PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 10/95

Converting Large Proof Structures into Natural Language Hypertext-like Explanations

Denise Aboim Sande e Oliveira
Clarisse Sieckenius de Souza
Edward Hermann Haeusler

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900
RIO DE JANEIRO - BRASIL

PUC RIO - DEPARTAMENTO DE INFORMÁTICA ISSN 0103-9741

Monografias em Ciência da Computação, Nº 10/95

Editor: Carlos J. P. Lucena May, 1995

Convertig Large Proof Structures into Natural Language Hypertext-like Explanations *

Denise Aboim Sande e Oliveira

Clarisse Sieckenius de Souza

Edward Hermann Haeusler

* This work has been sponsored by the Ministério de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

In charge of publications:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC Rio — Departamento de Informática
Rua Marquês de São Vicente, 225 — Gávea
22453-900 — Rio de Janeiro, RJ
Brasil
Tel. +55-21-529 9386 Telex +55-21-31048 Fax +55-21-511 5645
E-mail: rosane@inf.puc-rio.br

Converting Large Proof Structures into Natural Language Hypertext-like Explanations

Denise Aboim Sande e Oliveira

Clarisse Sieckenius de Souza

Edward Hermann Haeusler

Departamento de Informática - PUC-Rio
Rua Marquês de São Vicente 225
22453-900 - Rio de Janeiro, RJ
Brazil
email: {denise,clarisse,hermann}@inf.puc-rio.br

Keywords:

Natural Language Interfaces
Automated Reasoning

Converting Large Proof Structures into Natural Language Hypertext-like Explanations

Abstract

Logic-based reasoning systems present considerable challenges to interface designers. The generation of explanations from proofs is one of the most important of these. This task is often carried out by a conversion of the proof structure into a text plan and has a considerable degree of complexity derived from two facts: the intrinsic size and structure of proofs and the intended style of the text to be generated. We believe that proof-based explanations should be as readable as possible, which requires presenting proofs with enough information to allow for an easy reading/understanding. In this article we present a way of meeting this end by subdividing Natural Deduction (N.D.) proof structures into sub-proofs with an adequate level of information regarding their sub proof conclusions and the relationship between these, their premises, and the whole proof. The structure obtained from the hierarchical combination of sub proofs is called PLS (Proof-Like Structure). The definition of PLS takes advantage of the fact that proofs are in Normal Form at the time the relevant sub proofs are chosen. We show how to use PLS to build cognitively-motivated text plans and conclude that a considerable improvement has been obtained by comparison with previous frameworks for generating texts from N.D. proofs.

Keywords: Natural Language Interfaces, Automated Reasoning

Introduction

Logic-based reasoning systems present considerable challenges to interface designers. The advantages offered by a logic system as far as inferential consistency and truth maintenance are concerned bring about a number of obstacles for the generation of explanations. There are two especially critical obstacles to be overcome. One is related to the complexity of converting proof trees into text plans, a task which is often much more easily performed if knowledge representation (KR) is achieved via semantic networks, frames or other structuring techniques. The other is related to the size of the explanatory text, which can extend over dozens of paragraphs.

In existing text generation strategies (cf. [de Souza & Nunes 92]), explanatory text plans are recursively generated from an entire proof tree. Before being transformed into Rhetorical Structure Theory (RST) schemes [Mann & Thompson'87], the trees produced by a ND theorem prover go through two kinds of operations: Factorization and Derivation. These are meant to prune the trees and subtract information not relevant to the point being proved. The resulting proof is recursively transformed into a RST schema whose depth is determined solely by the size of the proof. As a result, explanations can be very long and difficult to understand despite their presentation in natural language (NL).

This paper introduces a set of transformations upon Natural Deduction (ND) proof trees which aim at improving the quality of textual explanations provided in integrated interface system for knowledge-based systems. Transformations tackle both the complexity and the size of texts which are to be generated generated by grammar rules extracted from Brazilian Portuguese, applied to text plans transformationally derived from large proof trees.

The effect of the new set of transformations presented here is (a) generality of the method; (b) recoverability of all the information from the proof; (c) much shorter explanations and (d) improved dialogue flow. The proposed transformations provide explanatory strategies more in line with what has been coined a "reactive approach" [Moore & Swartout'91] and has been shown to produce more agile conversation between user and system [Cawsey 92]. The process amounts to a recursive structuring of ND proof trees, where an original proof is split into a number of proof "pieces". Each piece is separately turned into an independent proof with an associated theorem, which becomes an anchor for recursively recovering the original information. Finally, a Proof-Like Structure (PLS) is achieved by means of substituting the proof pieces for a few rules, according to criteria we will specify in the following.

We provide a contrastive example of the kinds of explanations provided by both underlying models ([Nunes 91] and ours), and allow the reader to sense the effects of our current proposal. A point not to be disregarded in evaluating the type of text to be generated and interpreted is that, unlike some NL interface approaches which try to maximize resemblance of the system's performance to human performance, we deliberately seek to expose automatic reasoning idiosyncrasies to users. It is our belief that users should not be led into thinking that there are no differences between his/her way of reasoning and that of the system's. On the contrary, the more evidence he/she gets about the distinctive way of reasoning the system has, the more productive is the interaction [de Souza 93]. Thus, NL is used to make users aware of the existence of an underlying model of a domain and of its systematic manipulation by an inferencing machine. These interfaces are not actually (as none other is) "natural language" interfaces, but rather "artificial language" interfaces which attempt to incorporate *only* those NL features that are judged crucial for users to understand formal models of reality.

The following sections include a brief summary of the ND system, a description of the transformation method (in Section 3), a contrastive example between [Nunes 91] and our approach, and some concluding remarks about the possible impacts of our approach in terms of the systematicity of the method, computer resources and interface text quality.

Natural Deduction Proofs

Natural Deduction (ND) proofs can be viewed as structural descriptions of an argument. They are recursive and can be easily decomposed in sub proofs or composed to form new proofs. Their structure shows clearly the relations between the various facts lined up to support the conclusion and how they lead to it. The ND proof as a whole has a remarkable resemblance to the structure of human mathematical argumentation, although it lacks some kinds of human reasoning steps. This supports its choice as the basis for the generation of NL explanations. Here we present some of the terminology relating to ND proofs (cf. [Haeusler 90]) that is used in this article, as well as a few additional definitions.

The Natural Deduction proof system (cf. [Prawitz 65]) uses many rules of inference to derive formulas from other ones. The basic ND rule system consists of a set of *introduction* rules, one for each logical connective, used for building formulas from smaller ones, a related set of *elimination* rules for the same connectives, that extract formulas from larger ones, and a rule for *Classical Absurdity* (symbolized by $\perp$). The logical connectives are conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$), the universal quantifier ($\forall$) and the existential quantifier ($\exists$). Negation is represented as an implication, in the form $A\rightarrow\perp$, instead of $\neg A$. Additional connectives, such as $\leftrightarrow$ and the exclusive or (X), can be defined over these basic ones, and rules can be added to deal with them. However, they don't change significantly our results and therefore we won't deal with them here.

Every ND rule has one or more *premises* and one *conclusion*. Besides that, rules can *discharge* hypotheses that could have been used in the proof of one or more of their premises. In Figure 1 we show a list of all the basic ND rules for the First-Order Logic (FOL), in the form of schemes. Capital letters indicate formulas, "x" indicates a variable, "t" a term and "a" a parameter, which differs from all other variables and constants used in the proof. When a rule discharges a hypothesis, the schema shows it between brackets at the right side of the rule line and above the premise whose proof could have made use of it. Every rule is named after its connective and a letter "I" or "E" for introduction or elimination rules, respectively.

Figure 1:*Natural Deduction rules for First-Order Logic*

Introduction

$$\wedge\text{I: } \frac{A \quad B}{A \wedge B} \qquad \rightarrow\text{I: } \begin{array}{c}[A]\\ \Downarrow \\ \dfrac{B}{A \rightarrow B}[A]\end{array} \qquad \forall\text{I: } \frac{A(a)}{\forall x A(x)} \ (1)$$

$$\vee\text{I: } \frac{A}{A \vee B} \quad \frac{B}{A \vee B} \qquad \exists\text{I: } \frac{A(t)}{\exists x A(x)} \qquad \perp \text{(Classical Absurdity): } \begin{array}{c}[\neg A]\\ \Downarrow \\ \dfrac{\perp}{A}[\neg A]\end{array}$$

Elimination

$$\wedge\text{E: } \frac{A \wedge B}{A} \quad \frac{A \wedge B}{B} \qquad \rightarrow\text{E: } \frac{A \quad A \rightarrow B}{B} \qquad \forall\text{E: } \frac{\forall x A(x)}{A(t)}$$

$$\vee\text{E: } \frac{A \vee B \quad \begin{array}{c}[A]\\ \Downarrow \\ C\end{array} \quad \begin{array}{c}[B]\\ \Downarrow \\ C\end{array}}{C}[A][B] \qquad \exists\text{E: } \frac{\exists x A(x) \quad \begin{array}{c}[A(a)]\\ \Downarrow \\ C\end{array}}{C}[A(a)] \ (2)$$

Notes:
(*1*) In the ∀I rule the parameter *a* which is substituted by the variable *x* in the conclusion must not occur in any undischarged hypothesis in the premise proof.
(*2*) In the ∃E rule the parameter *a* that occurs in the discharged hypothesis must not occur in any other undischarged hypothesis in the minor premise (C) proof, nor in the premise C itself.
In both cases A(*a*) is the formula resulting from the substitution of *a* for *x* in A(*x*).

All introduction rule premises have the same status. The elimination rule premises, however, may have distinct roles in the inference. We call a *Major Premise* the premise in an elimination rule that contains the connective that is eliminated. In all elimination rules with a single premise, this is the major premise. In the ∨E rule the major premise is A∨B, and in the ∃E rule it is the formula ∃xA(x). Every premise of an elimination rule that is not a major one is called a *Minor Premise*. There are two kinds of minor premises, depending on their relation with the conclusion: *Unbounded Minor Premises* (UmP), which are identical to the conclusion, and *Bounded Minor Premises* (BmP), which correspond to a discharged hypothesis in the introduction rule (this is the case of the premise A of the →E rule).

A *deduction* is a tree-like structure with: (a) one or more *top-formulas* (which are not conclusions of rules); (b) rules with top-formulas and other rules' conclusions as premises; and (c) a single conclusion, its *root*, that is not premise to any rule. The rule application conditions must be satisfied. The top-formulas can be of the following types: *axioms* of the knowledge base, *theorems*

We also make some assumptions about the knowledge base. All the axioms in the knowledge base must be closed formulas, that is, with no free variable. They are supposed to be separated in two sets: one of *general axioms*, or simply *axioms*, and other of *particular axioms*, also called *facts*, and the latter normally refer to individuals, whereas the former refer to classes. The facts contain constants that can be generalized through substitution by a quantified variable resulting in a more general (although not necessarily true) statement.

Recursive Structuring of ND Proof Trees

Proof trees may be organized into more or less independent "proof pieces". Some of these could be omitted in a NL textual translation of the proof, but how can we do it? How can we break a proof into parts and then reorganize it into a new, recursive structure? In order to build a Proof-Like Structure (PLS) that reflects an "argumentative structure" we will follow a sequence of steps.

Identifying proof pieces

First of all, we must identify the proof pieces to be reorganized. Since we are talking about normal proofs in the Natural Deduction system, there are a number of formulas that present themselves as natural breaking points, because they serve as links between elimination and introduction rule chains. They are the minimal formulas, which we consider as our candidate breaking points. By doing so, we guarantee that the breaking points are exactly at the end of elimination rule chains, being the smallest formulas in the branch. There are however two exceptions to the rule: first, when the minimal formula is the absurdity ($\perp$), we take as breaking point the following formula in the branch; second, when it is a hypothesis discharged in a $\vee$E or $\exists$E rule, we take as breaking point the major premise of the rule. This can be justified by observing that the absurdity doesn't make sense by itself, but only connected to the formula that is derived from it, and in the case of the hypothesis, we are more interested in reflecting the two-dimensional structure of the proof, where the link is made between the major premise of those rules and their conclusion. All these candidate formulas can be thought of as partial conclusions, which summarize the content of the proof, viewed as an argumentative structure.

Definition: A *breaking point* in a Natural Deduction Proof is either a minimal formula or the major premise (which has the logical connective to be eliminated) of an $\vee$E or $\exists$E rule. The *proof piece* associated with the breaking point is the part of the proof above the breaking point. Within a proof piece, all hypotheses that remain undischarged at its breaking point are called *global hypotheses*, and all hypotheses that are discharged within it are called *local hypotheses.*

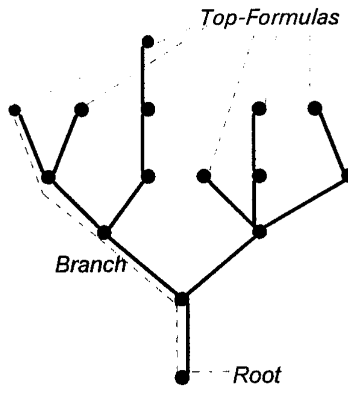
With these breaking points, our proof pieces consist of elimination rule chains together with introduction rule chains ending in minor premises of $\rightarrow$E rules. Each proof piece is a proof in itself, except for some occasional undischarged hypotheses. There is also an one-to-one mapping between proof pieces and breaking points, and an inclusion relation between the proof pieces, given by the order in which the breaking points appear in the proof. The largest proof pieces different from the proof itself are determined by the minimal formulas of the main introduction rule chains (which end in the proof conclusion itself).

Undoing the links

After breaking the proof, the next step is to make each proof piece a complete proof, and its root a theorem, so that we can manipulate it independently. In order to achieve this, we must do two things: we undo the unification steps made in the breaking point to link it with the rest of the proof deduced from the knowledge base and *hypotheses* which may or may not be discharged by some rule downward in the deduction. If the undischarged hypotheses are $A_1,...,A_n$ and the root is F, we say that the deduction is a *deduction of* F *from* $A_1,...A_n$.

A *proof* of a formula F in a given knowledge base is a deduction of F with no undischarged hypotheses. A proof or deduction has a form similar to the figure below (cf. Fig 2), in which formulas are represented by vertices, and edges link each premise of a rule to its conclusion.

Figure 2: *ND proof schema:*



A *branch* of a proof or deduction is a sequence of formulas $F_1,...F_n$ in which: (a) F_1 is a top-formula; (b) each F_i is either an introduction rule premise, a UmP of an elimination rule (when $F_i=F_{i+1}$) or a major premise when the rule has no UmP, with F_{i+1} the rule conclusion; and (c) F_n is either the root formula or a BmP of an elimination rule (→E). A major premise of a rule that has UmP (either a ∨E or a ∃E) is followed not by the rule conclusion, but by any one of the top-formulas of the deduction of one of its UmP, discharged at that given rule. Branches normally follow from premise to conclusion, "jumping" to a top-formula only at these points. Note that a formula may be part of many branches. The set of branches that end at a particular BmP or at the root form a part of the proof, a sub proof. Branches that end at the root formula are called *main branches*, composing the main part of the proof, that is, leading directly to the conclusion. The other, *secondary branches*, are part of the secondary proofs that lead to intermediary results.

A very important definition is the *Normal Proof.* A proof or deduction is said to be in *normal form* if and only if all its branches $F_1,...F_n$ can be divided in two parts, called *E-Part* and *I-Part*, surrounding a given formula A_k, so that: (a) the E-Part precedes the I-Part; (b) all formulas A_i of the E-Part except the last are major premises or UmP of elimination rules; and (c)all formulas A_j of the I-Part, except the last, are premises of introduction rules or UmP. A_k is called a *minimal formula*, because it is the smallest formula in the branch. Note that UmPs can occur anywhere in the branch, because they are identical to their conclusion. Except from these, all elimination rule applications precede the introduction rule ones.

Normal proofs are important because they guarantee that there is no unnecessary rule application in the proof, and so they can't be reduced. The FOL is a logic that has the *normal form property*, that is, for any theorem in FOL there is always a normal proof. The theorem prover may then restrict the search only to normal proofs. Normal proofs are also important because it is easy to analyze their structure. We can generalize the concept of I-Parts and E-Parts for parts of the whole proof, made by all the I-Parts (or E-Parts) of the branches with a given conclusion (or minimal formula).

"underneath" and discharge the not-yet-discharged hypotheses. The unification steps should be undone to make the proof piece as independent from the rest of the proof as possible, with as general a conclusion as can be made from the piece in hand. That is, the unification associated with the proof piece has to be the most general unification, determined solely by the substitutions needed to make the proof piece a deduction.

The first step amounts to replacing all terms in undischarged (global) hypotheses by new free variables. The axioms of the knowledge base cannot have free variables, and thus they remain unchanged. Any local hypothesis must also have its own terms replaced by new free variables, to guarantee a most general unification. The second step amounts to rebuilding the deduction from all the top-formulas by the application of the same rules in the same order. All rules apply, because their conditions are satisfied. The only rules that could cause problems are the $\forall$I and $\exists$E rules, since they have conditions upon terms. But their application is still valid, because the original parameter associated to one such rule has been substituted by a new free variable in every top-formula where it occurred. So through each branch, they must remain free variables and are unified, at any rule, only with other free variables originating at $\forall$E rules or at top-formulas where they substituted the same parameter. Otherwise the original proof wouldn't have been valid. The resulting deduction has a free variable (and the same one, after unification) in all places where the parameter was originally, and this variable occurs only in these places. We may then even restore the original parameter instead of it.

The resulting deduction is a valid ND deduction similar to the original one. Its root formula (partial conclusion) is also unifiable with the original. After this step, we have what we call an *independent deduction*, and the composition of the independent deductions with the main introduction rule chains (linked directly with the proof root), with the necessary substitutions, makes up the original proof.

Definition: The deduction resulting from the substitution of free variables for the terms in the hypotheses of a proof piece and the subsequent reconstruction of the deduction by the application of the same rules in the same order is called an *independent deduction.*

Note that all the terms carried from the rest of the proof into the proof piece have thus been eliminated from it, through the modification of all the hypotheses and the reconstruction of the conclusion, as we state in the next result:

Lemma: An independent deduction is a valid ND deduction. The unification associated with it is the most general that can be achieved, given the restriction imposed by the included axioms.

Building the Independent Proofs

The independent deductions, as long as they have undischarged hypotheses, are not yet full ND proofs. To make them so, we need to discharge the hypotheses, generating a new conclusion that is a theorem of the base. It is also desirable to close any free variable that may still remain in it after the unification. We can do so by a series of $\rightarrow$I rules, one for each undischarged hypothesis, followed by a series of $\forall$I rules, for each free variable in the resulting formula. These $\forall$I rules can be applied because there is no remaining undischarged hypothesis and no axiom contains a free variable. However, to simplify the resulting theorem, it is desirable to discharge all the pending hypotheses at once, through a single $\rightarrow$I rule and a conjunction of the hypotheses. The result is a theorem of the following form, where x represents the free variables, and F the conclusion of the independent deduction:

$$\forall x\ H_1(x) \wedge \ldots \wedge H_n(x) \rightarrow F(x)$$

Theorem: The deduction resulting from the application of rules discharging the pending hypotheses in an independent deduction and closing the free variables of its conclusion is a valid ND proof. We call it the *independent proof* associated with the independent deduction, and its conclusion the *associated theorem* of the independent deduction.

The proof is simple. The independent deduction built from the axioms and the modified hypotheses is, by construction, a correct ND deduction, as stated in the previous lemma. Since a proof is a deduction with no undischarged hypotheses, after the application of the →I rule we effectively arrive at a valid proof. The ∀I rules serve to build a theorem with no free variable whatsoever.

Further Generalization of the Theorems

We now observe that, although we made the independent deduction as general as it could be without modifying the original proof piece structure and axioms, it is not yet as general as we would want. We identify two kinds of axioms in the knowledge base: the general ones, which normally don't make reference to individuals except for attributing properties that are general to a class or for defining some general attribute that is represented as a constant, and factual assertions, which make reference to individuals of some of the main types of the base, and assign individual properties to them. Ideally, axioms of the second kind would be atomical, or at least be made of a conjunction, disjunction or negation of atomic formulas. We would like to generalize on these facts as well, to build a theorem that could refer to a whole class of individuals, instead of a single one. This has apparent advantages in explanations, because it is more informative to make a general assertion than an individualized one. It is also of logical interest, because it enables the system to extract intermediate results in the form of theorems that can be stored together with the knowledge base and reused.

To further generalize an independent deduction, we can choose some of its used axioms to serve as local hypotheses. The formulas should be chosen according to certain parameters. Specific facts about individuals, ideally identified at the time of the knowledge base construction, should be included whenever possible. The chosen formulas should all be "simple", which normally means that they do not include quantifiers and that they be of small size (small number of atomic components, preferably only one or two). Pragmatic considerations could also play a role in defining the set of chosen axioms, for example considering characteristics of the knowledge base, the size of the deduction, the number of candidates, the number and type of top-formulas with a given term etc. Having chosen a group of axioms as "generalizable" we will treat them as other undischarged hypotheses, substituting all their terms for free variables, in the process of building the independent deduction. They will, in the end, be discharged, and the proof conclusion will include them as further conditions in the theorem. The resulting theorem may then quantify over variables that would have been individualized otherwise.

Not all generalizations may be useful

Note that both here and in the original definition of independent deduction, it is not necessary and sometimes not desirable to replace all terms occurring in hypotheses for free variables. Substitution would be desirable when these terms refer to individuals of classes that are the "subject" of the proof piece. But there may be situations when the substitution of a term for a free variable may hamper the comprehension of a part of the proof. For example, when a term, which refers to some individual characteristic in a binary or n-ary predicate, but doesn't appear in the conclusion, is replaced by a free variable, a whole section of the proof may end up by containing a free variable that says nothing about the individual characteristic that is relevant in the context. So an important local contextual information is lost, with possibly no advantage in generalization, and this is certainly not cooperative (cf. Grice 75]).

It may be sometimes difficult to identify syntactically which terms shouldn't be changed in a deduction, in the absence of more information from the knowledge base. However, a strong clue can be given by observing terms that appear both in the original conclusion and in the chosen hypotheses, which indicates that the deduction in some way tells something relevant about that individual. If additional knowledge about a given domain could be collected, then other pragmatical rules might be defined for it, to choose which classes should be generalized according to context. Otherwise, it seems better to make a limited generalization that is assured to be relevant than make the most general one, that may leave parts of the proof too generic to be understandable. We therefore restrict here the substitution of terms for free variables to the ones that appear both in the conclusion of the deduction and in its hypotheses, when dealing with a generic domain.

Building Substitute Deductions

Now for each proof piece we have an associated theorem and its independent proof. But the original proof would be left incomplete if we discarded these proof pieces. So we're going to build a new deduction that could replace the proof piece in the original proof.

Taking any proof piece, its associated theorem contains a generalization of its conclusion. It has the following form, where A(x) represent the formulas that were originally axioms of the second class, and H(x) the original undischarged hypotheses:

$$\forall x A(x) \wedge H(x) \rightarrow F(x)$$

We may then use this theorem as a top-formula to deduce back the original breaking point formula F(t). Using the theorem prover to deduce it from the associated theorem, we will get the following deduction:

Figure 3: *Substitute Deduction:*

$$\dfrac{\dfrac{A(t)\quad [H(t)]}{A(t)\wedge H(t)} \qquad \dfrac{\forall x A(x)\wedge H(x)\rightarrow F(x)}{A(t)\wedge H(t)\rightarrow F(t)}}{F(t)}$$

Note that the substitution t/x restores all the original terms, by unifying the conclusion with the breaking point formula F(t). Note further that by the construction of the associated theorem itself, the formula(s) A(t) can be unified with axioms of the second kind, that is, the original axioms they represent, and the H(t) can be unified with some hypotheses that are valid in the breaking point, that is, that are discharged afterwards in the proof. The result is a substitution that undoes the generalization done in the process of generating the associated theorem and its independent proof.

This small deduction can replace the original proof piece while maintaining the correctness of the proof. In this way we can reduce significantly the length of the proof and its explanation, while being able to show some interesting intermediary results (the associated theorems). The result is a smaller proof, built to satisfy cooperative goals.

The Recursive PLS

We can now build a smaller proof with many other associated proofs for the intermediary results. All these can be organized in what we call a Proof-Like Structure. This consists of a proof that accepts theorems as some of its top-formulas, as long as proofs for these theorems are available. For each theorem, there is a "link" to its proof, and this can be similarly organized. This results in a recursive structure which may be used to structure the explanations of the main conclusion.

Having a PLS for a given question of the user, its "root" proof can be used to generate an explanation. The deduction that had replaced the original proof pieces, besides being already small, can be "explained", or better, enunciated, in a quite direct and convenient way, for example, stating the theorem and telling the axioms in order, then the hypotheses and at last the conclusion. Something like:

We know that every singing bird flies, PiuPiu is a bird and we are supposing that it sings, so we can conclude that it fliesr.

The theorem can be marked, as an anchor, so that if the user is not satisfied with the short explanation, he can further probe the theorem. Then the theorem's proof, which can itself be structured in PLS proof, would be presented in explanation form, and so on. This results in an hypertext-like structured explanation, that allows the user to focus his/her attention on one spot at a time, instead of having to look for the important information in the midst of a long complex proof.

Choosing the Breaking Points

In order to build a PLS, we must first choose some of the proof breaking points and use them to build the associated independent proofs. Every subset of the proof breaking points can serve as an initial set, with widely varying results. We must choose an adequate set of breaking points, following a number of pragmatic criteria. The criteria can depend on the knowledge domain and its representation, but nonetheless we can give some general guidelines, based on syntactical data, that should be verified while choosing the breaking points.

The set of breaking points should correspond to the specific set of information we want to convey in the explanation, with every minimal formula corresponding to a unit of information. So there are a number of criteria which refer to each individual choice, and others that refer to the whole set.

Each breaking point should be weighted according to its size. Too many variables, or logical connectives, or a large complexity, would normally not be desirable in a breaking point, especially in the first level of explanation. A small breaking point is preferable. Similarly, a breaking point with too many global hypotheses in its proof piece would result in a large theorem, so the number and complexity of the hypotheses should also be weighted.

The terms in the conclusion and the hypotheses of the proof piece should also be considered. There should be a match between them, that is, there should not be a term present in the conclusion which is not present in any chosen hypotheses. It would be very strange in an explanation to present an individual which "comes from nowhere". This can be resolved by picking another fact from the top-formulas to be made a new hypothesis and so appear in the theorem. If this is not possible or not desirable, then the breaking point should not be chosen. However, domain information can be used to identify some terms in some predicates to which this would not be a problem.

On the other hand, the largest the proof piece the better the breaking point, generally speaking. The chosen breaking points should have a large or complex proof piece, in order to simplify the resulting explanation. But some of the lowest minimal formulas, which have the largest proof pieces, sometimes may not be chosen, or else the explanation would result too small to be helpful. This is true if there is only one such minimal formula. So we must seek a balance between too many and too few information contained in the PLS first level.

An important guideline is to watch the terms that appear in the proof root (corresponding to the question being considered). If a breaking point contains some of these, and no other term (from the most important domains, at least), then it could be informative to present it, even if it is distant from the proof root. The same can be said of the terms already appearing in other chosen breaking points or their theorems. On the other way, if a breaking point contains terms not appearing elsewhere, and none or too few of the ones that appear, it may represent a side result with little importance to the conclusion. Domain information can also be used here, to relate the predicates of the conclusion and of the breaking points.

Building the PLS

With a set of chosen breaking points, we can proceed to build the PLS. As the independent proof is built from the whole proof piece above a given breaking point, and this proof piece can have another breaking point inside, the process result is dependent on the order. We start from the breaking points whose proof pieces do not contain any of the chosen breaking points. That is, we progress downwards from the top of the proof.

From a given proof piece, there may be two possibilities. If this proof piece is not part of any other chosen proof piece, we need only to obtain the associated theorem with its independent proof. Then we replace the proof piece with the deduction as described in the previous paragraph, linking the theorem with its proof.

However, the proof piece can be part of another chosen proof piece, and when this is collapsed the former theorem would disappear from the PLS. In this case, we would like to simply "cut" the breaking point, removing it from its place to link it directly with the PLS. The breaking point formula F would be left as a hypothesis on the larger proof piece, and when this on its turn is replaced by the small deduction from its associated theorem, F would reappear as a premise, possibly with some undone substitutions. Then its corresponding deduction from the former theorem would be added, as a proof of the premise F used by the larger proof piece.

But it is not so simple, since the proof piece can depend on hypotheses discharged on the larger one. Instead, we identify these hypotheses, local to the larger proof piece, and add them as premises for F, in the form L→F. The deduction of F from this formula is direct and can replace the original top-formula F (where L are valid local hypotheses), while L→F can be obtained from the first proof piece simply by discharging these hypotheses local to the larger piece. The theorem and its independent proof are then build considering L→F instead of F, and it can be used in the context of the larger proof piece substitution. The following figure shows the resulting substitution, where C is the conclusion of the larger proof piece, A_1 and H_1 are respectively the facts (made into hypotheses) and global hypotheses of the first proof piece, A_2 and H_2 are the same relating to the larger proof piece, and T is the conclusion of the proof. (In the original proof figure, we show only the branch containing L).

Figure 2:

$$\begin{array}{c}[L]\\ \vdots\\ F\\ \vdots\\ C\\ \vdots\\ T\end{array} \Rightarrow \begin{array}{c} \dfrac{\dfrac{\dfrac{\dfrac{A_1 \quad H_1}{A_1 \wedge H_1} \quad \dfrac{\forall x(A_1 \wedge H_1) \to (L \to F)}{(A_1 \wedge H_1) \to (L \to F)}}{L \to F} \quad A_2 \quad [H_2]}{(L \to F) \wedge A_2 \wedge H_2} \quad \dfrac{\forall x((L \to F) \wedge A_2 \wedge H_2) \to C}{((L \to F) \wedge A_2 \wedge H_2) \to C}}{C} \\ \vdots \\ T \end{array}$$

Once we have built the PLS by substituting all proof pieces for a corresponding small deduction from a theorem, the associated independent proofs remain linked to the PLS, and can themselves be structured in PLS form. However, we consider that their transformation is made only at the request of the user for an explanation of the corresponding theorem, which allows the consideration of other criteria concerning the user and the previous dialogue in the choice of the new set of breaking points.

The resulting PLS can then be expanded in a recursive form, allowing a flexible dialogue between the user and the system be construed for the explanation of a given complex theorem.

Examples of proof-derived text plans:[1]

1. Mixed Explanation from Reduced Proof ([Nunes 91, pg. 189]):[2]

The following is an example taken from [Nunes 91], generated from a RST derived from a ND proof tree after a series of transformations. This is the smallest possible text generated following Nunes's approach from the original proof tree. The text that differs from the one resulting from our approach is marked in bold. Note the contrapositive rule in the first marked text.

"O homem é caçador ou não preserva a natureza. Se ele é caçador, então ele é predador, pois, **se de um lado ele é caçador, de outro ele não o seria, caso não fosse predador**. Se ele não preserva a natureza, então ele é ignorante e predador, uma vez que todos os que não preservam a natureza são ignorantes e predadores. Logo, o homem é predador.
"Além disso, o homem ameaça o jacaré do Pantanal porque **ele tem valor** e tudo o que tem valor é ameaçado pelo homem.
"Assim, existe um predador que ameaça o jacaré pantaneiro."

2. Explanation resulting from the corresponding PLS, following the same textual realization:[3]

The following is the text that would result from the PLS derived from the original proof tree. The anchors representing the links to the recursive explanations are underlined.

"O homem é caçador ou não preserva a natureza. Se ele é caçador, então ele é predador, pois <u>todo caçador é predador</u>. Se ele não preserva a natureza, então ele é ignorante e predador, uma vez que todos os que não preservam a natureza são ignorantes e predadores. Logo, o homem é predador.
Além disso, o homem ameaça o jacaré do Pantanal porque <u>todo jacaré do Pantanal tem valor</u> e tudo o que tem valor é ameaçado pelo homem.
Assim, existe um predador que ameaça o jacaré pantaneiro."

[1] The texts are given in Brazilian Portuguese, because the text planning and realization is directed towards this language. Translations are given to English on the footnotes.
[2] Man is a hunter or he doesn't preserve the Nature. If he is a hunter, then he is a preyer, because, **if on one hand he is a hunter, on the other he wouldn't be, if he wasn't a preyer**. If he doesn't preserve the Nature, then he is ignorant and preyer, since everyone that doesn't preserve the Nature are ignorant and preyers. Then, man is a preyer.
Besides that, man threatens Pantanal alligator, because it is valuable and everything that is valuable is threatened by man.
Therefore, there is a preyer who threatens Pantanal alligator.
[3] Man is a hunter or he doesn't preserve the Nature. If he is a hunter, then he is a preyer, because <u>every hunter is a preyer</u>. If he doesn't preserve the Nature, then he is ignorant and preyer, since everyone that doesn't preserve the Nature are ignorant and preyers. Then, man is a preyer.
Besides that, man threatens Pantanal alligator, because <u>every Pantanal alligator is valuable</u>, and everything that is valuable is threatened by man.
Therefore, there is a preyer who threatens Pantanal alligator.

If the user probed the anchor, he would be given a subsequent explanation for the fact that Pantanal alligators are valuable. The same would be the case with the fact that every hunter is a predator. In both cases the deduction uses a contradiction, which in turn is hard to explain in NL.

Concluding Remarks

Whereas Nunes's pruning strategies [Nunes 91] scan the entire proof structure for varying proof patterns to be later transformed into derived rules, the approach presented above scans the proof for minimal formulas and transforms their sub proofs into a single short pattern of deduction. Consequently, it organizes the entire proof into a uniform structure which contributes to simplify explanations.

Moreover, by adopting a point of view in which users should have an intuition about how the system works, our approach reflects the hierarchy of the ND proof without compromising global cognitive goals. Although the text plan does not support the generation of a totally "natural" text, the output shows a coherent and cohesive set of inferences from knowledge pieces.

A comparison between texts that could be generated by Nunes's approach and ours, in the preceding section, shows that the quality of the explanation is improved in important ways. Firstly, Nunes's approach eliminates an entire section of the proof (saying why the Pantanal alligator is valuable), which originally contained important information concerning the individual. Her transformed proof tree does not contain the formulas that assert that the individual is a Pantanal alligator (Nunes 91, p. 189). This connection is supposedly recovered by the text generator from other parts of the proof. In our approach the information is readily available in the proof tree.

Secondly, we can notice that in our approach the text span that realizes the absurdity contained in the contrapositive is elegantly replaced by a direct inference (all hunters are preyers). This has an important cognitive effect on the quality of the text.

Thirdly, the information eliminated in Nunes's approach cannot be recovered for further probing, whereas ours is totally recoverable from the links between theorems and their proofs. Furthermore, this information has been generalized in our recursive explanation in order to allow for the correct inference to be made (all Pantanal alligators are valuable, and not just an individual alligator).

However, the most crucial difference between both approaches is that whereas Nunes's results have been considered only in small proofs of particular domains ([Nunes 91], p. 164), and may not reduce significantly the amount of explanation to be provided in a large proof, ours is general and constrains neither the size of the original proof, nor the quality and quantity of the reduction.

In computational terms, we also believe that our approach presents advantages relative to search space. Because we are searching for minimal formulas alone, very simple proof patterns, we spend much less time to build PLS's than if a much larger variety and complexity of patterns were searched for.

Future work should include research about the application of transformations upon the entire proof in order to simplify further the explanation of such difficult proof patterns as the Classical and the Intuitionistic Absurdity. There could be either a decrease in the number of applications of such patterns (by moving them to one specific point in the proof, like the last inference rule only), or a movement upwards in the proof structure and their application to the smallest number of axioms possible.

References

[Cawsey 92] Cawsey, A. *Explanation and Interaction. The Computer Generation of Explanatory Dialogues.*
MIT Press, Cambridge, Massachusetts, 1992.

[de Souza & Nunes 92] de Souza, C. S. & Nunes, M. G. V. *Explanatory Text Plannung in Logic Based Systems.*
Proceedings of the 15th International Conference on Computational Linguistics, COLING-92, 742-755, 1992.

[de Souza 93] de Souza, C.S. *The Semiotic Engineering of User Interface Languages.*
International Journal on Man-Machine Studies (1993) 39, 753-773.

[Grice 75] Grice, H. P. *Logic and Conversation.*
in P. Cole and J. Morgan (eds) Syntax & Semantics: Speech Acts, vol.3,
Academic Press, NY, 1975

[Haeusler 90] Haeusler, E. H. *Prova Automática de Teoremas em Dedução Natural - Uma Abordagem Abstrata.*
PhD. Thesis. Depto. de Informática, PUC/RJ, 1990

[Mann & Thompson 87] Mann, W. & Thompson, S. *Rhetorical Structure Theory: A Theory of Text Organization.*
In Polyani, L. (ed.) The Structure of Discourse. Ablex. 1987.

[Moore & Swartout 91] Moore & Swartout, W.R. *A Reactive Approach to Explanation: Taking the User's Feedback into Count.*
In Paris, C. L., Swartout, W. R. & Mann, W.C. (eds) Natural Language Generation in Artificial Intelligence and Computational Linguistics. Kluwer Academic Publishers, Massachusetts, 1991.

[Nunes 91] Nunes, M. G. V. *A Geraçao de Respostas Cooperativas em Sistemas Baseados em Lógica.*
PhD. Thesis. Depto. de Informática, PUC/RJ, 1991.

[Prawitz 65] Prawitz, D. *Natural Deduction - A Proof-Theoretical Study.*
Stockholm, 1965