"*.

- `game-result(winner,loser,score)` as in (9) *"the San Antonio Spurs defeated the Seattle Supersonics 121-111"* where the `winner` is *"the San Antonio Spurs"*, the `loser` is (10) *"the Seattle Supersonics"* and the `score` is *"121-111"*.

- `location` and `time`, the spatial and temporal circumstances of the reported game. Since these circumstances are implicitly shared by all predicates in the report, they are best viewed as independent propositions (as opposed to be considered as an argument of one of the other predicates describing the game).

The two remaining predicate-argument structures correspond to the two main types of historical proposition of the subdomain:

- `streak(author,aspect,result-type,number,location,absoluteness,adversary-scope)`

[64] but *not* as in *segregatory* coordinations like (3') *"Magic Johnson and Byron Scott scored 21 points apiece"* which realize *two distinct* simple-statistics that happen to have equal value and unit arguments.

- `extremum(carrier,aspect,direction,membership-scope,time-scope)`.

`streak(author,aspect,result-type,number,location,absoluteness,adversary-scope)` represents the historical property of a team to have had similar results in consecutive games:

- The team fills the `author` argument.
- The `aspect` argument specifies whether the last game-result prolonged the streak as in (17) *"the Chicago Bulls coasted past the Washington Bullets 118-94 [for their fourth straight win] "* or interrupted it as in (18) *" [the New York Knicks broke a three game losing streak] with a 112-97 victory over the Philadelphia 76ers"*.
- The `result-type` distinguishes between winning streaks as in (17) and losing streak as in (18).
- The `number` argument specifies the length of the streak in games.
- The `location` argument specifies whether the streak is only for home games as in (18) *"the Boston Celtics defeated the Phoenix Suns 132-103 [for their 15th consecutive home victory] "*, only for road games as in (19) *"the Boston Celtics triumphed 119-109 over the Miami Heat [who have lost 11 consecutive road games] "* or for both as in (17).
- The `absoluteness` specifies whether the streak runs since the inception of the season as in (20) [*"The Boston Celtics kept the Denver Nuggets winless on the road] with a 148-140 victory"*.
- The `adversary-scope` argument specifies whether the streak is only for games against a specific adversary as in *"Minnesota triumphed 100-94 over the Dallas Mavericks [who are 0 4 against the Timberwolves] "*[65].

`extremum(carrier,aspect,direction,membership-scope,time-scope)` represents the historical property for a given event to constitute an extrema in its class of events:

- The `carrier` argument specifies the event that constitutes an extremum. It is generally a simple-statistic as in (11) *"Kevin Johnson matched his career-high of 44 points"*, but it can also be a game-result as in (12) *"Michael Jordan had 25 points pacing the Chicago Bulls to a 122-87 victory over the Atlanta to hand the Hawks their worst defeat of the season"* or a streak as in (13) *"Michael Jordan scored 40 points leading the Chicago Bulls to a new club record 16th straight home victory"*.
- The `aspect` argument specifies whether the carrier breaks the current record as in (13) or just equals it as in (11).
- The `direction` argument distinguishes between maxima and minima.
- The `membership-scope` argument specifies the author reference set of the `carrier`. It can be either the carrier author alone as in (11), the set of his teammates as in (14) *"Armon Gilliam scored a franchise-record 39 points"* or the set of all the players in the league as in (15) *"John Stockton handed out a league-high 23 assists"*.
- The `time-scope` argument indicates the temporal reference set of the `carrier` filler. It can be either the current season as in (15) or the entire existence of the author referent set filling the `membership-scope` argument as in (10) and (16) *"Scott Skiles set an NBA record with 30 assists"*.

The distinction between the eight predicate-argument structures above defines the top-level proposition classes of the domain. Some arguments of these predicate-argument structures can be filled by domain entities of different ontological types (e.g., the `carrier` argument of an extremum predicate can be filled by either a simple-statistic, a game-result or a streak). The top-level proposition classes can thus be decomposed into subclasses, each corresponding to a different set of argument filler types. However, since the goal of this proposition classification was to identify the various realization patterns of each proposition class combination, the only relevant argument types distinctions are those that consistently influence surface realization. They are the following:

[65] The Timberwolves is the name of the basketball team based in Minneapolis.

- The `author` of the `simple-statistic` predicate classes. For this argument I distinguished among teams and player groups of different cardinality. Orthogonally, I also distinguished between winners and losers[66].
- The `type` of the `abstract-statistic` predicate class.
- The `carrier` of the `attributive` predicate class, for which I distinguished among player, simple-statistic and game-result fillers.
- The `carrier` of the `extremum` predicate class. For this argument I first distinguished among simple-statistics, game-results and streaks. I then further distinguished the different types of simple-statistics and streaks.
- The `aspect` of the `extremum` predicate class distinguishing between record-equalling events and record-breaking events.
- The `result-type` of the `streak` predicate class distinguishing between winning and losing streaks.

The combination of the distinctions in terms of predicate-argument structure with the distinctions in terms of argument filler types decompose the semantic domain of the corpus sentences into 42 proposition classes.

[66] which was possible because the game-result was one of the four propositions present in every subcorpus sentence

B Information-adding revision tools

A classification of the revision tools I identified through corpus analysis was presented in section 2.5.2. This classification starts by two important distinctions: (1) the distinction between simple and complex revision tools and (2) the distinction between the core transformations which characterize each revision tool and the side transformations which are used in combination with several revision tools. In section 2.5.2, I only discussed one simple revision tool, one complex revision tool and one side transformation.

In this appendix, I exhaustively present the simple revisions, complex revisions and side transformations I observed in the corpus. The presentation of each tool includes: (1) a figure showing its general structural transformation schema and (2) an example of surface increment pair <base-cluster, revised-cluster> from the corpus illustrating its usage.

The pictorial conventions used in the figures are the following:

- Constituents are circles or ovals.
- Structural relationships are lines.
- The lines corresponding to role relations[67] are labeled.
- Paratactic relation between two constituents is represented by grouping them under a "pseudo-head" tagged by an "&" sign[68].
- The elements **added** by the revision, either constituents or relations, are boldfaced.

The font conventions used in the surface increment pair examples are the following:

- The **additional constituent** is boldfaced.
- The *seed* constituent is in italics (unless it coincides with the entire base cluster).
- The SCOPE constituent is uppercased (unless it coincides with the base entire cluster).
- The <u>deleted constituents</u> are underlined.
- The `displaced constituents` are in a typewriter font.

B.1 Simple revisions

Procedurally, a revision tool is defined by a set of transformations to change the base pattern into the revised pattern. The first element of any such set is an *introductory* transformation, whose goal is to attach a new constituent onto the base pattern. This new constituent realizes the additional proposition introduced by the revision.

For *simple* revision tools, the set of transformations is reduced to a singleton consisting of this introductory transformation that does not affect the existing realizations of the base propositions.

I identified four main types of simple revisions: *adjoin, append, conjoin* and *absorb*. They differ from each other in terms of either the type of the base structure on which they can be applied or the type of revised structure they produce. I present each of them in turn in the following subsections.

B.1.1 Adjoin

Adjoin was already presented in section 2.5.2. It applies only to *hypotactically* structured bases and consists of the introduction of an additional constituent Ac appearing as an adjunct. Ac is thus *adjoined* to the base constituent under the base head Bh. The general schema of an adjoin is shown in Fig. 33.

[67] i.e. hypotactic relations, either argument or adjunct.

[68] In coordinations this pseudo-head is realized by a conjunction, in appositions simply by a comma.

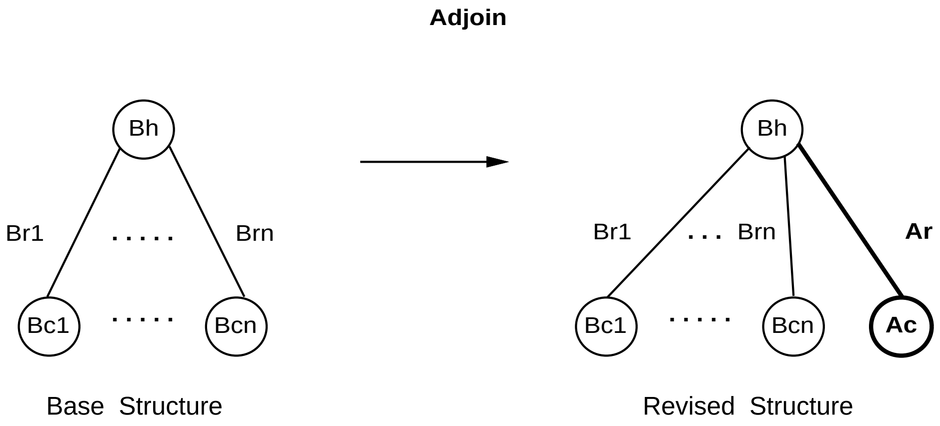

Figure 33: adjoin schema

In the corpus, adjoin is used at both the clause and nominal ranks. It is used for adding both historical and non-historical propositions. The surface increment pair (b1,r1) below provides an example of adjoin at the clause-rank adding an historical proposition:

(b1) "Chicago defeated Phoenix 99-91"

(r1) "Chicago defeated Phoenix 99-91 **for its third straight win**"

In this example, a streak proposition is introduced by simply adding the purpose PP adjunct - *"for its third straight win"* - to the base cluster.

Note that in this example, as well as in all the other examples of this appendix, only the revised cluster is an authentic corpus cluster. The base cluster has the grammatical structure of an authentic corpus cluster but not its original words. These original words have been substituted by those of the revised cluster. So for example an actual corpus cluster from which the structure of (b1) comes from is:

(b1') "Seattle defeated Sacramento 121-93"

Keeping the lexical material for the base and revised clusters in the presented examples makes the nature of the structural transformations involved more striking. This is desirable because I characterized the revision tools in a way that abstracts from lexical variations and focuses on structural transformations. The realization patterns of b1' and r1 were both given in Fig. 15 of section 2.5.1: R_b1 for b1' and R_r1 for r1.

B.1.2 Append

Append simply consists of adding an element Ac onto a *paratactically* structured base. Ac is thus *appended* to the base. There are two different types of such paratactic structures: coordinations and appositions. The general schema of an append is shown in Fig. 34.

The surface increment pair (b2,r2) below provides an example of revision by appending:

(b2) "Charles Smith scored 27 points and *Benoit Benjamin contributed 19 POINTS AND 16 REBOUNDS*"

(r2) "Charles Smith scored 27 points and Benoit Benjamin contributed 19 POINTS, 16 REBOUNDS AND **SIX BLOCKED SHOTS**"

The NP "six blocked shots" realizing the value and unit of a simple statistic of Benoit Benjamin is appended to a base NP coordination realizing the value and unit of two other simple statistics of that player.

The revision from b2 to r2 is an example of append at the nominal rank. Append is also used at the clause rank in the corpus. It is only used, however, on coordinations, not on appositions. One reason for this might be the general rarity of appositive constructions with more than two elements in English. Another

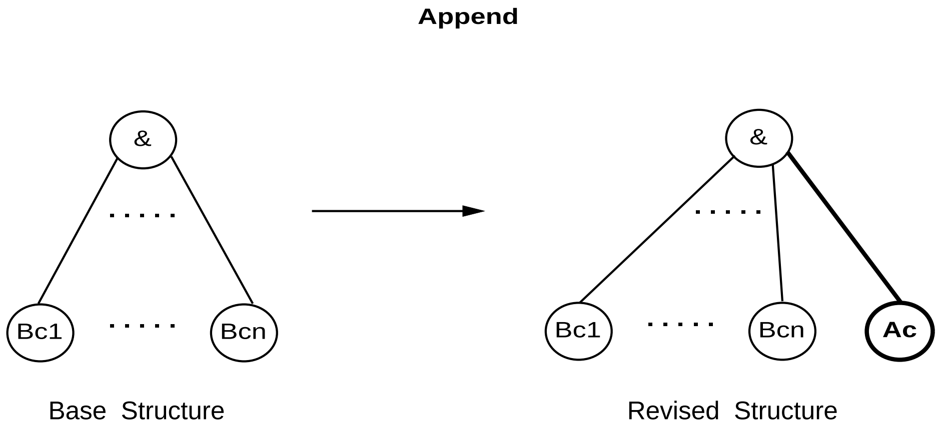

Figure 34: append schema

restriction on append that I observed in the corpus is that it was used only to add non-historical propositions.

B.1.3 Conjoin

Conjoin consists of replacing the base constituent Bhc by a *new paratactic* structure grouping this base constituent with the additional constituent Ac. Bhc and Ac are thus *conjoined* in this new paratactic structure. The general revision schema of conjoin is given in Fig. 35.

In the revised cluster, this new paratactic structure occupies whatever position the base constituent was occupying in the base cluster. Conjoin is thus a very versatile revision tool that can be applied to bases of all linguistic ranks[69], all internal structures, and all roles inside the base cluster. Moreover, it can be used to add either historical or non-historical propositions.

The surface increment pair (b3,r3) below provides an example of conjoin at the clause rank to add an historical proposition:

(b3) "to enable the Philadelphia 76ers *TO DEFEAT THE ATLANTA HAWKS 107 103*"

(r3) "to enable the Philadelphia 76ers TO DEFEAT THE ATLANTA HAWKS 107 103 **AND WIN THEIR SIXTH STRAIGHT GAME**"

[69] At least at all ranks where either coordination or apposition occurs.

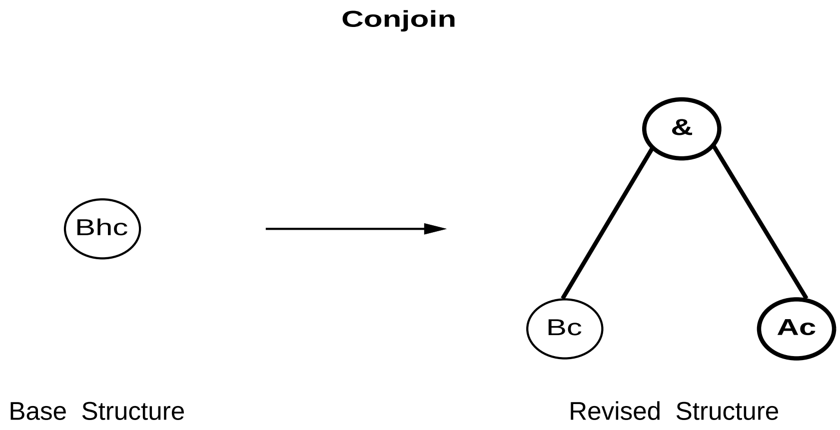

Figure 35: conjoin schema

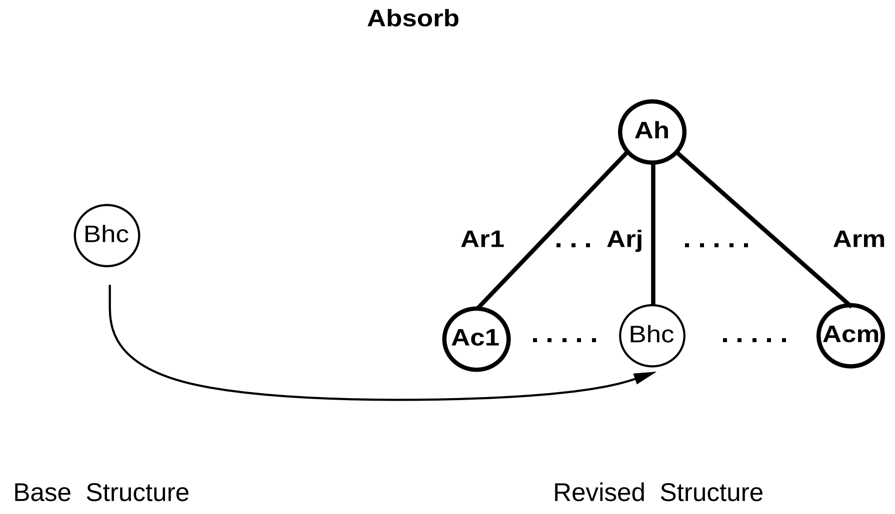

Figure 36: absorb schema

In this example, a streak proposition is introduced by coordinating an additional VP with the game result VP of the base cluster.

B.1.4 Absorb

Absorb consists of replacing the base constituent Bhc by a *new hypotactic* structure in which Bhc appears as a subconstituent. Bhc is thus *absorbed* by this new hypotactic structure. The general revision schema of absorb is shown in Fig. 36.

In the revised cluster, this new hypotactic structure occupies whatever position the base constituent was occupying in the base structure. In this respect, absorb is like conjoin and thus just as versatile in terms of base cluster structures on which it can be applied. Absorb differs from the three previously presented tools because it is the only one resulting in a revised cluster where the relation between the base constituent and the added constituent is hypotaxis *with the added constituent as head.*

The surface increment pair (b4,r4) below provides an example of absorb at the nominal rank:

(b4) "Larry Bird scored 29 points including *SEVEN 3 POINTERS*"

(r4) "Larry Bird scored 29 points including **MATCHING HIS OWN CLUB RECORD WITH** `SEVEN 3 POINTERS`"

In this example, the gerund expressing the additional extrema information absorbs the seed NP "seven three pointers" and replaces it as the PP complement of the cluster. In the gerund revised scope, the base seed appears as an instrument adjunct.

The corpus also contained absorb examples at the clause rank both for adding historical and non-historical information. In both cases, however, the core absorption transformation was accompanied by a side argument control transformation. The presentation of an example of such case is thus postponed to subsection on side transformations.

B.2 Complex revisions

In *complex* revisions a single transformation cannot account for the structural differences between the base pattern and the revised pattern; the introductory transformation attaching the additional proposition onto the base pattern is followed by *restructuring* transformations. The goal of these restructuring transfor-

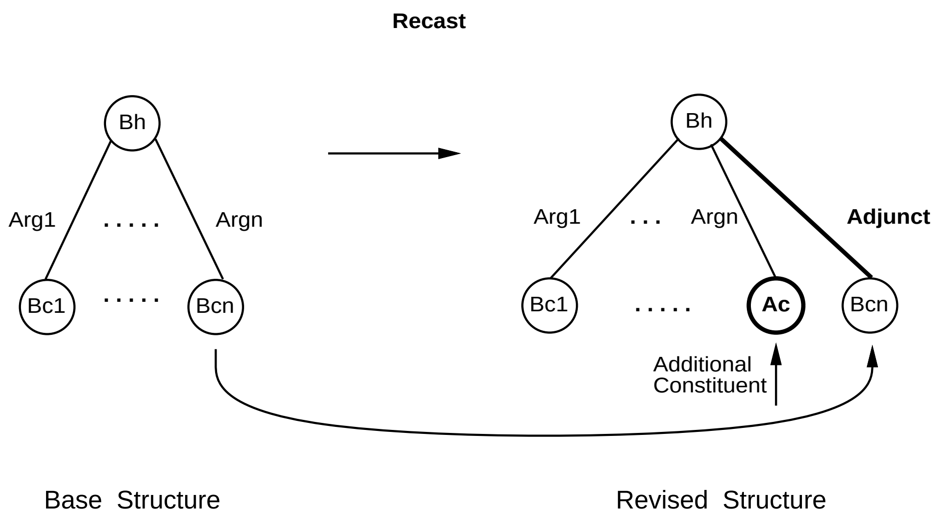

Figure 37: recast schema

mations is to maintain grammaticality despite the structural shake up brought about by the introductory transformation.

I identified six main types of complex revisions: *recast, argument demotion, nominalization, adjunctization, constituent promotion* and *coordination promotion*. Each type is characterized by a different set of restructuring transformations which involve displacing base constituent, altering the base argument structure or changing the base lexical head. These types of transformations also distinguishes complex revisions (as a whole) from simple revisions. Simple revisions conserve both the argument structure and the lexical head of the base and does not involve moving base constituents around. I present each type of complex revisions in turn in the following subsections.

B.2.1 Recast

A recast is any type of transformation that involves displacing a base subconstituent to accommodate an additional constituent while preserving both the argument structure and the lexical head of the base. The simplest of such transformation consists of having (1) an additional constituent Ac taking the place of a base constituent Bcn as a filler of an argument Argn of the base head Bh and (2) Bcn moved to an adjunct position. This transformation is illustrated in Fig. 37. Although in theory there might be more complex types of recast involving other constituents swapping argument position, all the recasts found in the corpus were following the simple schema of Fig. 37. Moreover, recast was used in a rather specialized way: only at the clause rank and only to add streak information.

The surface increment pair (b5,r5) below provides an example of recast:

(b5) "Charlotte took *a 123 111 victory over the Atlanta Hawks*"

(r5) "Charlotte took **its third straight win in** `a 123 111 victory over the Atlanta Hawks`"

b5 and r5 have the same head, the support verb "to take". However, in b5, the head verb "to take" does not realize any semantic element but just syntactically supports its range role, the seed NP "*a 123 111 victory over the Atlanta Hawks*" which realizes the game-result. In r5, "to take" now supports the NP "**its third straight win**" which realizes an additional streak proposition. This added NP replaced the seed NP as the range role filler. The seed NP is *recasted* as an adjunct PP complement. This revision is a recast because, while it involves moving a base subconstituent around (the seed NP), it preserves both the `process(agent,range)` argument structure of the base cluster and its head "to take".

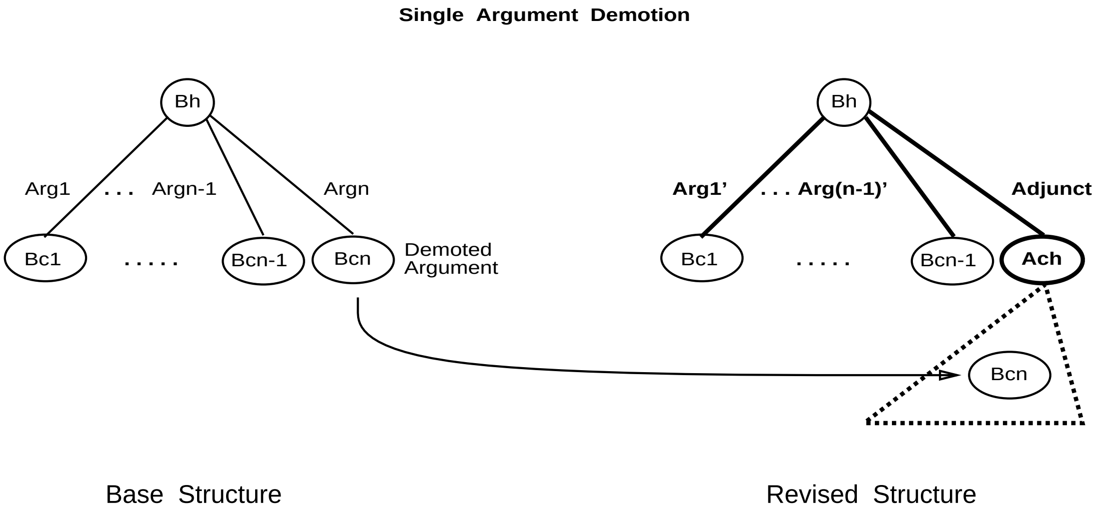

Figure 38: Single argument demotion schema

B.2.2 Argument demotion

An argument demotion is any type of transformation that involves displacing a top-level argument of the base to an embedded position inside the constituent realizing the additional proposition. An argument demotion preserves the lexical head of the base but shrinks its argument structure. This is possible because some lexical heads are compatible with argument sets of various length. I found two subcases of argument demotion in the corpus: single argument demotion and double argument demotion. In the single argument demotion case, the additional constituent appears as an adjunct as shown in the schema of Fig. 38.

The surface increment pairs (b6,r6) below provides an example of single argument demotion:

(b6) "the Golden State Warriors triumphed 110 105 over the Boston Celtics"

(r6) "the Golden State Warriors triumphed 110 105 **snapping** `the Boston Celtics'` **18 game home winning streak**"

In b6, "to triumph" is used as a prepositional verb taking two arguments: the NP "the Golden State Warriors" filling the agent role and the NP "the Boston Celtics" filling a prepositional affected role. In r6, "to triumph" is used as a transitive verb taking a single argument conflating both the agent and affected roles. An embedded adjunct clause realizes an additional streak proposition. The NP "the Boston Celtics" which filled the prepositional affected role of the base , has been *demoted* as the genitive determiner of this embedded clause affected role.

In the double argument demotion case, the additional constituent appears as an argument as shown in the schema of Fig. 39.

The surface increment (b7,r7) below provides an example of double argument demotion:

(b7) "helping the Houston Rockets to *a 108 100 victory over the Philadelphia 76ers*"

(r7) "helping the Houston Rockets **win for the first time in 24 games at the Spectrum with** `a 108 100 victory over the Philadelphia 76ers`"

In b7, the head verb "to help" is used as an composite material and locative process with three arguments: agent, affected/located and (metaphorical) location. The agent argument is controlled by the agent of an

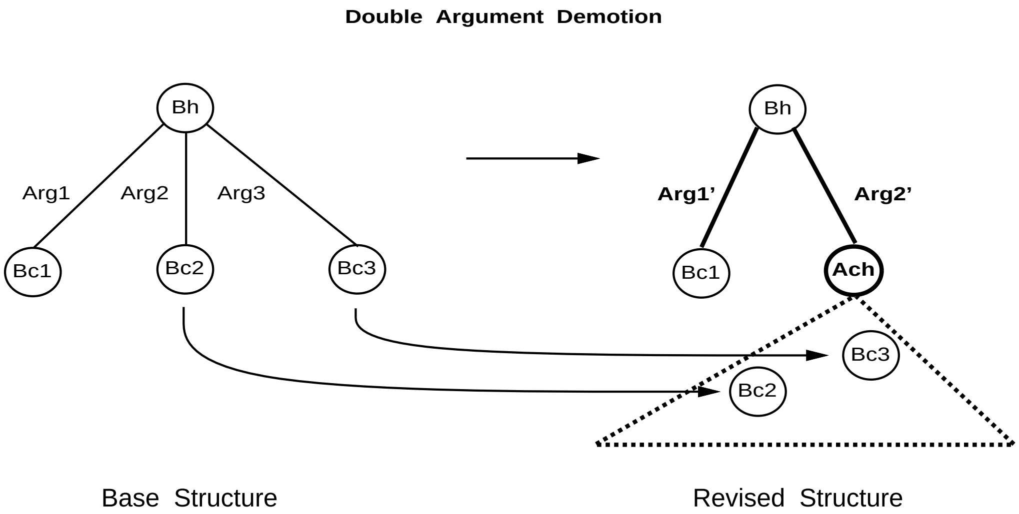

Figure 39: Double argument demotion schema

embedding prominent-statistic cluster[70], the affected/located argument is "the Houston Rockets" and the location argument "a 108 100 victory over the Philadelphia 76ers". In r7, the same head verb "to help" is this time used as a simple material process with only two roles: agent and affected[71]. The agent of this revised cluster remains the same controlled argument as in the base but the affected argument is now filled by an embedded clause realizing an additional streak proposition. The affected/located and location arguments of the base have been demoted, respectively as agent argument and instrument adjunct of this embedded clause.

Neither of the two forms of argument demotions I found in the corpus were used in a versatile fashion. They were both used only at the clause rank and only to add streak information.

B.2.3 Nominalization

Nominalization applies only to clausal bases. It consists of the replacement of a full head verb fV conveying a semantic element by a combination support verb sV and supported NP argument hN. Such a transformation expands the base argument structure and *nominalizes* the semantic element conveyed in the base by fV. This change in the realization pattern of the clause allows to realize an additional proposition by just adding premodifying constituents to the nominalized form. What makes such a nominalization convenient is that the type of additional information which can be conveyed by premodifiers at the NP rank cannot always be as concisely conveyed at the clause rank. The schema of a nominalization is shown in Fig. 40.

The surface increment pair (b8,r8) below provides an example of nominalization:

(b8) "the Seattle Supersonics <u>defeated</u> the Sacramento Kings 106 91"

(r8) "the Seattle Supersonics **handed** the Sacramento Kings **their 33rd straight road loss** 106 91"

In b8, the game-result is realized by the head verb "to defeat". In r8, it is realized by the head noun "loss" of the added NP. This NP is supported by the head verb of the revised cluster "to hand". The full verb pattern "X defeat Y" has thus been replaced by the synonymous support verb pattern "X hand Y DET loss". This nominalized form of the game-result is premodified by the complex determiner "their 33rd straight" and the classifier "road". These two premodifiers convey an additional streak proposition. The fact that no combination of clausal adjunct could realize this streak proposition in such a concise yet unambiguous way

[70] Recall from section 2.4.1 that many game-result clusters appear as adjuncts of a prominent-statistic cluster.
[71] Note these two usages of "to help" would correspond to two different entries in the lexicon of STREAK.

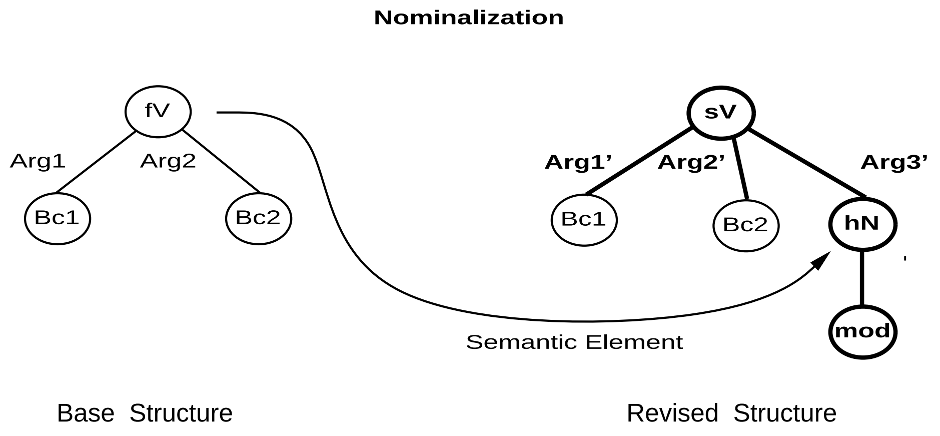

Figure 40: Nominalization schema

is the probable motivation underlying the use of nominalization in this case. In the corpus, nominalization was used only to add streak information.

B.2.4 Adjunctization

Adjunctization was already presented in section 2.5.2. It applies only to clausal bases headed by a support verb. In an adjunctization, the additional proposition is realized by a combination of two new constituents: a new head and a new argument. Deprived of the head verb supporting it, the supported argument of the base migrates to an adjunct position in the revised clause. The supported argument of base is thus *adjunctized*. The adjunctization schema is given in Fig. 41.

The surface increment pair (b9,r9) below provides an example of adjunctization:

(b9) "the Denver Nuggets <u>claimed</u> *a 124 110 victory over the Dallas Mavericks*"

(r9) "the Denver Nuggets **ended their three game losing streak with** `a 124 110 victory over the Dallas Mavericks`"

In b9, the italicized NP realizing the game-result is syntactically supported by the head verb "to claim". In r9 this support verb has been replaced by the full head verb "to end". This new head and the new boldfaced argument together realize an additional streak proposition. Deprived of its syntactic support, the

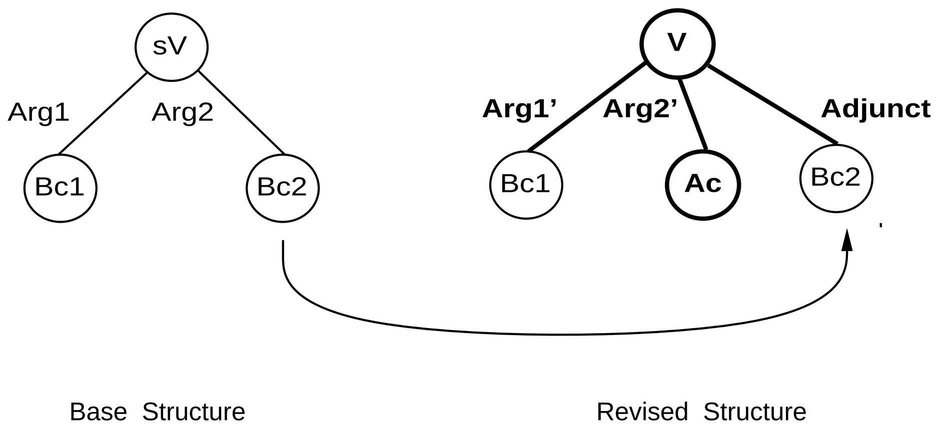

Figure 41: adjunctization schema

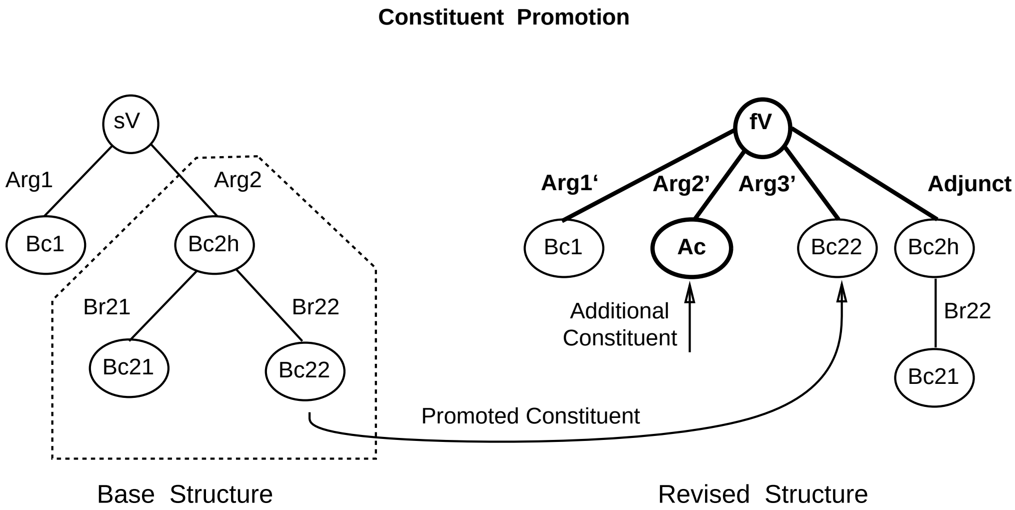

Figure 42: Constituent promotion schema

slanted NP realizing the game-result has migrated in adjunct position.

Adjunctization should not be confused with recast. Although both involve migration of a base cluster argument as a revised cluster adjunct, adjunctization is different because it changes the head and the argument structure of the base which recast preserves. Recast is also more versatile since it applies to any hypotactic base structure.

B.2.5 Constituent promotion

A constituent promotion is any transformation which involves a change of head and the promotion of an embedded base constituent at the top-level as an argument of the new head. The cases of argument promotions encountered in the corpus were special cases of adjunctization where a subconstituent of the adjunctized constituent was promoted at the top-level. The schema of such case of constituent promotion is shown in Fig. 42.

The surface increment pair (b10,r10) below provides an example of constituent promotion:

(b10) "the Boston Celtics <u>rolled to</u> *a 148 140 victory over the Denver Nuggets*"

(r10) "the Boston Celtics **kept** `the Denver Nuggets` **winless on the road with** `a 148 140 victory`"

In this example, the PP complement "the Denver Nuggets" embedded in the italicized range argument in the base cluster b10, is promoted as an affected/attribute argument in the top-level structure of the revised cluster r10.

B.2.6 Coordination promotion

A coordination promotion applies only to hypotactic base structures with a coordinated argument. It undoes the meaning factorization achieved by such structures and transforms it into a top-level coordinated structure of hypotactic elements. The embedded coordination in the base structure is thus *promoted* at the top-level in the revised structure. Accommodating the additional proposition requires to undo the meaning factorization achieved by the embedded coordination when one of its elements is part of the seed constituent[72] and the other is not. The general schema of a coordination promotion is shown in Fig. 43.

[72]Recall that by definition the seed is the base cluster constituent that is semantically related to the additional proposition.

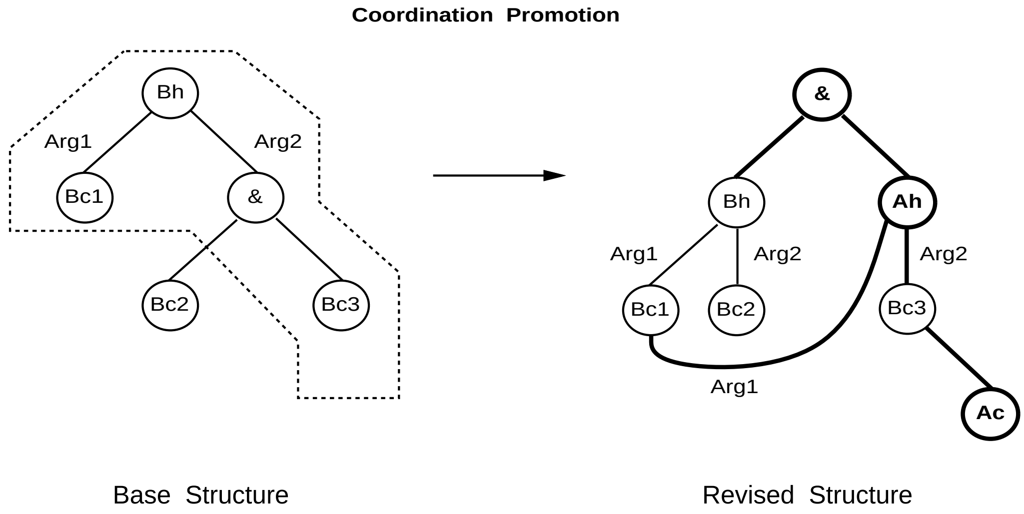

Figure 43: coordination promotion schema

The surface increment pair (b11,r11) below provides a coordination promotion example which illustrates the above schema:

(b11) "Otis Thorpe scored 27 points and grabbed 20 rebounds and *LARRY SMITH ADDED 12 POINTS AND 25 REBOUNDS*"

(r11) "Otis Thorpe scored 27 points and grabbed 20 rebounds and `LARRY SMITH ADDED 12 POINTS` AND **MATCHED HIS SEASON HIGH** `25 REBOUNDS`"

In b11, the uppercased seed constituent realizes two simple-statistics. Since the author of both is the same player this seed is structured a single clause with an NP coordination as its created argument. In r11, an additional extrema proposition is incorporated to that seed constituent. However, this extrema concerns only the rebounding element of the seed embedded coordination. The revision separates the elements of this embedded coordination into two different constituents and each of them fills the created argument of one element of a new top-level clause coordination in the revised seed.

In the corpus coordination promotion was used only for the record-equalling type of extrema. For the more common record-breaking type, the embedded coordination was preserved with the seed element simply premodified. On base cluster b11 this type of revision would yield the form:

(r11') "Otis Thorpe scored 27 points and grabbed 20 rebounds and `Larry Smith added 12 points and` **a season-high** `25 rebounds`"

Note that the added premodifier "**season-high**" does not explicitly precise whether the extrema is of type record-breaking or record-equalling. However, such form *implies* a record-breaking type because (1) this form is consistently used in the corpus for the record-breaking type and (2) this type is the most common and thereby default type. A priori, adjoining a premodifier in the seed element of the emdedded coordination could be used to add an extrema proposition of the record-equalling type. However, this modifier would have to explicitly precise that type, e.g., "a season high equalling 25 rebounds". Such a form is never used in the corpus, probably because it is a little too convoluted. Instead, the extrema type is conveyed by an additional head verb "to match". It is to accommodate for this additional head verb that coordination promotion is necessary.

B.3 Side transformations

The transformations the base pattern undergoes during the application of the revision tools presented in the two previous sections of this appendix follow one of two goals:

- To attach the additional proposition the base pattern (introductory transformation)
- To maintain grammaticality after the structural shake up brought about by the introductory transformation (restructuring transformation)

These transformations are the *core* transformations involved in information-adding revisions. However, they are sometimes accompanied by *side* transformations which satisfy other goals such as avoiding repetitions, avoiding ambiguities and enforcing collocation constraints. I identified six main types of side transformations: *reference abridging, ellipsis, argument control, scope marking, lexical adjustment* and *ordering adjustment*. The first four of these transformation are structural, *i.e.* they affect, like the core transformation, the structure of the base. The last two however, leaves that structured unchanged. They respectively act upon the lexical material and the constituent ordering of the base[73]. I present the six types of side transformations in turn in the following subsections.

B.3.1 Reference abridging

Reference abridging, the most widely used side transformation in the corpus, was already presented in section 2.5.2. Its goal is to suppress repetitions introduced by the attachment of the additional proposition. In the corpus, initial references use a set of default properties associated with the class of the referred entity in the domain ontology. For example initial references to players use their first and last name, initial references to teams use their name and location, etc. However, when a revision introduces a second reference, this set of property is *distributed* between these two references. The base reference is thus *abridged* by the revision[74].

The surface increment pair (b12,r12) below illustrate this phenomenon:

(b12) "to direct the Los Angeles Lakers past [*the* Washington *Bullets*] 87 72"

(r12) "to direct the Los Angeles Lakers past [Washington] 87 72 **handing** *the Bullets* **their ninth straight defeat**"

In the base pattern Rb3 both the name, "*the Bullets*" and the location, "Washington" of the losing team are used in the single reference to that team. When this cluster is revised the added constituent contains a second reference to this team. In this second reference the team name alone is used. As a side transformation, the first reference is abridged using the team location alone. Without such a side transformation, the team name would be repeated just three words after being introduced:

(r12') "to direct the Los Angeles Lakers past the Washington Bullets 87 72 **handing the Bullets their ninth straight defeat**"

B.3.2 Argument control

Argument control occurs in the corpus when a base constituent, during an absorb revision, becomes embedded under an added constituent that has one role in common with the base constituent. In such cases, a side transformation of the absorb revision, the common role in the base constituent is no longer realized, but instead becomes an omitted implicit argument, *controlled* by the corresponding argument is the added embedding constituent. This argument control side transformation avoids what would constitute a stylistically inappropriate repetition.

The surface increment pairs (b13,r13) below provides an example of absorb with argument control:

(b13) "<u>Minnesota</u> rolled to a 106 77 victory over the Charlotte Hornets"

(r13) "**Minnesota snapped a three game losing streak** `rolling to a 106 77 victory over the Charlotte Hornets`"

[73] Although the grammatical structure of the base largely constraints the ordering of its argument(s), the linear location of its adjunct(s) is highly underconstrained.

[74] Note that the use of this transformation is based on the assumption that the report targeted audience knows about every property in the default set, since otherwise it could not establish the co-reference link in the revised pattern

In r13, the top-level reference to Minnesota filling the agent argument of the added embedding constituent *controls* the omitted agent argument of the absorbed embedded clause (this absorbed clause is emphasized by a typewriter font).

B.3.3 Ellipsis

Ellipsis occurs as a side transformation in the corpus in conjunction with append and conjoin revisions. Stylistically, the appending of an additional element in a coordination sometimes allows for a more elliptical form for each element coordinated in the base.

The surface increment pair (b14,r14) below provides an example of append with ellipsis:

(b14) "Willie Anderson scored 25 points and Terry cummings <u>added</u> 24"

(r14) "Willie Anderson scored 25 points, Terry cummings 24 **and David Robinson 23**"

In b14, the second element of the conjunction has only the head of its created role (realizing the unit of the simple-statistic) ellided. In r14, the main verb of this second element is also ellided.

B.3.4 Scope marking

Consider the situation where a proposition P2 = `simple-statistic(author1,value1,unit1)` is to be added to a base cluster realizing a proposition P1 = `simple-statistic(author2,value1,unit1)`. The most concise way to add P2 is to conjoin a nominal realizing author2 with the nominal realizing author1 in the base. Using this tool achieves maximum meaning factorization. However, it also introduces an ambiguity: is the value cumulated for both authors *together* or for each author *separately*? Disambiguating such a coordination requires to add a *scope marker* in the emdedding clause.

The surface increment pair (b15,r15) below provides an example of conjoin with scope marking:

(b15) "[Magic Johnson scored 21 points]"

(r15) "Magic Johnson **and Byron Scott** scored 21 points **apiece**"

In this example, the segregatory coordination marker "apiece" is added at the top-level structure as a side of the coordination of the base agent with the additional boldfaced nominal. Embedded coordination disambiguation can also be achieved by adjusting the lexical head to a scope marking verb such as "to combine for". This lexical type of side transformation is presented in section B.3.5.

B.3.5 Lexical adjustment

The goal of lexical adjustment is to maintain collocation and selectional restriction constraints in the revised cluster after the introduction of a new constituent.

Consider the surface increment pair (b16,r16) below:

(b16) "Armon Gilliam <u>scored</u> 24 points"

(r16) "Armon Gilliam **had** 24 points **and 11 rebounds**"

In this example, a rebounding statistic is added to a clause realizing a scoring statistic of the same player. This addition is done by conjoining an NP "11 rebounds" to the NP "24 points". However, the collocation of the words "to score" and "rebound" in verb-object position is unfelicitous. As a side transformation, the embedding head verb is thus adjusted to the more versatile and collocationally neutral verb "to have".

B.3.6 Ordering adjustment

As noted in section B.3.4, the introductory transformation attaching the additional constituent can introduce ambiguities. These ambiguities can sometimes be circumvented by simply modifying the constituent ordering without altering the base structure.

The surface increment pair (b17,r17) below provides an example of adjoin with ordering adjustment:

(b17) "the Boston Celtics triumphed over the Miami Heat 119 109"

(r17) "the Boston Celtics triumphed `119 109` over the Miami Heat **who have lost 11 consecutive road games**"

In this example, a relative clause realizing an additional streak proposition is adjoined to the prepositional argument referring to the losing team. As a side transformation, the numerical manner adjunct "119 109" is moved in front of that prepositional argument. Without such a constituent re-ordering, the revised cluster would be:

(r17') ? "the Boston Celtics triumphed over the Miami Heat **who have lost 11 consecutive road games** 119 109

In r17', the attachment of the score constituent is ambiguous. It might be interpreted as meaning that all the games of the streak were lost on the very score of 119 109.

References

[Anderson 1985] D. Anderson. *Contemporary sports reporting.* Nelson-Hall, Chicago, IL, 1985.

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

[Bateman 1989] J.A. Bateman. Upper-modelling: current states of theory and practise - part1: foundations and related approaches -, 1989. Draft.

[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.

[Boyer and Lapalme 1985] M. Boyer and G. Lapalme. Generating paraphrases from meaning-text semantic networks. *Computational Intelligence*, (3&4):103–117, August-November 1985.

[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. Springler 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.

[Elhadad 1992] M. Elhadad. *Using argumentation to control lexical choice: a unification-based approach.* PhD thesis, Computer Science Department, Columbia University, 1992.

[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.

[Gross 1984] M. Gross. Lexicon-Grammar and the syntactic analysis of French. In *Proceedings of the 10th International Conference on Computational Linguistics*, pages 275–282. 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.

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

[Lenat and Guha 1989] D.B. Lenat and R.V Guha. *Building large knowledge base systems: representation and inference in the Cyc project*. Addison-Wesley, 1989.

[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 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.

[Roth *et al.* 1991] S. Roth, J. Mattis, and X. Mesnard. Graphics and natural language as component of automatic explanation. In J.W. Sullivan and S.W. Tyler, editors, *Intelligent user-interfaces*, ACM press frontier. Addison-Wesley, 1991.

[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.

[Smadja 1991] F. Smadja. *Retrieving Collocational Knowledge from Textual Corpora. An Application: Language Generation.* PhD thesis, Computer Science Department, Columbia University, 1991.

[Talmy 1975] L. Talmy. Figure and Ground in complex sentences. In C.E. Ferguson and E.A. Moravcsik, editors, *Syntax*, volume 4 of *Universals of human language*. Stanford University Press, 1975.

[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.