ISSN 1009 - 3095 *Journal of Zhejiang University SCIENCE* V.4, No.2, P.152 - 161, Mar. - Apr., 2003
http://www.periodicals.com.cn; http://www.zju.edu.cn/English
http://www.zjupress.com; http://lib.zju.edu.cn/eindex.htm; jzus@zju.edu.cn

A natural language user interface for fuzzy scope queries*

HUANG Yan(黄　艳), YU Hong-feng(俞宏峰), GENG Wei-dong(耿卫东)†, PAN Yun-he(潘云鹤)

(*College of Computer Science and Engineering, Zhejiang University, Hangzhou 310027, China*)

†E-mail: gengwd@cs.zju.cn

Received Dec.21, 2001, revision accepted May 6, 2002

Abstract: This paper presents a two-agent framework to build a natural language query interface for IC information system, focusing more on scope queries in a single English sentence. The first agent, parsing agent, syntactically processes and semantically interprets natural language sentence to construct a fuzzy structured query language (SQL) statement. The second agent, defuzzifying agent, defuzzifies the imprecise part of the fuzzy SQL statement into its equivalent executable precise SQL statement based on fuzzy rules. The first agent can also actively ask the user some necessary questions when it manages to disambiguate the vague retrieval requirements. The adaptive defuzzification approach employed in the defuzzifying agent is discussed in detail. A prototype interface has been implemented to demonstrate the effectiveness.

Key words: Integrated circuit information system, Natural language user interface, Grammar, Fuzzy rule, Matching degree
Document code: A **CLC number:** TP391.2; TP273.4

INTRODUCTION

An integrated circuit (IC) information system stores IC data in its database. This greatly benefits IC designers by allowing them to browse or query various IC information like logic functions, package types, pin descriptions, technical parameters, producer information and typical applications, etc. Unfortunately, relatively little attention has been paid to this area.

We built an IC information database on Unix Solaris platform that can be queried via a WWW browser using the client-server model. The query conditions are specified by either Boolean Language or Graphic User Interface (GUI). However, in practice, Boolean Language and GUI have limitations in IC information query process. IC queries mainly involve parameter retrieval, most of which can be categorized into scope queries. Conventional Boolean Language and GUI are suitable only for straightforward scope queries with precise boundaries. However, if fuzzy scope queries are desirable, they tend to fail since fuzzy conditions can hardly be accommodated in an adequate way in conventional mathematical formulae or placed in form slots or menu options. In IC queries, although experts always prefer straightforward scope queries, fuzzy scope queries are usually popular for non-expert users taking into consideration that they can only give succinct qualitative descriptions to specify target data and search conditions. A natural language interface can meet the requirement of non-expert users by allowing them to specify fuzzy query conditions in normal English sentence and leaving the understanding of natural language and the defuzzification of fuzzy conditions to computers.

Moreover, natural language input is more user friendly for a normal user whose only natural means of communication and expression is natural language. Hence, natural language interface is more attractive to novices by freeing novices from having to worry about the way of expressing queries or to learn an artificial query language. Pritchard-Schoch (1993) affirmed that the goal of many information retrieval experts is "to completely replace Boolean techniques with natural language techniques."

Early researches on natural language queries focused more on "examining natural-language-

* Project (No. 98 - 15) supported by the National Planning Committee of China

processing techniques as a way of improving retrieval performance when the stored information is largely textual" (Van Rijsbergen et al., 1993). They extracted keywords from natural language input and counted the occurrence of the terms in the stored documents. Such researches do not handle ambiguity, vagueness and imprecision inherent in natural language element, and therefore, are not suitable for the application of IC query interface which always accommodates fuzzy terms for queries exemplified by "List Flip-Flops with very high noise resistance ability". Among researches on fuzzy queries like (Kacprzyk et al., 1996; 2001; Zemankova et al., 1993; Bosc et al., 1998; Tahani, 1977), Kacprzyk et al. (2001) implemented FQUERY for Access, a user-friendly interface to Microsoft Access, which is a powerful set of fuzzy-logic-based tools for effective and efficient handling of imprecise elements of natural language. However, it skips the natural language parsing step by prohibiting users from inputting whole natural English sentences as query conditions, and only permitting the specification of fuzzy elements as SQL WHERE clauses via GUI, while our system aims at the entire pipeline from natural language processing to retrieved records ranking. Wang (1994; 2000) presented a natural language interface for geographic information system (GIS), which is very similar to our system. However, it mainly involved representing fuzzy spatial features and evaluating imprecise spatial relationships based on fuzzy grammar and possibility theory, and thus, cannot be directly applied to IC's scope queries. Moreover, in the aforementioned two systems, the defuzzification process was somewhat static since the interpretations of fuzzy terms were predefined by users or by program, and computers retrieved records based on these predefined static interpretations. Such approaches are not suitable for IC query interface in that the changeful IC data will often outdate the static interpretations. We enhance the query system's adaptability by employing an adaptive defuzzification technique. That is, the interpretation of a fuzzy term describing a certain attribute is dynamically customized based on the statistics of all comparable attribute values stored in the IC database.

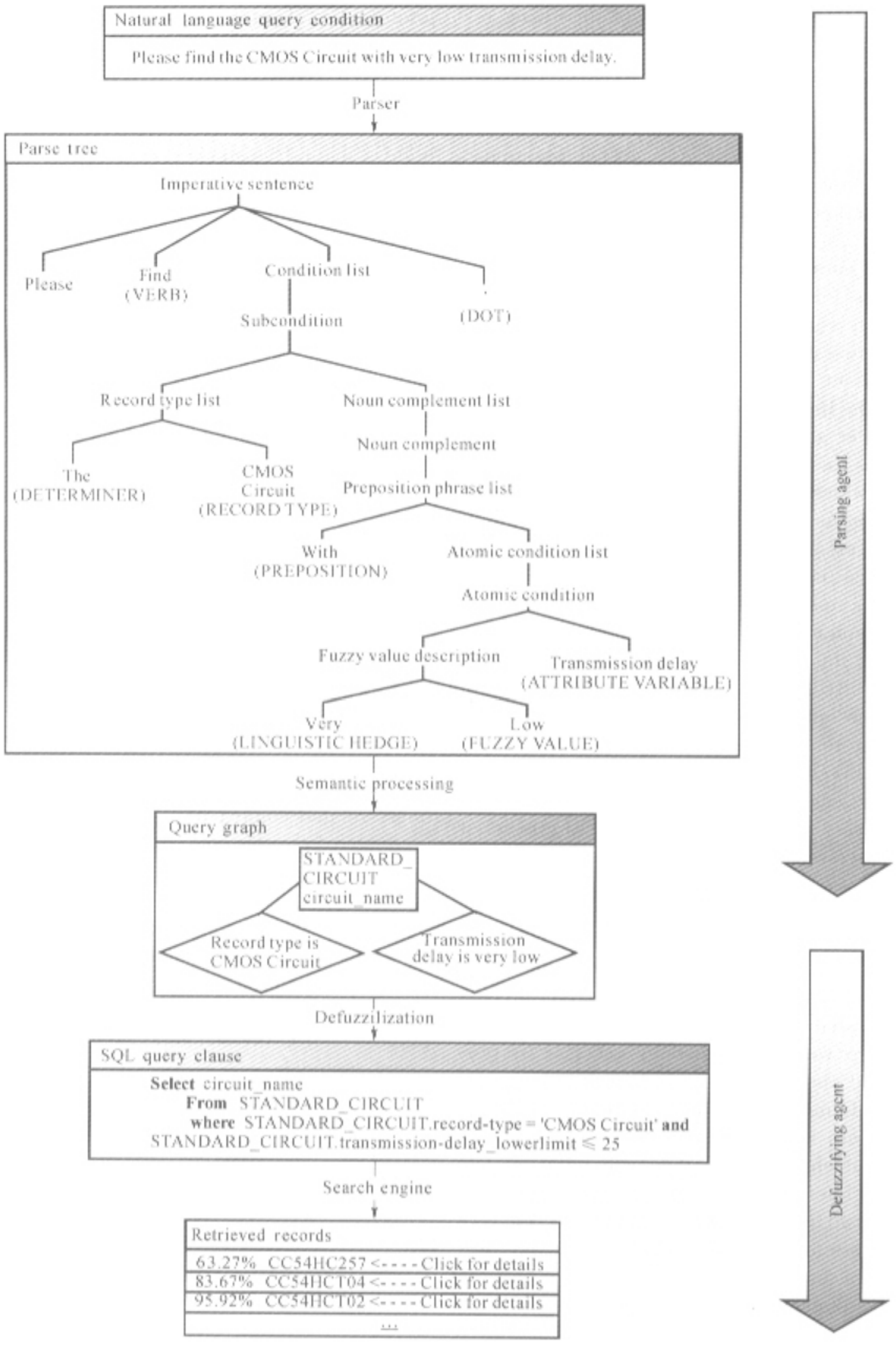
The work reported in this paper is an attempt towards considerably improving existing Boolean/GUI IC query interface by providing formal means to parse natural language and handle vagueness resulting from the use of natural language. We propose a two-agent framework to build a natural language user interface for IC information system. This framework equips IC information system with an effective front-end with three main functions: comprehension of natural language, adaptive fuzzy-rule-based transformation of a fuzzy linguistic element into its equivalent SQL query that can be executed by a computer, and fuzzy-rule-based computation of matching degrees of retrieved records. A prototype interface has been implemented. Currently, the interface can process scope queries as well as some relational queries of single English sentence that is grammatically correct.

OVERVIEW OF TWO-AGENT FRAMEWORK

The two-agent framework is shown in Fig.1.

Parsing Agent The task of this agent is to understand a natural language query sentence by syntactical processing and semantic interpretation, that is, to extract from the natural sentence the linguistic parts expressing fuzzy or crisp query conditions. Based on the predefined IC-query-related context-free grammar, a parse tree is created from the query sentence using bottom-up parsing technique. If more than one parse tree can generate the same sentence and can be reasonably interpreted semantically, the agent will actively ask the user to designate the right parse tree to disambiguate the parsing process. The parse tree is then converted into a query graph which is the semantic interpretation of the query sentence.

Defuzzifying Agent Firstly, this agent figures out what each fuzzy condition means by defuzzifying the imprecise conditions within the query graph into precise Boolean formula so that an executable SQL statement can be generated. To do this, we express the imprecise condition as a fuzzy set which, along with its adaptive membership function and a threshold, determines the precise query conditions. For each retrieved record, the matching degree is calculated based on the corresponding membership function.



```
Select circuit_name
    From STANDARD_CIRCUIT
    where STANDARD_CIRCUIT.record-type = 'CMOS Circuit' and
STANDARD_CIRCUIT.transmission-delay_lowerlimit ≤ 25
```

Fig.1 The two-agent framework

PARSING AGENT

To understand a natural language query condition, two successive processes, namely parsing and synthesizing, are involved. Parsing an English sentence, which is the focus of this section, is to first grammatically break it into component parts and then semantically interpret the component parts. The synthesizing process will convert semantic expression with fuzzy elements into precise executable image that is equivalent to the natural language condition. This process is decribed in the next section.

1. Syntactic processing

Grammar describes the internal structure of the language being compiled. With predefined grammar, an input query sentence can be parsed by a parsing technique. We employ context-free grammar (Allen, 1995) to control the structure of query sentences.

The IC-query-related context-free grammar we introduced is partially given in the appendix. It is represented as a variation of Backus-Naur form (BNF). By this grammar, we can express three types of English sentences, including declarative sentence exemplified by "I need pulse distributor whose pin number is less than ct54g110" imperative sentence exemplified by " Please find a noise reduction circuit with very high S/N radio and environment temperature," as well as WH interrogative sentence exemplified by "What is the manufacture technique of SC5156 encoder?" This grammar's productions contain both precise value phrase and fuzzy value phrase, and accordingly support not only precise attribute value queries, but also fuzzy scope or relational queries. This grammar covers most of the IC query patterns and can be further extended for practical use.

We employ the general bottom-up parsing technique, which is similar to the counter-reasoning process and is implemented by a push-down automation, to build the parse tree which is a structural representation of the sentence being parsed and shows how the starting symbol of the grammar derives the primitives in the language. An example of parse tree is shown in Fig.1.

However, it is not guaranteed that only one parse tree is generated. That is to say, the grammar may be ambiguous so that more than one parse tree can generate the same sentence and there is no way to predict, from the grammar itself, which one of these is in accordance with the searcher's intention. For example, for the query sentence "I want to know the working temperature of CD1452 and Phase_Loop_Lock or data selector with eight data input ports", two parse trees are possible. For one, "the working temperature of CD1452 and Phase_Loop_Lock" and " data selector with eight data input ports" are two sub-conditions respectively, while for another, " the working temperature of CD1452" and "Phase_Loop_Lock or data selector with eight data input ports" are two sub-conditions respectively. To overcome ambiguity, Wang (2000) proposed a possibility-theory-based fuzzy grammar to find the most possible parse tree. In our approach, we leave the issue of ambiguity to the subsequent semantic interpretation stage. That is, a parse tree that can be reasonably interpreted semantically is selected as the final parse tree. In the aforementioned example, the latter parse tree is semantically wrong since Phase_Loop_Lock has no attribute named " data input port", and is therefore discarded. If more than one parse tree are semantically reasonable, the parsing agent will actively ask the user to specify the right parse tree.

2. Semantic interpretation

Based on the affiliation table of record types, the parse tree generated from the bottom-up parser is converted into a query graph. The reasonability of the resulting query graph is then verified based on the attribute variable table of each record type. Affiliation table records the corresponding database table that each record type belongs to, e.g., record type "monofier" belongs to STANDARD_ CIRCUIT table.

Query graph is created from the parse tree under a recursive procedure. It is made up of the nodes representing the involved database tables that should be searched. Which database table involved for a record type is decided by looking up its affiliation table. Similar to the structure of query graph employed in Wang (2000), the attribute variables to be displayed are underlined, the attribute variables pertinent to relational operations are bold and underlined, and the relational operations are represented as edges. A node may be attached with one or more selection conditions and an edge is attached with a relational condition. The selection condition and relational condition can be either precise or fuzzy. For each query tree, a query graph is uniquely created to indicate the semantic interpretation. For example, the query graph for "Please list the monofier whose output voltage is high and environment temperature is lower than the environment temperature of ct54g81" is shown in Fig.2.

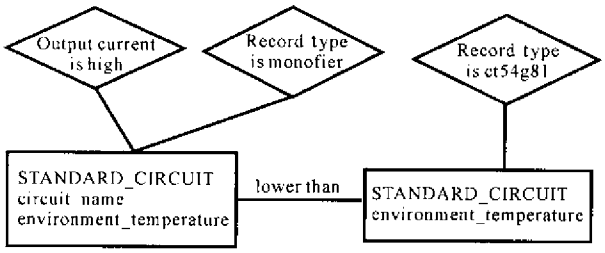

Output current is high
Record type is monofier
Record type is ct54g81
STANDARD_CIRCUIT circuit name environment_temperature
lower than
STANDARD_CIRCUIT environment_temperature

Fig.2 A query graph

As mentioned before, to deal with ambiguity, the validity of a query graph will be verified based on the attribute variables table of each record type, which, as its name indicates, includes all the attribute variables that a record type may contain. A query graph is valid if and only if each attribute variable involved in the selection or relational conditions of a record type has an entrance in the attribute variables table of that record type. In the example of "I want to know the working temperature of CD1452 and Phase_Loop_Lock or data selector with eight data input ports", since there is no entrance for "data input port" in the attribute variables table of the record type Phase_Loop_Lock, the parse tree that takes "Phase_Loop_Lock or data selector with eight data input ports" as a sub-condition is invalid. If more than one query graph converted from parse trees are valid, the parsing agent will actively ask the user to designate the right parse tree, and the corresponding query graph is selected out.

DEFUZZIFYING AGENT

Query graph can be conveniently interpreted as fuzzy SQL statement exemplified by "SELECT circuit_name FROM standard_circuit WHERE record_type is noise reduction circuit AND S/N_radio is very high AND environment_temperature is very high". Such fuzzy SQL statement cannot be directly executed by computers due to the fuzzy scope or relational linguistic terms included in the WHERE clause. The defuzzifying agent will build a bridge between imprecise and precise to transform a fuzzy SQL statement to the corresponding regular SQL statement that can be recognized by the search engine. This bridge is fuzzy rule that can model fuzzy terms and this process is referred to as defuzzification (Wang, 1994). In this process, fuzzy term is expressed as a fuzzy set which, along with its membership function and a matching degree threshold, determine the precise conditions corresponding to a fuzzy one. The definition of the membership functions is the keypoint in this process. Here, we propose a two-step adaptive customization approach to define membership functions. In the first step, the membership function corresponding to a fuzzy set is knowledge based customized, resulting a unified membership function on the interval [− 1, + 1] to allow for context-independent definition. In the second step, the variability interval of the described attribute is dynamically extracted from the database and then mapped onto the unified [− 1, + 1] interval.

1. Fuzzy sets

For a fuzzy formula "x is F" such as "output current is high", F is the restricting fuzzy set on x, associating x with a possibility distribution. That is, x can be partially matched with F for a possibility that can be decided by a membership function on the interval [0, 1]. Each fuzzy set is characterized by a membership function which can be defined as:

$$\mu_A(x):\ X \rightarrow [0,1] \qquad (1)$$

where 0 indicates "not belong to" and 1 "absolutely belong to", and elements in X are mapped to a close interval [0, 1], namely, $0 \leqslant \mu_A(x) \leqslant 1$. The value of the membership function, such as 0.5, is called matching degree.

2. Membership functions

IC query languages accommodate a large number of fuzzy terms that can be categorized into three types: linguistic values (i.e., high), linguistic modifiers (i.e., very) and linguistic relations (i.e., lower than), all of which can be described by fuzzy rules composed of fuzzy sets and corresponding membership functions. In fact, in Zadeh's paradigm of computing with words (Zadeh et al., 1999), there is another kind of fuzzy term called linguistic quantifiers. However, its chance to appear in an IC query sentence is so little that we leave it out of account.

In our approach, to define the membership function of a particular linguistic value or relation of the target attribute type and record type, two customizations must be applied. The first customization is performed based on knowledge, resulting in a unified membership function on the interval [− 1, + 1]. This membership function is context independent, that is, the same linguistic values or relations of different attribute type or record type will correspond to the same unified membership function. So far this customization process has not provided interface for users to customize according to their understandings. All customizations are pre-defined based on common sense. In the second customization, the minimum and maximum values of the described attribute within the comparable records are dynamically extracted from the database as the variability interval which is then mapped onto the unified [− 1, + 1] interval. To this end, the membership functions are context dependent. This process can enhance the system's ability to adapt to the changeful IC data. For linguistic modifier, it is always followed by a linguistic value or relation. We deal with it by applying concentration, dilation or filtration operations to the known unified membership function of the modified linguistic value or relation. How to obtain the unified membership functions in the first customization step is discussed below. With unified membership functions, the second customization process can be easily applied to them.

(1) Two basic functions

In our approach, two basic functions, S-function and Π-function, will be employed to define most of the membership functions. S-function, shown in Fig.3a is defined as:

$$S(x;\alpha,\beta,\gamma)=\begin{cases}0 & \text{if } x\leqslant\alpha\\ 2\left(\dfrac{x-\alpha}{\gamma-\alpha}\right)^2 & \text{if } \alpha<x\leqslant\beta\\ 1-2\left(\dfrac{x-\gamma}{\gamma-\alpha}\right)^2 & \text{if } \beta<x\leqslant\gamma\\ 1 & \text{if } x>\gamma\end{cases}$$

Π-function, shown in Fig.3 b is defined as:

$$\Pi(x;\beta,\gamma)=\begin{cases}S(x;\gamma-\beta,\gamma-\beta/2,\gamma) & \text{if } x\leqslant\gamma\\ 1-S(x;\gamma,\gamma+\beta/2,\gamma+\beta) & \text{if } x>\gamma\end{cases} \quad (3)$$

(2) Unified membership functions for linguistic values

Three kinds of unified membership functions are possible for linguistic values:

1) The matching degree increases with the increase of attribute value, exemplified by the membership function characterizing the fuzzy set "high" as shown in Fig.3c. In this case, we define the membership function as an S-function, $S(x;\alpha;\beta;\gamma)$, where α is 0, β is 0.5, and γ is 1.

2) The matching degree decreases with the increase of attribute value, exemplified by the membership function characterizing the fuzzy set " low" as shown in Fig.3d. In this case, we define the membership function as a reflection of S-function, $1-S(x;\alpha;\beta;\gamma)$, where α is -1, β is -0.5, and γ is 0.

3) The matching degree peaks when the attribute value locates at a certain point, but decreases when the attribute value deviates from the point, exemplified by the membership function characterizing the fuzzy set "medium" as shown in Fig.3e. In this case, we define the membership function as a Π-function, $\Pi(x;\beta;\gamma)$, where β is 1/3, and γ is 0.

(3) Unified membership functions for linguistic relations

Linguistic relation is represented by a binary fuzzy relation whose interpretation is similar to that for a linguistic value. The main difference is that in the case of linguistic value, only one universe of discourse (i.e., the set of possible values of a particular attribute) is employed, while in the case of a binary fuzzy relation, two attributes are involved. A natural approach is to assume the universe of discourse to be the set of possible values of the difference of the values of two attributes. Then a linguistic relation, FR, is equivalent to a fuzzy set FRS, that is, $\mu_{FR}(x,y)=\mu_{FRS}(x-y)$. Therefore, the membership function can be customized analogously for linguistic value.

Three kinds of unified membership functions are possible for linguistic relations:

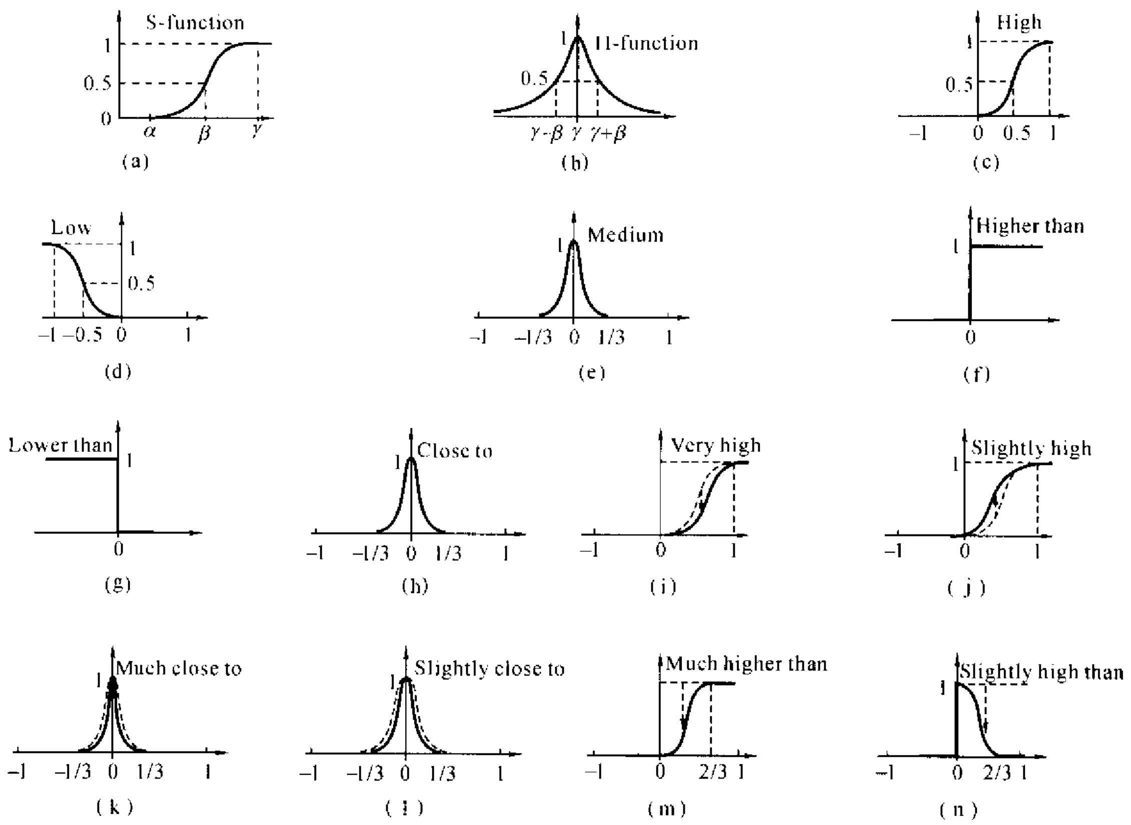


Fig.3 Example membership functions

(a) S-function; (b) Π-function; (c) high; (d) low; (e) medium; (f) higher than; (g) lower than; (h) close to; (i) very high; (j) slightly high; (k) much more close to; (e) slightly close to; (m) much higher than; (n) slightly higher than

1) The matching degree is either 1 (when the difference of two attribute values is bigger than zero) or 0 (when that difference is less than zero), exemplified by the membership function characterizing the fuzzy set "higher than" as shown in Fig.3f. In this case, we define the membership function as a pulse function.

2) The matching degree is either 0 (when the difference of two attribute values is bigger than zero) or 1 (when that difference is less than zero), exemplified by the membership function characterizing the fuzzy set lower than as shown in Fig.3g. In this case, we define the membership function as a pulse function.

3) Similar to the case in Fig.3e, the matching degree in this case peaks when the attribute value locates at a certain point, but decreases when the attribute value deviates from the point, exemplified by the membership function characterizing the fuzzy set close to as shown in Fig.3h. In this case, we define the membership function as the Π-function shown in Fig.3e.

(4) Unified membership functions for linguistic condition with linguistic modifiers

Linguistic modifier is always followed by a linguistic value or relation. We deal with it by applying proper operations to the known unified membership function of the modified linguistic condition. Three different operations are possible to apply.

1) In the case of "concentrative linguistic modifier + Fig.3 (c, d, e, h)" exemplified by the fuzzy set "very high" or "much more close to", concentration operation is applied to the unified membership function of the modified linguistic value, as shown in Fig.3 (i, k). This operation is defined as: $\mu_{\mathrm{CON}(A)}(x) = (\mu_A(x))^2$.

2) In the case of "dilative linguistic modifier + Fig.3 (c, d, e, h)" exemplified by the fuzzy set "slightly high" or "slightly close to", dilation oper-

ation is applied to the unified membership function of the modified linguistic value or relation, as shown in Fig.3j and 3l. This operation is defined as: $\mu_{\mathrm{CON}(A)}(x) = (\mu_A(x))^{0.5}$.

3) In the case of "filterable linguistic modifier + Fig. 3 (f, g)" exemplified by the fuzzy set "much higher than" or "slightly higher than", the unified membership function of the modified linguistic relation is filtered by an S-function as shown in Fig. 3 (m, n). For "much higher than", the filter is S ($3x/2$, 0, 0. 5, 1), while for "slightly higher than", the filter is $1 - S$ ($3x/2$, 0, 0.5, 1)

(5) Membership functions for combinational linguistic conditions

In this case, conjunctive composition (AND) or disjunctive composition (OR) of two or more atomic conditions are involved. To define the membership function, Max-min composition rules are employed. For the combinational fuzzy condition with the form of "C_1 AND C_2 AND ... AND C_n", min composition rule is applied and the membership function is defined as:

$$\mu_{C_1 \text{ AND } C_2 \text{ AND } \dots \text{ AND } C_n}(x) = \min(\mu_{C_1}(x), \mu_{C_2}(x), \dots \mu_{C_n}(x)). \tag{4}$$

For the combinational fuzzy condition with the form of "C_1 OR C_2 OR ... OR C_n", max composition rule is applied and the membership function is defined as:

$$\mu_{C_1 \text{ OR } C_2 \text{ OR } \dots \text{ OR } C_n}(x) = \max(\mu_{C_1}(x), \mu_{C_2}(x), \dots \mu_{C_n}(x)). \tag{5}$$

3. Defuzzification of fuzzy conditions

The defuzzification of fuzzy conditions involves construction of a precise SQL statement for execution by converting the fuzzy conditions to precise Boolean formulae based on the corresponding membership functions. After defuzzification, the fuzzy linguistic terms are eventually understood by computers. To implement the defuzzification process, a threshold θ is defined stating the minimum acceptable matching degree, i.e., only the record whose matching degree is possibly equal to or above θ is within the search range. Consequently, for a scope query, a record is within the search range only when $[R(AT)_{\mathrm{LL}}, R(AT)_{\mathrm{UL}}]$ has an intersection with the x range that satisfies $\mu_F(x) \geqslant \theta$, where $R(AT)_{\mathrm{UL}}$ is the upper limit of the values of attribute AT of record R and $R(AT)_{\mathrm{LL}}$ is the lower limit. The x range that satisfies $\mu_F(x) \geqslant \theta$ can be obtained by computing the inverse function $\mu_F{}^{-1}(\theta)$. Proper assignment to θ is necessary since it avoids low search efficiency and saves computing resources by preventing the handling of records with very low matching degrees. Once a precise SQL statement is constructed, it is passed to the search engine which will retrieve desired records.

4. Calculating matching degree

In this stage, the retrieved records will be displayed in a ranked order based on their matching degrees. The bigger the matching degree of a record, the higher its ranking. To compute the matching degree of a record with the input fuzzy condition, the fuzzy rules as analyzed in the previous section are employed. In IC query, the attribute values of a retrieved record are usually a scope rather than a single value. We assume the value distribution within that scope is even, and calculate the matching degree of the record by integrating the membership function within that scope.

For linguistic value, the matching degree, $md(.,.)$, of an atomic condition $AT = FV$ and a record R is

$$md(AT = FV, R) = \frac{\int_{R(AT)_{\mathrm{LL}}}^{R(AT)_{\mathrm{UL}}} \mu_{FV}(R(AT))\mathrm{d}(R(AT))}{R(AT)_{\mathrm{UL}} - R(AT)_{\mathrm{LL}}} \tag{6}$$

where $R(AT)_{\mathrm{UL}}$ is the upper limit of the values of attribute AT of record R, $R(AT)_{\mathrm{LL}}$ is the lower limit of the value of attribute AT of record R, μ_{FV} is the membership function of the linguistic value FV.

For linguistic relation, the compared objects can be two attribute values of a record, one attribute value of a record and one numerical value, or one attribute value of a record and one attribute value of another record. Without loss of generality, only the case of two attribute values of a record is discussed here. The matching degree, $md(.,.)$, of an atomic condition $FR(AV1, AV2)$ and a record R is:

$$md(FR(AT1, AT2), R) = \frac{\int_{R(AT1)_{LL}-R(AT2)_{UL}}^{R(AT1)_{UL}-R(AT2)_{LL}} \mu_{FRS}(R(AT1) - R(AT2))\mathrm{d}(R(AT1) - R(AT2))}{R(AT1)_{UL} + R(AT2)_{UL} - R(AT1)_{LL} - R(AT2)_{LL}} \tag{7}$$

where $R(AT1)_{UL}$ and $R(AT2)_{UL}$ are the upper limits of values of attributes $AT1$ and $AT2$ in R, $R(AT1)_{LL}$, $R(AT2)_{LL}$ are the lower limits of $AT1$ and $AT2$, μ_{FRS} is the membership function of the linguistic relation FR.

For linguistic value or relation with modifiers, the matching degree can be simply obtained by replacing the membership functions in Eq.(4) and Eq.(5) with the modified membership functions shown in this section.

For combinational linguistic condition, the matching degree of each atomic condition is calculated before applying Max-min composition rules to them.

SUMMARY AND DISCUSSIONS

In this paper, we present a two-agent framework to build a natural language querying interface to IC database, which can process scope queries as well as some relational queries of a single English sentence that is grammatically correct. We discuss the two agents in detail: a parsing agent that syntactically processes and semantically interprets a natural language sentence to construct a fuzzy SQL statement; and a defuzzifying agent that defuzzifies the imprecise part of a fuzzy SQL statement into its equivalent precise Boolean formula to create executable SQL statement based on fuzzy rules and calculates the matching degrees of retrieved records. The first agent can also actively ask a user to specify the right parse tree to disambiguate the parsing process. A prototype interface has been implemented using C++ on Unix Solaris platform.

This framework is a general framework for natural language query or understanding interface, not restricted to the particular applications of IC query. By replacing area-related grammars, semantic interpretation techniques and membership functions, this interface can be conveniently used in other applications.

However, in the defuzzifying agent, the first knowledge-based customization process is predefined and stored on the server side. Although this structure can ensure the correctness and consistency of fuzzy rules, it is by no means commodious for users to expand fuzzy rules and is cumbersome for personalization of fuzzy terms. Future work may involve providing users with interface to customize fuzzy rules and store fuzzy rules on both server and client side.

References

Allen, J., 1995. Natural Language understanding. 2nd edition, The Benjamin/Cummipngs Publishing Company, Inc., USA.

Bosc, P., Galibourg, M. and Hamon, G., 1998. Fuzzy querying with SQL: extensions and implementations aspects. *Fuzzy Sets Syst.*, **28**: 333 – 349.

Kacprzyk, J. and Zadrozny, S., 1996. A fuzzy querying interface for a WWW-server-based relational DBMS. 12Proceedings of IPMU'96 - Sixth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Granada, Spain, **1**: p. 19 – 24.

Kacprzyk, J. and Zadrozny, S., 2001. Computing with word in intelligent database querying: standalone and Internet-based applications. *Information Sciences*, **134**: 71 – 109.

Pritchard-Schoch, T., 1993. Natural language comes of age. *Onine*, **17**(3): 33 – 43.

Tahani, V., 1977. A conceptual framework for fuzzy query processing: a step toward very intelligent data systems. *Inf. Process. Manage*, **13**: 289 – 303.

Van Rijsbergen, C.J. and Agosti, M., 1993. The context of information retrieval. *The Computer Journal*, **35**(3): 193.

Wang, F.G., 1994. Towards a natural language user interface: an approach of fuzzy query. *Int. J. Geographic. Inform. Systems*, **8**: 143 – 162.

Wang F.G., 2000. A fuzzy grammar and possibility theory-based natural language user interface for spatial queries. *Fuzzy Sets and Systems*, **113**: 147 – 159.

Zadeh, Lotfi A., 1999. From computing with numbers to computing with words---from manipulation of measurements to manipulation of perceptions. *IEEE Transactions on Circuits and Systems*, **45**: 105 – 119.

Zemankova, M. and Kacprzyk, J., 1993. Introduction: the roles of fuzzy logic and management of uncertainty in building intelligent information systems. *Inform. Systems*, **2**: 311 – 317.

APPENDIX

1. < declarative sentence > : : =
   PRONOUN [AUXILIARY] VERB < condi-

tion list > DOT

2. < imperative sentence > : : =
[PLEASE] VERB < condition list > DOT

3. < wh interrogative sentence > : : =
{WHAT | WHICH | WHERE} BE < condition list > '?' |
{WHAT | WHICH} < record type list > VERB < atomic condition list >
< QUESTION MARK >

4. < condition list > : : =
< sub-condition > |
< sub-condition > {AND | OR} < condition list >

5. < sub-condition > : : =
< attribute list > OF < record type list > |
< record type list > < noun complement list >

6. < noun complement list > : : =
< noun complement > |
< noun complement > {AND | OR} < noun complement list >

7. < noun complement > : : =
< preposition phrase list > |
< adjective phrase list > | < relative clause list >

8. < preposition phrase list > : : =
PREPOSITION < atomic condition list >

9. < adjective phrase list > : : =
< adjective phrase > |
< adjective phrase > {AND | OR} < adjective phrase list >

10. < adjective phrase > : : =
< FUZZY VALUE > PREPOSITION < attribute list >

11. < relative clause list > : : =
< relative clause > |
< relative clause > {AND | OR} < relative clause list >

12. < relative clause > : : =
{THAT | WHICH} VERB < atomic condition list > |
WHOSE < attribute list > BE < atomic condition list >

13. < atomic condition list > : : =
< atomic condition > |
< atomic condition > {AND | OR}
< atomic condition >

14. < atomic condition > : : =
< fuzzy value description > [< ATTRIBUTE VARIABLE >] |
[DETERMINER] < ATTRIBUTE VARIABLE > BE < fuzzy value description > |
[DETERMINER] [< ATTRIBUTE VARIABLE >] < fuzzy relation description >
< attribute list > [OF < record type list >] |
[DETERMINER] [< ATTRIBUTE VARIABLE >] < fuzzy relation description >
< NUMBER >

15. < fuzzy value description > : : =
[< LINGUISTIC MODIFIER >] < FUZZY VALUE >

16. < fuzzy relation description > : : =
[< LINGUISTIC MODIFIER >] < FUZZY RELATION >

17. < attribute list > : : =
[DETERMINER] < ATTRIBUTE VARIABLE > |
[DETERMINER] < ATTRIBUTE VARIABLE > {AND | OR} < ATTRIBUTE VARIABLE >

18. < record type list > : : =
[DETERMINER] < RECORD TYPE > |
[DETERMINER] < RECORD TYPE >
{AND | OR} < RECORD TYPE >