Handling Dependencies in Reorganizing Content Specifications A Case Study of Case Analysis

Helmut Horacek
Universität des Saarlandes, FR 6.2 Informatik
Postfach 151150, D-66041 Saarbrücken, Germany
email: horacek@cs.uni-sb.de

Abstract

Handling the dependencies among alternatives in composing expressions in an efficient and qualitatively accurate manner is a fundamental problem of NLG. To pursue this goal effectively, simplifications are put forward in practical approaches, but also ambitious control regimes are tried out occasionally. However, neither of these is able to operate adequately on larger and involved structures. Approaching this issue in a methodological way, we present a case study from the area of mathematical proofs that illustrates the rhetorically motivated reorganization of machine-generated case analyses. Ingredients of this investigation are the design of optimization operations, effect-minimizing orderings on groups of operations, and tentative applications of local operations to test the effects of crucial dependencies. Our approach conceives NLG as a standard pipe-line architecture putting emphasis on orderings, with local revisions as a minor extension. This is particularly effective when text planning is organized as an optimization rather than as a construction process, such as for the presentation of mathematical proofs.

1 Introduction

Handling the dependencies among alternatives in composing expressions in an efficient and qualitatively accurate manner is a fundamental problem of NLG. To pursue this goal effectively, simplifications are typically put forward in practical approaches, following the standard, pipe-line architecture (Reiter, 1994). In order to handle dependencies in a more profound manner, some ambitious control regimes have been tried out to compute the best combination of alternatives within a local task that satisfies the constraints imposed by these dependencies. Tasks addressed include lexicalization (Beale, 1997) and referring expression generation (Gardent, 2002), which transfer the problem, making it accessible to a general and fast search procedure (on the lines of (Germann et al. 2001) for machine translation). However, neither of these approaches is able to operate adequately on larger and involved structures.

Approaching this issue in a methodological way, we present a case study from the area of mathematical proofs, that illustrates the rhetorically motivated reorganization of machine-generated case analyses. In order to adapt the structure of machine-generated proofs better on human needs, we have analyzed differences between internal proof representations and some structural properties underlying textbook proofs, focusing on case analyses. The differences observed inspired us in defining a set of restructuring operations for proof graph representations whose contextually motivated application enables expressing these reasoning structures in a rhetorically adequate manner. Ingredients of this investigation are the design of optimization operations, effect-minimizing orderings on groups of operations, and tentative applications of local operations to test the effects of crucial dependencies. Our approach conceives NLG as a standard pipe-line architecture putting emphasis on orderings, with local revisions as a minor extension. This is particularly effective when text planning is organized as an optimization rather than as a construction process, such as for the presentation of mathematical proofs.

This paper is organized as follows. We examine the role of case analysis in book presentations. We formalize a set of reorganization operations, and describe their contextually motivated application. We illustrate the effect of these operations by reorganizing a moderately complex proof for NL presentation. Finally, we discuss impacts of our approach.

2 Motivation – Needs of the Domain

Representations of results of automatically found proofs differ from rhetorically adequate natural language (NL) presentations of these proofs not only in their format and conventions but more fundamentally in their content and structure. While the adequacy of a presentation's content has been addressed by a number of transformation and abstraction techniques, a proof's structure is typically preserved by today's proof presentation systems. This is hardly surprising – preserving given structures in specifications is also what NL generation systems typically do.

If either a or b is 0, then both $|ab|$ and $|a||b|$ are equal to 0. If $a > 0$ and $b > 0$, then $ab > 0$ so that $|ab| = ab = |a||b|$. If $a > 0$ and $b < 0$, then $ab < 0$ so that $|ab| = -ab = a(-b) = |a||b|$. The other two cases are treated similarly.

Figure 1. The proof of the lemma $|ab| = |a||b|$ in an analysis textbook (Bartle, Sherbert, 1991)

Case analysis is a common reasoning structure in several areas of mathematics. It is applied to solve subproblems which require distinct inferencing in dependency of the values that some expression can take. The simplest and rather common distinction is truth or falsity of a logical expression. Another common situation is the distinction of values of a numerical expression when broken down into intervals. As an example, consider a proof of the elementary lemma $|ab| = |a||b|$. In an introductory book on analysis (Bartle, Sherbert, 1991), it runs as indicated in Figure 1. A verbalization in the style of the proof presentation system *P.rex* (Fiedler, 2001) is given in Figure 2. In contrast to the stereotype machine-generated text, the textbook version is better for two main reasons: (1) each case conveys a reasoning chain distinct from other cases, and (2) concise conditional clauses are formulated instead of overloaded textual markings of underlying case structures. Further effective presentation means used in the textbook are leaving out easily inferable intermediate steps ($|0b|$, $|a0|$, $0|b|$, $|a|0$ in the machine-generated version), referring to analogy, thereby trusting the addressee to be able to make use of it, and extraposing crucial justifications ($ab < 0$, $ab > 0$). In order to present case analyses of automatically found proofs in a similar manner, making adequate use of these presentation means is required. The inference issue, that is

To prove the lemma, let us consider the cases where $a = 0$, $a > 0$, and $a < 0$, respectively.

Case 1: $a = 0$. Then $|ab| = |0b| = 0 = 0|b| = |a||b|$.

Case 2: $a > 0$. Let us consider the cases where $b = 0$, $b > 0$, and $b < 0$, respectively.

Case 2.1: $b = 0$. Then $|ab| = |a0| = 0 = |a|0 = |a||b|$.

Case 2.2: $b > 0$. Then $|ab| = ab = |a||b|$.

Case 2.3: $b < 0$. Then $|ab| = -ab = a(-b) = |a||b|$.

Case 3: $a < 0$. Let us consider the cases where $b = 0$, $b > 0$, and $b < 0$, respectively.

Case 3.1: $b = 0$. Then $|ab| = |a0| = 0 = |a|0 = |a||b|$.

Case 3.2: $b > 0$. Then $|ab| = -ab = (-a)b = |a||b|$.

Case 3.3: $b < 0$. Then $|ab| = ab = (-a)(-b) = |a||b|$.

Figure 2. The proof of the lemma $|ab| = |a||b|$ in the style of the system *P.rex* (Fiedler, 2001)

crucial for many machine-found proofs, can be handled reasonably well (Horacek, 1999). Therefore, we can concentrate our effort on transducing case analysis structures underlying machine-found proofs into structures that mimic those found in textbooks. For this purpose, we have analyzed presentations of comparably simple proofs in two textbooks: a book on analysis (Bartle, Sherbert, 1991) and another one on algebra (Lamprecht, 1981). The proofs examined are likely to be subject to explanations or to exercises in tutorial sessions.

The central issue is to characterize situations for the use of implicit forms of expressing case analyses, in terms simple enough for estimating them on the level of the proof graph. Implicit forms may be sequences of conditional clauses, followed by stating that the conclusion is independent of the conditions, or preceding a critical case, introduced by “it remains to show that this also holds for <condition>”. Therefore, we have looked at complexities of the inference chains and expressions involved. For the formal account, we have approximated these properties in terms of the number of clauses (a series of equations was counted as a single clause). We found out that case analyses are expressed explicitly only if the presentation of two or more individual cases requires a certain degree of complexity. Therefore, we consider the length of the second largest branch a crucial parameter (P_1, default value 2) according to the results in Table 1 (26 examples found in the two books). Short case analyses marked explicitly contain untypically large expressions (we neglect this because of the low frequency), or they combine several cases (e.g., by the union of intervals).

Further characterizations gained from the textbook proofs are as follows: The case analyses contain mostly two, rarely more than three

length of inference chain	*per case*						*non-largest cases*					
number of clauses	1	2	3	4	5	> 5	1	2	3	4	5	> 5
frequencies implicit forms	12	12	3	1	1	1	8	6	0	0	0	0
frequencies explicit forms	1	9	5	2	4	4	3	5	0	3	2	2

Table 1. Textual forms of expressing case analyses and number of clauses

cases. Therefore, we limit the number of cases by P_2 (with default 5), for case analyses built by reorganization. Moreover, case expressions found are not more complex than a disjunction of two relations of compatible type (as in Figure 1). We take this into account by using a parameter for the maximum number of operators in case expressions, P_3 (with default 3). Furthermore, all case expressions with an equivalence treated as a conjunction of two implications with opposite directions were marked explicitly, which is easy to obey as a qualitative criterion. Finally, we encountered only one nested case analysis (a longer induction proof), although it is clear from the literature that this occurs more frequently in advanced proofs.

The estimated parameter values guide pursuing the requirements for a rhetorically adequate presentation of case analyses:

- Be economic by keeping the number of cases small.
- Produce structures that enable the use of implicit textual forms.
- Avoid nested case analyses, values of P_2 and P_3 permitting.

In order to fulfill these requirements, some measures are applied, including the reduction of inference chains within individual cases, aggregation of nearly identical cases, and the linearization of nested case analyses. The structures obtained can be explored in a human-oriented way by focusing on crucial distinctions, and they can be expressed more naturally and concisely by NL texts.

3 Restructuring Operations

As the texts in Figures 1 and 2 demonstrate, automatically generated proofs typically meet these requirements to an insufficient degree only. Since there are several logically equivalent forms of a proof, building alternative representations can be pursued that are preferable in rhetorical terms. This is done by condensing a proof, to meet the constraints just elaborated. Three operations aim at this goal, through reducing case analyses in terms of their *length*, *number* of cases, and *depth*:

Operation 1. The *length* of the reasoning line within a single case can be reduced if a subsequence heading this inference line is independent of the case assumption. This subsequence can be lifted out of the scope of the case analysis, and precede the case analysis in the presentation. This operation may be applicable several times, to multiple branches within the same case analysis. Lifting out subsequences is then delicate due to coherence matters, since reasoning chains that are originally sequential would become interleaved. Therefore, if several cases can be reduced by this operation, it is only applied to the one leading to the largest reduction, provided the second largest branch gets reduced enough so that its length falls below threshold P_1 (see the example in Section 5).

Operation 2. The *number* of cases can be reduced if the reasoning lines in some of the cases are identical but for references to the corresponding case assumptions. Such cases can be aggregated by building a disjunction of their assumptions, and by substituting this expression for the references to the original assumptions, provided that case expression is less complex than P_3 allows. Remaining within complexity limits may be possible by simplifications (e.g., union of overlapping or adjacent intervals). If not only some cases, but all cases can be aggregated this way, the entire case analysis collapses into a single inference chain. This can occur frequently in connection with calls to external systems, such as computer algebra systems – a case analysis has to be carried out for each value separately, but it turns out that the inference lines differ only in these values. We encountered such a situation for categorizing residue classes (Meier, Pollet, and Sorge, 2002). In our current context, operation 2 is applicable to cases 2.1 and 3.1 in the proof in Figure 2, provided these cases are lifted from the embedding case analyses (through operation 3). Operation 2 would also be applicable to any of these cases and case 1, provided the intermediate expressions (|*a*0], |*a*|0, |0*b*|, 0|*b*|) are left out – this illustrates the importance of handling the inferability issue.

Operation 3. The *depth* of a case analysis with further embedded case analyses can be reduced by lifting an embedded case analysis to the level of the embedding one. This is done by moving copies of the inferences of the embedding case that precede the embedded case analysis into that structure, and by merging the case assumption associated with the embedding case with each of the case assumptions associated with the embedded structure. Requirements for this operation are that the types of relations in the merged case assumptions are compatible, that the resulting expression is less complex than P_2 and P_3 allow, and that the original structures would be expressed explicitly; otherwise, nested structures are maintained. Moreover, if a length increase is the result of copying inference sequences, subsequent applications of operation 2 to cases from different levels in the original structure must yield reductions compensating that increase.

These operations are formally defined in Figure 3, using generalized case schemata with an arbitrary number of cases n:

$$\frac{\Delta \vdash \overset{i=1,n}{\vee F_i} \qquad \overset{[F_i]}{\Delta, F_i \vdash G}}{\Delta \vdash G}\ \text{CASE}$$

Δ represents a set of assumptions, and F, G, and H (with or without indexes) are metavariables representing expressions. The case schema represents the conclusion that G is derivable from the assumptions Δ ($\Delta \vdash G$), on the basis of two premises: 1) there is a case split, that is, one of the cases F_i holds, given the assumptions Δ ($\Delta \vdash \vee F_i$) for $1 \leq i \leq n$, and 2) asssuming an arbitrary one of these cases ($[F_i]$) enables the derivation of G from the assumptions Δ and the case assumption F_i ($\Delta, F_i \vdash G$).

In operation 1, the derivation $\Delta_0 \vdash F_0$, is lifted out of the scope of the embedding case analysis, since it is independent of the case assumption $[F_k]$. In operation 2, the two cases $[F_j]$ and $[F_k]$ are aggregated into a single case $[F_j \vee F_k]$, provided the associated derivation trees (tree($\Delta, F_j \vdash G$) and tree($\Delta, F_k \vdash G$), respectively), are identical but for the appearances of F_j and F_k, so that substituting $F_j \vee F_k$

Operation	*Original case schema*		*Modified case schema*	*Constraints*
1)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\Delta_0 \vdash F_0 \quad \dfrac{[F_k]}{\Delta, F_k \vdash \Delta_1}}{F_0, \Delta_1 \vdash G} \qquad \dfrac{[F_{i\neq k}] \quad \dots}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\dfrac{}{\Delta_0 \vdash F_0} \quad \dfrac{[F_k]}{\Delta, \{F_k\} \vdash \Delta_1}}{\Delta_1, F_0 \vdash G} \qquad \dfrac{[F_{i\neq k}] \quad \dots}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Delta \supseteq \Delta_0$
2)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \quad \dfrac{[F_j]}{\Delta, F_j \vdash G} \quad \dfrac{[F_k]}{\Delta, F_k \vdash G} \quad \dfrac{[F_{i\neq j,k}]}{\Delta, F_{i\neq j,k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \quad \dfrac{[F_j \vee F_k]}{\Delta, F_j \vee F_k \vdash G} \quad \dfrac{[F_{i\neq j,k}]}{\Delta, F_{i\neq j,k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	tree(Δ, $F_j \vdash G$) $[F_j \mid F_j \vee F_k]$ $\equiv$ tree(Δ, $F_k \vdash G$) $[F_k \mid F_j \vee F_k]$
3)	$\dfrac{\Delta \vdash \vee_{i=1,n} F_i \qquad \dfrac{\dfrac{[F_k]}{\dfrac{\Delta_0 \vdash G_0}{\vee_{j=1,m} G_j} \quad \dfrac{[G_j]}{G_0, G_j, F_k \vdash F_0}}}{\Delta, F_0 \vdash G} \qquad \dfrac{[F_{i\neq k}] \quad \dots}{\Delta, F_{i\neq k} \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$\Rightarrow$	$\dfrac{\Delta \vdash \vee_{i=1,n+m\ (i\neq k)} H_i \qquad \dfrac{[H_i] \quad \dots}{\Delta, H_i \vdash G}}{\Delta \vdash G}\ \text{CASE}$	$i=1,n\ (i\neq k)$: $\mathrm{H_i} \equiv F_i$ $i=n+1,n+m$: $\mathrm{H_i} \equiv F_k \vee G_{i-n}$ tree(Δ, $H_i \vdash G$) $\equiv$ Adjoin(tree($\Delta_0 \vdash G_0$), tree(G_i, G_0, $F_k \vdash F_0$, $\Delta \vdash G$))

Figure 3. Proof reorganizing operations that reduce case analyses in terms of length (1), number of cases (2), and depth (3)

for each of them leads to identical derivation trees. Finally, operation 3 merges the cases $[G_j]$ with the embedding case $[F_k]$, which replaces case F_k by m new cases in the embedding case analysis. The new case assumptions are $F_k \vee G_{i-n}$, and the derivation trees (tree($\Delta_0 \vdash G_0$) from F_k, and tree(G_0, G_j, $F_k \vdash F_0$, $\Delta \vdash G$) from each G_j) are composed through adjoinment.

4 Organizing the Reorganization

Judging the adequacy of individual operation applications in context is delicate since it is not locally clear whether or not a case analysis can ultimately be expressed by a concise sentence pattern. There are two reasons for this difficulty, which are typical for processes in NL generation and organizing their specifications:

External dependencies, which appear in terms of how inference chains within a case analysis are verbalized. This is determined by decisions about the content (building abstractions due to the inferability issue) and about form, the verbalization proper.

Internal dependencies, since reductions are usually brought about by the composition of several operation applications.

In order to handle external dependencies, the content-related processes (here: the abstraction process as defined in (Horacek, 1999)) are carried out prior to dealing with reorganization, since this usually leads to significant length reductions that are unforeseeable otherwise. Doing this independently of case analysis reorganizations is also possible, because the inferability addresses sequences of inferences rather than tree-like structures. Conversely, effects of proper verbalization have to be anticipated, because interleaving them with reorganization would be very expensive – verbalizing some substructures several times would be required, since the reorganization process uses local backtracking (see below). Therefore, length estimates are required for the resulting structures. We obtain them by simply counting the number of inference steps, adding one point for required structural markings. The numbers obtained this way are comparable to the threshold parameters estimated on the basis of the empirical analysis.

Concerning internal dependencies, determining the local suitability is only difficult for operation 1. In theory, multiple applications to the same case may be possible and required to be tested; in practice, it is mostly a single application. However, testing multiple applications in an efficient manner is not trivial. It is done by traversing the proof subgraphs of each case separately, starting from the leaf nodes, until a use of the case assumption is encountered; the size of the remaining portion of the proof

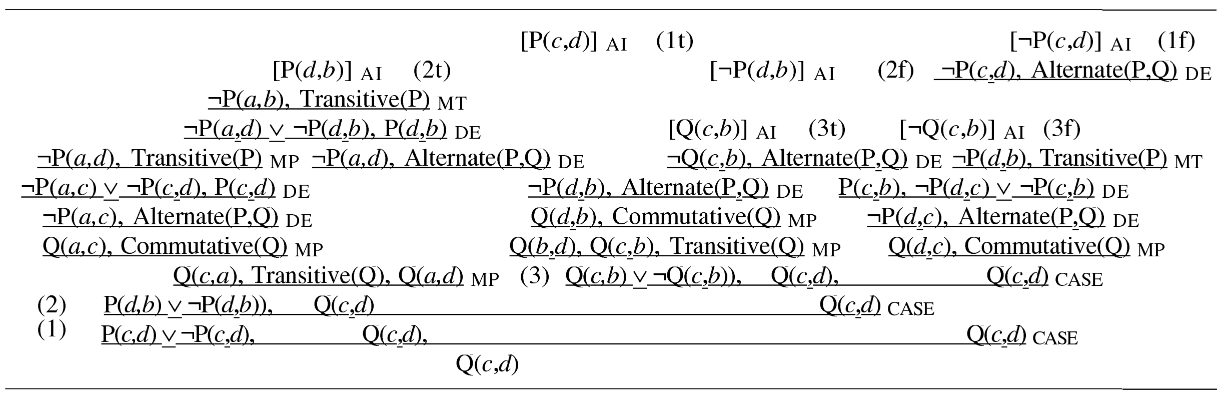
[P(c,d)] AI (1t)
[¬P(c,d)] AI (1f)
[P(d,b)] AI (2t)
[¬P(d,b)] AI (2f)
¬P(c,d), Alternate(P,Q) DE
¬P(a,b), Transitive(P) MT
¬P(a,d) ∨ ¬P(d,b), P(d,b) DE
[Q(c,b)] AI (3t)
[¬Q(c,b)] AI (3f)
¬P(a,d), Transitive(P) MP
¬P(a,d), Alternate(P,Q) DE
¬Q(c,b), Alternate(P,Q) DE
¬P(d,b), Transitive(P) MT
¬P(a,c) ∨ ¬P(c,d), P(c,d) DE
¬P(d,b), Alternate(P,Q) DE
P(c,b), ¬P(d,c) ∨ ¬P(c,b) DE
¬P(a,c), Alternate(P,Q) DE
Q(d,b), Commutative(Q) MP
¬P(d,c), Alternate(P,Q) DE
Q(a,c), Commutative(Q) MP
Q(b,d), Q(c,b), Transitive(Q) MP
Q(d,c), Commutative(Q) MP
Q(c,a), Transitive(Q), Q(a,d) MP
(3) Q(c,b) ∨ ¬Q(c,b)), Q(c,d), Q(c,d) CASE
(2) P(d,b) ∨ ¬P(d,b)), Q(c,d) Q(c,d) CASE
(1) P(c,d) ∨ ¬P(c,d), Q(c,d), Q(c,d) CASE
Q(c,d)

Figure 5. Initial representation of the proof, abstracted to applications of commutativity, transitivity, and alternativity axioms

subgraph is then compared with P_1. That leaves the dependencies among these operation to be dealt with. Within a full proof, this is done by traversing the entire proof graph starting from its leaf nodes. If a case analysis is encountered, operation 2 is tried first, for all pairs of cases (quadratic complexity). Next, operation 1 is tested for each branch, and the suitability can be decided due to the prior application of operation 2. Finally, linearization with eventually embedded case analyses is treated. If this application of operation 3 leads to a number of cases larger than P_2, it is applied tentatively, testing the effect of operation 2 for the combinations of cases not yet considered (one from each of the case analyses combined tentatively). If this leads to a sufficient reduction, the process is resumed with the structures built tentatively; otherwise, with those built before.

5 A Moderately Complex Example

In this section, we illustrate the capablities of our method by a proof to the problem specified in Figure 4. The abstracted proof representation is given in Figure 5, with abbreviations for the use of axioms: the predicates *transitive*, *commutative*, and *alternate* (of P or Q) stand for the instantiated forms of these axioms in the specific proof line. Moreover, MP, MT, DE, AI, and CASE stand for modus ponens, modus tollens, disjunction elimination, assumption introduction, and case analysis, respectively. This form is obtained from the machine-oriented proof by abstracting to the *partial assertion representation level* according to (Horacek, 1999), that is, applications of axioms, unless their use is considered cognitively difficult (e.g., modus tollens, as motivated in (Johnson-Laird, Byrne, 1990)).

The proof contains three case analyses, discriminating according to the truth values of P(*c*,*d*) (1), P(*d*,*b*) (2), and Q(*c*,*b*) (3), with true (t) and false (f), labeled accordingly in Figure 5. Due to their nestings, a completely explicit verbalization of this structure would result in a

Let Q be a transitive and symmetric relation and P a transitive relation. Let P and Q hold 'alternatively', i.e., P(*x*,*y*) ∨ Q(*x*,*y*) for all *x* and *y*. Let also ¬P(*a*,*b*) for some arbitrary *a* and *b*.
Then Q(*c*,*d*) holds for arbitrary *c* and *d*.

Figure 4. Problem definition

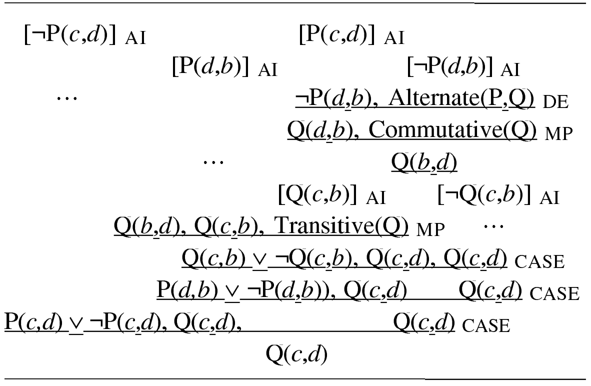
[¬P(c,d)] AI
[P(c,d)] AI
[P(d,b)] AI
[¬P(d,b)] AI
...
¬P(d,b), Alternate(P,Q) DE
Q(d,b), Commutative(Q) MP
...
Q(b,d)
[Q(c,b)] AI
[¬Q(c,b)] AI
Q(b,d), Q(c,b), Transitive(Q) MP
...
Q(c,b) ∨ ¬Q(c,b), Q(c,d), Q(c,d) CASE
P(d,b) ∨ ¬P(d,b)), Q(c,d) Q(c,d) CASE
P(c,d) ∨ ¬P(c,d), Q(c,d), Q(c,d) CASE
Q(c,d)

Figure 6. Final representation of the proof, focusing on modified parts

rather cumbersome text. Since operation 2 is not applicable to any substructure here, we focus on the other operations. Starting with the most embedded case analysis, (3), the applicability of operation 1 is tested for both of its branches. In (3t), the first three inference steps are independent of Q(c,b), while it is only the first one in (3f) for ¬Q(c,b). Since the structure resulting from lifting the longer inference subsequence (of (3t)) out of the scope of (3) remains within the limits P_2 and P_3, operation 1 is applied within (3t). Moving on to the embedding case analysis, (2), operation 1 is only applicable to the first inference step in (2t), but with an insufficient reduction. Operation 3 is not applied because of the expectation that the embedded case analysis can be expressed in an implicit form. Conversely, this operation is also not applied in (1), because this expectation holds for the embedding case analysis. Figure 6 illustrates the structural changes imposed on the proof graph in Figure 5, and the later reordering of (1). Figure 7 gives a possible verbalization. It is based on the assumption that axiom uses, unless being in modus tollens direction, are inferable and can be omitted, and that facts used in inferences need to be reintroduced after not being mentioned for some time (see the model in *P.rex*). Altogether, the example demonstrates that only a few of all possible operator applications are

---

If ¬P(c,d) holds, Q(c,d) follows. Hence, it remains to show, that Q(c,d) also holds if P(c,d). To prove that, let us consider the cases where P(d,b) and ¬P(d,b) hold, respectively.

1. P(d,b) holds. Then ¬P(a,b) implies ¬P(a,d) ∨ ¬P(d,b) due to transitivity, which implies ¬P(a,d) and further Q(a,d). ¬P(a,d) implies ¬P(a,c) ∨ ¬P(c,d) due to transitivity, and further ¬P(a,c) due to P(c,d). This implies Q(a,c) and further Q(c,a). Then Q(a,d) implies Q(c,d).
2. ¬P(d,b) holds. This implies Q(d,b) and further Q(b,d). If Q(c,b) holds, then Q(c,d) is valid. If ¬Q(c,b) holds, then P(c,b). ¬P(d,b) implies ¬P(d,c) ∨ ¬P(c,b) due to transitivity. This implies ¬P(d,c) and further Q(c,d).

Hence Q(c,d) irrespective of the truth of P(d,b).

Figure 7. A possible verbalization of the proof sketched in Figure 6

truly effective, guided by anticipations about options available in subsequent processing.

6 Discussion

Proof presentation systems, as NLG systems in general, tend to express all specifications explicitly and in a form widely corresponding to these specifications. Text planning, notably approaches based on Rhetorical Structure Theory (RST) (Mann, Thompson, 1983) address the choice of connectives and clause structures, such as subordinates versus embedding. Exceptions deal with differences between intentional and ideational structures (Maier, 1985) and elaborations on sequences (Horacek, 1998). The latter measure is also incorporated into skillful presentations of the specific form of series of inequations, for remarks on individual steps in such a series (Fehrer, Horacek, 1997). Altogether, no approach to proof presentation or to NLG in general is able to deal with structures of a case analysis in a rhetorically adequate manner.

The task of proof presentation in natural language can be considered a special form of NLG with restricted language and embedding of formulas, where the machine-generated proof is interpreted as a complete and correct, but rhetorically inadequate text plan. Consequently, reorganizations are essential contributions for making machine-generated solutions (here, proofs) better accessible for tutorial purposes (Melis, Horacek, 2000). The modified proofs indicate more adequately which distinctions to make and what cases to consider first – these are essential aids for supporting humans when searching for a proof in a tutorial environment.

The use of reorganization operations can be considered a first step towards a new approach to text planning. It is conceptualized as an optimization process to some externally given highly structured specification, rather than a homogeneous process (Marcu, 1997) that is applied to sets of facts and relations. Many choices are motivated by soft, size-related criteria, as opposed to the more usual qualitative rhetorical preferences. Text planning is broken into a sequence of subphases, so that the most intertwined dependencies can be dealt within the same subphase in a computationally reasonable manner. The position of a subphase is determined by assessing whether results from other subphases should be available, or estimating them is sufficient for well-motivated choices. Hence, we would incorporate lemmatization into our present model between the inferability and the case analysis task. The deletions obtained by the inferability task are the most crucial information, while the effects of lemmatization are more global than those of handling case analyses. This process organization mediates between opportunistic application of operators and a systematic procedure (as (Dalianis, 1999) and (Shaw, 1998) for aggregation, respectively). It constitutes a compromise between potentially considering all possible combinations and a strict pipe-line.

7 Conclusion

In this paper, we have addressed dependencies among alternatives in composing expressions, for larger and involved structures. We have presented a case study from the area of mathematical proofs that illustrates the rhetorically motivated reorganization of machine-generated case analyses. A set of restructuring operations is defined on the proof graph whose contextually motivated application enables expressing these reasoning structures in a rhetorically adequate manner. Operations include shortening inference chains within individual cases, aggregation of nearly identical cases, and the linearization of nested case analyses. We will soon integrate the stand-alone process into the system *P.rex*. Our approach shows a new kind of handling dependencies in text planning, and it contributes to expressing mathematical proofs for didactic purposes.

References

Bartle, R., Sherbert, D. 1991: Introduction to Real Analysis.

Beale, S. 1997: HUNTER-GATHERER: Applying Constraint Satisfaction, Branch-and-Bound and Solution Synthesis to Computational Semantics. PhD thesis, Carnegie-Mellon University.

Dalianis, H. 1999. Aggregation in Natural Language Generation, *Computational Intelligence* 15(4).

Fehrer, D., Horacek, H. 1999. Presenting Inequations in Mathematical Proofs. In *Information Sciences* 116, pp. 3-23.

Fiedler A. 2001. Dialog-driven Adaptation of Explanations of Proofs. In *Proc. of IJCAI-2001,* pp. 1296-1300.

Gardent, C. 2002. Generating Minimal Definite Descriptions. In Proc. of *ACL-2002*, pp. 96-103.

Germann, U., Jahr, M., Knight, K., Marcu, D., Yamada, K. 2001. Fast and Optimal Decoding for Machine Translation. In Proc. of *ACL-2001*, pp. 228-235.

Horacek, H. 1998. Generating Inference-Rich Discourse Through Revisions of RST-Trees. In *Proc. of AAAI-98,* 814-820.

Horacek, H. 1999. Presenting Proofs in a Human-Oriented Way. In *Proc. of CADE-99*, pp. 142-156.

Johnson-Laird, P., Byrne, R. 1990. Deduction. Ablex Publishing.

Lamprecht E. 1981. Einführung in die Algebra, Birkhäuser Verlag.

Maier, E. 1985. Textual Relations as Parts of Multiple Links Between Text Segments. In Trends in Natural Language Generation – An Artificial Intelligence Perspective, pp. 68-87.

Mann, W., Thompson, S. 1983. Rhetorical Structure Theory: A Theory of Text Organization. Technical Report, ISI/RR-83-115, ISI at University of Southern California.

Marcu, D. 1997. From Local to Global Coherence: A Bottom-Up Approach to Text Planning. In Proc. of *AAAI-97*, pp. 629-635.

Meier, A., Pollet, M., and Sorge V. 2002. Comparing Approaches to Explore the Domain of Residue Classes. *Journal of Symbolic Computation*, 34(4), pp. 287-306.

Melis, E., Horacek, H. 2000. Dialog Issues for a Tutor System Incorporating Expert Problem Solvers. AAAA-Fall Symposium on Building Dialog Systems for Tutorial Applications.

Reiter, E. 1994. Has a Consensus Architecure Appeared, and is it Psycholinguistically Plausible? In Proc. of the *7th International Workshop on Natural Language Generation*, pp. 163-170.

Shaw, J. 1998. Clause Aggregation Using Linguistic Knowledge. In Proc. of the *9th International Workshop on Natural Language Generation*, pp. 138-147.