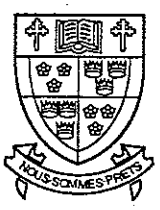
The First Conference of the Pacific Association for Computational Linguistics

Proceedings of the Conference

April 21 – 24, 1993

The Harbour Centre Campus
of
Simon Fraser University
Vancouver, British Columbia
Canada



Natural Language Explanation of Natural Deduction Proofs

Andrew Edgar
Dept. of Computing Science
Univ. Alberta
Edmonton, Alberta
Canada T6G 2E1

Francis Jeffry Pelletier
Depts. of Computing Science & Philosophy
Univ. Alberta
Edmonton, Alberta
Canada T6G 2E1

1. Some uses of automated theorem proving.

Automated theorem proving has been seen as having a wide variety of applications to issues of computational interest. Perhaps one of the first uses that comes to mind is *program verification*: an automated theorem prover would mechanically prove that a given program satisfies some specification, or produces the same output as some another program, or can be executed within certain time and space bounds. To do this, there must be a formal program semantics given for the programming language, and an automated program verifier reduces the question of whether the program has such-and-such a property to the question of whether so-and-so formulas are theorems of the formal semantics [Sterling & Bundy 1982, Dijkstra & Scholten 1990]. Along these same lines, one might wish to present a formal statement of an algorithm and have an automated theorem prover generate explicit code in a programming language; that is, develop an *automatic programmer* [Manna & Walinger 1977]. Or even more ambitiously, one might wish only to specify the relations that are to hold between the input and output variables, and to have an automated theorem prover try to generate a proof of $(\forall x)(Px \rightarrow (\exists y)Qxy)$....where x ranges over input variables to the program and P is the predicate that the input is expected to satisfy, y ranges over output variables and Q specifies the relation between each input variable and its associated output variable after execution of the desired program. Given sufficient other axioms so that this formula is provable, the proof of the formula can be converted into a program by rather well-understood mechanisms. This is *program generation* [Gumb 1989].

Another group of uses to which automated theorem proving is put contains question/answer systems. Not only can theorem provers provide answers to yes/no questions posed to a database, but long ago it was shown how to use theorem proving techniques to extract answers from a declarative database [Green 1969a, 1969b]. And a slight generalization of the method can be used to answer 'how'-questions that require the specification of a sequence of actions to perform a task [Lehnart 1977]. And this last use has suggested how to use automated theorem proving to automatically specify a "plan", and this [STRIPS/ ABSTRIPS] robotic planning methodology is seen to apply widely to all types of mechanical generation of a sequence of actions which will accomplish a goal which was stated at a very high level of generality [Fikes & Nilsson 1971, Fikes *et al* 1972, Sacerdoti 1974].

A third area of application of some ideas taken from automated theorem proving is diagnosis and knowledge-based expert systems generally). In this area there is generally a "background theory" which gives a group of typical (and overlapping) causes for an observed even or symptom. Having generated a number of premises from which the symptom could be deduced, the inference engine then has the task of "paring down" the number of possible explanations of that fact by requesting further information which would be consistent with only some of these possible hypotheses. [See Reiter 1987 and Shapiro 1981, for example.)

Finally there are the more "cognitive science" applications of automated theorem proving. One area has been the use of automated theorem provers as an aid in mathematics (and other fields). The idea is that the person will guide the computer, making overall strategic suggestions and allowing the computer to organize and prove lower level portions of the proofs [Guard *et al* 1969, Bundy 1983]. In the realm of language understanding (by humans), it has been claimed that people engage in (sub-conscious) theorem proving in order to draw inferences from "what is said" to "what is meant"[Grice 1975, Hirst 1987]; that they must draw inferences (again un- or sub-consciously) on a very wide range of topics so as to be able to know (for example) what events occur before which events in a tale that is narrated to them; and that they draw a wide variety of inferences based on "world knowledge" together with what is said in order to be able just to minimally understand what the speaker intends [Moore & Paris 1991, Allen 1991, Schubert & Hwang 1990]. Therefore, any computer natural language understanding system must be able to invoke a theorem prover in order merely to attain minimal competence. Of course, once one has this sort of information, we would perhaps be able to construct or generate a text [Hovy 1988]. Furthermore, since rational reasoning has long been seen as what sets people apart from other animals, investigations into computerized theorem proving has been sometimes thought to be relevant to understanding

human psychology [Johnson-Laird & Byrne 1991 Chap. 9, Rips 1984, Boden 1977].

We thus see that automated theorem proving has classically been viewed as having a wide range of applications: from program verification and generation, through question/answer systems and robotic planning systems, through man-machine cooperative systems, and finally to the investigation of human cognition and natural language understanding systems.

2. Why a natural language "back end" is desirable.

Such uses of automated theorem proving would be a great boon if they were successfully completed, but the fact of the matter is that they are currently only marginally acceptable; and even if a scientific breakthrough in the field were to happen, it is surely the case that the general population (or indeed, even the scientific community) would justifiably be wary of any such claim, and would demand some ability to investigate whether any particular use was in fact correct. We are all familiar with such challenges as "Would you trust a never-tested train routing program if it were automatically verified?" or "Do you believe this nuclear power plant to be safe on the basis of this automated theorem prover's proof that its circuitry has the predicate 'fail-safe' truly applied to it?" And we all recall the reaction of the computer science community to the claim that the space lasers of the Strategic Defense Initiative would be guided by a program too large to be comprehended by a person but which could be automatically verified.

But we needn't go to such extremes to find cases where we would like the option of having humans be able to inspect and understand the proofs produced by an automated theorem prover. Certainly in the setting of a mathematician interacting with an automated theorem prover, it is quite certain that there should be an ability for the mathematician to be able to inspect the "low level" portion of any proof that the automated theorem prover asserts is successful. In the area of program generation, if a theorem prover alleges that its program is able to perform some task, it would be the height of folly not to inspect the background proof. And surely one quite often wants to be able to know how a question-answer system arrived at its answer, especially in those cases where it is using quite a lot of theorem proving power in order to generate the answer. If we are truly interested in people's cognitive abilities, we are not really interested in the mere assertion that a theorem prover might make to the effect that such-and-so formula is a theorem, but rather we are interested in *how* the formula was proved to be a theorem and whether this corresponds to how people do similar proofs.

When a natural language understanding system is employed to represent the "underlying meaning" of a story or other text, we certainly would like to have the ability to inspect for ourselves whether we think the system has hit upon the correct understanding of the text as we know it. Along these same lines, it has long been recognized that knowledge-based systems require a "back end" so that the user can understand *why* the system came to the diagnosis it did. (Much work has been done in this area. Different styles of explanation have been proposed, and different purposes to use the explanations have been identified. For important highlights of this area see Davis 1980, Swartout 1983, Clancey 1983, Hasling *et al* 1984, Moore & Swartout 1989, Moore & Paris 1991. Many of the ideas we apply to formal-logic proofs can be found in a somewhat less formal setting in these works. Indeed, the main difference seems to be the domain under consideration: we are directly concerned with an explicitly-given proof, rather than with a more comprehensive theory that might incorporate things that look like parts of proofs.)

Finally, when a robotic planning system is used there is a special problem in that the actions might have unforeseen side-effects. Consider the various nightmare scenarios mentioned in the science fiction literature: a robotic planning system is asked to solve the overcrowding situation in some major city, and its proof mechanism reasons that if there were fewer people in the area then there would be less overcrowding. So it makes the city very unpleasant to live in -- perhaps it even explodes a nuclear device in the region. Or consider a medical planning system which is asked to formulate a plan to save a seriously injured person with a very rare blood type. The system reasons that if this person is to be saved there must be at least four quarts of this rare blood available; and so it searches its databanks to find a suitable donor and arranges to have this one donor supply the entire four quarts. This might make the initial plan succeed, but it would have a very unpleasant (to the donor) side effect -- and this might not be a relevant consideration to the computerized planning system. Not only are there these science-fiction nightmares, but also it seems that the trouble would raise its head even more insideously when the side effects are extremely subtle. Arguably, the true difficulty with any AI program is the recognition of subtle unwanted behavior rather than exaggerated unwanted behavior. In any case, though, we would like to be able to inspect the proof which generated the plan.

What is the best way for us to inspect a robotic plan (or any other application of automated theorem provers)? This of course depends upon the particular area under investigation and upon the background of the inspector. But for the types of examples mentioned above we would like to be able to investigate the background proof that is generated. We would like to be able to see whether it really does prove the hidden sublemma; or whether it will result in the destruction of the city, or in the death of the donor; and in other applications we might merely be curious as to how the system came up with *that* answer to our query.

if such a formula is already in the proof, one can "break it down" to get at the component parts of the formula; and if it is not in the proof but (for whatever reason) we desire it to be in the proof, there is a method for introducing the formula.

In any of the various natural deduction systems, these above-mentioned ways of operating with a formula amount to saying: "if formulas Φ_1, Φ_2,.... are already in the proof, then we are entitled to add formula Ψ to the proof". Such operations are called *rules of inference*. More strikingly, and the feature that many think of as defining natural deduction, is another way of operating with formulas, *subproof generation*. This is to put the main portion of the proof on "temporary hold" while attempting to show that some other formula is provable (provable based on what has already gone on in the proof so far). When one starts such a subproof one is allowed to make a temporary assumption (the particular assumption being determined by what formula one is trying to prove), and is allowed to use this assumption together with earlier parts of the proof to try to prove the subproof's goal formula. When this subproof call succeeds, then the formula which was to be proved becomes a part of the outside, main proof; but the portion of the proof which justified our claim that this subgoal is provable is no longer available to the main proof. (The reason for this is that this portion depends upon the assumption which was made, and that assumption is no longer valid.) Since the main conclusion to be proved is itself considered a subproblem, the problem is solved when this main subgoal is proved.

The method discussed in the later sections (and the examples we present) concerns restating natural deduction proofs as a natural language argument. Therefore we need to give enough information about the natural deduction system we employ so that one can follow the natural language explanations. The underlying logic system used here is that of Kalish, Montague, Mar [1980], which has been implemented as a program called THINKER [Pelletier 1982, 1987]. In a natural deduction system there are many Rules of Inference; the retention of the "natural form" requires that there needs to be Rules describing what can be done with each different type of formula. Writing a natural deduction automated theorem proving system is in large measure a matter of organizing the application of all these Rules so as to efficiently generate proofs. There are 25 Rules of Inference used in THINKER for the full first-order logic with identity. To give their flavor but to conserve space, we mention only five of them. Each of these rules has some preconditions in terms of formulas that must already be in the proof and must be *antecedent* (a technical term explained below). These preconditions are stated to the left of the '==>'. When these preconditions are met, the formula to the right of the '==>' *may* be introduced into the proof (along with an *annotation* --a justification in terms of the Rule of Inference employed and the locations [line numbers in the proof] of the preconditions).

RULE	NAME
$(\Phi\rightarrow\Psi),\Phi \Longrightarrow \Psi$	MP (Modus Ponens)
$(\Phi\rightarrow\Psi),\neg\Psi \Longrightarrow \neg\Phi$	MT (Modus Tollens)
$(\forall\alpha)\Phi \Longrightarrow \Phi'$	UI (Universal Instantiation)
$\Phi' \Longrightarrow (\exists\alpha)\Phi$	EG (Existential Generalization)
$\Phi\alpha,\alpha=\beta \Longrightarrow \Phi\beta$	LL (Leibniz's Law)

In the quantifier rules, Φ' is the result of replacing all free occurrences of α in Φ (that is, the ones that are bound by the quantifier phrase in $(\forall\alpha)\Phi$ or $(\exists\alpha)\Phi$) with some term (constant or variable) in such a way that if it is a variable it does not become bound by any other quantifier in Φ'. Furthermore, in the case of EI, the new term must be a variable, and this variable must be entirely new to the proof as thus far constructed.

An argument in general has premises and a conclusion. Premises can be entered into a proof at any time (with the annotation PREM). What makes natural deduction systems distinctive is the idea of *subproofs* -- "smaller" or "simpler" subproblems which, if proved, can be "put together" to constitute a proof of the main problem. Quite obviously, a major factor in natural deduction theorem proving (whether automated or not) is the timely selection of appropriate subproofs to be attempted. In Kalish, Montague, Mar, one indicates that one is about to attempt a subproof of formula Φ by writing "show Φ". (This is called "setting a (sub)goal", and the line in the proof which records this is called a show-line). Since the main conclusion to be proved, C, is itself considered a subproblem, the first line of a proof will be "show C". One is allowed to write "show Φ" for any Φ at any further stage of the proof. Intuitively, one is always "working on" the most recently set subgoal -- although one can of course set a further sub-subgoal (and then "work on" that newly-set sub-subgoal). The formula following the "show" on one of these lines which initiates a subproof is not really a part of the proof proper (until it has been proved). The technical term for this is that it is *not antecedent*, and the effect of being not antecedent is that this formula is unavailable for use in the Rules of Inference.

Setting a subgoal allows one to make an *assumption*. The form of the assumption depends on the form of the formula whose proof is being attempted, and these assumptions can only be made on the line immediately following a show line. (They are annotated ASSUME). There are three types of assumptions allowed:

show $(\Phi\rightarrow\Psi)$	show $\neg\Phi$	show Φ
Φ ASSUME	Φ ASSUME	$\neg\Phi$ ASSUME

The final concept required here is that of *(sub)proof completion* -- the conditions under which a (sub)goal

For the majority of these uses, it is insufficient that a printout of the generated, background proof be available. Perhaps such a document would be useful to some logician or to the creator of the system; but what is desired is that the immediate user be able to know the "reasoning" or "logic behind" the answer. And given that such users are unlikely to be formal logicians with a computer background, a detailed formal proof will be of very limited use. Rather, we want something which captures all the background reasoning involved in the application, but which presents this information to the user at the user's own level of understanding and in a language that the user can understand. The most plausible candidate for a representation of the proof (which is being used by the system) would be one that is presented in natural language. This would obviate the need for any special understanding of the problem on the part of the user. Of course, the user must still be able to follow the presentation of the proof; but we presume that some things would be much more striking when presented in natural language. For example, were the robotic planning system to say that the traffic problem would be lessened if there were fewer automobiles on the roads, and that this could be achieved if there were fewer roads, and that could be achieved by placing two tonnes of dynamite at every intersection, even the most naive user would recognize that this solution was not something that they had in mind when the problem was presented to the system. This is the sort of work alluded to earlier that is carried out in the explanation literature of knowledge-based expert systems, when the goal is to produce believable justifications for system behavior--and often this involves tracing the "reasoning" being used by the system.

It therefore seems that the ability to re-present a proof which was generated by an automated theorem prover into a natural language representation would be useful.Or at least, having this ability would alleviate some of the qualms that people have in trusting automated proof generation. (Of course, it would have to do the re-presentation in an explicit and careful manner. Unless they were convinced of the presence of this feature no one would trust it even if the explanation were carried out in natural language.) We call this ability of a computational system to "explain itself" or to "justify itself" by means of a natural language representation of the internal proof generated and used, a ***natural language back end.*** (It is not a front end, since it is not for use in communicating with the system so as to set goals or tasks, but rather it is used by the system in explaining what it has concluded.)

As we can see from the variety of different uses to which such a back end might be put, there ought to be a variety of different *styles* in which the natural-language presentation of the proof can be given. The system to be described in the following pages is interesting in at least two distinct directions. One is merely the demonstration that (reasonably) easy-to-understand natural language versions of formal logical proofs *can* be mechanically generated from a well-established proof system. The other direction is our theoretical discussion about the various *dimensions of explanation style* that can be relevant to the natural language presentation of a proof. Due to space limitations we will not be able to demonstrate in any detail the first of these two directions. We present only one example at the end. For a series of detailed examples, including the output of the EXPLAIN program applied to a wide variety of proofs, see Edgar [1991]. For the present, more theoretically-oriented paper, our discussion of the dimensions of explanation style can be found in Section 4. We think that these dimensions will show up in *any* natural language back end to a formal logic engine. Each of the dimensions corresponds to certain interests that some users of such a system will have, and therefore each of the dimensions will be required if the system is to be of any use. Of course, some of this discussion has already taken place within the expert systems literature, where there is interest in constructing explanations of this knowledge-based system behavior. The construction of a natural language back end for logic proofs seems to have a small literature: Chester 1976, Felty & Hager 1988, Huang 1989, McDonald 1985,.

The rest of this paper is an investigation into the topic of natural language explanations of proofs that were automatically generated.

3. Natural deduction theorem proving.

Although resolution-based inference is perhaps the industry standard in automated theorem proving, there have always been systems that employed a different format. Even in the late 50's and early 60's there were different systems: the Logic Theorist [Newell *et al* 1957] produced proofs using an axiomatic method, and the actual output of the program would be considered legitimate axiomatic proofs; Wang's systems [Wang 1960a,b] employed a Gentzen-sequent proof strategy [Gentzen 1934/35]; Beth's systems [Beth 1958] employed his semantic tableaux method; and Prawitz's systems [Prawitz 1960, Prawitz *et al* 1960] seem to employ a natural deduction format.

There are many differences between a natural deduction style of logic and a resolution style. One difference that immediately strikes anyone is that natural deduction does not convert formulas to any normal form (e.g., negated-conclusion clause form) but instead works with them in their original, "natural" form. (But this is not such an important difference since resolution-like strategies can and have been developed which also do resolution on "natural form" formulas. And in any case, sequent proof strategies and semantic tableaux methods both operate on "natural form" formulas). More important is the method of developing a proof in natural deduction systems. The fundamental idea is that for each type of ("natural") connective there are two ways to operate with a formula that has it as a main connective:

can be considered to have been proved. The following is a summary of all the ways to complete a subproof. We suppose that the last portion of the proof so far constructed has the form:

show Φ
X_1
...
...
X_n

Then we can change this part to ("complete the subproof", alternatively called "box and cancel"):

* show Φ
| X_1
| ...
| ...
| X_n

if (a) There are no "uncancelled shows" amongst $X_1 \ldots X_n$, and (b) one of the following situations hold: (b_1) Φ occurs "unboxed" amongst $X_1 \ldots X_n$, (b_2) both Θ and $\neg\Theta$ occur "unboxed" amongst $X_1 \ldots X_n$ for some formula Θ, (b3) Φ has the form $(\Psi_1 \rightarrow \Psi_2)$ and Ψ_2 occurs "unboxed" amongst $X_1 \ldots X_n$, or (b_4) Φ has the form $(\forall\alpha)\Phi\alpha$, $\Phi\alpha$ occurs "unboxed" amongst $X_1 \ldots X_n$, and α is not free in any line antecedent to this show line.

When this happens, the lines $X_1 \ldots X_n$ are said to be *boxed* (indicated by the vertical scope line) and are thus no longer antecedent (in the technical sense), while the "show" is said to be *cancelled* (indicated by the *-sign) and the formula Φ is now antecedent (in the technical sense). The boxed lines $X_1 \ldots X_n$ constitute a proof of Φ, but having been used in establishing this goal, they are no longer valid to use (i.e., no longer antecedent in the technical sense) -- the reason being that they may have "depended upon" an assumption made in the course of that subproof and this assumption is no longer in force now that the goal has been proved.[1]

A proof of Φ from a set of premises Γ is a sequence of lines of formulas and scope lines constructed in accordance with the above, in which Φ occurs unboxed and in which there are no uncancelled show lines. We present here four example proofs to show what proofs look like in the Kalish, Montague, Mar system:

1.	*show $(\forall x)(Fx \rightarrow Gx) \rightarrow ((\forall y)Fy \rightarrow (\forall z)Gz)$	
2.	$(\forall x)(Fx \rightarrow Gx)$	ASSUME
3.	*show $(\forall y)Fy \rightarrow (\forall z)Gz$	
4.	$(\forall y)Fy$	ASSUME
5.	*show$(\forall z)Gz$	
6.	$Fz \rightarrow Gz$	2,UI
7.	Fz	4,UI
8.	Gz	6,7MP

1.	* show $(P \vee \neg\neg\neg P)$	
2.	$\neg(P \vee \neg\neg\neg P)$	ASSUME
3.	*show $\neg P$	
4.	P	ASSUME
5.	$(P \vee \neg\neg\neg P)$	4,ADD
6.	$\neg(P \vee \neg\neg\neg P)$	2,R
7.	$\neg\neg\neg P$	3,DN
8.	$(P \vee \neg\neg\neg P)$	7,ADD

1.	* show $(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$	
2.	$(P \rightarrow Q)$	ASSUME
3.	*show $(\neg Q \rightarrow \neg P)$	
4.	$\neg Q$	ASSUME
5.	$\neg P$	2,4 MT

1.	*show $(Faaa \ \& \ a=b) \rightarrow Fbab$	
2.	$Faaa \ \& \ a=b$	ASSUME
3.	$Faaa$	2,S
4.	$a=b$	2,S
5.	$Fbab$	2,3LL

There are many more technical details, both about the Kalish, Montague, Mar system and about THINKER's adaptation of this system, which could be mentioned. However, space limitations prevent us from getting into these exciting details. Interested readers can consult Pelletier [1982, 1987].

4. Explaining proofs: dynamics, representations, levels, and methods.

There are four fundamental dimensions along which different types of natural language explanations of natural deduction proofs (of the sort outlined in the preceding section) might vary. These four dimensions seem to us to be independent of one another, and the different varieties of one dimension can pretty freely co-occur with any variety from any of the other dimensions.

The first dimension is a measure of how much one is to explain the strategy behind the construction of the proof vs. a simple recounting of the proof as it stands. That is, this dimension has a dynamic account of the proof construction process at one end and a static account of the generated proof at the other end. (This is akin to the idea of explaining the "control structure" of

[1] Two things here should be noted. First, when a subgoal is proved, *all* lines after the show line are boxed -- even if the "reason for the cancelling" is not the last line of the proof as thus far constructed. (This is required for soundness.) Secondly, although it is common and expected that the method of proof completion will "match" the type of assumption made -- e.g., assuming the antecedent of a conditional should allow one to cancel because of generating the consequent of the conditional -- this is not required.

a knowledge based expert system's behavior. In that literature, it is strongly argued that we want a "dynamic" explanation of the behavior. See Swartout 1983, Clancey 1983, Moore & Paris 1991, Moore & Swartout 1989). This dimension is particularly salient in natural deduction systems, and especially when it comes to setting subgoals. At the dynamic side of this dimension we would expect the natural language explanation to say such things as "At this stage of the proof we set the subgoal on line 6 because we notice that line 4 has such-and-such property and if we could prove line 6 then we could do so-and-so to lines 4 and 6. Furthermore it looks like we probably *can* prove line 6 because of blah-blah...." At the static side of this dimension we would see such explanations as "At line 6 such-and-so subgoal was set and was proved by blah-blah. Once line 6 was proved it was used with line 4 to do so-and-so."

The thrust of our program is to the static side of this dimension. The task that we set ourselves was to examine the proof which was produced by THINKER, and to give an explanation of it. Our natural language generator had no private information about THINKER's internal proof development strategies, and therefore was not in a position to try to give a dynamic account of proof construction. To add this ability, we would have to somehow be able to read off of the final proof the actual heuristic THINKER used at a particular point; and then it would need to have some insight into THINKER's author's mind in order to see why this or that heuristic was thought to be useful at the particular place in the proof where it was employed. As the writers mentioned earlier in this section are at pains to point out, this is probably the most important aspect of an explanation. It is, however, one that was not implemented here.

A second dimension has to do with the representation of the lowest level items being explained. In the sample proofs of the last section, the item on any line was an uninterpreted formula -- a string of symbols. And an explanation *could* simply refer to these formulas: " 'q' then would follow from our earlier 'p→q' and 'p' ." We call this end of the second dimension "the abstract representation". However, the connectives in these formulas are usually taken to represent certain well-known English phrases: 'if--then', 'or', 'and', 'for every', and the like. It might sometimes be useful to state the various formulas on lines of a proof as asserting these kinds of relationships amongst (uninterpreted) predicates. In such an explanation we might find: "The 'q' on line 15 is generated because it is the consequent of the conditional on line 10, and this conditional's antecedent is on line 4." Or perhaps we would see "Line 7 says that something has the property of being both an F and a G, so on line 12 we will introduce an arbitrary thing -- call it r_0 -- and say that it is both F and G." We call this position along the present dimension "the representation of the connectives." Finally along this dimension, we note that quite a few of the applications mentioned in Section 1 above actually inject a particular meaning into the abstract predicates that are found in the formulas of a proof. That is, although the formal proof is carried out with just abstract symbols, in the specific application these predicate symbols have a definite meaning or representation....they stand for such things as "clear traffic off of---" or "drop__on---" and the like. And it might sometimes be of particular interest --for example when having a robotic planner explain its proof-- to have an explanation of the proof-where the connectives and the predicates are fully represented as natural language. We call this position on the second dimension "the representation of predicates". At this position on the second dimension we might have presented to us such proof parts as: "if there is mass destruction there will be less traffic; and if an atomic bomb explodes over the downtown there will be mass destruction. I therefore can achieve the top-level goal of having less traffic by implementing this plan: Drop atomic bomb on Vancouver."

The third dimension has to do with the *level* at which an explanation is to take place. One level, the lowest level, would be to parrot the exact proof: "line 43 follows from lines 39 and 27 by an application of Modus Ponens". But one can imagine higher levels of explanation in which not every detail in the proof will be explained. For example, in most proofs there are a few major subgoals which need to be solved in order to justify or explain the top level goal. So a very high level of explanation would just refer to them, and would not further explain how they were generated. Instead, they are just asserted to be "low-level provable" and won't be explained. More generally, one can imagine setting a variable that marks the "level of explanation" and then to have the natural language generator explain only those lines which are "above" this level. (As a detail, it might turn out that although certain lines are themselves above this level, they are used only to justify a line which is below this level. And therefore they also would not be mentioned, even though they are above the level.) Such a higher-level explanation would give the human an overview of how the proof is done without "completely" explaining it. But this raises the question of what we should use as a measure of the "level". It is here that one of the advantages of natural deduction theorem proving comes to the fore; for we can get a very good approximation to what humans would mean by this concept when we measure depth of embedded subgoals. The main goal is the least embedded; the justifications for this are (usually) at the first level of embedding, and the justifications for these are (usually) at the next level of embedding, etc. A proof which is explained by having the level set to a few embeddings truly gives a high-level overview of the proof.....an "executive summary", as it were.

The fourth and final dimension along which we can vary explanations of proofs is what we call the *style* of explanation. And by this we mean a dimension which at one end is "top-down" and at the other end is "bottom-up". The middle is taken with various "mixed explanations". To explain a proof bottom-up one would search the proof for lines which do not depend on other lines of the proof--premises or assumptions, for example. (But recall that if the third dimension has the level of explanation set higher, then those lines to be explained will be ones that are considered not to depend on other lines.) These are just asserted to be facts or assumptions, in the explanation. Further lines of the proof are mentioned only when all the lines they depend upon have been explained. So this style develops by explaining simple lines first and then those lines derivable from simple lines, etc. Top-down explanations state the main thing to be proved and then to state the formulas upon which it depends. And this explanation is carried out recursively until the bottom-level formulas (assumptions, premises, etc.) are reached. Having done this for the formulas upon which the top level goal depends, we then explain how these formulas generate the top level goal.

Each of these styles of explanation has advantages and has disadvantages. One advantage to the bottom-up style is that the easy-to-understand lines are dealt with first -- these being the lines which have no further explanation. And after these are explained, the lines that are "next hardest" are mentioned: those that depend on the just-explained lines. In other words, the proof explanation goes from simple to complex. But a disadvantage to this style of explanation is that the goal (or a subgoal) which is being explained is not mentioned at the beginning, and so the reader of the explanation never knows *why* something is being cited or *where* the proof is going. To help with this problem, our bottom-up method first states the main goal of the proof as an introductory sentence at the beginning of the explanation, for otherwise this main goal would not appear in the explanation until the final sentence.

One advantage of the top-down explanation is that the goal (or subgoal) to be explained is stated at the beginning of its explanation; and this leads to a much easier-to-understand, "goal-driven" explanation. Along these same lines another reason that these explanations are easier to follow is that, in the initial portion of the explanation, the subgoals that the main goal requires are stated immediately after the main goal. And for whatever reason, people seem to like this sort of explanation. But a disadvantage is that the simplest lines of the proof are not explained until the end....and so the hearer is expected to be able to understand why certain high-level goals are set without knowing what their bases are nor whether they are reasonable to expect a proof of.

Two technical details ought to be mentioned here. First, in either of these methods it is possible that there will be redundancy in explanation. For, in a natural deduction proof, a given line might have more than one use -- it might be used to justify more than one line. But we would not wish to burden the listener with two explanations of the same line. Therefore, we need to keep track of whether a line has already been explained, and just remind the hearer of the previous explanation of that line. Secondly, there is a special problem about the order of introduction of free variables by existentially quantified formulas. The restriction in a proof is that such variables be new to the proof when they are introduced, and it is only after this introduction that other formulas (such as universally quantified ones) should be instantiated to these variables. The point is that variables introduced from existentially quantified formulas have to be ***arbitrary***, and that means they cannot be already in the proof. But the explanation of a proof does not proceed in the same order as the proof itself did. If we are doing a top-down explanation then we are doing some sort of depth-first look at the proof--and this does not respect the order of lines introduced in the proof. Therefore, the English explanation might have the effect of mentioning these "arbitrary variables" before it is actually explained how they got into the proof. And this in fact has the effect of making the proof seem to be invalid when it is really valid. We therefore introduced a check to see whether any variable being appealed to in the explanation came from an existential instantiation which has not yet been mentioned. If so, then at that time there is added a side explanation of this introduction.

As mentioned before, it is possible to combine the top-down and bottom-up styles. This combination would explain part of the proof in a top-down fashion and other parts in a bottom-up fashion. Our implementation of this combined method is to require a stipulation by the user for a "major style" and a "minor style". The major style is the type of explanation used to state the high level goals and subgoals of the proof, while the minor style is used to explain the lower levels. There is a variable which can be set so as to state when to change from one style to the other (as a function of the depth of embedding of the subgoal level). Any show line above that level will be explained by the main style and anything below that level will be explained by the minor style. (But keep in mind that there is also the dimension of "level", and that a proof need not be explained "all the way to the bottom".) Thus something like "major: top-down; minor: bottom-up; depth 3" is a style. And this being so, it can be a major or minor style. This allows for (what is admittedly only a curiosity) the change back and forth of top-down and bottom-up styles as one gets deeper and deeper into the proof. Combined explanations try to retain the good points of both the top-down and the bottom-up styles. If the top level is explained bottom-up, and then there is a change to top-down, then the subgoals will be like simple lines in a

proof and will quickly lead to the primary goal. When top-down is used as the main method, the combined explanation states the primary goal at the beginning, so that the reader has a sense of the direction the proof is to take. For small subproofs the bottom-up method is advantageous since the simple lines will be explained first and will quickly lead to the subgoal; but since it is a short subproof, the reader is not lost because the goal comes quickly.

5. Summary and Examples.

We have explained some uses of automated theorem proving and tried to motivate (from the point of such uses) the idea that there are times at which it would be desirable to have a natural language explanation of these proofs. We then explained one type of proof theory, natural deduction, and showed what sort of considerations are relevant to explaining proofs in this theory. We outlined four dimensions along which explanations of proofs might vary (independently of one another), and we picked out two variants along dimension 1 (the dynamic/static dimension), three variants along dimension 2 (the representations), and an arbitrarily large number of variants along both of dimensions 3 (levels of explanation depth) and dimension 4 (styles of explanation). If we hold dimension 3 to just two levels ("explain everything" vs. "cut off explanation at some point") and hold dimension 4 to four styles (top-down, bottom-up, top-down/bottom-up, bottom-up/top-down), then there are 96 different kinds of explanation under discussion here. As mentioned, our system does not do the dynamic explanation, and so we are able to give examples of only 48 of the possible explanation types. But space considerations limit us even further. What follows are a few examples of explanations. The proof as produced by THINKER is displayed and then some different kinds of explanations are shown.

References:[2]

Allen, J. (1991) "Natural Language, Knowledge Representation and Logical Form" in M. Bates & R. Weischedel *Challenges in Natural Language Processing* (Cambridge: CUP).

Beth, F. (1958) "On Machines Which Prove Theorems". Reprinted in Siekmann & Wrightson 1983, pp. 79-90.

Boden, M. (1977) *Artificial Intelligence and Natural Man* (NY: Basic Books)

Bundy, A. (1983) *The Computer Modelling of Mathematical Reasoning* (London: Academic Press).

Chester, D. (1976) "The Translation of Formal Proofs into English" *Artificial Intelligence* 7: 261-278.

Clancey, W. (1983) "The Epistemology of a Rule-based Expert System: A Framework for Explanation" *Artificial Intelligence* 20: 215-251.

Davis, R. (1980) "Metarules: Reasoning About Control" *Artificial Intelligence* 15:179-222.

Dijkstra, E. & C. Scholten (1990) *Predicate Calculus and Program Semantics* (Berlin: Springer-Verlag)

Edgar, A. (1991) "Natural Language Generation of Natural Deduction Proofs". MSc Thesis, Univ. Alberta, Dept. Computing Science.

Felty,A. & G.Hager (1988) "Explaining Model Logic Proofs" *Proc. IEEE Conf. on Systems, Man, and Cybernetics* Vol I, pp. 177-180.

Fikes, R., P.Hart, N.Nilsson (1972) "Learning and Executing Generalized Robot Plans" *Artificial Intelligence* 3:251-288.

Fikes, R. & N.Nilsson (1971) "STRIPS: A New Approach to Application of Theorem Proving in Problem Solving" *Artificial Intelligence* 2:189-208.

Gentzen, G. (1934/35) "Investigations into Logical Deduction". Translation printed in M. Szabo *The Collected Papers of Gerhard Gentzen* (Amsterdam: North-Holland) 1969, pp.68-131.

Green, C. (1969a) "The Application of Theorem Proving to Question-Answering Systems" *IJCAI-69* pp.219-237.

Green, C. (1969b) "Theorem Proving by Resolution as a Basis for Question Answer Systems" in B. Meltzer & D. Michie (eds.) *Machine Intelligence 4* (Edinburgh: Univ. Edinburgh Press) pp.183-205.

Grice, H. (1975) "Logic and Conversation" in P.Cole & J.Morgan (eds) *Syntax and Semantics III: Speech Acts* (NY: Academic Press) pp.41-58.

Guard,J., F. Oglesby, J.Bennett, L.Settle (1969) "Semi-Automated Mathematics" *JACM* 16:49-62.

Gumb, R. (1989) *Programming Logics* (NY: John Wiley).

Hasling, D., W. Clancey, & G. Rennels (1984) "Strategic Explanations for a Diagnostic Consultation System" in M.J.Coombs (ed) *Developments in Expert Systems* (London: Academic Press).

Hirst, G. (1987) *Semantic Interpretation and the Resolution of Ambiguity* (Cambridge: CUP).

Hovy, E. (1988) "Planning Coherent Multisentential Text" *ACL-88* 163-169.

Huang,J. (1989) "A Human Oriented Proof Presentation Model" SEKI SR-89-11, Kaiserslautern Univ.

Johnson-Laird, P., & R. Byrne (1991) *Deduction* (Hillsdale: Lawrence Erlbaum).

[2]This research was supported in part by the Canadian NSERCgrant OPG5525. The author is a member of the Institute for Robotics and Intelligent Systems and wishes to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, NSERC, and the participation of PRECARN Assiciates Inc. Correspondence concerning this work should be directed to Pelletier.

Kalish, D., R.Montague, G.Mar (1980) *Logic: Techniques of Formal Reasoning* 2nd ed. (NY: Hartcourt, Brace, Javonovich).

Lehnert, W. (1977) *The Process of Question Answering* (Yale Univ. Comp.Sci. Tech Report #88).

Manna,Z., & R.Wallinger (1977) *Studies in Automatic Programming Logic* (NY: Academic Press)

McDonald, D. (1985) "Natural Language Generation as a Computational Problem" in M. Brady & R. Berwick *Computational Models of Discourse* (Cambridge: CUP).

Moore, J. & C. Paris (1991) "Requirements for an Expert System Explanation Facility" *Computational Intelligence* 7: 367-370.

Moore, J. & W.Swartout (1989) "A Reactive Approach to Explanation" *IJCAI-89*, 1504-1510.

Newell, A., J. Shaw, & H. Simon (1957) "Empirical Explorations with the Logic Theory Machine: A Case Study in Heuristics". Reprinted in Siekmann & Wrightson 1983, pp. 49-73.

Pelletier, F.J. (1982) *Completely Non-Clausal, Completely Heuristic Driven, Automated Theorem Proving*. Univ. Alberta, Dept. Computing Science, Tech Report 82-7.

Pelletier, F.J. (1987) *Further Developments in THINKER, an Automated Theorem Prover.* Australia National Univ., Project on Automated Reasoning, Tech Report 87-16.

Prawitz,D. (1960) "An Improved Proof Procedure". Reprinted: Siekmann & Wrightson 1983 pp.162-199.

Prawitz,D., H.Prawitz, & N.Voghera (1960) "A Mechanical Proof Procedure and its Realization in an Electronic Computer". Reprinted in Siekmann & Wrightson 1983 pp. 202-228.

Reiter, R. (1987) "A Theory of Diagnosis from First Principles" *Artificial Intelligence* 32: 57-95

Rips, L. (1984) "Reasoning as a Central Intellectual Ability" in R. Sternberg (ed) *Advances in the Study of Human Intelligence* (Hillsdale: L. Erlbaum).

Sacerdoti, E. (1974) "Planning in a Hierarchy of Abstraction Spaces" *Artificial Intelligence* 5:115-136.

Schubert,L. & C-H.Hwang (1990) "An Episodic Knowledge Representation for Narrative Texts". Report TR 345, Dept. Computer Science, Univ. Rochester.

Shapiro, E. (1981) "An Algorithm that Infers Theories from Facts" *IJCAI-81* 446-451.

Siekmann, J. & G.Wrightson (1983) *Automation of Reasoning: Classical Papers on Computational Logic. Vol. I: 1957-1966.* (Berlin: Springer-Verlag).

Sterling, L. & A.Bundy (1982) "Metalevel Inference and Program Verification" in D.Loveland *Proceedings of CADe 6* (Berlin: Springer-Verlag) 144-150.

Swartout, W. (1983) "XPLAIN: A System for Creating and Explaining Expert Consulting Programs" *Artificial Intelligence* 21: 285-325.

Wang, H. (1960a) "Towards Mechanical Mathematics". Reprinted in Siekmann & Wrightson 1983, pp. 244-264.

Wang, H. (1960b) "Proving Theorems by Pattern Recognition". Reprinted in Siekmann & Wrightson 1983, pp. 229-243.

Output of THINKER, piped to Natural Language Explainer. The attached are unmodified output of these programs.

INPUT:
Goal: $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$

OUTPUT:

1	*show* $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$	7,9,CONTRA
2	| $(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$	ASSUME
3	| $(B(r_1)\&(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$	2,EI
4	| $(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1))$	3,S
5	| $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$	4,UI
6	| $(\neg P(r_1,r_1)\rightarrow P(r_1,r_1))$	5,BC
7	| *show* $P(r_1,r_1)$	6,8,MP
8	| | $\neg P(r_1,r_1)$	ASSUME
9	| $\neg P(r_1,r_1)$	5,7,EQ

THE EXPLANATION IN BOTTOM-UP FASHION

The goal of the proof is to prove $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$. We assume $(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$. To get $(B(r_1)\&(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$ we substitute the new constant r_1 for y_1 in $(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$. We set the variable x_1 to equal r_1 in $(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1))$. Assume $\neg P(r_1,r_1)$. $(\neg P(r_1,r_1)\rightarrow P(r_1,r_1))$ was proven earlier. And using the last two formulas, we can get $P(r_1,r_1)$ since the antecedent of the conditional is true. We previously proved $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$. Using the last two formulas, since one side is true and the equality is true we can derive the consequent of $\neg P(r_1,r_1)$. We have generated a contradiction $(P(r_1,r_1))$ and $(\neg P(r_1,r_1))$ and therefore we can conclude that $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$.

THE EXPLANATION IN TOP-DOWN FASHION

To prove that $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$ is true we want to generate a contradiction. We can get $P(r_1,r_1)$ if we can prove both $\neg P(r_1,r_1)$, and $(\neg P(r_1,r_1)\rightarrow P(r_1,r_1))$. To show $(B(r_1)\&(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$ is true we have to prove $(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$ and then we can instantiate the variable y_1 with the new constant r_1. Assume $(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$. To get $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$ we must prove that $(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1))$ is true and then substitute r_1 for x_1. As we saw earlier $(B(r_1)\&(\forall x_1)(P(r_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$ is true. Assume $\neg P(r_1,r_1)$. We can now get $P(r_1,r_1)$ because we assumed $\neg P(r_1,r_1)$, and proved $(\neg P(r_1,r_1)\rightarrow P(r_1,r_1))$ and so we can get the conclusion of the conditional statement. In order to prove the statement $\neg P(r_1,r_1)$ we will show that $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$ and $P(r_1,r_1)$ are both true. We previously proved $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$. $P(r_1,r_1)$ was proven earlier. Because the biconditional, $(P(r_1,r_1)\leftrightarrow\neg P(r_1,r_1))$, was proven and one side is true $(P(r_1,r_1))$ the other side $(\neg P(r_1,r_1))$ must also be true. Since we have generated a contradiction $(P(r_1,r_1))$ and $(\neg P(r_1,r_1))$ we can conclude $\neg(\exists y_1)(B(y_1)\&(\forall x_1)(P(y_1,x_1)\leftrightarrow\neg P(x_1,x_1)))$.

EXPLANATION AT THE LEVEL OF THE CONNECTIVES

The goal of the proof is to prove (there is no constant y_1 that gives us (both $B(y_1)$ and (for every constant x_1, we get ($P(y_1,x_1)$ follows from ($P(x_1,x_1)$ is false) and vice versa)))). Now, let's suppose (there is a constant y_1 that gives us ($B(y_1)$ and (for each constant x_1, (if we have $P(y_1,x_1)$ we get ($P(x_1,x_1)$ is not true) and vice versa)))). We replace y_1 by a new constant r_1 in (both $B(r_1)$ and (for any constant x_1, (we get $P(r_1,x_1)$, if and only if we have, ($P(x_1,x_1)$ is false)))). We get (if we have $P(r_1,r_1)$ we get ($P(r_1,r_1)$ is false) and vice versa) by substituting r_1 for x_1 in the statement (for any constant x_1, (we get, $P(r_1,x_1)$, if and only if we have, ($P(x_1,x_1)$ is not true))). Now, let's suppose ($P(r_1,r_1)$ is false). As we saw earlier (if ($P(r_1,r_1)$ is false), then $P(r_1,r_1)$) is true. And using the last two formulas, since both the antecedent and the conditional are true we can derive the consequent of $P(r_1,r_1)$. Earlier in the proof we proved ($P(r_1,r_1)$ follows from ($P(r_1,r_1)$ is not true) and vice versa). From the previous two, we can get ($P(r_1,r_1)$ is not true) since one side of the equality is true. We have generated a contradiction ($P(r_1,r_1)$) and (($P(r_1,r_1)$ is not true)) and therefore we can conclude that (there is no constant y_1 that gives us (both $B(y_1)$ and (for any constant x_1, (we get, $P(y_1,x_1)$, if and only if we have, ($P(x_1,x_1)$ is false))))).

EXPLANATION AT THE LEVEL OF THE PREDICATES

The main goal is to prove it is false that somebody is a barber and he shaves all and only those people who don't shave themselves. Now, let's suppose someone is a barber and he shaves all and only those people who don't shave themselves. Let's select somebody arbitrarily and call him Fred. Fred is a barber and he shaves all and only those people who don't shave themselves would follow from the fact that somebody does this and Fred is arbitrary. We can get that Fred shaves himself if and only if Fred doesn't shave himself by particularizing the earlier claim that Fred shaves all and only those people who don't shave themselves. Now, let's suppose Fred doesn't shave himself. It was proved before that Fred is shaved by himself follows from Fred doesn't shave himself. And from these two we derive that Fred shaves himself because the antecedent being true implies that the consequent is also true. We proved earlier in the proof that Fred shaves himself if and only if Fred doesn't shave himself. It follows from these two since one side is true and the equality is true we can derive the consequent of Fred doesn't shave himself. We have generated a contradiction (Fred is shaved by himself) and (Fred doesn't shave himself) and therefore we can conclude that it is not true that somebody is a barber and he shaves all and only those people who don't shave themselves.

A Flow Chart for EXPLAIN

Set parameters (from command line)
If predicate explanation READ lexical file
Remove unused lines from the linked list of the proof
If bottom-up explanation call pick_rule_BU
If top-down explanation call pick_rule_TD

pick_rule_BU

If the current line is the first line state it as the main goal
If current line's level is the current switching level call pick_rule_TD with current line and set the current switching level to the next level where switching is to take place
If current line's level is the cut-off level explain it as provable
If justification 1 exists call pick_rule_BU with justif.1
If justification 2 exists call pick_rule_BU with justif. 2
Call line to explain current line's inference rule
Set current line as explained at time X

pick_rule_TD

If current line's level is the current switching level call pick_rule_BU with current line and set the current switching level to the next level where switching is to take place
If current line's level is the cut-off level explain it as provable
Call line to explain current line's inference rule
Set current line as explained at time X

BU inference rule functions

Produce the linking phrase (depending on when the justifications were explained)
Produce the explanations of the formulas to be used
Pick an explanation phrase for the inference rule

TD inference rule functions

If the line has been explained previously restate the line and return
Produce the explanations of the formulas to be used
State the current line as the goal
State the subgoal to be proved
Call pick_rule_TD with the justification lines
(optional) Explain how the justifications are combined to get the goal

Formula Explanation

If to the level of the inference rules return as is
If to the level of the connectives
- Call fcn to explain the main connective (if it exists)
- Explain the left subformula (if it exists)
- Explain the right subformula (if it exists)
- Pick a connective phrase to combine left and right explanations
- If no main connective return the predicate as is

If to the level of the predicates
- Call fcn to explain the main connective (if it exists)
- If Universal of Existential quantifier
  - Make an entry in the constant list for the variable with a suitable phrase such as "everybody" or "someone"
  - Explain the right subformula
- Explain the left subformula (if it exists)
- Pick a connective phrase to combine left and right explanations
- If no connective call predicate explanation

Predicate explanation

If one argument
- Should the negative phrase be used? (set by ¬ connective explanation)
- Can a pronoun be used?
- Is the phrase to be plural or singular?

If 2 arguments
- Get the object phrase
- Get the subject phrase
- Check if self referral (i.e.; Fred shaved himself)
- Positive or negative?
- Any pronouns?
- Singular or plural?
- Active or passive? (randomly chosen)