

Conceptual Revision for Natural Language Generation

Ben E. Cline

Department of Computer Science

Blacksburg, VA 24061

benjy@vtvml.cc.vt.edu

Traditional natural language generation systems are based on a pipelined architecture. A conceptual component selects items from a knowledge base and orders them into a message to address some discourse goal. This message is passed to the stylistic component that makes lexical and syntactic choices to produce a natural language surface text. By contrast, humans producing formal text typically create drafts which they polish through revision [Hayes and Flower 1980]. One proposal for improving the quality of computer-generated multisentential text is to incorporate a draft-and-revision paradigm.

Some researchers have suggested that revision in generation systems should only affect stylistic elements of the text [Vaughan and McDonald 1986]. But human writers also engage in conceptual revision, and there is reason to believe that techniques for conceptual revision should also be useful for a generation system producing formal text. Yazdani [1987] argues that both stylistic and conceptual revisions are useful. This paper extends those arguments and provides further evidence for the usefulness of conceptual as well as stylistic revision. We present strategies for identifying situations applicable to conceptual revision and techniques for effecting the revision.

Why is revision important for a natural language generation system? First, Hayes and Flower suggest that revision reduces the cognitive strain of an author by postponing the need to make some decisions while concentrating on others. A generation system can reduce complexity in the same way. By using revision, generation modules can be simpler. Second, inspection of surface text is necessary to determine whether the generated text is ambiguous. Ambiguities result not only from the words used at the surface level but from their relationships to other words in the text. To detect ambiguities, the surface text must be read. If that process reveals ambiguity, the text can be regenerated using different words or syntax. A revision component is the ideal location for reading generated text and identifying ambiguities.

The Kalos system being developed by the author is designed to perform both stylistic and conceptual revision. Kalos will generate portions of a draft user's guide for a microprocessor from an abstract architectural descrip-

tion of the microprocessor. The system achieves conceptual generation using a discourse schema system [McKewon 1985, Paris 1985]; stylistic generation will be rule-based. The revision component will review the generated text and produce recommendations to the conceptual and stylistic components as to how to improve the text.

Kalos takes a knowledge intensive approach. Each component of the system, including conceptual generation, stylistic generation, and revision, has access to the full knowledge of the system, and they use the same inference system. This use of a unified knowledge base lets the revision component identify easily both the concepts and schema slots from which the surface string was generated. This type of association is crucial for a revision system. In systems where knowledge is localized, it is difficult or impossible to determine the deep level knowledge responsible for a particular subtext.

In Kalos, conceptual revision will be applied to at least three situations. First, the Kalos revision module will detect situations where a preferred word or phrase will improve the text. Second, it will detect the need for an example to produce clearer text. Third, it will attempt to identify paragraphs that are too short or too long.

Kalos generates text aimed at engineers and others experienced with microprocessors, using preferred words and phrases common to user's guides covering various microprocessors. The revision module will manage use of preferred words for two reasons. First, performing preferred word processing in the revision component reduces the complexity of the generation components. Second, using preferred phraseology can affect both the conceptual and the stylistic components, so placing the logic for handling preferred words and phrases in the revision component localizes the necessary knowledge structures for easier maintenance and expansion.

For example, consider a description of the address bus of the Zilog Z-80 microprocessor: "The address bus of the Zilog Z-80 microprocessor is sixteen bits wide." Using the preferred phrase "address space", the same fact can be restated as follows: "The Zilog Z-80 has a sixty-four kilobyte address space."

The first sentence relates an attribute of the address bus, while the second sentence makes a statement directly about the processor. The second sentence both uses a preferred way of describing the processor's maximum memory size and gives an important feature of the microprocessor. It is thus desirable to include it in an overview paragraph of the microprocessor rather than in a following paragraph describing its buses.

Kalos will contain rules indicating preferred phrases for the discourse goal of describing a microprocessor. In this example, the relevant rule states that if the size of the address bus is described, replace the sentence with a description of the address space of the microprocessor. As noted above, Kalos will have a representation of the surface sentence which includes the surface representation and associations to the concepts and schemata from which the sentence was generated. By inspecting the underlying concept, Kalos can determine that the rule should be applied. It can then locate the schemata responsible for the text and make the revision.

The revision component of Kalos will be used to suggest at which points in the text an example is appropriate. This processing is placed in the revision module to reduce the complexity of the conceptual generation module. Examples will sometimes be included in the text in the description of individual instructions. Instructions that are straightforward do not require an example. Consider the add instruction of typical microprocessor. A typical reader of a microprocessor user's guide will gain little or no information from an example of the add instruction after reading the description of the register transfers involved. This is not the case, however, with more complicated instructions involving several registers and register transfers.

In Kalos, the process schema selects the knowledge structures needed to describe the actions of an instruction. This schema has an optional example slot which will initially be left empty by the conceptual generation module. The Kalos revision module inspects the underlying conceptual structures of instruction descriptions to determine if an instruction is complicated, based on the number of register transfers and the number of registers involved. When a complicated instruction is identified, the revision module will suggest that the generation module expand the text by filling the example slot of the process schema. It is then the task of the conceptual generation component to construct an example.

Kalos's third type of conceptual revision relates to the size of generated paragraphs. Extremely short or long paragraphs are sometimes appropriate, but they are suspect and will be examined by the revision component for possible restructuring.

Kalos will attempt to expand small paragraphs by suggesting revisions that fill optional schema slots when the

text is regenerated. In Kalos, text can be expanded by adding an example or comparing and contrasting the object being described to another object. The suggestions to add text will be inspected by the generation module and implemented if they meet two criteria. First, the knowledge base must contain the information necessary to fill the optional schema slot. Second, the inclusion of the additional knowledge must pass a test for salience. Salience will be based in part on deviation from typicality [Cline and Nutter, 1989].

The revision module will also try to restructure long paragraphs. It will look at both the surface text and the underlying concepts from which the text was generated in order to produce suggestions for the revision. To reduce the amount of text, the revision component will suggest that the generation component either remove an optional schema slot or take a different choice point in a schema. Targets for removal include embedded compare and contrast schemata and example slots in process schemata. The revision module may also select a different choice point in the constituency schema to list part categories rather than parts. For example, an overview of a typical microprocessor would do better to list instruction categories than to list over a hundred instructions. In reducing long paragraphs, the revision module will have some simple characterizations as to how important the removed information is. Based on these measures, the revision component may decide to retain the lengthy paragraph.

Cline, B. E. & Nutter, J. T. (1989) Implications of natural categories for natural language generation. In: *Proceedings of the First Annual SNePS Workshop*.

Gregg, L. W. & Steinberg, E. R. (Eds.) (1980) *Cognitive Processes in Writing*. Hillsdale, NJ: Erlbaum.

Hayes, J. R. & Flower, L. S. (1980) Identifying the Organization of Writing Processes. In: Gregg & Steinberg.

Kempen, G. (Ed.) (1987) *Natural Language Generation*. Dordrecht: Martinus Nijhoff Publishers.

McKeown, K. R. (1985) *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text*. Cambridge: Cambridge University Press.

Paris, C. L. (1985) Description Strategies for Naive and Expert Users. In: *Proceedings of the 23rd Annual Meeting of the Association of Computational Linguistics*. Chicago, Ill.

Vaughan, M. M. & McDonald, D. D. (1986) A Model of Revision in Natural Language Generation. In: *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics*. New York.

Yazdani, M. (1987) Reviewing as a Component of the Text Generation Process. In: Kempen