USING A NATURAL-ARTIFICIAL HYBRID LANGUAGE FOR DATABASE ACCESS

Teruaki AIZAWA and Nobuko HATADA

NHK Technical Research Laboratories

1-10-11, Kinuta, Setagaya, Tokyo 157, Japan

In this paper we propose a natural-artificial hybrid language for database access. The global construction of a sentence in this language is highly schematic, but allows expressions in the chosen language such as Japanese or English. Its artificial language part, SML, is closely related to our newly introduced data model, called scaled lattice. Adopting Japanese as its natural language part, we implemented a Japanese-SML hybrid language processing system for our compact database system SCLAMS, whose database consists of scaled lattices. The main features of this implementation are (1) a small lexicon and limited grammar, and (2) an almost free form in writing Kana Japanese.

1. Introduction

Various query languages for database access have been developed, among which unambiguous artificial ones are better adapted to computers. For man, on the other hand, it would be more convenient to communicate with computers in a natural language. The possibility of man-machine communication in a natural language has been one of the main concerns in the field of artificial intelligence, and considerable results have been obtained specifically in research into natural language access to a database.[1-5] These results, however, seem to be too complex and inflexible for practical application to general-purpose database systems.

We will propose in this paper a "natural-artificial hybrid" language for database access. The global construction of a sentence in this language is highly schematic but allows expressions in the chosen language such as Japanese or English. A Japanese version of this language has been implemented for our compact database system SCLAMS[6,7] (SCaled LAttice Manipulation System). The main features of this implementation are:

(1) Use of only a small lexicon and limited grammar so that they are quite easy to implement, and

(2) Allowance of almost free form in writing Kana Japanese.

Feature (1), which will be achieved also when using other languages like English, French, and so on, is one of the most noticeable merits obtained by using such a natural-artificial hybrid language for database access.

We begin with an explanation of our basic logical unit of data, Scaled Lattice, or S.L. for short, since the proposed language is closely related to this unit.

2. SML:Scaled lattice manipulation language

2.1 Scaled lattice as a data model

What the normalization theory in the relational data model tells us can be stated very loosely as "one fact in one place".[8] The concept of Scaled Lattice, or S.L. for short, also goes along this direction.

Roughly speaking an S.L. is a multi-dimensional table, and is defined as a collection of data of one species arranged at multi-dimensional lattice points corresponding to the combinations of attribute values. Fig. 1 shows a graphical image of S.L. which represents population data by year, prefecture, and sex.

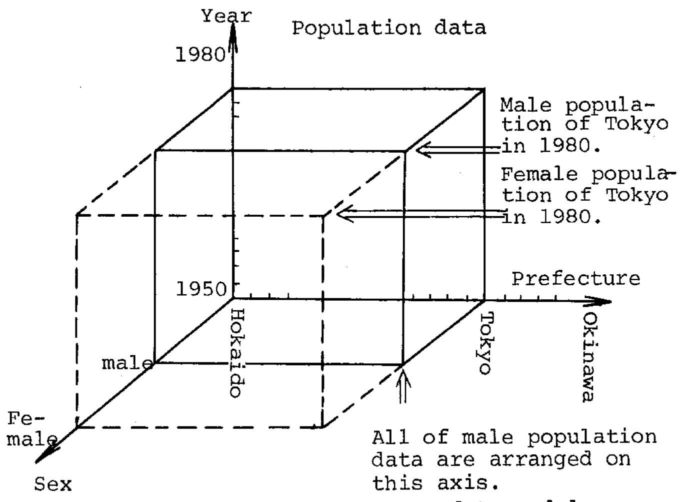

Year
Population data
1980
1950
Male population of Tokyo in 1980.
Female population of Tokyo in 1980.
Prefecture
Hokaido
Tokyo
Okinawa
male
Female
Sex
All of male population data are arranged on this axis.

Fig. 1 Graphical image of S.L. data model

This is an example of three dimensional S.L's, which can be furthermore regarded as a mapping or a function with three variables in the mathematical sense. Let S1, S2, and S3 be finite sets such as

$$S1 = \{ 1950, 1951, \ldots , 1980\},$$

$$S2 = \{ \text{Tokyo, Osaka, Nagoya, } \ldots\},$$

and

$$S3 = \{ \text{male, female} \}.$$

Also let A be an appropriate set having enough elements to represent values of population. Then the above S.L. can be naturally regarded as a mapping:

$$F : S1 \times S2 \times S3 \rightarrow A, \qquad (1)$$

which associates any triple (x, y, z) of attribute values in S1 x S2 x S3 with the corresponding population value F(x, y, z). Thus, for example,

F (1980, Tokyo, male)

denotes the male population of Tokyo in 1980.

Generally an S.L. is a mapping F of the direct product of finite sets S1, ..., Sn into an appropriate set A denoted by

$$F : S1 \times \ldots \times Sn \rightarrow A. \qquad (2)$$

These sets S1, ..., Sn and their elements will be sometimes called root words and leaf words respectively.

The following are the advantages of this data model:

(1) Data contained in an S.L. can be displayed exactly in the two-dimensional table form, which is visually very understandable.

(2) In order to display data in table form, it is necessary to cut out an appropriate two-dimensional cross section from the S.L., or more precisely to select two appropriate scales on which the table is constructed, and, at the same time, to fix the remaining scales at some attribute values. This is nothing but a retrieval operation. Cutting out such a section is very easy, which means that certain retrieval operations are also easy.

(3) Since an S.L. is regarded as a mapping, precise and powerful notations concerning "sets and mappings" are directly applicable for manipulation of the S.L. data.

2.2 Brief outline of SCLAMS

We have implemented a compact database system SCLAMS (Scaled lattice manipulation system), whose database consists of S.L.'s.[6,7] SCLAMS has the following three major modes:

(1) Storage mode: Storage of data as a set of S.L.'s editing from any file into the database.

(2) Retrieval mode: Selection of one or more suitable S.L.'s from the database.

(3) Manipulation mode: Data extraction from the above S.L.'s and some operation on the data.

Thus, a retrieval operation according to a user's query is divided into two modes: Retrieval and Manipulation. Retrieval mode is similar to the document retrieval system, and Manipulation mode to the database system, in a narrow sense, regarding each S.L. as a small file. The main concern of our design of SCLAMS was to combine effectively these two modes, in other words, to integrate the function of document retrieval systems and that of database systems.

2.3 Manipulation of scaled lattices by SML

In this paper we will focus our attention exclusively on Manipulation mode of SCLAMS. The major function of this mode is to manipulate S.L.'s in a variety of ways such as extraction of data satisfying specified conditions, join of more than two S.L.'s data, elementary calculations for extracted data, etc. These operations are done through a query language for end users, named as SML (Scaled lattice Manipulation Language).

We now show a few examples to illustrate some aspects of SML. Let F1 and F2 be two S.L.'s, i.e. two mappings such as

$$F1 : S1 \times S2 \times S3 \rightarrow A1, \text{ and} \qquad (3)$$

$$F2 : S1 \times S2 \rightarrow A2, \qquad (4)$$

where S1 = Year scale

$$= \{ 1950, 1951, \ldots, 1980\}, \qquad (5)$$

S2 = Prefecture scale

$= \{$ Tokyo, Osaka, Nagoya,.. $\}$, (6)

S3 = Sex scale

$= \{$ male, female $\}$, (7)

A1 = Set of population values,

A2 = Set of numbers of TV subscribers.

These S.L.'s may be considered as an output of Retrieval mode.

Each example below consists of an informal query and the corresponding formal one expressed by SML. Notice that the SML expressions contain the mathematical notations to describe sets and mappings.

Example 1. List the male population of Tokyo in 1980.

```
LIST A;

  A = F1(1980, Tokyo, male);
```

Example 2. List names and the number of prefectures in which the male population in 1980 is greater than one million.

```
LIST B, C;

  B = <X:F1(1980, X, male)>
      1,000,000>;

  C = COUNT (B);
```

In this example B is defined as the set of prefecture X's with the population value F1(1920, X, male) > 1,000,000, and C as COUNT of B, where COUNT is one of aggregate functions prepared in SCLAMS.

Example 3. List numbers of TV subscribers in 1980 of prefectures in which the female population in 1975 is less than one million.

```
LIST NUM;

  NUM = F2(1980, P);

  P = <X:F1(1975, X, female)
      <1,000,000>;
```

In this example two S.L.'s F1 and F2 are related by a common scale S2.

General format of a query or a sentence by SML is shown in Fig. 2.

```
LIST a1, a2, ..., am;
  b1 = expression 1;
  b2 = expression 2;
   .
   .
   .
  bn = expression n;
```

Fig. 2 General format of a query by SML

In this format each of variables a1,..., am is equal to one of those b1, ..., bn; and the order of b1, ..., bn is arbitrary. The types of expressions can be classified into the following six categories:

1) Numeral or literal constants; e.g.

   1980, Tokyo, male, etc.

2) Aggregate function values; e.g.

   COUNT (x), SUM (y), etc.

3) S.L.'s values; e.g.

   F(x1, ..., xn), etc.

4) Set operation formulas; e.g.

   x & y, x|y, x-y, etc.

5) Set definition formulas; e.g.

   <3, 5, 7, 11>, <Tokyo, Nagoya, Osaka>,
   <xi:F(x1,...,xi,...,xn)<y>, etc.

6) Abbreviate notations for elements of a scale, i.e. leaf words; e.g.

   S.1, S.11-20, etc.

   The latter, for example, represents from 11th to 20th elements of a scale S.

It would be easily seen, from the above explanation, that a query by SML is expressed basically as a set of "non-procedural" local queries, and thus the query as a whole has also of non-procedural nature.

3. Hybridization of SML with a natural language

3.1 An illustrative example

We have assured that our query language SML is sufficiently flexible and has strong expressive power, specifically for those who are familiar with mathematical notations concerning sets and mappings. However, we can also say that SML is less convenient than a natural language which seems to be best suited for casual users. We therefore tried to hybridize SML with a natural language like English, Japanese, etc., believing that such a natural-artificial hybrid language should be one of the milestones to a realization of database systems wholly accessible via unrestricted natural languages.

The next example, closely related to Example 2 in the last section, will show us how to hybridize SML with a natural language, say English.

<u>Example 4</u>. List names and the number of prefectures in which the male population in 1980 is less than the female population of Tokyo in 1970.

Now we consider the following two types of expressions for this query.

<u>Type I</u> (Original formal expression by SML)

```
LIST A, B;

  A = <X:F1(1980, X, male) < C >;

  B =  COUNT (A);

  C =  F1(1970, Tokyo, female);
```

<u>Type II</u> (Extended new expression)

```
LIST A, B;

  A = Names of prefectures in which
      the male population in 1980 is
      less than C;

  B = Number of elements of A;

  C = Value of the female population
      of Tokyo in 1970;
```

The features of Type II expressions are:

(1) The global construction is quite similar to that of Type I expression, but it allows us to write phrases in the chosen natural language for definitions of variables such as A, B, and C. (If necessary, some of the variables may retain the original formal definitions.)

(2) Notice that variable symbols such as A and C can be embedded in ordinary English phrases, so that the original query expressed as a complex sentence is divided into some simple queries. This contributes to readability of queries both for man and computer.

3.2 <u>Features of a Japanese-SML version</u>

We have implemented a "Japanese-SML" hybrid language processing system, as an extension of SCLAMS. The major design goal was to be practical rather than just ambitious. The processing system, which will be called <u>Translator</u>, is essentially a translator of a Japanese phrase into the corresponding SML expression, or in the above terminology, of a Type II expression into its Type I equivalent. The main process of Translator is shown in Fig. 3.

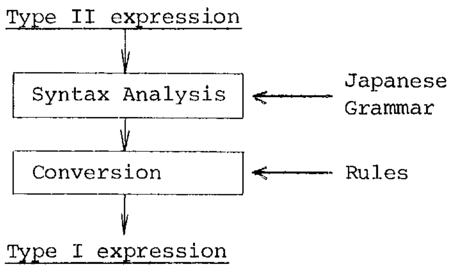


Fig. 3 Process of Translator

Some considerations in achieving practicability of the implemented system are:

(1) In our implementation a Japanese sentence or phrase can be written as a string of only Kana characters, in which case it is desirable, for convenience, to guarantee freedom from segmentation as much as possible. Our system indeed allows the free writing of a Kana sentence, as long as the leaf words (the elements of scales) cause no confusion with the reserved words in the lexicon.

(2) It is desirable to keep the grammar as compact as possible to save storage space and processing time. This was done by restricting forms of possible Type II expressions.

4. <u>Translation of Japanese into SML</u>

4.1 <u>Micro-grammar for Japanese</u>

As mentioned in Section 2.3, the set of all Type I expressions are classified into six categories 1)∿6). Then the possible Type II expressions, which our Translator can accept, are restricted to those corresponding to the categories 2), 3), and a part of 5), i.e. the so-called implicit set definitions. It should be noticed that expressions belonging to the other categories are neatly expressed rather by Type I forms.

We now show the lexicon and the grammatical rules prescribing these Type II expressions.

Lexical items and their categories. There are 12 categories of lexical items.

1) Num : Numbers, e.g.

 12, 165.3, -0.137, etc.

2) Naux: Auxiliary numbers, e.g.

 hyaku, byaku, pyaku, sen, man (hundred, thousand, million), etc.

3) Agg : Names of aggregate functions, e.g.

 kosu, souwa, saidai, heikin (count, sum, maximum, average), etc.

4) eq : Equality words or copulas, e.g.

 no, dearu, deatte, nihitoshii, nihitoshiku (is equal to), etc.

5) comp 1: Words for comparison, e.g.

 ijo, ika, miman, igo (more, less, later), etc.

6) Comp 2: Particle for comparison, i.e.

 yori, yorimo (∿ than).

7) adj : Adjectives, e.g.

 ookii, hayai, shouno, daino (large, early, small, wide), etc.

8)* Root : Root words, i.e. names of scales, e.g.

 nen, ken (year, prefecture), etc.

9)* Leaf : Leaf words, i.e. elements of scales, e.g.

 1980, Tokyo, otoko (male), etc.

10)* Unit: Words for data units, e.g.

 en, nin, km (Yen, person, kilometer), etc.

11)* SL : Names of S.L.'s representing the sort of the S.L. data, usually given at Storage mode, e.g.

 jinko, TV keiyakusha (population, TV subscriber), etc.

12)** Var: Variable names such as

 A, B, KEN, etc.

The items in the categories marked by one asterisk are automatically added to the lexicon at the beginning of Manipulation mode in order to cover those S.L.'s which are passed from Retrieval mode, and deleted after use. They are thus highly application oriented.

The lexicon would become very large if it included the items in Leaf category. We tried to exclude them from our lexicon by contriving a recognition method of them from the contexts, so that the lexicon contains only about 100 application independent items plus application oriented ones.

Var category marked by two asterisks was also excluded from our lexicon, since the formation rules of this category is well-defined and easily programmed.

Grammatical rules. It was sufficient to prepare merely a dozen grammatical rules expressed as context-free-like productions with conditions of application.

1) Initial production

$$S \rightarrow \left\{ \begin{matrix} R \\ D \\ V \end{matrix} \right\}$$

2) Range-of-S.L. phrase

$$R \rightarrow \left\{ \begin{matrix} \text{Var} \\ \underbrace{\text{Mod Mod} \ldots \text{Mod}}_{n} \quad \text{SL} \end{matrix} \right\}$$

Condition: n = dim(SL), where the right-hand side of the equality denotes the dimension of S.L. represented by SL.

3) Root modifier

$$\tilde{R} \rightarrow \underbrace{\text{Mod Mod} \ldots \text{Mod}}_{n} \ \text{SL}$$

Condition: n = dim(SL)-l.

4) Modifier

$$\text{Mod} \rightarrow \left\{ \begin{matrix} (\text{Root } \underline{\text{ga}}) \ \text{Leaf} \\ \text{D} \end{matrix} \right\} \text{eq}$$

5) Domain-of-S.L. phrase

$$\text{D} \rightarrow \left\{ \begin{matrix} \text{Var} \\ (\tilde{R} \ \underline{\text{ga}} \ \text{cond}) \ \text{Root} \end{matrix} \right\}$$

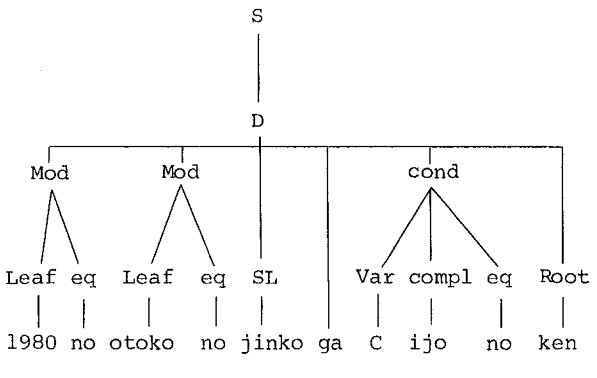
6) Numeric value

$$\text{V} \rightarrow \left\{ \begin{matrix} \text{Var} \\ \text{Num (Naux)(Unit)} \\ \left\{ \begin{matrix} \text{R} \\ \text{D} \end{matrix} \right\} \left\{ \begin{matrix} \underline{\text{no}} \\ \underline{\text{nitaisuru}} \end{matrix} \right\} \text{Agg} \end{matrix} \right\}$$

7) Condition

$$\text{cond} \rightarrow \text{V} \left\{ \begin{matrix} (\text{comp 1}) & \text{eq} \\ \text{comp 2} & \text{adj} \end{matrix} \right\}$$

An example of parsing trees by this grammar is given in Fig. 4. We assume that 'jinko' S.L. is of dimension three.



(Prefectures in which the male population in 1980 is greater than C.)

Fig. 4 Example of a parsing tree

4.2 Translation into SML

Translation from Type II expressions in Japanese into Type I expressions in 'pure' SML is performed by using two fundamental tools: a word-for-word conversion table and a conversion procedure.

Word-for-word conversion table. This is prepared for the following five categories of lexical items:

Agg, compl, adj, Root*, SL*.

For the asterisked categories the table is made up whenever Manipulation mode is invoked. A portion of the conversion table is shown in Table 1.

Table 1 Word-for-word conversion table (a part)

Category	Words (Items)	
	source	target
Agg	kosu	COUNT
	souwa	SUM
	saidai	MAX
compl	ijo	>=
	miman	<
adj	ookii	>
	hayai	<
	daino	>
Root	nen	S1
	ken	S2
SL	jinko	F1
	menseki	F2

Conversion procedure. Since the proposed grammar is so compact, we considered that the conversion procedure including syntax analysis would be best realized through a general-purpose programming language, say PL/I, rather than a comprehensive grammar-writing system like ATN.[9] This will also contribute to a portability of the system.

The programming considerations were:

(1) To insure a free writing of a Japanese Kana phrase, we adopted a left-to-right parsing, predicting the succeeding category. However, the lexicon does not include the leaf words, we had to impose the restriction that any leaf word should be enclosed by a space or an apostrophe.

(2) An SML expression is generated, by introducing a new variable symbol in the form 'SYS**', whenever a partial result of parsing becomes sufficient to do so. (This point can be best illustrated by the

example given below.)

(3) Two important steps in a parsing flow are the decisions:

a) Which of the initial productions can be applied; $S \rightarrow R$, $S \rightarrow D$, or $S \rightarrow V$?

b) Which phrase actually appears, R or $\tilde{R}$?

4.3 An example

We now return to Example 4 in Section 3.1. That query will be written in Type II form in Japanese as follows. (We adopt here a real notation of our system using Kana characters.)

Example 5. (A Japanese translation of Example 4).

```
LIST A, B;

  A = '1980'ノ'オトコ'ノジンコウガC イジョウノケン;

  B = Aノコスウ;

  C = 1970 ノ トウキョウ ノ オンナ ノ ジンコウ;
```

This Type II expression will be translated into the following Type I equivalent.

```
LIST A, B;

  SYS01 = '1980';

  SYS02 = 'オトコ';

  A = <X:F1(SYS01, X, SYS02) < C > ;

  B = COUNT (A);

  SYS03 = '1970';

  SYS04 = 'トウキョウ';

  SYS05 = 'オンナ';

  C = F1(SYS03, SYS04, SYS05);
```

5. Conclusions

Our compact database system SCLAMS with a translator from Japanese into SML has been implemented for IBM 370/138. The translator is a PL/I program consisting of about 500 statements including the lexicon and the grammatical rules themselves. The overall performance of the translator seems to be sufficient for practical use. In fact, the translation time of each Type II expression is about 1 second.

We believe, from our experiences, that a natural-artificial hybrid language like ours will be a practical step to explore the better languages for database access, specifically for casual users.

Acknowledgement. The authors wish to express their gratitute to Y. Suzuki, the former Deputy-Director of NHK Technical Research Laboratories and M. Machida, Head of Information Processing Research Group of the Laboratories for encouragement and guidance. They are also grateful to J. Kutsuzawa, Senior Research Engineer of our group for his valuable comments concerning the implementation of the system.

References

1. W.A. Woods et al.: The luner sciences natural language information system. BBN Rep. 2378, Bolt Beranek and Newman, Cambridge, Mass., 1972.

2. E.F. Codd: Seven steps to rendezvous with the casual user. In "Data base management", J.W. Klimbie et al., eds., North-Holland, Amsterdam, 1974, pp. 179-200.

3. L.R. Harris: User oriented data base query with the ROBOT natural language query system. Proc. 3rd VLDB, Tokyo, Oct. 1977.

4. G.G. Hendrix et al.: Developing a natural language interface to complex data. ACM Trans. on Database Systems, Vol. 3, No.2, June 1978, pp. 105-147.

5. M. Sibuya et al.: Noun-phrase model and natural query language. IBM J. RES. DEVELOP., Vol. 22, No.5, Sep. 1978, pp. 533-540.

6. T. Aizawa et al.: SCLAMS - a data processing system (in Japanese). Preprint of WGDBMS of IPSJ, Tokyo, July 1979.

7. T. Aizawa (ed.): SCLAMS - a user's manual. NHK Res. Lab., Tokyo, Apr. 1980.

8. C. J. Date: An introduction to database systems, 2nd ed.. Addison-Wesley, 1977.

9. P.H. Winston: Artificial intelligence, Addison-Wesley, 1977.