THE XCALIBUR PROJECT:

A Natural Language Interface To Expert Systems

Jaime G. Carbonell, W. Mark Boggs, Michael L. Mauldin
Computer Science Department, Carnegie-Mellon University,
Pittsburgh, PA 15213

PeterG. Anick
Digital Equipment Corporation
Hudson, MA 01749

Abstract

The inevitable proliferation of expert systems underscores the need for robust, friendly interfaces requiring minimal user training. The objective of the XCALIBUR project is to meet this need by providing natural comprehension and generation in the context of a focused mixed-initiative dialog. The XCALIBUR architecture is discussed, including its three central components (parser, generator and information manager), its methods of handling ellipsis and imperfect input, and its relation to the underlying expert system.

1. Introduction

As expert systems mature into software products for a wide spectrum of users, the availability of flexible user interfaces becomes critically important. Natural language interfaces become particularly appropriate for the following reasons:

- © No extra cognitive demands are placed on the user, allowing him or her to focus on the problem at hand.
- The domain specificity of most expert systems provides strong constraints on the vocabulary, actions and concepts that may be expressed in natural language. This renders an otherwise open-ended task tractable.
- Recent advances in multi-strategy parsing, fail-soft recovery techniques, ellipsis handling and the like, help to make flexible natural language communication a practical reality [2, 6].

This paper describes the XCALIBUR project, whoso objective is to provide flexible natural language access (comprehension and generation) to the XSEL expert system [11]. XSEL, the Digital Equipment Corporation's automated salesman's assistant, advises on selection of appropriate VAX components and produces a sales order for automatic configuration by the R1 system [10]. Part of the XSEL task is to provide the user with information about DEC components, subsuming the data-base query task. However, unlike a pure data base query system, an expert system interface must also interpret imperatives requesting actions, understand assertions of new information, and carry out task-oriented dialogs (such as those discussed by Grosz [5]). XCALIBUR, in particular, deals with commands to modify an order, as well as information requests about its present task or its data base of VAX component parts. XCALIBUR will also process clarificational dialogs when the underlying expert system (i.e. XSEL) requires additional information or advice, as illustrated in the sample dialog below:

>What is the largest 11730 fixed disk under $40,000?

The rp07 aa is a 516 MB fixed pack. disk that costs $38,000.

>Add /wo rp07-aa disks to my order.

Line item 1 added: (2 rp07-aa)

>Add a printer with graphics capability

fixed or changeable font?

>fixed font

lines per minute?

>make it at least 200, upper/lowercase.

Ok. Line item 2 added: (1 lxy11 -sy)

>Tell me about the lxyl 1

The lxyl 1 is a 240 LPM line printer with plotting capability

Except for the system-driven clarification interchange, which is beyond XCALIBUR's present capabilities, this dialog is indicative of the present state of our system.

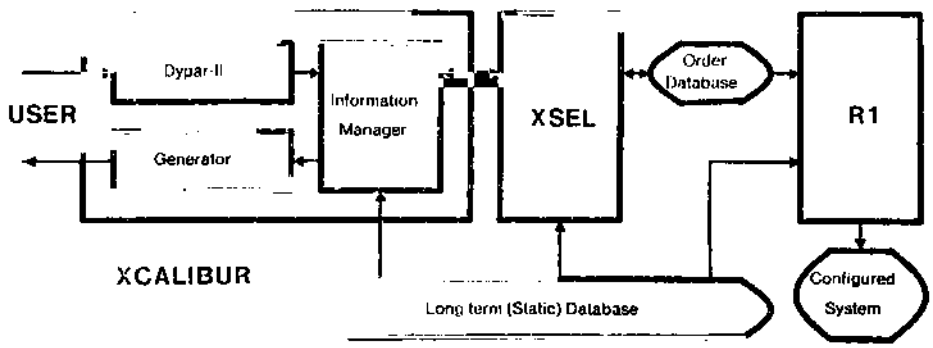

Figure 1: Overview of XCALIBUR

The rest of this paper discusses the major modules of XCALIBUR: the DYPAR II parsing system, the information manager and the natural language generator. Since the major contribution of XCALIBUR thus far is perhaps the integration of diverse techniques into a working system, we focus more on the engineering aspects of our project. We do mention, however, some advances we have not reported elsewhere, including expectation-based error-recovery, case-frame ellipsis resolution, and focused natural language generation. Figure 1 provides a schematic view of XCALIBUR, and the following sections present some details of the internal workings of XCALIBUR as it processes a sample information request.

2.The DYPAR-II Module

The original DYPAR natural language parser [2] was built to test the feasibility of incorporating multiple parsing strategies into a uniform computational framework. DYPAR-II is an extension of the DYPAR parser that incorporates recursive case-frame instantiation, in addition to the semantic-grammar, pattern matching and global transformation strategies present in the original system. Through multi-strategy parsing, a measure of flexibility, robustness and conciseness can be achieved that is not exhibited in more rigid single-strategy systems.[1] In this section we mention some of the highlights of the DYPAR II system.

2.1. The Recursive Case Frame Method

A case frame consists of a header pattern and a set of marked or positional cases with (semantic and syntactic) expectations for possible case fillers. Cases and case frames correspond to semantic units in the problem domain. Although different classes of case frames, such as *action-verb case frames* and *complex-noun case frames,* play different syntactic roles, all are recognized by the same interpreter.

Efficiency and robustness is achieved by combining bottom up recognition of semantic patterns (corresponding to case headers, for instance), with top down expectation-driven instantiation once a case-frame has been recognized. Moreover, discourse level expectations and constraints can easily be integrated into the expectation and recognition mechanisms.

The parsing method used by DYPAR-II is summarized below:[2]

1. If dialog expectations exist at the case-frame level, activate those case frames and skip to step 3. (Then, if step 3 fails — i.e., no expectations are confirmed — return to step 2).
2. Apply a fast scanning unanchored match of the input (or input fragment) against a set of header patterns, which have been cross-referenced and precompiled into a discrimination network for efficiency. Each pattern is associated with one or more case frames, and serves to stait a complete parse attempt in that frame's context. Failure of this step may indicate that the utterance is an ellipsed phrase.
3. If steps 1 or 2 succeeded, use the Case frame (and syntactic transformations) in a top down fashion to generate syntactic and semantic expectations to recognize the rest of the input. Here is where the system recurses into embedded patterns or case frames and where expectation-driven error recovery comes into play [1].
4. Finally, the completed parse is mapped into a form suitable for the information manager. This second mapping gives the system a measure of portability to different domains.

2.2. Expectation-Based Error Recovery

In the absence of a globally consistent parse of the input utterance, an expectation-based error recovery process is triggered, similar to the methods proposed by Granger [4] and Carbonell [I]. Errors can range from ungrammaticalities and interjections to unknown words and misspellings. Our scheme calls for attempting corrections that satisfy pending expectations — and these can be ranked according to the strength of the expectation and the likelihood of occurrence for that particular error. Thus far, spelling correction is our only fully operational recovery strategy. A word that can be corrected to fulfill an expectation receives much higher priority than a context-free spelling correction. For instance, the correction *prot -> port* in "Add a dual prot disk" is made because a disk descriptor is expected.[3] Furthermore, semantically anomalous but "correctly spelled" words can be considered as candidates for correction (as in "Copy the flies in my directory to the backup tape"), but these corrections are more risky.

3. The Information Handler

The information handler mediates the communications between the parser, the underlying expert system, and the natural language generator. Currently, the underlying system is a stripped-down version of the real XSEL, including access to a relational database of component descriptions, and the capability to create and modify a dynamic database of ordered "line-items" (quantily-partname pairs), but lacking the expertise to suggest components to the user or check an order for configurability.

3.1. Internal Representation

When XSEL is ready to accept input, the information handler is passed a message indicating the case frame or class of case frames expected as a response. For our example, assume that a command/query is expected, the parser is notified, and the user enters

>What is the price of the 2 largest dual port fixed media disks?

The parser returns:

```
[QUERY (OBJECT (SELECT (disk
                        (ports (VALUE (2)))
                        (disk-pack-type (VALUE (fixed)))
                (OPERATION (SORT
                             (TYPE (*descending))
                             (ATTR (size))
                             (NUMBER (2)))
                (PROJECT (price)) )
        (INFO-SOURCE (*default)))))]
```

This representation embeds the case frame information extracted during the parse within primitives understood by the information-handling routines and XSEL data adapter. The SELECT field describes the selection criteria for a set of database items; the OPERATION field describes operations to be performed on the set, such as ordering and truncating, the PROJECT field contains the attribute(s) of principal interest; and the INFO-SOURCE field contains the database from which the objects are to be selected. In the example, the query does not explicitly name an INFO-SOURCE, which could be the component database, the current set of line-items, or a set of disks brought into focus by the preceding dialog.

The reader is referred to [7] and [6] for a discussion of the multi-strategy approach. DYPAR-II is a step on the way to the as yet unfinished MULTIPAR system, which carries the multi strategy approach to its logical conclusion — applying problem solving techniques to select dynamically the appropriate knowledge and strategy to apply as the parse unfolds.

2 The computational mechanisms and knowledge structures of DYPAR-II are reported in greater depth in [7] and [3].

Using a small dictionary, the TOPS20 SPFLL program generated 13 possible corrections to the word "prot" Clearly expectations reduce search, especially in the presence of compound errors or potentially ambiguous input.

The information handler is responsible for filling in defaults, modifying, and adding fields to the parser output to satisfy the needs of the expert system routines that execute the commands. It contains tables for mapping ambiguous attribute names (eg. "size", "speed") into the field names appropriate to the selected object case frames, and for applying default database matching functions when these are not explicitly mentioned in the input. For example, in most contexts, "300 MB disk" means a disk with "greater than or equal to" 300 MB, not strictly "equal to". A "large" disk refers to ample memory capacity in the context of a functional component specification, but to large physical dimensions during site planning. We plan to extend the knowledge sources available to the information handler to support anaphora resolution and the more subtle pragmatic decisions that interfaces to expert systems require. We are also in the process of augmenting the internal representation language with recursion to handle joins/composition of attributes, as in "the cost of the controller for the disk."

4. Case-frame ellipsis

The XCALIBUR parser handles ellipsis at the case-frame level. Its coverage appears to be a superset of the LIFER/LADDER system [8, 9] and the PLANES ellipsis module [13]. Although it handles most of the ellipsed utterances we encountered, it is not meant to be a general linguistic solution to the ellipsis phenomenon.

4.1. Examples

The following examples illustrate the types of sentence fragments our case-frame algorithm can parse. For brevity, assume that each sentence fragment occurs immediately following the initial query

INITIAL QUERY: "What is the price of the three largest single port fixed media disks?"

"Speed?"

"Two smallest?"

"How about the price of the two smallest?"

"smallest with dual port"

"Speed with two ports?"

"Disks with two ports."

In the representative examples above, punctuation is of no help, and syntax alone is of limited utility. For instance, the last three phrases are syntactically similar (indeed, the last two are indistinguishable), but each requires that a different substitution be made on the preceding query. All three substitute the number of ports in the original SELECT field, but the first substitutes "ascending" for "descending" in the OPERATION field, the second substitutes "speed" for "price" in the PROJECT field, and the third merely repeats the case header of the SELECT field filler.

4.2. The Ellipsis Resolution Methods

». Ellipsis is resolved differently in the presence or absence of strong discourse expectations. In the former case, the discourse expectation rules are tested first, and, if they fail to resolve the sentence fragment, the contextual substitution rules are tried. If there are no strong discourse expectations, the contextual substitution rules are invoked directly.

Exemplary discourse expectation rule:

IF: The system generated a query for confirmation or disconfIrmation of a proposed value of a filler of a case in a case frame in focus,
THEN: EXPECT one or more of the following:
1) A confirmation or disconfIrmation pattern.
2) A different but semantically permissible filler of the case frame 1n question (optionally naming the attribute or providing the case marker).
3) A comparative or evaluative pattern.
4) A query for possible fillers or constraints on possible fillors of the case in question. [If this expectation is confirmed, a sub-dialog is entered, where previously focused entities romain in focus.]

The following dialog fragment, presented without further commentary, illustrates how these expectations come into play in a focused dialog:

>Add a line printer with graphics capabilities.

Is 150 lines per minute acceptable?

>No, 320 is better		Expectations 1, 2 & 3
(or)	*other options for the speed?*	Expectation 4
(or)	*Too slow, try 300 or faster*	Expectations 2 & 3

The contextual substitution rules exploit the semantic representation of queries and commands discussed in the previous section. The scope of these rules, however, is limited to the last user interaction of appropriate type in the dialog focus, as illustrated in the following example:

Exemplary Contextual Substitution Rulo:

IF: An attribute name (or conjoined list of attribute names) is prosent without any corresponding filler or case header, and the attributo is a semantically permissible descriptor of the case frame 1n the SELECT field or the last query in focus,
THEN: Substitute the new attribute name for the old filler of the PROJECT field of the last quory.

For example, this rule resolves the ellipsis in the following utterances:

>What is the size of the 3 largest single port fixed media disks?

>And the price and speed?

XCALIBUR currently has eight rules similar to the one above, and we have found several additional ones to extend the coverage of ellipsed queries and commands (see [3] for a more extensive discussion). It is significant to note that a small set of general rules exploiting the case frame structures covers most instances of commonly occurring ellipsis, including all the examples presented earlier in this section.

5. The Natural Language Generator

Generation proceeds in three phases: (1) a request from the parser or information handler is converted into a conceptual dependency graph, (2) the verb is selected and the slots of the CD graph are mapped into a case frame, and (3) the case frame is rendered into English. Only the first stage of this process is domain dependent. Stage two is performed by a case frame builder similar to that in Goldman's BABEL system [12]. The third stage includes dialogue modeling of objects already mentioned, and pronominal references are built for noun phrases that have already been said. Throughout these stages, focus information is used to guide the generation process.

5.1. Why Natural Language Output

XCALIBUR chooses sentential output over a tabular form when a table would be degenerate. For example, suppose the user requests the price of all 120 volt graphics terminals costing less than 3200 dollars. There is only such terminal, the vt105-ma, so XCALIBUR prints:

The vt105-ma is a 120 volt terminal with graphics capability that costs 3100 dollars.

5.2. An Example

Consider the following user input:

> "What is the price of the largest dual port fixed media disks?"

After parsing and database look up, the information handler passes the following request to the generator:

```
(render-result
     (class (disk))
     (actor-list (rp07 ba))
     (projection-attr (price))
     (focus (ports disk-pack-type
            number-of-megabytes)))
```

The domain adaptor then constructs the following conceptual dependency graph. CD macros such as CD-BE or CD-CONNECT are used to represent concepts that are not CD primitives. The following CD graph is generated:

```
(cd-be
  (actor (rpO7-ba (ref (def))))
  (object
    (disk (focus (ports media number-of-megabytes price))
          (rel (cd-cost (actor (disk))
               (object (dollar (count (43140))
                       (focus (count))))))))
  (tense (pres)))
```

To prevent noun phrases from accreting large adjective lists, some information is placed into relative clauses. The generator uses focus information to determine the placement of various slot fillers. Hers the price of the disk has been moved into a relative clause for emphasis. Currently there are three levels of focus: out of focus, in focus, and explicitly requested by user.

After the request structure has been converted into a CD graph, the case frame builder selects a verb using rules similar to Goldman's BABEL generator, and maps the CD slot fillers into individual cases. The sentence structure is then generated using a recursive ATM grammar. Finally the individual noun phrases are filled out, and any relative clauses are generated by recursive calls to the generator.

> *The rp07-ba is a dual port fixed pack 516 MB disk that costs $43.140 dollars.*

5.3. Pronominalization

Simple pronominalization is performed by the noun phrase builder. As each object is rendered into a noun phrase, a database of all object/noun phrase mappings is checked, and the shortest unique form of the noun phrase is used if the information has already been rendered in the current session. Thus a vt100-aa might be represented by any cf the following noun phrases:

- the soft copy terminal with optional advanced video.
- the soft copy terminal.
- the terminal.
- it.

6. Concluding Remark

XCALIBUR is still in its early stages of development, but it already surpasses the capabilities of most if not all existing natural language systems as a flexible expert systems interface. Future developments are governed by the dual (and complementary) objectives of providing the requisite functionality for a robust expert system interface, and investigating focused task-oriented mixed initiative dialogs with users of differing abilities and interests.

7. References

1. Carbonell, J. G., "Towards a Self-Extending Parser," *Proceedings of the 17th Meeting of the Association for Computational Linguistics,* 1979, pp. 3-7.

2. Carbonell, J. G. and Hayes, P. J., "Dynamic Strategy Selection in Flexible Parsing," *Proceedings of the 19th Meeting of the Association for Computational Linguistics,* 1981.

3. Carbonell, J. G., Boygs, W. M., Mauldin, M. L. and Anick, P. G., "XCALIBUR Progress Report '/ 1: Overview of the Natural Language Interface," Tech. report, Carnegie-Mellon University, Computer Science Department, 1983.

4. Granger, R., "FOUL-UP: A Program that Figures Out Meanings of Words from Context," *Proceedings of IJCAI-77,* 1977, pp. 172-178.

5. Grosz, B. J., *The Representation and Use of Focus in Dialogue Understanding,* PhD dissertation, University of California at Berkeley, I977, SRI Tech. Note 151.

6. Hayes. P. J., and Carbonell, J. G., "Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence,* August 1981, pp. 432-439.

7. Hayes. P. J. and Carbonell, J. G., "Multi-Strategy Parsing and it Role in Robust Man-Machine Communication," Tech. report CMU-CS-01-118, Carnegie-Mellon University, Computer Science Department, May 1981.

8. Hendrix, G. G., Sacerdoti, E. D. and Slocum, J., "Developing a Natural Language Interface to Complex Data," Tech. report Artificial Intelligence Center., SRI International, 1976.

9. Hendrix, G. G., "The LIFER Manual: A guide to Building Practical Natural Language Interfaces," Tech. report Tech. note 138, SRI, 1977.

10. McDermott, J., "R1: A Rule-Based Configurer of Computer Systems," Tech. report, Carnegie-Mellon University, Computer Science Department, 1980.

11. McDermott, J., "XSEL: A Computer Salesperson's Assistant," in *Machine Intelligence 10,* Hayes, J., Michie, D. and Pao, Y-H., eds., Chichester UK: Ellis Horwood Ltd., 1982", pp. 325-337.

12. Schank, R. C, *Conceptual Information Processing,* Amsterdam: North-Holland, 1975.

13. Waltz, D. L. and Goodman, A. B., "Writing a Natural Language Data Base System," *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* 1977, pp. 144-150.