Dynamically Improving Explanations: A Revision-Based Approach to Explanation Generation*

Charles B. Callaway
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
USA
cbcallaw@eos.ncsu.edu

James C. Lester
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206
USA
lester@eos.ncsu.edu

Abstract

Recent years have witnessed rapid progress in explanation generation. Despite these advances, the quality of prose produced by explanation generators warrants significant improvement. *Revision-based* explanation generation offers a promising means for improving explanations at runtime. In contrast to single-draft explanation generation architectures, a revision-based generator could dynamically create, evaluate, and refine multiple drafts of explanations. However, because of the inherent complexity of revision, previous multi-sentential revision-based approaches have not scaled up. We have developed a scalable revision-based model of explanation generation that dynamically improves multi-sentential explanations. By operating on *abstract discourse plans* encoded in a minimalist representation, it combats both the conceptual complexities and the efficiency problems posed by revision. This approach has been implemented in REVISOR, a unification-based revision system. Evaluations of REVISOR's performance in generating a corpus of extended multi-sentential scientific explanations yielded encouraging results.

1 Introduction

Automatically generating natural language explanations of complex phenomena is a critical functionality for advisory systems and knowledge-based learning environments. In recent years, explanation generation capabilities have advanced steadily [Cawsey, 1992; Mittal, 1993; Moore, 1995; Lester and Porter, 1997; Suthers, 1993]. Despite these advances, the quality of prose produced by current explanation systems calls for significant improvement. One approach to improving explanations is to expand the capabilities of discourse planners and realization systems. However, because discourse planners and realization systems already consider a complex matrix of syntactic, semantic, and pragmatic constraints to make appropriate decisions about content selection, clause structure, phrase structure, and lexical choice, adding to their complexity is problematic.

Revision offers a promising alternative for significantly increasing the quality of explanations without increasing the complexity of either the discourse planning or realization components. Just as human writers employ revision to cope with the complexities of composition [Hayes and Flower, 1986], an explanation system can employ an iterative process of composing drafts, evaluating them, and revising them. In contrast to classical explanation systems that operate on a single draft, a revision-based system can operate on multiple drafts over time. Because revision is a generate-and-test method, a system that dynamically edits and revises explanations presents itself with a series of opportunities for considering the myriad constraints that bear on generation decisions. However, previous efforts to develop computational models of revision have not scaled up. Many were never implemented [Vaughan and McDonald, 1986; Yazdani, 1987], and of those that were, most were highly restricted and operated on only a few examples [Gabriel, 1988; Inui *et al.*, 1992; Mann and Moore, 1981]. By far the most successful revision-based approach to date [Robin, 1994] suffers from serious efficiency problems and operates only on single sentences.

To dynamically improve explanations while at the same time combatting the problems of complexity and efficiency, we have developed a revision-based model of explanation generation that operates on *abstract discourse plans*. Encoded in a minimalist representation consisting of only those features that are most critical for making revision decisions, abstract discourse plans promote efficiency by reducing the complexity of drafts. This model of revision-based generation has been implemented in REVISOR, a revision system that employs a non-monotonic unification framework to compose and edit multiple drafts of explanations. REVISOR focuses on clause-aggregation revisions, a problem that has been the subject of increasing attention from a variety of perspectives in the natural language generation community [Dalianis and Hovy, 1993;

*Support for this work was provided by the IntelliMedia Initiative of North Carolina State University.

Robin, 1994].

Given an initial *ground level* discourse plan produced by a discourse planner, REVISOR transforms the ground level plan to an abstract discourse plan, which contains only the most critical lexical, syntactic, and semantic features needed for revision. This abstract discourse plan represents the first draft. REVISOR then iteratively refines draft after draft by applying and revision operators and evaluating the resulting discourse plans with respect to discourse and style constraints. When a final draft has been constructed, it is transformed to a ground level discourse plan which is then realized in natural language.

To evaluate the quality of explanations produced by the revision-based model, REVISOR was introduced into a full-scale explanation system that includes a discourse planner, a functional sentence planner, and a systemic/functional surface generator. In a blind comparative study, a corpus of scientific explanations produced by two versions of the explanation system, one without REVISOR and one with REVISOR, was evaluated by a panel of judges. The results of the study are encouraging and suggest that revision-based generation is a promising paradigm for explanation systems.

2 Revision-Based Generation

Classically, explanation generation is decomposed into three subtasks:

- *Discourse planning:* Constructing a discourse plan that specifies the propositions to be communicated and their organization.
- *Sentence planning:* Mapping semantic specifications to syntactic roles.
- *Surface generation:* Mapping syntactic roles to natural language.

Each module must in one pass take into account a multitude of complex constraints to interpret the specifications given to it and to factor in constraints at its own level.

To illustrate, consider the issue of clause aggregation, a central problem in multi-sentential explanation generation. Suppose an explanation system for the domain of botanical anatomy and physiology is given the task of constructing an explanation of the structure of plant ovaries. Expressed individually, the propositions extracted from a knowledge base on botanical anatomy and physiology might be realized in the text shown in Figure 1. Although this passage accurately communicates the selected propositions, the terseness of each sentence makes the overall effect disjointed. Merely irritating in one example, explanations such as this would be intolerable for extensive user interactions with an explanation system.

To avoid producing a series of abrupt sentences, a discourse planner could be assigned the task of predicting how particular concepts will be realized in order to optimize clause aggregation decisions. However, this approach violates modularity considerations and does not scale well: it significantly complicates the design of the discourse planner by forcing it to attend simultaneously to content selection, discourse organization, and complex syntactic issues. Alternatively, the propositions could be grouped by a single-pass realization system. This approach is also ineffective. Reorganizing, aggregating, and realizing the specifications in a single pass poses innumerable difficulties: the realizer would somehow have to anticipate the cumulative effects of all aggregation decisions with regard to grammaticality, subordination, and lexical choice.

```
The ovary is a kind of plant reproductive
structure.  It is below the style.  It is the
location of double fertilization.  It is located
in the flower.  It is found in angiosperms.  It is
contained in the gynoecium.  It is connected to
the style.  It contains several ovules.  The
subregions of the ovary include the ovarian locule,
the ovarian wall and the funiculus.  The ovary is
the origin of the fruit.  It is organized as a
layer of cells.  It is shaped like a sphere.
```

Figure 1: Initial explanation of ovary

```
The ovary is a plant reproductive structure that
is the location of double fertilization and that
is below the style.  It is located in the flower
in angiosperms.  It is contained in the gynoecium
and it is connected to the style.  It contains
several ovules, it includes the ovarian locule,
the ovarian wall and the funiculus, and it is the
origin of the fruit.  It is organized as a
spherical layer of cells.
```

Figure 2: Revised explanation of ovary

In contrast to single-pass generation, revision-based generation temporally distributes opportunities for making decisions. It is hypothesized that human writers produce extended texts in multiple drafts to help them cope with the enormous complexities of writing [Hayes and Flower, 1986]. Introducing a revision component[1] into explanation architectures can accrue the same benefits. By interposing a revision system between an explanation system's sentence planner and its surface realizer, we can enable the explanation system to construct texts by iteratively considering individual decisions, building drafts that incorporate those decisions, evaluating the drafts, and making and retracting additional decisions until an acceptable text has been constructed. For example, an explanation system with a revision component could iteratively refine the choppy text shown in Figure 1 to produce the more fluid prose in Figure 2.

However, because of the enormous number of factors that contribute to each revision decision, previous ef-

[1] Revision-based generation is similar to incremental generation, e.g., [Smedt, 1990], but incremental generators are typically monotonic, i.e., they do not retract decisions.

forts to develop computational models of revision have not fared well in these respects. Some never yielded implemented systems [Vaughan and McDonald, 1986; Yazdani, 1987]. One (YH) operates in a toy domain with highly domain-specific revision operators [Gabriel, 1988]. Others produced restricted proof-of-concept systems that work on only a few examples and do not include facilities for performing multi-sentential revisions, e.g., WEIVER [Inui *et al.*, 1992], or a principled formalism for non-monotonically retracting revision decisions, e.g., KDS [Mann and Moore, 1981]. Attacking the full complexities of revision, STREAK [Robin, 1994] is by far the most successful revision-based project to date. It generates texts that are both clear and fluid, but it operates only on single (albeit very complex) sentences and suffers from serious efficiency problems, requiring, for example, more than 2 hours to produce one particular sentence.

3 Revising in Abstraction Spaces

Revising multi-sentential explanations can be cast as a search problem: alternative discourse plans are states; editing actions that manipulate discourse plans are operators. Given an initial discourse plan, a revision system iteratively selects an operator to create a revised discourse plan which is then evaluated against the given stylistic criteria. Discourse plans which are judged to produce superior texts are selected and further revised. Revision-based generators that operate in interactive environments must provide efficient solutions to evaluating candidate operators and representing revision histories.

To address these problems, we have developed REVISOR, a revision system that dynamically improves multi-sentential explanations by searching through an abstraction space of discourse plans. Rather than enacting revisions by reasoning about all of the syntactic and semantic details of discourse plans, it abstracts away all but the most essential aspects of a discourse plan and performs all manipulations on drafts encoded in the abstracted representation. By conducting its search through this abstraction space, it efficiently evaluates candidate revision operators, applies selected operators to create new drafts, and retracts operators to return to previous drafts. REVISOR (Figure 3) dynamically improves the explanation with the following three-phase process:

1. **Discourse Plan Abstraction:** It maps the initial *ground level* discourse plan to an *abstract discourse plan* by excising (and storing away for future use) all but the most essential syntactic and semantic features of each sentential specification. Discourse constraints, which specify inter-sentential organizational requirements, as well as focus constraints, are passed directly to the revision system.
2. **Abstract Discourse Revision:** It iteratively applies revision operators to the abstract discourse plan to create a *draft tree*, where each node is a draft that was derived from its parent by applying a revision operator.

```
((cat clause)
 (proc
  ((type locative) (lex ``organize'')
   (voice passive) (mode equative)
   (passive-prep ((lex ``as'')))))
 (agentless no)
 (partic
  ((location
    ((cat common) (countable no) (lex ``cambium'')
     (describer === ``fascicular'')))
   (located
    ((cat common) (lex ``layer'') (definite no)
     (qualifier
      ((cat pp) (prep ((lex ``of'')))
       (np ((cat common) (number plural)
            (countable no) (lex ``cell'')))))))))
 (syntax independent-sentence)
 (subj-path {partic location})
 (semantics organizational-clause)
 (obj-path {partic located})
 (subject-index fascicular-cambium)
 (object-index layer)
 (fd-index spec-52) (spec organ-desc))
```

Figure 4: Example sentential specification

3. **Discourse Plan Grounding:** It reconstitutes abstract discourse plans as ground level discourse plans by first locating the syntactic and semantic information that was stored away during the abstraction phase and then integrating it into the structures specified by the abstract discourse plan.

The revised ground level discourse plan is then passed to a surface generator, which produces the final explanation.

3.1 Abstraction

Given a ground level discourse plan, the first task is to remove the syntactic and semantic features that are not essential for revision operations. Because revision systems require knowledge about the syntax of elements in the discourse plan, discourse plans must include some features of the sentential specifications produced by a sentence planner. For example, Figure 4 shows the sentential specification for the sentence, "Fascicular cambium is organized as a layer of cells." Discourse plans house a collection of such sentential specifications, one for each *protosentence*[2] to be generated. The approach of revising with abstract discourse plans is based on the observation that it is possible to enact a broad range of text-improving revisions by considering only a small subset of features encoding protosentences. It is therefore possible to temporarily extract these details, manipulate abstracted versions of the representation, and then re-introduce the details prior to realization.

[2]We refer to the typically simple sentence that would be generated from a single sentential specification in an unrevised discourse plan as a protosentence.

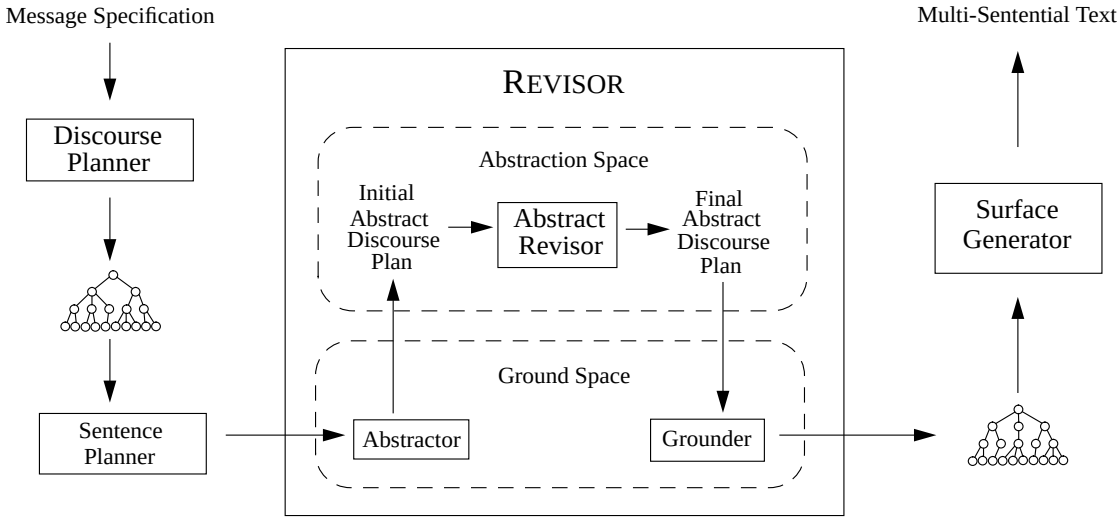

Figure 3: The revision-based explanation architecture

To illustrate, suppose a revision system is given the task of optimizing explanations with respect to clause aggregation. In enacting clause aggregation revisions, it need not consider the full collection of features such as those in Figure 4. For example, to combine the clauses "fascicular cambium is connected to extraxylary fibers" and "fascicular cambium is generated from procambium" into the clause "fascicular cambium is connected to extraxylary fibers and is generated from procambium", a revision operator need not consider adjectival modifiers such as "extraxylary." Nor must it consider the semantic role played by fascicular cambium in each clause, the tense of the verbs, or specific lexical knowledge. This is not to say that this knowledge is not ultimately required to generate the final sentences (it is), nor that it could not in principle be used in some revision decisions (it could). Rather, the decision to combine the clauses with a particular technique (in this case, conjunctivization) can be made without detailed knowledge of the constituents.

Revision systems must retain all sentential knowledge that is critical for making revisions but can excise all features that are not essential. Because REVISOR focuses on clause aggregation revisions, it requires knowledge of a discourse plan's clausal semantics, fundamental syntactic type, and the concepts holding the subject and object roles. Hence, for the sentential specification shown in Figure 4, it retains the features specifying that the clause is an `organizational-clause`, that it is currently expressed as an `independent-sentence`, and the identities of its subjects and objects. It abstracts away features representing:

- *Semantic roles:* Agent, possessor, identifier, etc.
- *Verb features:* Voice, tense, particles, etc.
- *Modifiers:* Noun modifiers, adjectival modifiers, prepositional phrase modifiers, etc.

Given:
- D_0, the initial draft
- C_D, discourse constraints
- C_F, focus constraints
- C_G, global revision constraints

Do:
$D_i \leftarrow D_0$
$CurrentClauseElement \leftarrow$ first clause-element of D_i
while $\neg\exists$ unrevised clause elements in D_i **do**
  if no revisions have been performed on D_i
  and $\exists R$,a revision operator such that:
    R has not been considered
      for $CurrentClauseElement$
    and applying R does not violate
      any constraints in C_D, C_F, or C_G
  then apply R to create draft D_{i+1}
    $CurrentClauseElement \leftarrow$
      next clause element of draft D_{i+1}
  else if $\neg\exists R$ **and** $i \neq 0$
    then backtrack to draft D_{i-1}

Figure 5: Revision Algorithm

- *Lexical features:* Number, lexemes, etc.

To ensure that it can later reconstitute a fully specified ground level discourse plan, it retains *anchors* for the excised semantic, syntactic, and lexical details. Depending on the genre of the text to be revised, large fractions of discourse plans' sentential specifications can be excluded from consideration by revision operators. For example, REVISOR, which operates on scientific expository texts, typically abstracts away 80% of the features in sentential specifications.

3.2 Abstract Discourse Revision

The Abstractor provides the Abstract Revisor with four sets of information. First, it supplies an initial draft

D_0, which consists of an ordered sequence of *clause elements*, where each clause element represents a protosentence. Second, it supplies a set of *discourse constraints* C_D, which specify a partial order on the clause elements. These are passed directly from the discourse planner. For example, the discourse planner might pass a constraint stating that, for an object explanation, all clause elements that communicate detailed propositions about an object must follow all other clause elements. Third, the Abstract Revisor is given a set of *focus constraints* C_F, which restrict marked concepts to occupying subject or subject-modifying positions. Fourth, the Abstract Revisor employs its own *global revision constraints* C_G. These impose stylistic restrictions, e.g., limiting sentence complexity by not permitting any sentence to express more than n propositions.

To dynamically improve an explanation, the revision system composes a tree of drafts. Beginning with the initial draft, it conducts a depth-first exploration of the abstract discourse space. Each iteration of the revision algorithm (Figure 5) consists of selecting a candidate revision operator R and evaluating its pre-conditions against the current draft D_i. If R's pre-conditions are satisfied, the revision system determines whether a new draft D_{i+1} would violate any discourse constraints, focus constraints, or global revision constraints. If so, it considers other operators or backtracks; if not, it performs R's actions to construct D_{i+1}. In the worst case, i.e., when the system is unable to construct an acceptable revision, this process continues until all revision operators have been considered for all unrevised clause elements in the current draft.

Revision operators (Figure 6) consist of (1) *preconditions*, which are predicates on the syntactic and semantic types of the clause elements in a draft that must hold if the operator is to be applied, and (2) *actions*, which specify the changes to be enacted to create a "child" draft from a "parent" draft. The revision system employs two types of revision operators:

- *Transformational operators:* Aggregate adjacent elements of abstract discourse plans.
- *Migrational operators:* Permute adjacent elements of abstract discourse plans to enact reorganizations.

Given two clause elements C_1 and C_2 in a draft, transformational operators aggregate them by modifying their features to create C_1' and C_2', where either C_1' is subordinate to C_2' or vice versa. For example, suppose the `Aggregate-Organizational-With-Perceptual` operator is applied to two clauses, where C_1 = "the ovary is organized as a layer of cells" and C_2 = "the ovary is shaped like a sphere." Because classifying adjectives that modify objects of clauses always immediately precede the objects, the operator will produce an aggregation consisting of a C_1' and a C_2' that (if no further revisions are applied) will be realized as "the ovary is organized as a spherical layer of cells." By interleaving the application of transformational and migrational operators, the revision system can explore a large portion of the abstract discourse space.

```
Aggregate-Organizational-With-Perceptual Operator
  Type: Transformational
  Pre-Conditions:
     (Syntactic-type ?C1 Independent-Sentence)
     (Semantic-type ?C1 Organizational-Clause)
     (Syntactic-type ?C2 Independent-Sentence)
     (Semantic-type ?C2 Perceptual-Clause)
     (Subject-concept ?C1 ?S1)
     (Subject-concept ?C2 ?S2)
     (= ?S1 ?S2)
  Actions:
     (Remove
       (Syntactic-type
           ?C2 Independent-Sentence))
     (Add
       (Syntactic-type
           ?C2 Object-Adjectival-Modifier))
     (Make-Subordinate ?C2)
```

Figure 6: An abstract revision operator

The constraints passed to the revision system restrict the application of revision operators in two ways. First, by employing focus constraints and global revision constraints, the revision system avoids drafts that would be created by the inappropriate application of transformational operators. Second, by employing discourse constraints, it avoids drafts that would be created by the inappropriate application of migrational operators. Because discourse constraints induce "boundaries" in drafts across which clause elements cannot be transported, they prevent the revisor from degrading the organizational structure specified by the discourse planner.

3.3 Grounding

When the abstract revision phase is complete, the Grounder reconstitutes the abstract discourse plan as a ground level discourse plan by (1) locating the syntactic and semantic information that was stored away during the abstraction phase and (2) integrating it into the structures specified by the abstract discourse plan. Because it stored away *anchors* for the detailed information about the constituents of the clause elements during the Abstraction phase, it can properly restructure the pieces according to the new organization specified in the final abstract discourse plan.

To reconstitute the corresponding ground level discourse plan, it iterates across the clause elements of the final draft of the abstract discourse plan. Recall that clause elements are organized as a sequence of independent sentences, each of which may be accompanied by some number of modifiers. As the grounding algorithm iterates through each clause element C, it restores the sentential specification for C by following the anchor link stored in the revised abstract discourse plan for C. In addition, it retrieves the sentential specifications for each modifier of C through its anchor links, and restructures it based upon the new syntactic and semantic type tags

in the revised abstract discourse plan. This is accomplished by applying a series of sentential specification *rewriting* operators, each of which is targeted toward restructuring a sentential specification of a particular syntactic type. The rewriting operators remove subsets of the subordinate sentential specifications and insert them into the appropriate locations in the superordinate specification.

The net result of the grounding phase is a sequence of modified sentential specifications whose organization corresponds to the revised abstract discourse plan. These specifications are then passed to the surface generator for realization. After applying a simple pronominalization strategy (i.e., much less sophisticated than that presented in [Dale, 1992]), the surface generator constructs the final text.

4 A Unification-Based Implementation

The revision-based approach to explanation generation has been implemented in REVISOR, a revision system that improves explanations by creating multiple drafts represented in abstract discourse plans. REVISOR operates in concert with a full-scale discourse planner (KNIGHT, [Lester and Porter, 1997]), a robust sentence planner (FARE, [Callaway and Lester, 1995]), a surface generator with a large systemic/functional grammar (FUF, [Elhadad, 1991]), a large-scale knowledge base in the domain of botanical anatomy and physiology [Porter *et al.*, 1988], and a rich lexicon with an average of approximately 10 systemic features per entry. It employs 37 revision operators to enact semantics-preserving clause aggregations. Through both transformational and migrational revision actions, it creates a variety of aggregations with prepositional phrases, coordinations, contrastive conjunctions, adjectival modifications, nonrestrictive subject relative clauses, and restrictive relative clauses.

The greatest challenge in implementing a scalable revision-based explanation system, i.e., a system that can generate a variety of explanations rather than one or two examples, is efficiently representing the draft tree. To address this problem, REVISOR is implemented in a unification formalism which permits it to efficiently represent multiple competing drafts: the state of unification at each point in the computation represents the draft tree implicitly. In particular, REVISOR is implemented in NONMON-FUF, a non-monotonic version of FUF [Elhadad, 1991] devised by the authors to accelerate revision decisions. In contrast to classical (monotonic) unification, non-monotonic unification allows features to be removed from the cumulatively formed unification expressions, thereby permitting greater flexibility in dynamically modifying drafts during revision.

REVISOR dynamically improves explanations by rewriting choppy explanations consisting of many terse sentences to compose fluid explanations with fewer but more complex sentences. For example, when requested to explain the structure and function of plant ovaries, the explanation system without REVISOR generates the explanation shown in Figure 1. However, with REVISOR, it considers approximately 90 drafts, finally producing the more fluid explanation shown in Figure 2.

5 Evaluation

To evaluate the revision-based approach to explanation generation, we conducted an empirical study of REVISOR's performance on the task of generating scientific explanations. In this study, REVISOR generated a corpus of multi-sentential scientific explanations by operating with the discourse planner, sentence planner, and surface generator noted above. It produced 24 explanations about concepts in the domain of botanical anatomy and physiology (e.g., endodermis, central mother cells, microspore), each spanning on average 12 sentences.

Efficiency. REVISOR's execution times were measured as it generated the 24 explanations. Running on a DEC Alpha, it required on average less than 1 minute to generate each explanation. Of this time, revision *per se* required 1–2 seconds.

Domain-Specific Scalability. We undertook an informal two-phase *training-and-test* experiment, analogous to studies conducted in machine learning research. First, we developed REVISOR's revision operators with a set of "training" concepts. In this training phase, we experimented with different versions of the operators to generate a variety of explanations about 8 concepts. We then froze the operators and observed their effects on the remaining 16 concepts. It was found that the revision operators were applicable to other concepts in the domain, as judged by their ability to perform their intended function of clause aggregation. REVISOR revised initial "choppy" explanations from the "test" set with an average of 12 sentences into improved smoother explanations with an average 6 sentences.

Prose Quality. We conducted a formal *blind prose study* with a panel of judges. First, the REVISOR-less explanation system generated explanations of the 24 concepts. Second, the explanation system with REVISOR generated explanations of the same concepts. Finally, the 24 pairs of explanations were presented to a panel of 4 judges, none of whom had specialized knowledge of the domain or were aware of the purpose of the study. The following conditions held throughout the study. (1) No modifications were made to the revision operators after their initial development on the first 8 concepts. Hence, 16 of the 24 revised explanations presented to the judges were produced with revision operators created by analyzing only the first 8 concepts. (2) None of the explanations were marked for the judges as being "unrevised" or "revised." (3) The order of explanations in each pair, i.e., whether the unrevised explanation preceded the revised explanation or vice versa, was randomized. Judges

were asked to indicate the superior explanation for each pair, yielding the following results:

- 3 of the 4 judges preferred the revised explanations for most (79% or more) of the pairs.
- 3 of the 4 judges preferred the revised explanations for most (75% or more) of the 16 "test" (i.e., "non-training") pairs.
- 2 of the 4 judges preferred the revised explanations for *every* one of the 24 pairs.

6 Conclusion

Revision-based explanation generation offers a promising means of dynamically improving explanations. With virtually no increase in runtime, introducing a well-designed revision component into an explanation system can significantly increase the quality of its prose. To combat the complexity of revision, a revision system can search through an abstraction space of discourse plans to efficiently compose and evaluate a large number of drafts. This approach has been implemented in a non-monotonic unification formalism and empirically evaluated in a scientific domain. Results of this study suggest that revision-based explanation systems can (1) dynamically improve explanations, as demonstrated by a blind experiment by a panel of judges, and (2) they can do so in an efficient manner. This work represents a promising step toward the goal of interactive explanation systems that produce human-quality prose while operating under realtime constraints. One of the greatest challenges ahead lies in expanding realtime revision techniques to support more sophisticated lexical choice. We will be exploring these issues in our future research.

Acknowledgements

The authors wish to thank: Bruce Porter for making available the Botany Knowledge Base; Michael Elhadad for creating and generously assisting us with FUF: and Stuart Towns for insightful comments on an earlier draft of this paper.

References

[Callaway and Lester, 1995] Charles Callaway and James Lester. Robust natural language generation from large-scale knowledge bases. In *Proceedings of the Fourth Bar-Ilan Symposium on the Foundations of Artificial Intelligence*, pages 96–105, 1995.

[Cawsey, 1992] Alison Cawsey. *Explanation and Interaction: The Computer Generation of Explanatory Dialogues.* MIT Press, 1992.

[Dale, 1992] Robert Dale. *Generating Referring Expressions.* MIT Press, 1992.

[Dalianis and Hovy, 1993] Hercules Dalianis and Eduard Hovy. Aggregation in natural language generation. In *Proceedings of the Fourth European Workshop on Natural Language Generation*, Pisa, Italy, 1993.

[Elhadad, 1991] Michael Elhadad. FUF: The universal unifier user manual version 5.0. Technical Report CUCS-038-91, Department of Computer Science, Columbia University, 1991.

[Gabriel, 1988] Richard P. Gabriel. Deliberate writing. In David D. McDonald and Leonard Bolc, editors, *Natural Language Generation Systems*, pages 1–46. Springer-Verlag, New York, 1988.

[Hayes and Flower, 1986] John R. Hayes and Linda S. Flower. Writing research and the writer. *American Psychologist*, 41:1106–1113, 1986.

[Inui *et al.*, 1992] Kentaro Inui, Takenobu Tokunaga, and Hozumi Tanaka. Text revision: A model and its implementation. In Robert Dale, Eduard Hovy, Dietmar Rosner, and Oliviero Stock, editors, *Aspects of Automated Natural Language Generation*, pages 215–230. Springer-Verlag, 1992.

[Lester and Porter, 1997] James C. Lester and Bruce W. Porter. Developing and empirically evaluating robust explanation generators: The KNIGHT experiments. *Computational Linguistics*, 23(1):65–101, 1997.

[Mann and Moore, 1981] William C. Mann and James A. Moore. Computer generation of multiparagraph english text. *American Journal of Computational Linguistics*, 7(1):17–29, 1981.

[Mittal, 1993] Vibhu O. Mittal. *Generating Natural Language Descriptions with Integrated Text and Examples.* PhD thesis, University of Southern California, September 1993.

[Moore, 1995] Johanna D. Moore. *Participating in Explanatory Dialogues.* MIT Press, 1995.

[Porter *et al.*, 1988] Bruce Porter, James Lester, Kenneth Murray, Karen Pittman, Art Souther, Liane Acker, and Tom Jones. AI research in the context of a multifunctional knowledge base: The botany knowledge base project. Technical Report AI Laboratory AI88-88, University of Texas at Austin, Austin, TX, 1988.

[Robin, 1994] Jaques Robin. *Revision-Based Generation of Natural Language Summaries Providing Historical Background.* PhD thesis, Columbia University, December 1994.

[Smedt, 1990] Koenraad De Smedt. IPF: An incremental parallel formulator. In Robert Dale, Chris Mellish, and Michael Zock, editors, *Current Research in Natural Language Generation*, chapter 7. Academic Press, London, 1990.

[Suthers, 1993] Daniel D. Suthers. *An Analysis of Explanation and Its Implications for the Design of Explanation Planners.* PhD thesis, University of MA, February 1993.

[Vaughan and McDonald, 1986] Marie M. Vaughan and David D. McDonald. A model of revision in natural language generation. In *Proceedings of the 24th Annual Meeting*, pages 90–96, Columbia University, 1986. Association for Computational Linguistics.

[Yazdani, 1987] Masoud Yazdani. Reviewing as a component of the text generation process. In Gerard Kempen, editor, *Natural Language Generation*, pages 183–190. Martinus Nijhoff, Dordrecht, The Netherlands, 1987.