An Explanation Facility for a Grammar Writing System

Loong Cheong TONG
Institute of Systems Science,
National University of Singapore,
Kent Ridge,
SINGAPORE 0511.
E-mail: ISSTLC@NUSVM.BITNET

Abstract

Explanation has become a standard feature in many expert systems today. Adapting from this work, a study was made to determine the types of explanation required in a grammar writing system and to investigate design and implementation issues. The first version of this explanation facility is based on a derivational history of the inferencing process, and although no supplementary knowledge is used, this explanation facility is able to furnish answers to the traditional *why*, *how* and *what* type of queries, and even the *what-if* (simulation) query. The explanation is also enhanced through the use of special files containing canned-text for describing grammar rules and variables.

1. Introduction

Explanation for expert systems has been studied very extensively, and has become a standard feature in these systems. Its objectives are to enable users to understand how conclusions are reached, to be convinced that these conclusions are reasonable, and to debug the knowledge base and possibly problem solving behavior as well. Viewed in a more general context, research into explanation is an essential component of the study into the symbiosis between man and machine, supported by the empirical fact that most knowledge-based systems are intended to assist human endeavor and are almost never intended to be autonomous agents.

Grammar writing systems (GWS) used in Natural Language Processing (NLP) work have been compared to expert systems (Boitet and Gerber, 1984) by equating the knowledge base to the grammar, and the problem-solving activity to the transformation performed on the text representation structure. Existing GWSs do not provide any explanation because their usage is usually confined to expert users who can understand low-level programs or rule traces, and moreover, the inferencing process for NLP applications is normally carried out in batch mode. If users do interact with the system, it is usually for the purpose of resolving ambiguities (for example), which is obviously quite different from explanation. The presence of explanation is one of the main reason why expert systems have found a substantial degree of success with users, so perhaps NLP systems could also benefit from such a service, especially in gaining end-user acceptance.

In this investigation, we are not studying the nature of explanation per se, but rather, starting from an existing system, determine what explanations can be provided and how. The system under consideration is the GWS associated with a machine translation environment known as JEMAH (Tong, 1988; 1989).

Previous studies on explanation (see, for example, Swartout, 1984, and Chandrasekharan, 1989) have shown that providing effective explanation frequently requires *supplementary* knowledge in addition to the existing knowledge base. For example, the metamorphosis from MYCIN to NEOMYCIN (Clancey, 1983) requires the addition of meta-rules to explicitly represent the diagnostic strategy and the relationship between rules. The research described in this paper will ascertain the level and degree of explanation which can be provided without resorting to the use of supplementary knowledge, more or less along the same lines as Wick and Slagle (1989), who studied the type of explanation facility that can be provided based on current expert system methodologies where supplementary knowledge is not available.

Besides the knowledge content required for such explanation, the other side issues related

to explanation like presentation, user modelling based on goals and background knowledge, and dialogue structure are not included in this paper. This first prototype of an explanation facility for the JEMAH system uses simple menus and pre-defined dialogs.

2. Explanation in Expert Systems

There are 3 major types of explanation which have been studied for expert systems: (a) explaining the (dynamic) inferencing on a specific input data set, (b) explaining the (static) knowledge base itself, and (c) explaining the control strategy. The first type of explanation, based on the original work in MYCIN (Scott et al., 1977), has been adopted by almost all commercial expert system shells. Such trace-based explanation will answer queries on *why* (is a question being asked?), *how* (is this conclusion reached?) and *what* (is the current variable/rule?).

The second type of explanation is exemplified by the Xplain system (Swartout, 1983), whose implementation-level knowledge base is compiled from an explicitly-represented deep knowledge model. Xplain can explain its knowledge base by referring to this deep knowledge, as well as knowledge about the compiling process itself. For grammar writing systems, such a scheme corresponds to the static and dynamic grammars in Vauquois and Chappuy (1985) and the meta and (low-level compiled) object grammars in Bogureav et al. (1988). Most commercial expert system shells only explain its implementation level knowledge base through the use of canned-texts associated with rules and variables.

NEOMYCIN (Clancey, 1983) generalizes the diagnostic problem by explicitly representing the diagnostic tasks, whose strategy can then be explained for any one data set, and this implements the third type of explanation mentioned above. Another example is the generalized task-based approach adopted by Chandrasekaran (1989).

This paper will concentrate mainly on trace-based explanation, since the other two obviously require supplementary knowledge. The execution trace of the processing of a NL text contains a wealth of information, but this crude data must first be transformed into a more well-ordered structure and then rearranged into a form suitable for explanation purpose. Hence, explanation is viewed here as an *information refining* process (analogizing on an oil refinery).

3. Explanation in a Grammar Writing System

Any explanation system must first address the question of who its users are, and very frequently, a user classification based on user expertise, which covers the spectrum from novice to expert, is used for this purpose. Knowledge of a user's class will influence the amount of explanation to be provided, as well as its form and content. In the JEMAH grammar writing system, we have identified 3 groups of users: the translator, the linguist and the grammar writer (corresponding to the end-user, the expert and the knowledge engineer in an expert system). The grammar writer may use the explanation facility to debug the grammar related to the rules and the control strategy, the linguist to study the various linguistic phenomena which are associated with the translation process, and the translator to better understand the translation output, and hence, better able to edit it.

For the grammar writer and the linguist, the detailed trace of execution provided by the JEMAH system contains superfluous unstructured information. One way of controlling the level of detail and the form of information is through the use of abstraction hierarchies (Clancey, 1983), and one method of implementation is to attach numeric markers to each rule to denote its level of importance and complexity. A similar scheme has been adopted by the JEMAH explanation facility.

The following examples will illustrate the types of explanations encountered in grammar writing systems.

(a) *"Why is the rule XYZ applied?"*
"Why is the noun phrase ... attached to the verb?"

This is explaining the inferencing process and answers can be found directly in the execution trace.

(b) *"What other rules affect noun phrases?"*
"What is the function of the rule XYZ?"

This is explaining the grammar itself and answers can be generated using canned-text and abstraction hierarchies.

(c) *"Why is the rule PQR not applied?"*

This is explaining the control strategy.

(d) *"What if the rule ABC is applied first?"*

This is the simulation type of explanation.

4. Design and Implementation

In most commercial expert system shells, explanations are provided *during* the inferencing process. For grammar writing systems in machine translation, this is performed at the end, after storing all relevant derivational (or inferencing) history. Therefore, the most important design decision is to determine how to represent this derivational history, and what to store in it. In the JEMAH system, the derivational history consists of a sequence of snapshots of the tree structure taken after each transformation. Backtracking through this derivational history will provide all necessary information on how the translation was carried out. JEMAH also provides the *what-if* explanation (simulation) through its restart capability; i.e. JEMAH can backtrack to a certain point in the translation process, reset specific values, and then restart the translation process from that point.

Explaining the control strategy is obviously much more difficult, depending on whether the control information is explicit or implicit. In ROBRA (Boitet et al., 1978), where this strategy is explicit (user-defined), any explanation will involve explaining the flow of control within its transformational subsystem. If implicit (as in JEMAH), then explanation has to be hard-coded into inference engine, as done in SOPHIE (Brown et al., 1982).

The derivational history of a translation carried out by the JEMAH system is represented as a sequence of **events**. Each event is a single transformation on the tree structure resulting from the successful application of a grammar rule. The information content of each event consists of:

TREE	resultant tree structure,
RULE-NAME	applied rule,
PIVOT-NODE	pivot node,
DELETED-NODES	list of deleted nodes,
NEW-NODES	list of created nodes,
ACTIVE-NODES	list of modified nodes and their annotations.

The full derivational history of a translation from source text to target text is stored in 3 sequences of events (see Fig. 1), corresponding to the analysis (*AS-TRACE*), transfer (*TS-TRACE*) and generation phase (*GS-TRACE*), as well as the final event (*GM-TRACE*). Each sequence has a start event (E_0) and a final event (E_n). A start event differs from the other events in that it has null values for RULE-NAME, PIVOT-NODE, and DELETED-NODES, and all nodes of TREE are included in NEW-NODES and in ACTIVE-NODES; this represents the initial conditions for each event in a phase.

The computations involved in the 3 dictionary processes of morphological analysis, lexical analysis and morphological generation are not recorded in the derivational history. Each of these phases is considered as a single transformation; for example, lexical analysis causes the transformation from E_n of *AS-TRACE* to E_0 of *TS-TRACE*. Since there is no dictionary process between structural transfer and structural generation, the event E_n of *TS-TRACE* is the same as the event E_0 of *GS-TRACE*.

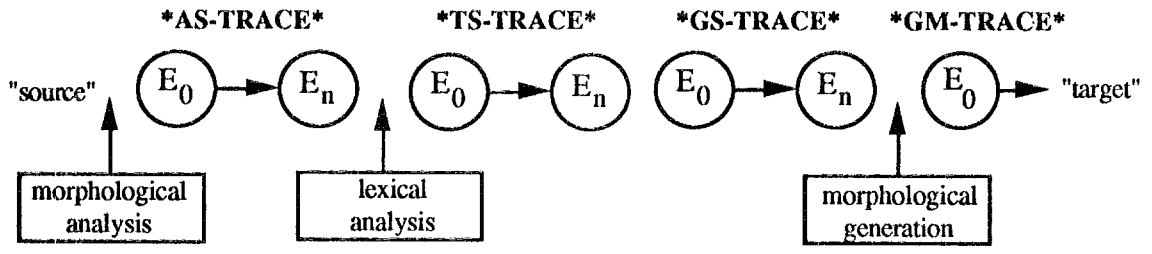

Figure 1. Events in the Derivational History of the Translation Process

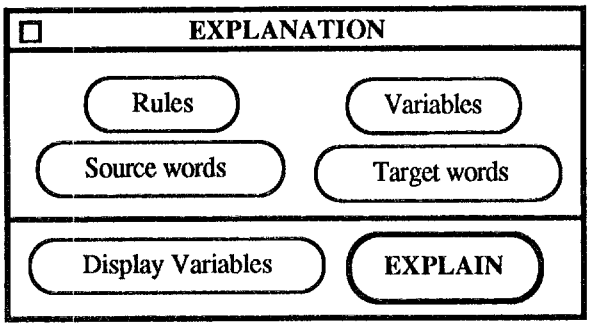

Figure 2. Explanation Facility User Interface

Fig. 2 shows the simplified user interface dialog that controls the explanation facility of JEMAH. The 4 buttons on the top half of the dialog represent the 4 query elements: Rule, Variable, Source Text, Target Text. Selecting (clicking on) any of these 4 buttons will automatically pop-up a list of all rules, variables, source words or target words respectively. User-selected items will then be used to determine the content and form of the explanation provided when the user click on the EXPLAIN button. The user can also reduce the amount of details by selecting only the interested variables for display by using the "Display Variables" button. Once items in any of these 4 query elements have been selected by the user, the system will extract the relevant events from the derivational history and then generate an output text based on a pre-defined template structure.

All explanation output texts will involve describing rules and variables. Instead of using a single word (rule name or variable name) to describe each of them, special supplementary files consisting of canned-text have been created to provide a more meaningful explanation. For example, the grammar in each translation phase is associated with a supplementary file describing all rules in that grammar, and each record has the following fields: rule-name, author, date (of last modification), rule-type, keywords, description. The rule-type field is to identify a rule's level of importance and its relevance to a particular group of users, while the keywords field is used for indexed retrieval as well as grammar partitioning. The description field is divided into 2 sub-fields, an abstract and a main text, to provide 2 levels of explanation details.

A similar canned-text approach is used for variables, each of which is described with the following fields: variable-name, keywords, type (morphological, syntactic, semantic, etc.), description (2 levels), rule-name (rules that either test or assign values to that variable).

A sample explanation of the translation of the word "activate" in the sentence "This information is the input to activate the inventory control application." (translated into Malay as "Maklumat ini adalah input untuk menggerakkan {menghidupkan giat} penerapan kawalan inventori.") is shown below.

"menggerakkan" is translated from "activate",
analyzed morphologically as lexical unit ACTIVATE
with information (CAT V SUBV VB VL1 N TENSE PRES NUM PLU VEND 2).

In structural analysis, it is processed as follows:
Rule ELEVATE-VCL constructs a VCL from a verbal.
with changes: (SF GOV)
Rule TO-VCL constructs an infinitive clause from TO + VCL.
with changes: (K PARTCL SVL I SLOCK IMP)
Rule PCL-NP absorbs a following NP into a PARTCL.
with changes: (AVL1 N SEM1 ABST)
Rule VCL-PCLCIRC absorbs the right PARTCL as circumstantial into the clause.
with changes: (SF CIRC)

In lexical transfer, it is translated to target GERAK
with alternative translations (MENGHIDUPKAN GIAT).

In structural transfer, it is processed as follows:
Rule ASP-TENSE->ASPEK maps the English TENSE, VOICE and ASPect to Malay ASPEK.
with changes: (ASPEK NORMAL)

In structural generation, it is processed as follows:
Rule CL-ARG1=ACT positions ARG1 to the right of the governor of an active clause.
with changes: (RSV CAUSE)

5. Supplementary Knowledge

The JEMAH system explanation facility is very tightly coupled to the execution trace, and hence, to the grammar knowledge base. Advanced work with explanation systems have strongly indicated the need for supplementary knowledge to improve the explanation provided, and lately, some experts have even argued for the complete decoupling of knowledge used by the explanation system from the knowledge used by the expert system. Future work on the explanation facility of JEMAH will include supplementary knowledge to improve the quality and range of explanations provided.

In describing NEOMYCIN, Clancey and Letsinger (1984, pg 380) stated that:

> "To explain diagnosis, it is useful to have psychological model of problem solving. In particular, we need to incorporate into our model the medical knowledge and strategies an expert uses for initial problem formulation."

In the grammar writing system context, there is a similar need for a linguistic knowledge model, as well as a parsing/generation strategy model. Normally, the linguistic knowledge refers to knowledge about the specific grammar under consideration, and does not include knowledge about linguistics in general, although this possibility should not be ruled out in future work. The pre-compilation process in the JEMAH system (see Tong, 1989) extracts information about rules' relationships, and this can be considered as a kind of meta-knowledge about the grammar. Furthermore, since this meta-knowledge is used in JEMAH to optimize the control strategy, it seems fair to assume that it can also contribute towards explaining the parsing/generation strategy. This single concept is obviously insufficient to provide comprehensive explanation to the user, but does serve as an example of the kind of supplementary knowledge required.

A second feature which is already available within the JEMAH system is that of grammar partitioning. Rules are grouped according to certain linguistic properties, and this can be used as a source of supplementary knowledge in providing explanations. A more elaborate method would be to construct a classification scheme in the form of a hierarchical tree structure, like the refinement structure in Xplain (see pg. 392 in Swartout, 1984).

The variables and their values used in a grammar obviously play a very important role in explanation; in NEOMYCIN, for example, relationships between variables and new variables are created solely for the purpose of explanation. In JEMAH, this type of meta-knowledge about variables may include, for example, information on mutual exclusion (between morphosyntactic class and word category), equivalent set of values (valencies 1 and 2) and hierarchical relationship (subcategory SUBA of category A).

6. Conclusion

This paper has described the explanation facility for a grammar writing system in a machine translation environment, including its design and implementation which is based on previous work on explanation for expert systems. The current system makes use of the derivational history of the translation process, and future work will concentrate on enhancing it with supplementary knowledge.

References

Boitet, C.; Guillaume, P.; and Quezel-Ambrunaz, M. 1978. Manipulation d'arborescences et parallelisme: le systeme ROBRA. Proc. COLING-78, Bergen.

Boitet, C. and R. Gerber. 1984. Expert systems and other new techniques in MT. Proc. COLING-84, Stanford University, Calif., 468-471.

Boguraev B., J. Carroll, T. Briscoe and C. Grover. 1988. Software support for practical grammar development. Proc. COLING-88, Budapest: 54-58.

Brown, J.S., R.R. Burton and J. de Kleer. 1982. Pedagogical, natural language and knowledge engineering techniques in Sophie I, II and III. in Intelligent Tutoring Systems, D. Sleeman and J.S. Brown, eds., Academic Press, London, UK: 227-282.

Chandrasekaran, B., M.C. Tanner and J.R. Josephson. 1989. Explaining control strategies in problem solving. *IEEE Expert*, 4(1): 9-24.

Clancey, W.RJ. 1983. The epistemology of a rule-based expert system - a framework for explanation. *Artificial Intelligence*, May: 215-251.

Clancey, W.J. and R. Letsinger. 1984. NEOMYCIN: reconfiguring a rule-based expert system for application to teaching. In W.J. Clancey and E.H. Shortliffe (eds.), **Readings in Medical Artificial Intelligence**. Addison-Wesley, 361-381.

Scott, A.C., W.J. Clancey, R. Davis and E.H.Shortliffe. 1977. Explanation capabilities of knowledge-based production systems. *American Journal of Computational Linguistics* Microfiche 62.

Swartout, W.R. 1983. Xplain: a system for creating and explaining expert consulting programs. *Artificial Intelligence*, Sept: 285-325.

Swartout, W.R. 1984. Explaining and justifying expert consulting programs. In W.J. Clancey and E.H. Shortliffe (eds.), **Readings in Medical Artificial Intelligence**. Addison-Wesley, 382-399.

Tong, L.C. 1989. A Data-Driven Control Strategy for Grammar Writing Systems. *Machine Translation*, 4(4) Dec 1989: 177-193.

Tong, L.C. 1988. The JEMAH System Reference Manual. CAT Project Tech. Report, Universiti Sains Malaysia.

Vauquois, B. and Chappuy, S. 1985. Static grammars. A formalism for the description of linguistics models. International Conf. on Theoretical and Methodological Issues in Machine Translation of Natural Language, Colgate University, 14-16.

Wick, M.R. and J.R. Slagle. 1989. An explanation facility for today's expert systems. *IEEE Expert*, **4**(1): 26-36.