Corpus Analysis for Revision-Based Generation of Complex Sentences

Jacques Robin and Kathleen McKeown
Department of Computer Science
Columbia University
New York, NY 10027
{robin,kathy}@cs.columbia.edu

Abstract

The complex sentences of newswire reports contain *floating* content units that appear to be opportunistically placed where the form of the surrounding text allows. We present a corpus analysis that identified precise semantic and syntactic constraints on where and how such information is realized. The result is a set of revision tools that form the rule base for a report generation system, allowing incremental generation of complex sentences.

Introduction

Generating reports that summarize quantitative data raises several challenges for language generation systems. First, sentences in such reports are very complex (*e.g.*, in newswire basketball game summaries the lead sentence ranges from 21 to 46 words in length). Second, while some content units consistently appear in *fixed* locations across reports (*e.g.*, game results are always conveyed in the lead sentence), others *float*, appearing anywhere in a report and at different linguistic ranks within a given sentence. Floating content units appear to be opportunistically placed where the form of the surrounding text allows. For example, in Fig. 1, sentences 2 and 3 result from adding the same streak information (*i.e.*, data about a series of similar outcomes) to sentence 1 using different syntactic categories at distinct structural levels.

Although optional in any given sentence, floating content units cannot be ignored. In our domain, they account for over 40% of lead sentence content, with some content types *only* conveyed as floating structures. One such type is historical information (*e.g.*, maximums, minimums, or trends over periods of time). Its presence in all reports and a majority of lead sentences is not surprising, since the relevance of any game fact is often largely determined by its historical significance. However, report generators to date [Kukich, 1983], [Bourbeau *et al.*, 1990] are not capable of including this type of information due to its floating nature. The issue of optional, floating content is prevalent in many domains and is receiving growing attention (cf. [Rubinoff, 1990], [Elhadad and Robin, 1992], [Elhadad, 1993]).

1. *Draft sentence*:
   "San Antonio, TX – David Robinson scored 32 points Friday night lifting the San Antonio Spurs to a 127 111 victory over the Denver Nuggets."
2. *Clause coordination with reference adjustment*:
   "San Antonio, TX – 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**".
3. *Embedded nominal apposition*:
   "San Antonio, TX – 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 1: Attaching a **floating content unit** onto different draft sentence SUBCONSTITUENTS

These observations suggest a generator design where a draft incorporating fixed content units is produced first and then any floating content units that can be accommodated by the surrounding text are added. Experiments by [Pavard, 1985] provide evidence that only such a revision-based model of complex sentence generation can be cognitively plausible[1].

To determine how floating content units can be incorporated in a draft, we analyzed a corpus of basketball reports, pairing sentences that differ semantically by a single floating content unit and identifying the minimal syntactic transformation between them. The result is a set of *revision tools*, specifying precise semantic and syntactic constraints on (1) where a particular type of floating content can be added in a draft and (2) what linguistic constructs can be used for the addition.

The corpus analysis presented here serves as the basis for the development of the report generation sys-

[1] cf. [Robin, 1992] for discussion.

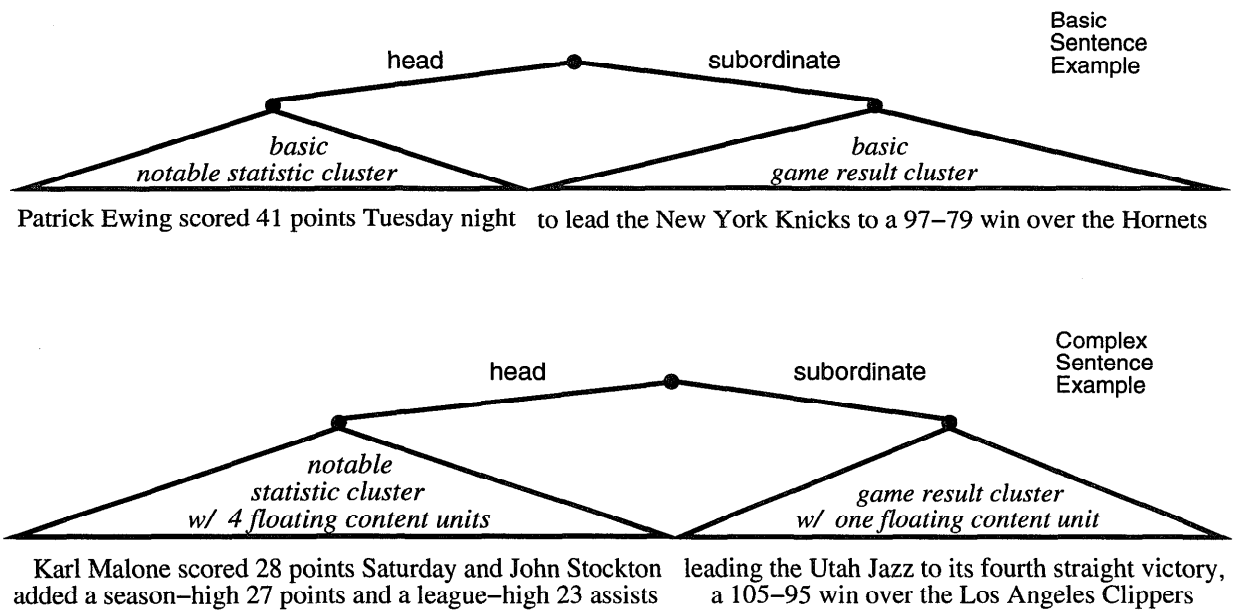

Figure 2: Similarity of basic and complex sentence structures

tem STREAK (Surface Text Revision Expressing Additional Knowledge). The analysis provides not only the knowledge sources for the system and motivations for its novel architecture (discussed in [Robin, 1993]), but also with means for ultimately evaluating its output. While this paper focuses on the analysis, the on-going system implementation based on functional unification is discussed in [Robin, 1992].

After describing our corpus analysis methodology, we present the resulting revision tools and how they can be used to incrementally generate complex sentences. We conclude by previewing our planned use of the corpus for evaluation and testing.

Corpus analysis methodology

We analyzed the lead sentences of over 800 basketball games summaries from the UPI newswire. We focused on the first sentence after observing that all reports followed the *inverted pyramid structure with summary lead* [Fensch, 1988] where the most crucial facts are packed in the lead sentence. The lead sentence is thus a self-contained mini-report. We first noted that all 800 lead sentences contained the game result (*e.g.*, *"Utah beat Miami 105-95"*), its location, date and at least one final game statistic: the most notable statistic of a winning team player. We then semantically restricted our corpus to about 300 lead sentences which contained only these four fixed content units and zero or more floating content units of the most common types, namely:

- Other final game statistics (*e.g.*, *"Stockton finished with 27 points"*).
- Streaks of similar results (*e.g.*, *"Utah recorded its fourth straight win"*).
- Record performances (*e.g.*, *"Stockton scored* **a season-high** *27 points"*).

Complex Sentence Structure We noted that *basic* corpus sentences, containing only the four fixed content units, and *complex* corpus sentences, which in addition contain up to five floating content units, share a common top-level structure. This structure consists of two main constituents, one containing the notable statistic (the *notable statistic cluster*) and the other containing the game result (the *game result cluster*), which are related either paratactically or hypotactically with the notable statistic cluster as head. Hence, the only structural difference is that in the complex sentences additional floating content units are clustered around the notable statistic and/or the game result. For example, the complex sentence at the bottom of Fig. 2 has the same top-level structure as the basic sentence at the top, but four floating content units are clustered around its notable statistic and a fifth one with its game result. Furthermore, we found that when floating elements appear in the lead sentence, their semantics almost always determines in which of the two clusters they appear (*e.g.*, streaks are always in the game result cluster).

These corpus observations show that any complex sentence can indeed be generated in two steps: (1) pro-

duce a basic sentence realizing the fixed content units, (2) incrementally revise it to incorporate floating content units. Furthermore, they indicate that floating content units can be attached within a cluster, based on *local* constraints, thus simplifying both generation and our corpus analysis. When we shifted our attention from whole sentence structure to internal cluster structures, we split the whole sentence corpus into two subsets: one containing notable statistic clusters and the other, game result clusters.

Cluster structure To identify syntactic and lexical constraints on the attachment of floating content units within each cluster, we analyzed the syntactic form of each cluster in each corpus lead sentence to derive *realization patterns*. Realization patterns abstract from lexical and syntactic features (e.g., connectives, mood) to represent the different mappings from semantic structure to syntactic structure. Examples of realization patterns are given in Fig. 3. Each column corresponds to a syntactic constituent and each entry provides information about this constituent: (1) semantic content[2], (2) grammatical function, (3) structural status (*i.e.* head, argument, adjunct etc) and (4-5) syntactic category[3]. Below each pattern a corpus example is given.

Realization patterns represent the structure of entire clusters, whether basic or complex. To discover how complex clusters can be derived from basic ones through incremental revision, we carried out a differential analysis of the realization patterns based on the notions of *semantic decrement* and *surface decrement* illustrated in Fig. 4.

A cluster C_d is a semantic decrement of cluster C_i if C_d contains all but one of C_i's content unit types. Each cluster has a set of realization patterns. The surface decrement of a realization pattern of C_i is the realization pattern of C_d that is structurally closest. Figure 3 shows a semantic decrement pairing C_d, a single content unit, with C_i, which contains two content units. Both clusters have two realization patterns associated with them as they each can be realized by two different syntactic structures. These four syntactic structure patterns must be compared to find the surface decrements. Since R_{d1} is entirely included in R_{i1}[4] it is the surface decrement of R_{i1}. To identify the surface decrement of R_{i2}, we need to compare it to R_{d2} and R_{d1} in turn. All the content units common to R_{i2} and R_{d2} are realized by identical syntactic categories in both patterns. In particular, the semantic head (game-result) is mapped onto a noun (*"victory"* in R_{d2}, *"triumph"* in R_{i2}). In contrast, this same semantic head is mapped onto a verb (*"to defeat"*) in R_{d1}. R_{d2}, rather than R_{d1} is thus the surface decrement of R_{i2}.

We identified 270 surface decrement pairs in the corpus. For each such pair, we then determined the structural transformations necessary to produce the more complex pattern from the simpler *base* pattern. We grouped these transformations into classes that we call *revision tools*.

Revisions for incremental generation

We distinguished two kinds of revision tools. *Simple* revisions consist of a single transformation which preserves the base pattern and adds in a new constituent. *Complex* revisions are in contrast non-monotonic; an *introductory* transformation breaks up the integrity of the base pattern in adding in new content. Subsequent *restructuring* transformations are then necessary to restore grammaticality. Simple revisions can be viewed as elaborations while complex revisions require true revision.

Simple revisions We identified four main types of simple revisions: `Adjoin`[5], `Append`, `Conjoin` and `Absorb`. Each is characterized by the type of base structure to which it applies and the type of revised structure it produces. For example, `Adjoin` applies only to hypotactic base patterns. It adds an adjunct A_c under the base pattern head B_h as shown in Fig. 5.

`Adjoin` is a versatile tool that can be used to insert additional constituents of various syntactic categories at various syntactic ranks. The surface decrement pair $< R_{d1}, R_{i1} >$ in Fig. 3 is an example of `clause rank PP adjoin`. In Fig. 6, the revision of sentence 5 into sentence 6 is an example of `nominal rank pre-modifier adjoin`: *"franchise record"* is adjoined to the nominal *"sixth straight home defeat"*.

In the same figure, the revision of sentence 2 into sentence 3 is an example of another versatile tool, `Conjoin`: an additional clause, *"Jay Humphries added 24"*, is coordinated with the draft clause *"Karl Malone tied a season high with 39 points"*. In general, `Conjoin` groups a new constituent A_c with a base constituent B_{c1} in a new paratactic[6] complex. The new complex is then inserted where B_{c1} alone was previously located (cf. Fig. 5). Note how in Fig. 1 paraphrases are obtained by applying `Conjoin` at different levels in the base sentence structure.

Instead of creating a new complex, `Absorb` relates the new constituent to the base constituent B_{c1} by demoting B_{c1} under the new constituent's head A_h which is inserted in the sentence structure in place of B_{c1} as

[2] An empty box corresponds to a syntactic constituent required by English grammar but not in itself conveying any semantic element of the domain representation.

[3] The particular functions and categories are based on: [Quirk *et al.*, 1985], [Halliday, 1985] and [Fawcett, 1987].

[4] Remember that we compare *patterns*, not sentences.

[5] Our `Adjoin` differs from the adjoin of Tree-Adjoining Grammars (TAGs). Although, TAGs could implement three of our revision tools, `Adjoin`, `Conjoin` and `Append`, it could *not* directly implement non-monotonic revisions.

[6] Either coordinated or appositive.

C_i: <game-result(winner,loser,score),**streak(winner,aspect,type,length)**>.
C_d: <game-result(winner,loser,score)>.

$R_{i1}(C_i)$:

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

$R_{d1}(C_d)$ **surface decrement of** $R_{i1}(C_i)$:

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

$R_{i2}(C_i)$:

winner	**aspect**		**type**	**streak**	**length**		score	game-result	loser
agent	process	affected/located			location	means			
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	triumph	over Denver

$R_{d2}(C_d)$ **surface decrement of** $R_{i2}(C_i)$:

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 3: Realization pattern examples

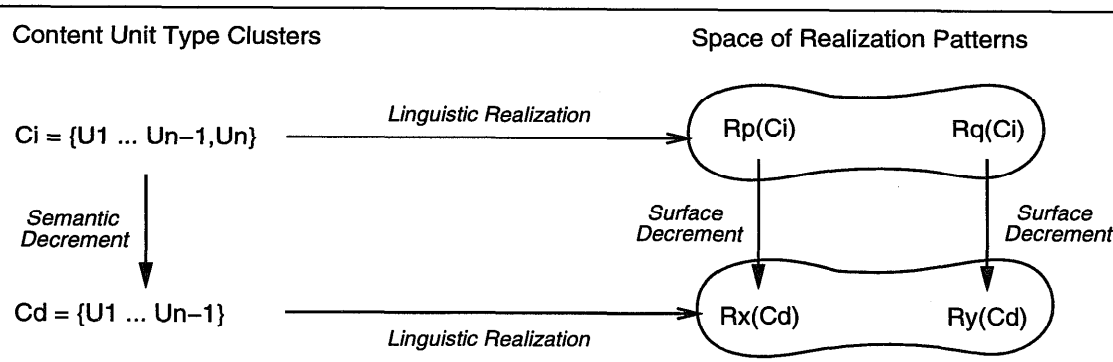

Figure 4: Differential analysis of realization patterns

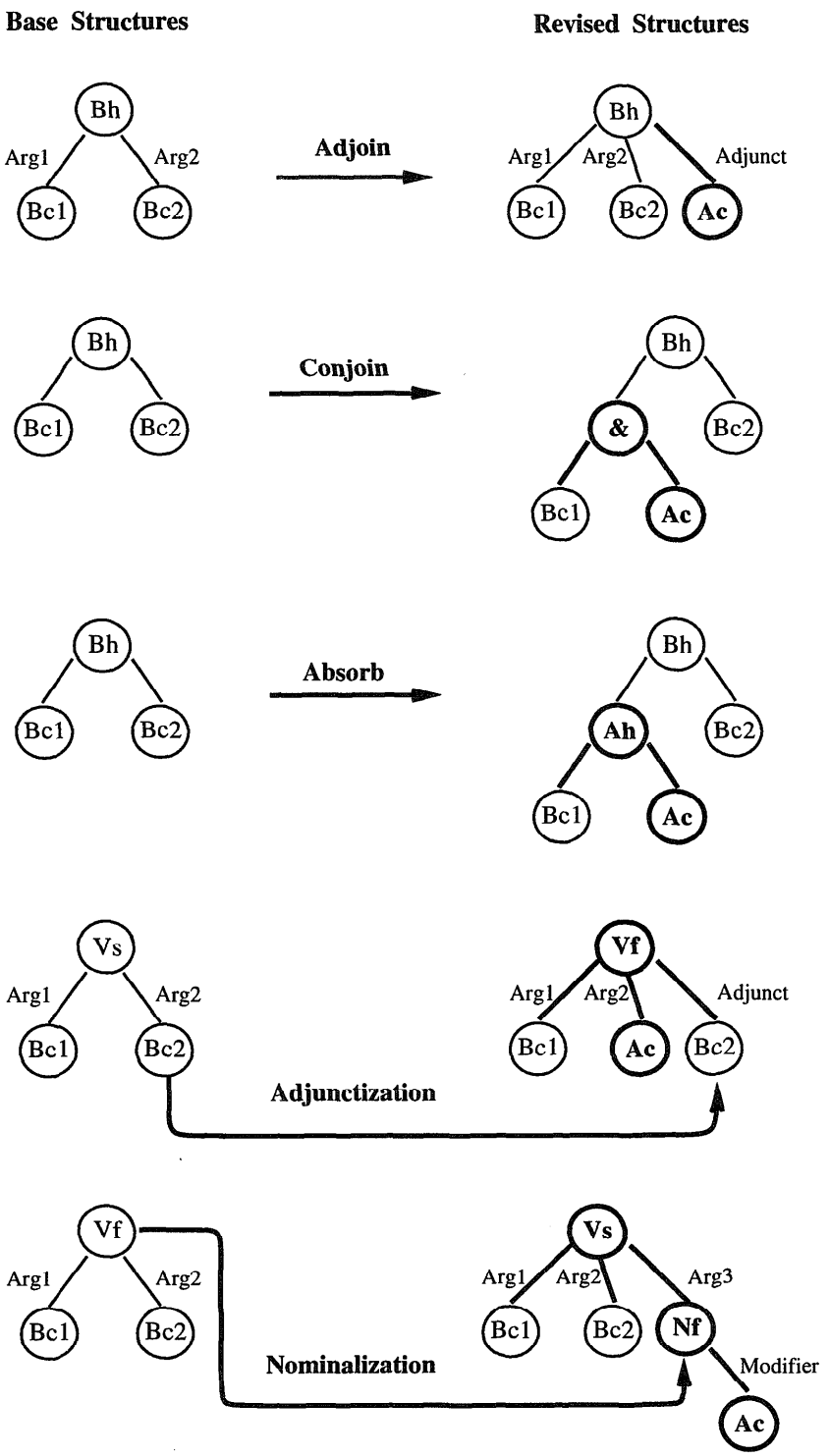

Figure 5: Structural schemas of five revision tools

1. **Initial draft (basic sentence pattern):**
   "Hartford, CT – Karl Malone scored 39 points Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
2. **adjunctization:**
   "Hartford, CT – Karl Malone **tied a season high with** *39 points* Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
3. **conjoin:**
   "Hartford, CT – Karl Malone tied a season high with 39 points **and Jay Humphries added 24** Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
4. **absorb:**
   "Hartford, CT – Karl Malone tied a season high with 39 points and Jay Humphries **came off the bench** *to add 24* Friday night as the Utah Jazz defeated the Boston Celtics 118 94."
5. **nominalization:**
   "Hartford, CT – Karl Malone tied a season high with 39 points and Jay Humphries came off the bench to add 24 Friday night as the Utah Jazz **handed** the Boston Celtics **their sixth straight home** *defeat* 118 94."
6. **adjoin:**
   "Hartford, CT – Karl Malone tied a season high with 39 points and Jay Humphries came off the bench to add 24 Friday night as the Utah Jazz handed the Boston Celtics their **franchise record** sixth straight home defeat 118 94."

Figure 6: Incremental generation of a complex sentence using various revision tools

shown in Fig. 5. For example, in the revision of sentence 3 into sentence 4 Fig. 6, the base VP *"added 24"* gets subordinated under the new VP *"came off the bench"* taking its place in the sentence structure. See [Robin, 1992] for a presentation of **Append**.

Complex revisions We identified six main types of complex revisions: **Recast, Argument Demotion, Nominalization, Adjunctization, Constituent promotion** and **Coordination Promotion**. Each is characterized by different changes to the base which displace constituents, alter the argument structure or change the lexical head. Complex revisions tend to be more specialized tools than simple revisions. For example, **Adjunctization** applies only to clausal base patterns headed by a support verb V_s. A support verb [Gross, 1984] does not carry meaning by itself, but primarily serves to support one of its meaning bearing arguments. **Adjunctization** introduces new content by replacing the support verb by a full verb V_f with a new argument A_c. Deprived of its verbal support, the original support verb argument B_{c2} migrates into adjunct position, as shown in Fig. 5.

The surface decrement pair $< R_{d2}, R_{i2} >$ of Fig. 3 is an example of **Adjunctization**: the RANGE argument of R_{d2} migrates to become a MEANS adjunct in R_{i2}, when the head verb is replaced. The revision of sentence 1 into sentence 2 in Fig. 6 is a specific **Adjunctization** example: *"to score"* is replaced by *"to tie"*, forcing the NP *"39 points"* (initially argument of *"to score"*) to migrate as a PP adjunct *"with 39 points"*.

In the same figure, the revision of sentence 4 into sentence 5 is an example of another complex revision tool, **Nominalization**. As opposed to **Adjunctization**, **Nominalization** replaces a full verb V_f by a synonymous <support-verb,noun> collocation $< V_s, N_f >$ where N_f is a nominalization of V_f. A new constituent A_c can then be attached to N_f as a pre-modifier as shown in Fig. 5. For example, in revising sentence 4 into sentence 5 (Fig. 6), the full-verb pattern *"X defeated Y"* is first replaced by the collocation pattern *"X handed Y a defeat"*. Once nominalized, *defeat* can then be pre-modified by the constituents *"their"*, *"sixth straight"* and *"home"* providing historical background. See [Robin, 1992] for presentation of the four remaining types of complex revisions.

Side transformations Restructuring transformations are not the only type of transformations following the introduction of new content in a revision. Both simple and complex revisions are also sometimes accompanied by *side* transformations. Orthogonal to restructuring transformations which affect grammaticality, side transformations make the revised pattern more concise, less ambiguous, or better in use of collocations.

We identified six types of side transformations in the corpus: **Reference Adjustment**, **Ellipsis**, **Argument Control**, **Ordering Adjustment**, **Scope Marking** and **Lexical Adjustment**. The revision of sentence 1 into sentence 2 in Fig. 1 is an example of simple revision with **Reference Adjustment**. Following the introduction of a second reference to the losing team *"the Nuggets ..."*, the initial reference is abridged to simply *"Denver"* to avoid the repetitive form *"a 127 111 victory over the Denver Nuggets, handing the Denver Nuggets their seventh straight loss"*. See [Robin, 1992] for a presentation of the other types of side transformations.

Revision tool usage Table 1 quantifies the usage of each tool in the corpus. The total usage is broken

		side transformations		scope ranks		streak	extrema	game stat
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 1: Revision tool usage in the corpus

down by linguistic rank and by class of floating content units (*e.g.*, **Adjoin** was used 88 times in the corpus, 23 times to attach a streak at the nominal rank in the base sentence). Occurrences of side transformations are also given.

Figure 6 illustrates how revision tools can be used to incrementally generate very complex sentences. Starting from the draft sentence 1 which realizes only four fixed content units, five revision tools are applied in sequence, each one adding a new floating content unit. Structural transformations undergone by the draft at each increment are highlighted: deleted constituents are underlined, added constituents boldfaced and displaced constituents italicized. Note how displaced constituents sometimes need to change grammatical form (*e.g.*, the finite VP "*added 24*" of (3) becomes infinitive "*to add 24*" in (4) after being demoted).

Conclusion and future work

The detailed corpus analysis reported here resulted in a list of revision tools to incrementally incorporate additional content into draft sentences. These tools constitute a new type of linguistic resource which improves on the realization patterns traditionally used in generation systems (*e.g.*, [Kukich, 1983], [Jacobs, 1985], [Hovy, 1988]) due to three distinctive properties:

- They are compositional (concerned with atomic content additions local to sentence subconstituents).
- They incorporate a wide range of *contextual* constraints (semantic, lexical, syntactic, stylistic).
- They are abstract (capturing common structural relations over sets of sentence pairs).

These properties allow revision tools to opportunistically express floating content units under surface form constraints and to model a sublanguage's structural complexity *and* diversity with maximal economy and flexibility. Our analysis methodology based on surface decrement pairs can be used with any textual corpus.

Revision tools also bring together incremental generation and revision in a novel way, extending both lines of research. The complex revisions and side transformations we identified show that accomodating new content cannot always be done without modifying the draft content realization. They therefore extend previous work on incremental generation [Joshi, 1987] [De Smedt, 1990] that was restricted to elaborations preserving the linguistic form of the draft content. As content-*adding* revisions, the tools we identify also extend previous work on revision [Meteer, 1991] [Inui *et al.*, 1992] that was restricted to content-*preserving* revisions for text editing.

In addition to completing the implementation of the tools we identified as revision rules for the STREAK generator, our plans for future work includes the evaluation of these tools. The corpus described in this paper was used for acquisition. For testing, we will use two other corpora. To evaluate completeness, we will look at another season of basketball reports and compute the proportion of sentences in this test corpus whose realization pattern can be produced by applying the tools acquired in the initial corpus. Conversely, to evaluate domain-independence, we will compute, among the tools acquired in the initial corpus, the proportion of those resulting in realization patterns also used in a

test corpus of stock market reports. The example below suggests that the same floating constructions are used across different quantitative domains:

- " Los Angeles – John Paxson hit 12 of 16 shots Friday night to score a season high 26 points **helping the Chicago Bulls snap a two game losing streak** with a 105 97 victory over the Los Angeles Clippers."
- "New York – Stocks closed higher in heavy trading Thursday, as a late round of computer-guided buy programs tied to triple-witching hour **helped the market snap a five session losing streak.**"

Although the analysis reported here was carried out manually for the most part, we hope to automate most of the evaluation phase using the software tool CREP [Duford, 1993]. CREP retrieves corpus sentences matching an input realization pattern encoded as a regular expression of words and part-of-speech tags.

Acknowledgments

Many thanks to Tony Weida and Judith Klavans for their comments on an early draft of this paper. This research was partially supported by a joint grant from the Office of Naval Research and the Defense Advanced Research Projects Agency under contract N00014-89-J-1782, by National Science Foundation Grants IRT-84-51438 and GER-90-2406, and by New York State Center for Advanced Technology Contract NYSSTF-CAT(92)-053 and NYSSTF-CAT(91)-053.

References

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

De Smedt, K.J.M.J. 1990. Ipf: an incremental parallel formulator. In Dale, R.; Mellish, C.S.; and Zock, M., editors 1990, *Current Research in Natural Language Generation.* Academic Press.

Duford, D. 1993. Crep: a regular expression-matching textual corpus tool. Technical Report CUCS-005-93, Columbia University.

Elhadad, M. and Robin, J. 1992. Controlling content realization with functional unification grammars. In Dale, R.; Hovy, H.; Roesner, D.; and Stock, O., editors 1992, *Aspects of Automated Natural Language Generation.* Springler Verlag. 89–104.

Elhadad, M. 1993. *Using argumentation to control lexical choice: a unification-based implementation.* Ph.D. Dissertation, Computer Science Department, Columbia University.

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

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

Gross, M. 1984. Lexicon-grammar and the syntactic analysis of french. In *Proceedings of the 10th International Conference on Computational Linguistics.* COLING. 275–282.

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

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

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

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

Joshi, A.K. 1987. The relevance of tree-adjoining grammar to generation. In Kempen, Gerard, editor 1987, *Natural Language Generation: New Results in Artificial Intellligence, Psychology and Linguistics.* Martinus Ninjhoff Publishers.

Kukich, K. 1983. The design of a knowledge-based report generation. In *Proceedings of the 21st Conference of the ACL.* ACL.

Meteer, M. 1991. The implications of revisions for natural language generation. In Paris, C.; Swartout, W.; and Mann, W.C., editors 1991, *Natutal Language Generation in Artificial Intelligence and Computational Linguistics.* Kluwer Academic Publishers.

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

Quirk, R.; Greenbaum, S.; Leech, G.; and Svartvik, J. 1985. *A comprehensive grammar of the English language.* Longman.

Robin, J. 1992. Generating newswire report leads with historical information: a draft and revision approach. Technical Report CUCS-042-92, Computer Science Department, Columbia Universtity, New York, NY. PhD. Thesis Proposal.

Robin, J. 1993. A revision-based generation architecture for reporting facts in their historical context. In Horacek, H. and Zock, M., editors 1993, *New Concepts in Natural Language Generation: Planning, Realization and Systems.* Frances Pinter, London and New York.

Rubinoff, R. 1990. Natural language generation as an intelligent activity. PhD. Thesis Proposal, Computer Science Department, University of Pennsylvania.