Structured Argument Generation in a Logic-Based KB-System

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

Abstract

This paper presents a method for the transformation of Natural Deduction proofs into hypertext-like structured explanations to be provided by knowledge-based systems. First-Order Logic and a variant of the Natural Deduction system are taken as a basis for representing knowledge and reasoning over a generic classification domain. Information selection and restructuring are used to convert proofs into argumentation schemata. Our goal is to produce structured arguments that can be converted into Natural Language explanatory hypertexts which satisfy some pragmatical constraints. Thus, pragmatically motivated rules are incorporated to the semantic component of a friendly user interface language in which explanations are requested and conveyed.

1 Introduction

Today's booming information technology brings about a renewed interest in the construction of information systems which not only store and retrieve information but also deal with them in intelligent (or rational) ways. Knowledge-based systems (kb-systems) are being used in a great variety of fields to enhance the processing of information. The impact of their role in this scenario is conditioned by their ability to present and justify their reasoning to the human user. As a result, an increasing research effort is being placed in the development of intelligent multimodal interfaces [Maybury'93] and explanatory text generation components [Cawsey'92, Moore'95].

In an effort to improve the legibility of artificial reasoning patterns, we have explored the idea that a logic proof can be used as an argument structure [Huang'94]. Elaborations on proofs should produce human-readable arguments [Toulmin'64] that are adequate for a set of domains. We have chosen First Order Logic (FOL) as a flexible and powerful representation language to account for generic classification domains. Logic representation and deductive reasoning, having been derived from actual human argumentation practice, present some attractive features to support the automatic generation of quality explanatory texts. By the same token, Natural Deduction (ND), derived from human mathematical proof practice, produces bidimensionally-structured tree-like proofs that are arguably easier to understand than other methods such as resolution. ND tends to produce direct proofs, notwithstanding the maintenance of full structural integrity of the original sentences. It is also important to note that, although we have chosen FOL to begin with, any other logic that can be adapted to ND reasoning would be equally acceptable as long as it had a set of ND rules that users would be able to take as "natural."

In conversational computer applications such as KB-systems, a number of pragmatic rules should be observed for interaction to proceed productively. They have been first organized by Grice into what he has called the "Co-operative Principle" [Grice'75]. His four maxims that characterize the dimensions of logic in conversation have been summarized by McCawley as: "you should assert neither more nor less than is appropriate for the purpose at hand" (Quantity); "what you assert should be true, and you should have adequate grounds for holding it to be true" (Quality); "what you mention should be relevant for the purpose at hand" (Relation); and "you should use linguistic means no more elaborate than what is needed to convey what you are asserting" (Manner) [McCawley'81, p.217].

Direct observation of logic proofs, whatever the method used to generate them, gives clear evidence for much needed transformations on them so that they can be used to provide support in human-computer explanatory dialogues. Grice's maxim of Quality is rigorously accounted for in logic-based systems, but proofs contain lots of distracting details and apparently irrelevant detours compared to "natural reasoning". The maxims of Quantity and Relevance interact in complicated ways, especially when disjunctions are used throughout the proof. Converting a ND proof into an argument schema thus involves a considerable amount of focal and topical evaluations, along with pruning machinery that contributes to maintain the structural elements within highly cohesive boundaries. The requirements on Manner are self-evidently numerous in that the linguistic means to convey assertions must be something other than FOL for the vast majority of KB-systems' users.

Moreover, conventional implicatures entailed by lexical items alone, sentences, or text spans, represent a major obstacle to meeting the requirements of Quantity and Relevance. The examples provided in this paper (such as the explanation of why cats don't live in the ocean, in sections 4 and 6) refer us to common sense knowledge residing in the misty frontier between language and cognition. They illustrate that there are implicatures in the textual explanations which suggest that some kind of logic reasoning should be present in the linguistic interface of KB-systems, and not only in the knowledge bases. For example, the text:

Cats don't live in the ocean, because they have paws and walk.

is better than the automatically generated text:

Cats don't live in the ocean, because they have four paws. Anyone who has four paws walks, so cats walk. Since anyone who walks does not live in the ocean, then cats don't live in the ocean (either).

The last phrase is implied by the use of the two previous phrases. In this case, we feel that there are strong indications of a regularity which can be mapped in the text generator, but which cannot be accounted for at the level of the reasoning module, since without the last phrase the argument is incomplete.

Our research is aimed at designing a user interface language that is easy to use and understand, and applicable to KB-systems in generic classification domains. Its formal specification includes lexical [Dias'94], syntactic [Quental'95], semantic, and discourse [García'95] components, but no independent pragmatic component. Following patterns of typical computer language interpreter design, our processor additionally includes (a) a generation module, and (b) a discourse control module, to account for continuous interaction between users and system..

In this paper we explore aspects of the semantic component of such a user interface language, from a generation point of view. We show the method we have used to project some pragmatic principles of linguistic interactions onto the semantic structures manipulated by the reasoning machinery, in order to structure proof trees and select information to be conveyed to users. Most of the ideas presented here are implemented in a working prototype which is now in its final integration stage. LINX [de Souza et al.'96] is an interactive environment for logic-based KB-systems which allows users to converse with a theorem prover in natural language and/or by means of other typical widgets of graphical user interfaces. It includes all the components mentioned above, and many of the algorithms and pragmatical criteria presented in this article. The prototype is currently being used to test the various hypotheses involved in the research, using a small knowledge base. Textual interaction is based on Brazilian Portuguese. User input can be directly typed ahead or gradually built from menu-selection steps. Both forms are analysed by a grammar consistently derived from models of domain objects and relations. Users' questions trigger proof processes which produce argument structures. Such structures result from transformations over ND proofs, and are converted into recursive rhetorical schemata, further translated into hypertext-like Natural Language explanations.

The interactive model is designed so as to maintain language consistency throughout the whole interaction. It is also intended to clearly show to the user all of the system's limitations and regularities, both in terms of reasoning and communicative abilities. This should dispell the impression that the system is a replica of a human processor, and emphasize the notion that the theorem prover and its interface are not but software artifacts [de Souza'93].

In the following sections we will briefly examine the question of using pragmatical principles for the structuring of argumentative texts, make some considerations about the knowledge base and the logical proof, and then proceed to show our proof transformation methods. We finish with a detailed example of proof transformation and corresponding explanatory texts.

2 The Generation of Cooperative Explanations

Our objective is to convert a large and complex logical proof into an amenable, human-oriented, easy-to-understand, organized explanation. From considering Grice's Maxims, we arrive at four principles that we can use to design explanations:

1. *The exact way in which knowledge is coded and structured in the knowledge base is irrelevant for the user.*

2. *All information accidental to the proof process should be omitted from the explanation.*

3. *The user should be left to achieve deduction steps in a simple and direct inferential process, provided that he/she knows the premises, in their correct order, and the conclusion.*

4. *The amount of information contained in any explanation step should be limited to the amount that can be simultaneously visualized and processed by a human being without great effort.*

All the above principles translate into a set of transformations which convert a proof into a human-oriented argument structure. By masking the exact coding of the base whenever possible and adequate, by pruning extraneous references in the proof, by resuming deductive steps, and by restructuring them into smaller-sized proof "chunks", we aim to try and satisfy the general principles stated above, without additional domain-specific information.

3 Some Considerations about the Knowledge Base and Natural Deduction

We use the FOL with the Natural Deduction system [Prawitz 65] as the basis for the system's inference engine. Although it has the advantage that in answering questions the system generates justifications as a kind of by-product, which minimizes the problem of having to deal with two separate knowledge bases, we still have to provide it with a way of computing inferences in ND. In practice, it amounts to designing and implementing an automated theorem prover for it, built from a variant of Natural Deduction so defined as to be computable. Our automated theorem prover ([Oliveira'92], cf. also [Paz&Haeusler'94]) not only generates valid Natural Deduction proofs, but also generates them in Normal Form, which gives us a minimal formula at every proof branch, separating the elimination and introduction instances of rules in the branch. This characteristic is key to our treatment, as the Normal Form offers a simple pattern for all proofs. Any logic that can be expressed in ND in a compatible computable form could be used instead of FOL, although the impact of their new connectives and ND rules on the human understanding of system's reasonings should also be taken into account.

The quality of the knowledge base also has a direct impact on the quality of the generated proof. There are many ways of imposing structure onto logical representations of a given domain in order to improve the quality of the resulting proofs. However, we have chosen a single structure over the whole knowledge base, which is convenient when converting proofs into argument schamata and imposes no real restriction upon knowledge base engineering. We make a distinction between general axioms and particular facts in knowledge representation. General axioms are those logical formulas that, relative to the main categories of the domain, express general characteristics, attributes or relations. On the other hand, a particular fact is a formula that expresses a characteristic, attribute or relation of a particular individual instance of one of the main categories of the domain. In the absence of a previous distinction made during the knowledge engineering process, we may suppose that general axioms are all those formulas that have a quantifier as the main operator, whereas all the others are particular facts. This distinction is important when considering the premises over which we would like to generalize (see Section 6).

Moreover, we have also slightly modified the concept of "logical proof" in order to satisfy our general cooperative principles stated in the previous section. In order to hide the actual formulas in the knowledge base from explanatory texts provided to users, we extend the concept of valid proof to accept as proofs the deductions which have as undischarged top-formulas not only axioms or facts of the knowledge base, but also theorems deducible from it, as long as proofs for them are already available in the system. In this way, as we shall see, the first principle is already partially met. The proofs of those theorems can then be linked to their occurrences in the main proof, creating what we will call a Proof-Like Structure (PLS - see below). We have also extended some of the ND rules, namely accepting sets of discharged hypotheses instead of a single hypothesis whenever it is a conjunction, and accepting extended versions of the conjunction and disjunction rules to deal with conjunctions or disjunctions of more than two subformulas. We also allow deduction steps that are more complex than the basic ND introduction/elimination rules, as long as they remain logically valid (that is, that can be reconstructed as a sequence of basic ND rules).

Finally, we would like to introduce some definitions and notations. We use either $\wedge$-Intro or $\wedge$I as notation for the ND rule for the introduction of the conjunction, and similarly with all other logical connectives. A top-formula of a proof is a formula which is not a conclusion of any rule, but either an axiom, a hypothesis or a theorem (anchor for a secondary proof in a PLS). A branch of a ND proof tree is a sequence of formulas which begins with a top-formula and follows from rule premise to conclusion ending on either the proof root (the proved formula) or a secondary goal: either the minor premise of a $\rightarrow$-Elim (modus ponens) rule or the major premise (the one containing the eliminated connective) of a $\vee$-Elim or $\exists$-Elim rule.

With these considerations in mind, we proceed on to the transformation methods.

4 Eliminating Irrelevant Information

As the first principle does not have an immediate impact on the proof, let us consider the second one. In fact, in a generic local knowledge base, the axioms could well express many facts in a single formula, for example: Cats have tail and four paws, which simultaneously says two attributes of the same class cat. Let us suppose that this axiom would be used in a proof which however requires only the fact that a cat has four paws, as in the following example:

Working Example 1:

Axioms:

$\forall x\ \mathsf{cat}(x) \rightarrow \mathsf{has}(x, \mathsf{paws}) \wedge \mathsf{has}(x, \mathsf{paws}(4))$

$\forall x\ \mathsf{has}(x, \mathsf{paws}(4)) \rightarrow \mathsf{locomotion}(x, \mathsf{walks})$

$\forall x\ \mathsf{locomotion}(x, \mathsf{walks}) \vee \mathsf{locomotion}(x, \mathsf{flies}) \rightarrow \neg\mathsf{lives}(x, \mathsf{ocean})$

Question:

$\forall x\ \mathsf{cat}(x) \rightarrow \mathsf{lives}(x, \mathsf{ocean})$

Answer:

$No.\forall x\ \mathsf{cat}(x) \rightarrow \neg\mathsf{lives}(x, \mathsf{ocean})$

A possible explanatory text:

Cats don't live in the ocean, because they have tail and four paws. Anyone who has four paws walks so, cats walk. Since anyone who either walks or flies does not live in the ocean, then cats don't live in the ocean.

We notice that the fact that it has tail is discarded along the way, by an elimination rule. However, it is present in the proof before that point. If the proof is translated directly into text, this fact would be expressed in the explanation and would mislead the reader into supposing that the fact that cats have tails is somehow important to the conclusion, although the relation is nowhere clear. This happens because this information introduces a new theme in the explanation text, so humans suppose that it is somehow relevant to the conclusion. The logical relation in the proof, however, derives from the precise way in which information was coded in the knowledge base, and this is irrelevant to the human user. So, according to the second principle, that information should never be mentioned.

Examining the standard ND rules for FOL, we can verify that there are two rules that either eliminate or introduce information locally irrelevant to the deduction itself, and which are more related to the actual form of the logical sentences. These two rules are the elimination of the conjunction (as in the example above) and the introduction of the disjunction (as when concluding, from cats are felines, that cats are felines or herbivores). These two rules should be avoided in general. One way could be the modification of

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{[cat(a)] \quad \dfrac{\forall x\ cat(x) \rightarrow has(x,paws(4)) \wedge has(x,tail)}{cat(a) \rightarrow has(a,paws(4)) \wedge has(a,tail)}}{has(a,paws(4)) \wedge has(a,tail)}}{has(a,paws(4))} \quad \dfrac{\forall x\ has(x,paws(4) \rightarrow locom(x,walks)}{has(a,paws(4)) \rightarrow locom(a,walks)}}{locom(a,walks)}}{locom(a,walks) \vee locom(a,flies)} \quad \dfrac{\forall x\ locom(x,walks) \vee locom(x,flies) \rightarrow \neg lives(x,ocean)}{locom(a,walks) \vee locom(a,flies) \rightarrow \neg lives(a,ocean)}}{\neg lives(a,ocean)}}{cat(a) \rightarrow \neg lives(a,ocean)}[cat(a)]}{\forall x\, cat(x) \rightarrow \neg lives(x,ocean)}$$

Figure 1: A formal Proof

∞

$$\dfrac{\dfrac{\dfrac{\dfrac{[cat(a)] \quad \dfrac{{}^*\forall x\ cat(x) \rightarrow has(x,paws(4))}{cat(a) \rightarrow has(a,paws(4))}}{has(a,paws(4))} \quad \dfrac{\forall x\ has(x,paws(4)) \rightarrow locom(x,walks)}{has(a,paws(4)) \rightarrow locom(a,walks)}}{locom(a,walks)} \quad \dfrac{{}^*\forall x\ locom(x,walks) \rightarrow \neg\, lives(x,ocean)}{locom(a,walks) \rightarrow \neg\, lives(a,ocean)}}{\dfrac{\neg\, lives(a,ocean)}{cat(a) \rightarrow \neg\, lives(a,ocean)}[cat(a)]}}{\forall x\ cat(x) \rightarrow \neg\, lives(x,ocean)}$$

Figure 2: The transformed proof

$$
\begin{array}{c}
\forall x\, cat(x) \rightarrow has(x, paws(4)) \wedge has(x, tail) \\
\Pi_1 \\
\wedge E\,\dfrac{has(a, paws(4)) \wedge has(a, tail)}{has(a, paws(4))} \\
\Pi_2 \\
\forall x\, cat(x) \rightarrow \neg lives(x, ocean)
\end{array}
\quad \Rightarrow \quad
\begin{array}{c}
\forall x\, cat(x) \rightarrow has(x, paws(4)) \\
\Pi_1' \\
has(a, paws(4)) \\
\Pi_2 \\
\forall x\, cat(x) \rightarrow \neg lives(x, ocean)
\end{array}
$$

Figure 3: An edited proof

the base by splitting all the formulas. But we consider highly inconvenient to change the form of the sentences in the base, because in the event that the user wants to know exactly what was asserted by the domain specialist, it would be very hard to find out. The other is the editing of the proof to eliminate all unnecessary occurrences of these rules, with a corresponding propagation throughout the proof tree to eliminate the vestiges of the extraneous information (cf. Figure 1).

Figure 1:

Occurrences of these rules are either linked to occurrences of disjunctions and/or conjunctions in the axiom base or in the user question itself. In the first case, the exclusion of the extraneous information can be ignored by the reader at this level, and is in accordance with our first principle. In the second, the modification of the conclusion can be used by the dialogue system to identify and correct mistaken presuppositions, and to generate answers which are more informative, even without the need of presenting the explanation (as in: "Are cats felines or herbivores? Yes, cats are felines.").

Tha validity of our transformation is stated in the next theorem:

Theorem 1 *Given a normal ND proof of a formula F, it's possible to build an extended proof with similar structure but containing no $\wedge$-Elim or (unnecessary) $\vee$-Intro rules, whose top-formulas are either the same or implied by the original ones, and whose conclusion F' implies F.*

We'll describe the algorithm for the construction of the extended proof (or, rather, the tranformation of the original proof) that satisfies the above theorem.

We are supposing that all hypotheses-discharging rules are modified to discharge sets of hypotheses whenever the original hypothesis is a conjunction, so there is no hypothesis that has as main connective a conjunction. This helps to reduce the number of $\wedge$-Elim cases.

When the proof is complete, we can proceed to eliminate the irrelevant information rules from the proof. Each undesired rule can be removed, and its branch reconstructed, by the application (or reverse application) of the same original rules (except the one being removed) to the formula in question, while the other branches remain the same. In the case of an elimination rule (the $\wedge$-Elim) the branch has to be reconstructed upwards from the rule conclusion to its top-formula, while in an introduction rule (the $\vee$-Intro) it has to be reconstructed from the single premise downwards to the branch conclusion. Anyway, we need to examine only the resulting top-formula(s) or conclusion. All the rule applications are symmetrical and can be inverted to generate secondary proofs of the modified top-formulas from the original ones, or, conversely, of the original conclusion from the new one. So the new top-formula can be used as an extended-proof anchor for a secondary proof where it's derived from the original one. The conclusion is still valid (since the validity of the deduction was unaffected) and the original conclusion can be obtained from the new one.

Note that some rules (the $\vee$-Elim rule) propagate a downward transformation to other parallel branches upward from the rule, and if one of those other branches has at least one introduction rule then the rule removal that originated the transformation (necessarily the introduction rule: $\vee$-Intro) is not possible. In this case, we can try to push the $\vee$-Elim rule upwards through identical parallel branch sequences, until the propagation is possible in all branches. If it's not possible, this means that the information being eliminated may not be irrelevant after all.

Table 1: The transformation propagates upwards from elimination rules, and downwards from introduction rules. The arrows indicate the direction of the propagation of the rebuilding process.
(*) if this propagation upwards in the parallel branches is not possible (when one of the other branches has at least one introduction rule) then the optional propagation or rule removal which originated the process is invalid and should not be done.

If the original top-formula was a hypothesis, and the rebuilding occurred in all its instances in exactly the same way, then the transformation can be propagated to the discharge point, by rebuilding the corresponding branch, in the same way we described above. If it succeeds that eliminates the need for the secondary proof. If the conclusion of a branch is rebuilt, the transformation should be propagated to this next branch, in the same way. In this case the affected rule has to be a $\rightarrow$-Elim rule, in which case the rebuilding is done upwards towards its own top-formula (the rule can't be a $\vee$-Elim or $\exists$-Elim rule, since in these cases the branch ending on the ma-

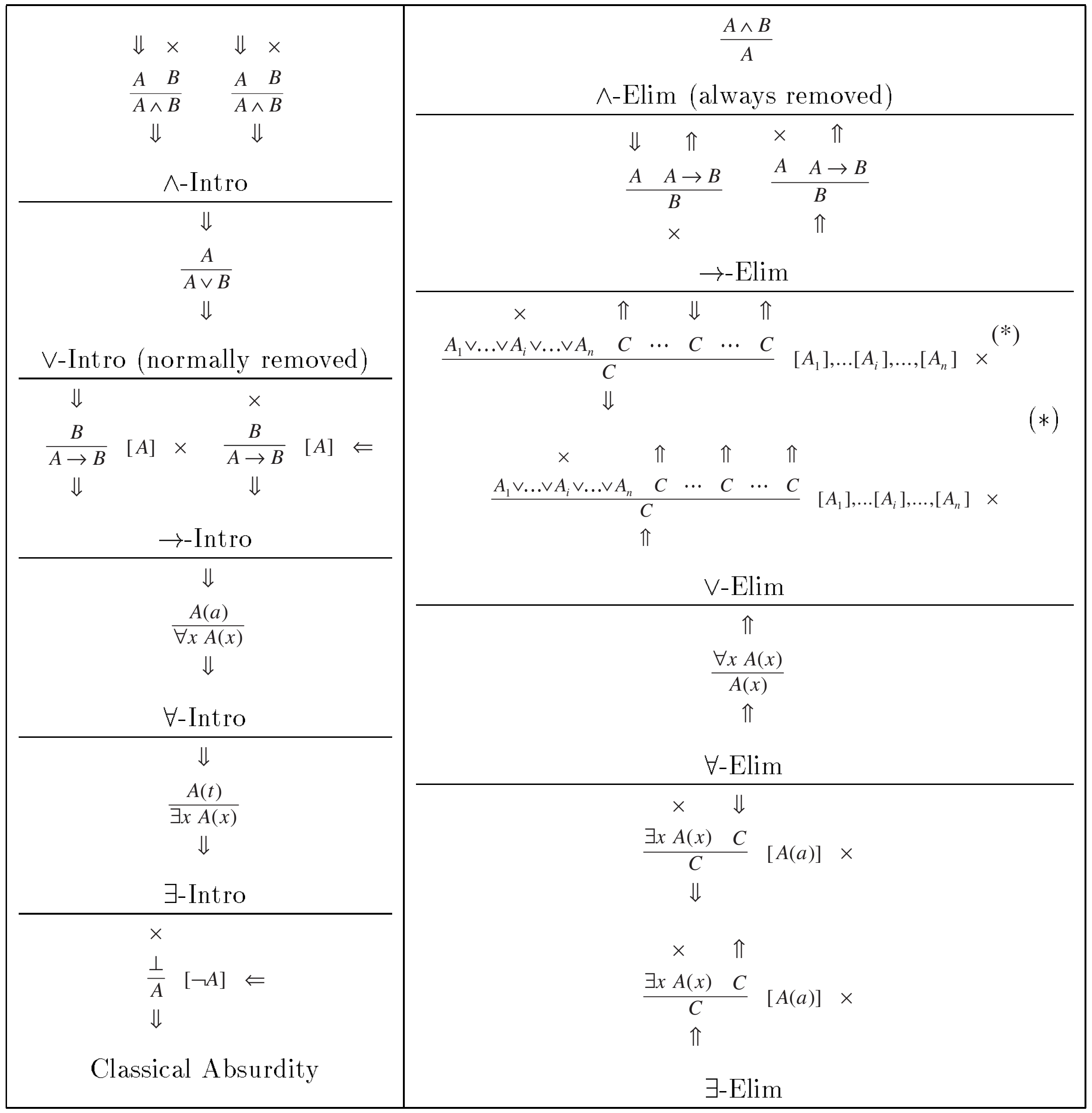

Figure 4: The transformation propagates upwards from elimination rules, and downwards from introduction rules. The arrows indicate the direction of the propagation of the rebuilding process

jor premise contains only elimination rules, which don't propagate downwards). In this way the proof rebuilding can affect large portions of the original proof. The propagation through hypotheses links is optional but recommended whenever possible, and force the reconstruction upwards in elimination rules ($\vee$-Elim and $\exists$-Elim, to the major premise branch) and downwards in introduction rules ($\rightarrow$-Intro and Classical Absurdity).

Finally, when a given hypotheses-discharging rule discharges a hypothesis (from a set of conjuncts) that is not used throughout the proof, the corresponding conjunction can be modified through the exclusion of the unused hypothesis and the modification propagated upwards or downwards through the branch, in the same way as above. In this way large conjunctions can be modified to satisfy our goal.

By reconstructing the whole proof, we arrive on a similar valid deduction. This may differ from the former on some top-formulas and the conclusion, but these top-formulas can be derived from the former ones by proofs which can be easily constructed from the original rule sequences, and the conclusion implies the former conclusion in the same way. So this derived deduction is in a sense *smaller* although logically equivalent to the former. As the new premises are theorems of the knowledge base, this new deduction fits in the extended proof definition above, and so is acceptable from the point of view of the first principle, at the same time that it satisties the second.

The above proof, transformed accordingly, would be as below, where the top-formulas marked with (*) are the ones modified as above (which have secondary proofs that derive them from the original top-formulas is shown in Figure 2

Here is a possible explanatory text:

Cats don't live in the ocean, because they have four paws. Anyone who has four paws walks, so cats walk. Since anyone who walks does not live in the ocean, then cats don't live in the ocean (either).

5 Simplifying Proof Steps

From the point of view of a Normal Form Natural Deduction proof, the third principle can be restated as saying that in a sequence of rule applications of the same character (introduction or elimination) of a given proof, you may state the premises of the uppermost rule(s) and the conclusion of the last, because all the intermediate rules are syntactical reduction or merging rules. You may also do this with rules of different characters and even different branches. In the end you can reduce the whole proof to a single deduction step. But this is not what we are looking for, and it may result in deduction steps whose complexity can be very large. What we want to do is group rule applications in composite proof steps while keeping them simple enough to be easily understood, as in principle 4.

We can start by considering the size of the formulas in the proof. But the largest formulas in the proof are either axioms of the base, or the question submitted to the prover, since we are dealing only with Normal proofs. So the syntactical complexity of the sentences necessary to articulate the argument is at most the same as the complexity of the largest formulas in the knowledge base or the question itself, and normally smaller. Therefore we can assume that the knowledge representation language, the system interface and the knowledge-base builder only produce formulas of "acceptable" complexity, that is, that can be expressed in a way the normal human user can understand without (great) effort. If that's not so, the problem lies in the system itself, and no manipulation of the proof will be able to completely eliminate it.

Having established this, we consider the proof itself. Since we don't want to collapse the whole proof into a trivial text with no argument, we want to divide the proof into parts that can be considered separately. We choose to divide a proof branch at its minimal formula, since this formula is relevant to the transformation of the next section and shouldn't disappear. So the proof branches are divided into their elimination and introduction segments.

We then look at the relative complexity of each basic ND rule. Here we don't have experimental data to support our suggestions, so our method is subject to future verification. We consider the elimination rules with a single premise (the $\wedge$-Elim, which is removed by the previous section method, and the $\forall$-Elim) as the simplest and most obvious rules. The enunciation of the premise already announces its conclusion. So all elimination rules with a single premise should be collapsed with the subsequent rule in their branch, with the premise taking the place of the conclusion. The case of the other elimination rules ($\rightarrow$-Elim (modus ponens), $\vee$-Elim and $\exists$-Elim) involve either more than one premise or a variable substitution, both of which need to be examined in the proof to ascertain its validity, and which require an equivalent care in the textual argument, or we may end with an unclear argument. Even so, a sequence of identical rules coming from a single major premise (a large disjunction $A \vee B \vee C$ or a causal chain $A \rightarrow B \rightarrow C$, for example) can be collapsed into a single rule application, as long as the text generator is able to take advantage of the collapsed rule by expressing it in a resumed way.

On the introduction rules, the case is different. The $\forall$-Intro rule is very sensitive to the interface language, to the way it is able to express the difference between specific and generic individuals and classes. In our current knowledge base, since there are no individuals, only classes, the $\forall$-Intro rule can be mostly ignored in the text generation, with the system using the parameter that replaces the variable to identify the generic case. In such bases, a sequence of $\forall$-Intro rules can be collapsed into a single instance, with few if any problems to the text generator. In other bases, however, it may be necessary to keep the $\forall$-Intro rules separated. A similar but simpler case happens with the $\exists$-Intro rule. In general, we assume that both quantifier introduction rules can be collapsed into a single rule instance whenever they occur in sequences.

The case of the $\rightarrow$-Intro rule is also special because this rule discards a hypothesis. Depending on the kind of knowledge representation language chosen for the domain, there can be large sequences of these rules which can be collapsed without problems, or almost no case of a sequence of two $\rightarrow$-Intro rules. Since this rule (and the Absurdity) discards one or more hypotheses, and the hypotheses links are one of the main rhetorical structures to be considered when building the text, we assume that those rules should not be collapsed when they appear in sequence, but that they can be collapsed with a subsequent quantifier introduction rule (producing derivations of the form "supposing A(x) we derive B(x) so $\forall x A(x) \rightarrow B(x)$").

There remains only the $\wedge$-Intro rule, since the $\vee$-Intro rules are (almost always) removed by the algorithm of section 4. Sequences of $\wedge$-Intro rules can be very common, and they present no problem for comprehension. So we assume a sequence of $\wedge$-Intro rules is always collapsed into a single step, either in the proof process or here in the transformation process. More, we can safely assume that the step itself is redundant, and collapse the step into the one below it in the proof tree, by writing the premises in the place of the conclusion.

We enumerate below the transformations that can take place with each of the basic ND rules, regarding sequences of rules of the same type and the collapse of different rule types. Note that a rule always collapse with the one below it, that is, replacing its conclusion with its premises:

$\forall$-Intro: a sequence of rules can collapse into a single step

$\exists$-Intro: a sequence of rules can collapse into a single step

a single rule can collapse with a $\forall$-Intro rule below it

$\rightarrow$-Intro :a sequence of rules don't collapse a single rule can collapse with a quantifier rule below it

$\wedge$-Intro: a sequence of rules always collapse into a single step

a sequence of one or more rules always collapse into the rule below it

$\vee$-Intro: normally removed from the proof. Does not collapse

$\perp$ (Classical Absurdity): follows the same case as the $\rightarrow$-Intro rule.

$\forall$-Elim: a sequence of rules always collapse into a single step

a sequence of one or more rules always collapse with the rule below it.

$\exists$-Elim: a sequence of rules can collapse into a single step

$\rightarrow$-Elim: a sequence of rules can collapse into a single step

$\wedge$-Elim: always removed from the proof

$\vee$-Elim: a sequence of rules can collapse into a single step.

In this simple way, a great reduction in size can be achieved in the proof, which also puts in evidence the kind of structural information that is really relevant in an argument the way in which many different pieces of information are combined and lead to the conclusion at hand. The reduction could take place at the text generator, but by transforming the proof at the logical level, we can ensure its consistency and the systematicity of the process.

6 Proof Restructuring

The proof restructuring is the most extensive proof transformation within our method. The result is what we call a Proof-Like Structure (PLS) which reflects an *argumentative structure*. To achieve it we follow a sequence of steps, resumed in the following algorithm:

`Proof-Restructuring (P):`
`Apply the algorithm for the removal of Irrelevant Information to P;`
`For each Minimal Formula` F_i `in P do:`
`If it is the Absurdity (`$\perp$`),, consider instead its conclusion;`
`Mark it as a breaking point and select its subtree` TF_i`;`
`Apply recursively Proof-Restructuring to` TF_i`;`

```
   Identify hypotheses and select top-formula facts and anchors
to be discharged
in TF_i, following pragmatical criteria to limit their number;
   Undo the unifications bounding F_i and TF_i to the rest of the
proof,
by rebuilding the deduction from the hypotheses with freed variables;
   Discharge hypotheses and top-formula anchors generating a new
conclusion F_i',
and a complete proof PF_i' of F_i', from TF_i;
   Select the set of Breaking Points F_j whose substitution would
satisfy the pragmatical ccriteria;
   For all the Breaking Points F_j in the selected set, do:
   replace the subtree with the deduction of F_j from F_j',
   place an anchor for PF_j' from F_j';
   Apply the algorithm for Simplifying Proof Steps to P.
```

First, we remove the irrelevant information from the proof, possibly generating some secondary proofs in the process. Then we need to identify proof pieces that represent inference chunks whose internal structure is less important to argumentation than the relations maintained between them. Since we are talking about normal proofs in a ND system, the minimal formulas constitute natural candidates for breaking points between the chunks. We prefer to have as the conclusion of a proof piece the smallest possible formula, as they represent information 'atoms' that are reorganized in order to reach the conclusion. Minimal formulas are then our choice of breaking points, except in one case. The case is when the minimal formula is the absurdity ($\perp$); we then take as breaking point the subsequent formula in the branch, which is proved through contradiction. Note that in the case of the hypothesis-discharging elimination rules (of the existential $\exists$ and the disjunction $\vee$), whose conclusions are identical to one or more of the premises, the major premise (which differs from the rule conclusion) is the minimal (and last) formula of the branch, which has no introduction rule (in this we differ from the definitions in [Oliveira'92]). The proof piece is thus the whole subtree determined by a given breaking point. In this way all proof pieces have as their conclusion meaningful formulas (not the absurdity) and are valid deductions (although maybe not complete proofs) in themselves.

The second step is to recursively apply the proof restructuring to each proof piece. This can be optimized by a good selection of the first breaking points, by considering the lowest minimal formulas and checking whether the remaining deduction (the proof without their subtrees) satisfies our pragmatical criteria on the size of the resulting explanation. We consider that the pragmatical criteria have to be tailored to the specific knowledge-representation language chosen and the characteristics of the text generator. But, as a general rule, we suggest counting the number of predicates occurring in the deduction and the number of "objects" referred to by the terms in those predicates. Both those numbers (and even their sum) should be lower than the "magical number" (5 to 9) [Miller'56] of items a human being can simultaneously consider. However, this pragmatical criterium has to be validated and more experimentation should be done before we arrive on a way to optimize the criteria to specific knowledge-based systems. We are here more interested in proposing mechanisms that can be used to implement specifical criteria than concluding exactly whose criteria should be applied in each case.

Then we make each proof piece a complete proof in itself, thus generating a new conclusion for the proof piece, which will be a theorem of the knowledge base. This is done by first undoing the unification steps necessary to link the breaking point formula to the rest of the proof, which turns the proof piece into a more general deduction. The idea here is to convey all information stored in the structure of the deduction, derived from its top-formulas, regardless of the possible instantiations that could be necessary to apply it to the proof. The links to the main proof are done through unification between terms in the main proof and free variables in the conclusion, and also through the use of (undischarged) hypotheses in the deduction. Since the unification is done to the whole deduction, we must rebuild it from its top-formulas. The undischarged hypotheses will be substituted for local hypotheses, so their terms can also be turned into free variables. We should however substitute only terms that appear in the conclusion of the deduction (the breaking point), since it will be the focus of the argumentative text, and the point where the generalization should take place.

So before undoing the unification steps we must select the hypotheses in the proof piece, and identify in them the terms which appear both in a hypothesis and in the proof piece conclusion (the breaking point). These terms can be replaced with free variables, in all the top-formulas in which they appear, that is, all the hypotheses and theorem anchors have to be examined. Axioms shouldn't have free variables, so they would never need a substitution. The deduction is then rebuilt so that the conclusion has as many free variables as possible given the argument structure. The result is what we call an independent deduction, which retains no links to the previous proof.

We take a further step in the generalization of the proof piece by allowing the consideration of any particular fact or "factual theorem" (a top-formula anchor to another proof which has the same structure as a fact) as an (undischarged) hypothesis, and substituting its terms for free variables when they coincide with the conclusion, before rebuilding the deduction. The number of predicates and terms in the hypotheses and the conclusion are subject to limitation from our pragmatical criteria, that is, they should be limited by our "magical number", since they would all be mentioned in the proof conclusion. An independent deduction with a conclusion that is too complex should be avoided, in general. The characteristics of the knowledge representation language and the domain can also be used to identify terms that should (or should not) be generalized, helping the identification of the facts that can or should be considered as hypotheses. Another consideration is that it is often undesirable to include additional terms in the conclusion, by choosing as hypothesis a fact which has another term that is not being generalized. This would introduce in the main discourse another object that was previously not mentioned, and unless the domain asks for it, it should be avoided. A possible exception is when the proof structure is complex enough (through the use of "difficult" rules, for example, as the classical absurdity, or an irreducible sequence of rules with too many top-formulas and intermediate results) to warrant an alternate presentation. Then the identification of those "hypotheses-like" facts can indeed provide an explanation without using the complex axioms that originated the proof piece structure (our example in section 7 has one such piece that is more complex than the average argument structure).

We can then discharge the not-yet-discharged hypotheses, through an implication ($\rightarrow$) introduction rule discharging their conjunction. Universal quantification introduction rules are also applied to bound the variables that are free in the conclusion. The result is a new formula which is a theorem. It is as general as we can (or want to) make it, which is advantageous from the point of view of explanations. Normally, it is more informative to make a general assertion than an individualized one, because by making a general assertion, the system avoids implying to the user that the assertion would not be generalizable, and states exactly what the proof implies (that is, that the argument is valid in that range of cases). However, pragmatical criteria, as well as the coherence of system behavior, may suggest otherwise, and the generalization of facts and "factual theorems" should be subject to these considerations. Anyway, it is also of logical interest, because it enables the system to extract and identify frequently used intermediate results in the form of theorems that can be stored and reused.

For each proof piece there is now an associated theorem and its proof. We can then substitute the whole proof piece in the original proof by a new deduction of the breaking point from the associated theorem, which is a simple enough direct deduction recurring to locally available hypotheses and particular facts and anchors. For example, when the breaking point is $F(t)$, the undischarged hypothesis is $H(t)$ and the particular fact is $A(t)$, the proof piece can be substituted by the following deduction, where the main top-formula is a theorem of the base:

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

This small deduction can replace the original proof piece while maintaining the correctness of the proof. This is an extended proof, and associated to the theorem used in the deduction there is its own proof, which can be examined to assert the validity of the argument. So we have a recursive structure, the PLS, which may be used to structure the textual justification of the main conclusion.

The last step in the algorithm is to apply the simplification of proof steps to the resulting PLS, so that the number of formulas and rules (and the maximum size of the resulting text) is reduced while preserving the clarity of the argument. In fact, the simplification of proof steps can also be made before the proof structuring, since the relevant characteristics (minimal formulas, top-formulas, conclusion) are preserved.

What remains to be done is to establish the criteria through which we will choose the appropriate breaking points in the original proof. Although every subset of breaking points would result in a PLS, these may not satisfy our cooperativity principle above. The PLS is an instrument to structure the proof, which must be applied according to pragmatical criteria, in order to produce satisfactory results.

We identify a series of criteria which are domain-independent, based on syntactical data with pragmatic considerations in mind, that should be verified when choosing the breaking points. These can be adapted to a particular domain, if neccessary, as well as being applied together with additional domain-specific criteria.

Each breaking point should be weighted according to the size of its theorem, as well as the resulting proof size, as we said above. A small theorem is preferable. This may conflict with the generalization over facts and top-formula anchors described above. These criteria can be tailored to the average formula complexity of the domain, which is also a purely syntactical information, and be used to weight the many breaking point candidates, while guiding the generalization choices in each proof piece and excluding the breaking points with unnacceptably large theorems.

In general, the largest the proof piece the better the breaking point, as long as the resulting proof retains a manageable amount of information to be presented. The number of predicates and terms making up the proof should be within the acceptable limits (around 7, cf. [Miller'56]), whenever possible.

A structural criteria to identify desirable breaking points is to look at the terms that appear in the proof conclusion. If a breaking point contains one or more of these, and none (or only a few) of other ones, it bears a more or less direct relation, in content, to the conclusion of the proof, and should be preferred to the ones that present too much new information.

In this way we strive to limit the amount of information contained in each proof making up the PLS. As the structure is recursive, each proof subsumed in a theorem should also be transformed into a PLS, so the result allows for a flexible dialogue between the user and the system, each time focusing on a specific point of the argument, until the user is satisfied.

It is evident that, in the presence of additional cognitive information related to a specific domain, new criteria could be drawn that would enhance the breaking point selection, allowing for a better argumentative focus. We therefore emphasize the flexibility of this tool for the generation of structured explanations in logic-based knowledge systems.

7 A Working Example

In this section we present an example chosen to be complex enough to show the techniques developed in this article. The knowledge base over which it is based is still under construction, and will be used for future experimentation of the algorithm together with the whole inteface. The text generator is not ready yet for this base, so the textual explanations are hand-generated.

Suppose we have the following axioms in our knowledge base (in fact, a possible future section of our Brazilian Fruits KB, in development), where "aceroleira" is a Brazilian tree of the Malpighia genus whose fruit is a small red berry, and Paranagua is a city in the Brazilian state of Parana:

Facts:

1. $\mathsf{Malpighia}(\mathsf{aceroleira})$

2. $\mathsf{city}(\mathsf{Paranagua}, \mathsf{Parana})$

3. $\mathsf{part\text{-}of}(\mathsf{Parana}, \mathsf{SouthRegion})$

4. $\mathsf{in}(\mathsf{Paranagua}, \mathsf{CoastalArea})$

Axioms:

1. $\forall x\ \mathsf{Malpighia}(x) \rightarrow \mathsf{tropical\text{-}cycle}(x) \wedge \mathsf{MinAdTemp}(x, 10C) \wedge \mathsf{MinAdRain}(x, 1000)$
2. $\forall x\ \mathsf{tropical\text{-}cycle}(x) \rightarrow \neg\mathsf{dormant\text{-}state}(x)$
3. $\forall x \forall t\ \mathsf{MinAdTemp}(x, t) \wedge t > 0C \wedge \neg\mathsf{dormant\text{-}state}(x) \rightarrow \mathsf{sensitive\text{-}to}(x, \mathsf{frost})$
4. $\forall x \forall y \forall z\ (\mathsf{sensitive\text{-}to}(x, y) \vee \mathsf{illness}(y, x)) \wedge \mathsf{occurs}(y, z) \rightarrow \neg\mathsf{do\text{-}well}(x, z))$
5. $\forall x \forall y\ \mathsf{city}(x, y) \rightarrow \mathsf{in}(x, y)$
6. $\forall x \forall y\ \mathsf{part\text{-}of}(x, y) \rightarrow \mathsf{in}(x, y)$
7. $\forall x \forall y \forall z\ \mathsf{in}(x, y) \wedge \mathsf{in}(y, z) \rightarrow \mathsf{in}(x, z)$
8. $\forall x\ \mathsf{in}(x, \mathsf{SouthRegion}) \rightarrow \exists t \mathsf{MinAdTemp}(x, t) \wedge t < 0C$
9. $\forall x\ \mathsf{in}(x, \mathsf{CoastalArea}) \rightarrow \mathsf{humid}(x)$
10. $\forall x \forall t\ \mathsf{MinTemp}(x, t) \wedge t < 0C \wedge \mathsf{humid}(x) \rightarrow \mathsf{occurs}(\mathsf{frost}, x)$

Then the question: $\mathsf{do\text{-}well}(\mathsf{aceroleira}, \mathsf{Paranagua})$ has a negative answer, but the conventional ND proof of the negation is large (55 formula instances and 36 rule applications), and the textual explanation would require a series of complex paragraphs.

Applying the irrelevant information algorithm to the proof we eliminate three rule instances arriving in a reduced proof with 52 formula instances and 33 rule applications, together with 3 secondary proofs deriving simplified instances of the axioms 1 and 4.

After applying the algorithm for simplified proof steps, we arrive at the following proof, with only 31 formula instances and 12 rule applications, with the same 3 secondary proofs (we have broken the proof and abbreviated predicate names to reduce the proof size):
(a): Proof of $\mathsf{in}(\mathsf{Paranagua}, \mathsf{SouthRegion})$: see Figure 5.
(b): Proof of $\mathsf{occurs}(\mathsf{frost}, \mathsf{Paranagua})$ using (a): see Figure 6.
(c): Proof of $\mathsf{sensitive\text{-}to}(\mathsf{aceroleira}, \mathsf{frost})$ with anchors to secondary proofs $\langle 1 \rangle$ and $\langle 2 \rangle$.
Secondary proof $\langle 1 \rangle$ (proof $\langle 2 \rangle$ is similar to $\langle 1 \rangle$, from the same axiom):

$$\dfrac{\dfrac{city(P, Parana) \quad \forall x \forall y\ city(x,y) \rightarrow in(x,y)}{in(P, Parana)} \quad \dfrac{part-of(Parana, SR) \quad \forall x \forall y\ part-of(x,y) \rightarrow in(x,y)}{in(Parana, SR)} \quad \forall x \forall y \forall z\ in(x,y) \wedge in(y,z) \rightarrow in(x,z)}{in(P, SR)}$$

Figure 5: Proof of in(Paranagua, SouthRegion)

$$\dfrac{\dfrac{\overset{(a)}{in(P,SR)} \quad \forall x\ in(x,SR) \rightarrow \exists t\ MT(x,t) \wedge t < 0C}{\exists t\ MT(P,t) \wedge t < 0C} \quad \dfrac{[MT(P,t_1)] \quad [t_1 < 0C] \quad \dfrac{in(P,CA) \quad \forall x\ in(x,CA) \rightarrow h(x)}{h(P)} \quad \forall x \forall t\ MT(x,t) \wedge t < 0C \wedge h(x) \rightarrow occ(f,x)}{occ(f,P)}}{occ(f,P)} [MT(P,t_1)],[t_1 < 0C]$$

Figure 6: Proof of occurs(frost, Paranagua)

$$\dfrac{\dfrac{[M(a_1)] \quad \forall x\ M(x) \rightarrow tc(x) \wedge MAT(x,10C) \wedge MAR(x,1000)}{tc(a_1) \wedge MAT(a_1,10C) \wedge MAR(a_1,1000)}}{\dfrac{MAT(a_1,10C)}{\forall x\ M(x) \rightarrow MAT(x,10C)}[M(a_1)]}$$

(d): Proof of $\neg$do-well(aceroleira, Paranagua) using (b) and (c) and secondary proof $\langle 3 \rangle$:

$$\dfrac{\overset{(c)}{st(a,f)} \quad \overset{(b)}{occ(f,P)} \quad \overset{\langle 3 \rangle}{\forall x \forall y \forall z\ st(x,y) \wedge occ(y,z) \rightarrow \neg dw(x,z)}}{\neg dw(a,P)}$$

Secondary proof $\langle 3 \rangle$:

$$\dfrac{\dfrac{\dfrac{[st(a_1,b_1)]}{st(a_1,b_1) \vee ill(b_1,a_1)} \quad [occ(b_1,c_1)] \quad \forall x \forall y \forall z\ (st(x,y) \vee ill(y,x)) \wedge occ(y,z) \rightarrow \neg dw(x,z)}{\neg dw(a_1,c_1)}}{\forall x \forall y \forall z\ st(x,y) \wedge occ(y,z) \rightarrow \neg dw(x,z)}[st(a_1,b_1)],[occ(b_1,c_1)]$$

This breakdown of the main proof at the minimal formulas, resulting in smaller chunks, can be reflected in an explanation text divided in many paragraphs. In fact, here we have a selection of breaking points that serve to begin the proof-restructuring process.

The next step is the generalization over the proof pieces. The proof piece in (a) could be generalized by considering city(Paranagua, Parana) as a hypothesis, but this would introduce an additional term (Parana) in the conclusion, resulting in the theorem

$$\forall x\ \mathsf{city}(x, \mathsf{Parana}) \rightarrow \mathsf{in}(x, \mathsf{SouthRegion}).$$

This alternative is generally unfavored, and the context provides nothing to support it. Moreover, the predicate in normally does not require elaboration or explanation, so in this knowledge base we have a pragmatical criterium that prevents the complexification of this breaking point. So the proof piece (a) is left as it is.

The proof piece in (b) is very complex, and the alternative is to generalize it to try to identify the relevant pieces of information. It can be generalized by taking as minimal formula the conclusion itself, and selecting as hypotheses the facts in(Paranagua, CoastalArea) and in(Paranagua, SouthRegion) (the latter is an anchor to (a)). In this case, the generalization results in the formula: $\forall x$ in(x, CoastalArea) $\wedge$ in(x, SouthRegion) $\rightarrow$ occurs(frost, x), which is not prevented in our base. So we have two versions of the same proof piece: one the original (b) with occurs(frost, Paranagua) as anchor, and the new one with a theorem linking the occurrence of frost to the location in the coastal area of the south region of Brazil.

$$
\dfrac{
\dfrac{M(a) \quad \overset{\langle 1 \rangle}{\forall x\ M(x) \rightarrow MAT(x{,}10C)}}{MAT(a{,}10C)}
\qquad 10C > 0C \qquad
\dfrac{\dfrac{M(a) \quad \overset{\langle 2 \rangle}{\forall x\ M(x) \rightarrow tc(x)}}{tc(a)} \quad \forall x\ tc(x) \rightarrow \neg ds(x)}{\neg ds(a)}
\qquad \forall x \forall t\ MAT(x,t) \wedge t > 0C \wedge \neg ds(x) \rightarrow st(x,f)
}{st(a,f)}
$$

Figure 7: Proof of sensitive-to(aceroleira, frost)

However, when we try to apply the new theorem as anchor in the main proof, the new facts and terms mentioned make it too complex to be desirable, so the simple ungeneralized version is preferred. For the explanation of the secondary proof of occurs(frost, Paranagua), in its turn, the new generalized version can be used, to try to convey useful information without having to follow all the complex details of the original proof piece.

The third proof piece, (c), lends itself to a straightforward generalized version, with

$$\forall x \ \mathsf{Malpighia}(x) \rightarrow \mathsf{sensitive\text{-}to}(x, \mathsf{frost})$$

as theorem. Another minimal formula ($\neg$dormant-state(aceroleira)) results in an additional secondary proof with $\forall x$ Malpighia$(x) \rightarrow \neg$dormant-state(x) as theorem.

The fourth proof piece is the remaining main proof, and the resulting PLS is as below:

i) Main Proof, with 4 predicates and 3 objects:

$$\frac{\dfrac{M(a) \quad \overset{\langle 4\rangle}{\forall x\, M(x) \rightarrow st(x,f)}}{st(a,f)} \quad \overset{\langle 5\rangle}{occ(f,P)} \quad \overset{\langle 3\rangle}{\forall x \forall y \forall z\, st(x,y) \land occ(y,z) \rightarrow \neg dw(x,z)}}{\neg dw(a,P)}$$

A corresponding text would be as following (in parenthesis a textual implicature that could be removed by the text generator, as long as the anchor in this case can be considered unnecessary):

The Aceroleira does not do well in Paranagua, because they're from the Malpighia genus, Malpighia genus plants are sensitive to frost, frost occurs in Paranagua (and plants sensitive to something that occurs in a given place don't do well there).

Underlined we see the anchors for the secondary proofs:

ii) Secondary proof corresponding to Anchor 4 (see Figure 8):

A corresponding text would be:

Plants of the Malpighia genus are sensitive to frost because their Minimum Adequate Temperature is 10 degrees Celsius, which is above 0, and they don't have a dormant state (and plants whose Minimum Adequate Temperature is above 0 degrees Celsius and which do not have a dormant state are sensitive to frost).

iii) Secondary proof corresponding to Anchor 5:

$$\frac{in(P,CA) \quad \overset{\langle 8\rangle}{in(P,SR)} \quad \overset{\langle 7\rangle}{\forall x\, in(x,CA) \land in(x,SR) \rightarrow occ(f,x)}}{occ(f,P)}$$

$$\cfrac{\cfrac{\overset{\langle 1 \rangle}{[M(a_1)] \quad \forall x\ M(x) \to MAT(x,10C)}}{MAT(a_1,10C)} \quad 10C > 0C \quad \cfrac{\overset{\langle 6 \rangle}{[M(a_1)] \quad \forall x\ M(x) \to \neg ds(x)}}{\neg ds(a_1)} \quad \forall x \forall t\ MAT(x,t) \wedge t > 0C \wedge \neg ds(x) \to st(x,f)}{\cfrac{st(a_1,f)}{\forall x\ M(x) \to st(x,f)}[M(a_1)]}$$

Figure 8: Secondary proof corresponding to Anchor 4

$$\cfrac{\cfrac{[in(p_1,SR)] \quad \forall x\ in(x,SR) \to \exists t\ MT(x,t) \wedge t < 0C}{\exists t\ MT(p_1,t) \wedge t < 0C} \quad \cfrac{[MT(p_1,t_1)] \quad [t_1 < 0C] \quad \cfrac{[in(p_1,CA)] \quad \forall x\ in(x,CA) \to h(x)}{h(p_1)} \quad \forall x \forall t\ MT(x,t) \wedge t < 0C \wedge h(x) \to occ(f,x)}{occ(f,p_1)}}{}[MT(p_1,t_1)],[t_1 < 0C]$$

Figure 9: Secondary proof corresponding to Anchor 7

A corresponding text would be:

Frost occurs in Paranagua because Paranagua is a place in the Coastal Area and is on the South Region, and places in the Coastal Area of the South Region are subject to frost.

Note that here we must present explicitly the anchor for $\langle 7 \rangle$, so the implicature can't be used to reduce the text.

iv) Secondary proof corresponding to Anchor 6:

$$\cfrac{\cfrac{\cfrac{[M(a_1)] \quad \overset{\langle 2 \rangle}{\forall x\, M(x) \rightarrow tc(x)}}{tc(a_1)} \qquad \forall x\, tc(x) \rightarrow \neg ds(x)}{\neg ds(a_1)}}{\forall x\, M(x) \rightarrow \neg ds(x)}\,[M(a_1)]$$

A corresponding text would be:

Plants of the Malpighia genus don't have dormant states, because they are tropical-cycle plants (and tropical-cycle plants don't have dormant states).

v) Secondary proof corresponding to Anchor 7:

$$\frac{occ(f, p_1)}{\forall x\, in(x, CA) \wedge in(x, SR) \rightarrow occ(f, x)}\,[in(p_1, CA)], [in(p_1, SR)]$$

A corresponding text would be:

Frost occurs in the Coastal area of the South Region, because in every place in the South Region the Minimum Temperature is below 0 degrees Celsius, and if it's in the Coastal Area it's humid too, and frost occurs in places where the Minimum Temperature is below 0 degrees Celsius and where it's humid.

vi) Secondary proof corresponding to Anchor 8: it's the same deduction as in (a) above.

A corresponding text would be:

Paranagua is in the South Region because it's a city of Parana, (so it's in Parana), and Parana is part of the South Region (so it's in the South Region).

The transitivity rule should not be written since it's part of the meaning of the word "in", and the phrases inside parenthesis should also be omitted, since they represent part of the meaning of "city" and "part of".

The only proofs remaining correspond to anchors $\langle 1 \rangle$, $\langle 2 \rangle$, and $\langle 3 \rangle$. We think it's unnecessary to explain such proofs, the explanation in those cases would simply be the enunciation of the corresponding axiom.

The combination of the above explanation texts allow the interface to establish an explanation dialogue to the user, in which the system uses small paragraphs whenever possible, and the user can ask for further detailing of the argument when necessary. The use of small paragraphs of text is very attractive for most knowledge-based system applications, and the ability to present the complete argument with all the details whenever necessary has the ability to make explanations useful in those systems. The problem then moves from the explanatory system to the knowledge base itself, since the quality of the system's explanations is now limited by the quality of the domain representation.

8 Concluding Remarks

In this paper we have presented a method for transforming normalized Natural Deduction proof trees into proof structures adequate to be mapped onto text structures to render good quality Natural Language explanations for knowledge-based systems. The method is inspired in Toulmin's definition of what an argument is and in Grice's Maxims, both being renowned attempts at providing a formal basis for mutual intelligibility.

One of the main features of the resulting explanatory structure from our method is a hypertext-like structure, in which chunks of sensible information are provided to users and expanded on demand. This is an important attemp at pragmatic adequacy [Leech'83] that we do; all transformations upon ND proofs are so done that the remaining logical formulas contain information that users can bind together in an inferential process that is natural to them. Breakdowns in this process can be handled by expanding proof nodes in accordance to proof-like structures that account for the argumentative thread in the system's reasoning.

Another noteworthy feature is our attempt to keep only focal information in the proofs. This is achieved through an elimination of all information that is not directly related to the ones present in the conclusion. Cohesive and coherent knowledge representation in multimodal interfaces and textual explanations are a mandatory quality without which users are easily led astray in their interpretation of artificial intelligent agents. Although cohesion-oriented transformations can be carried out at text-structure level, our decision for performing them at proof level guarantees coherence and soundness of conclusions (while merely textual manipulations have been historically observed to produce sophisms rather than logical arguments).

We are using only the standard First-Order Logic and a Natural Deduction theorem prover, with minimal assumptions over the structure of the knowledge base and other components of the system. The overall results are encouraging enough to allow the conclusion that simple but systematic approaches can be successful in generating good quality interactions that satisfy the cooperativity goals while not attempting the perfection of human-like dialogues.

The Natural Deduction theorem prover and the proof transformation algorithm presented here are currently parts of a larger system, LINX (cf. [de Souza et al.'96]), a prototype integrating research results in many different areas. It uses a subset of Portuguese for the textual interaction. The system is in the final phases of implementation and will be used to test the many theories involved in its development.

Our ongoing research aims to furthering logic knowledge representation processing so that optimal textual and hypertextual explanations can be generated to improve the usability of knowledge-based systems in classificatory domains. Our prototype will be used to test the use of pragmatical criteria in an extended knowledge base being built, which should be large enough to provide a number of interesting proof examples. We are also investigating the demands our methods impose over the knowledge base, such as the restrictions in the complexity of the axioms, the increased demand for cohesion and coherence in the knowledge representation, and the different levels of detail in the logical relations between the concepts. We also intend to produce a stand-alone version of the theorem prover, together with the proof transformation module, to be made available to other researchers in the same field, and to be used for systematical experimentation and validation of the pragmatical criteria.

Recent results include a method for interpreting the results of the theorem prover in the case of partial or negative answers, with an analysis of the impact of the use of logical connectives in the context of cooperative interaction (cf. [Oliveira et al.'98]). We observe that the exact meaning of the logical connectives has to be taken into account when generating natural language answers from formal logical systems in order to avoid misleading the user. We have also investigated how multimodal reasoning and explanation could be implemented within the basic framework of our system, taking as reference the Hyperproof system (cf. [Barwise&Etchemendy'94]).

Long-range goals are directed towards ensuring automatic knowledge acquisition and usage as independently as possible from human engineering, the use of automated knowledge acquisition to guarantee an acceptable quality on the knowledge base, the investigation of alternative and/or expanded logics and the integration of multiple representation systems to account for complex domains with a classification aspect.

References

[Barwise&Etchemendy'94] Barwise, J. & Etchemendy, J. Hyperproof. CSLI, Stanford, distributed by Cambridge University Press, 1994.

[Cawsey'92] Cawsey, A. Explanation and Interaction. Cambridge,Ma. MIT Press, 1992.

[de Souza'93] de Souza, C.S. The Semiotic Engineering of User Interface Languages. International Journal of Man-Machine Studies no.39, pp.753-773. November 1993.

[de Souza et al.'96] de Souza, C.S., Garcia,L.S., Dias,M.C.P. & Quental,V.S.D.B. LINX: An Integrated Interface Environment for Knowledge-Based Systems (in Portuguese). Proceedings of the II Meeting on Computational Processing of Written and Spoken Portuguese, of the XIII Brazilian Symposium on Artificial Intelligence, pp. 29-39. Curitiba, Brazil, 1996.

[Dias'94] Dias,M.C.P. The Lexicon in Automated Text Analysis and Generation Systems in Portuguese Language (in Portuguese). Ph.D. Dissertation. Departamento de Letras, PUC-Rio. Rio de Janeiro, Brazil, 1994

[García'95] Garcia, L.S. An Integrated Interface Environment for Knowledge-Based Information Systems (in Portuguese). Ph.D. Dissertation. Departamento de Informática. PUC-Rio. Rio de Janeiro, Brazil, 1995

[Grice'75] Grice, H. P. Logic and Conversation. In Cole, P. & Morgan, J. L. (eds) Syntax and Semantics, vol. 3: Speech Acts. Academic Press, New York, 1975.

[Huang'94] Huang, X. Planning Argumentative Texts. In Proceedings of COLING94 Kyoto, Japan, 1994.

[Huang'96] Huang, X. Human Oriented Proof Presentation: A Reconstructive Approach. Ph.D. Dissertation, DISKI 112, Infix, Sankt Augustin, Germany, 1996.

[Leech'83] Leech, G. The Principles of Pragmatics. Longman, New York, 1983.

[Maybury'93] Maybury, M.T. (Editor) Intelligent Multimedia Interfaces. AAAI Press. Menlo Park, 1993.

[Miller'56] Miller, G. The Magical Number, 7 plus or minus two: Some limits on our capacity for processing information. Psychological Review no. 63 pp.81-97. 1956

[Moore'95] Moore, J.D. Participating in Explanatory Dialogues. MIT Press. Cambridge,Ma. 1995

[Oliveira'92] Oliveira, D.A.S. A Natural Deduction Theorem Prover Capable of Complementing its Knowledge (in Portuguese). Master Dissertation. Departamento de Informática. PUC-Rio. Rio de Janeiro, Brazil, 1992.

[Oliveira et al.'98] Oliveira, D.A.S, de Souza, C.S., Haeusler, E.H. Dealing with Negative Answers in Logic. Communication presented at Logic in Natural Language, LINGUA'98, Recife, Brazil, 1998.

[Paz&Haeusler'94] Paz, A.C. & Haeusler, E.H. A Generator of Interactive Natural Deduction Theorem Provers. Proceedings of the XI Brazilian Symposium on Artificial Intelligence pp.557-571, Fortaleza, Brazil, 1994.

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

[Quental'95] Quental, V.S.D.B. A Grammar for Automatic Text Comprehension and Generation in Portuguese Language (in Portuguese). Ph.D. Dissertation. Departamento de Letras, PUC-Rio. Rio de Janeiro, Brazil, 1995.

[Toulmin'64] Toulmin, S. E. The Uses of Argument. Cambridge University Press, 1964.