Proc. 14th Int'l Conf. on Data Engineering (ICDE'98),
Feb., 23-27, 1998, Orlando, Florida.

Fuzzy Triggers: Incorporating Imprecise Reasoning into Active Databases

Antoni Wolski Tarik Bouaziz†

Technical Research Centre of Finland (VTT)
VTT Information Technology
P.O. Box 1201, FIN-02044 VTT, Finland
{Antoni.Wolski, Tarik.Bouaziz}@vtt.fi
http://www.vtt.fi/tte/projects/tempo/

Abstract

Traditional Event-Condition-Action triggers (active database rules) include a Boolean predicate as a trigger condition. We propose fuzzy triggers whereby fuzzy inference is utilized in the condition evaluation. This way, approximate reasoning may be integrated with a traditional crisp database. The new approach paves the way for intuitive expression of application semantics of imprecise nature, in database-bound applications. Two fuzzy triggers models are proposed. Firstly, a set of fuzzy rules is encapsulated into a Boolean-valued function called a rule set function, leading to the C-fuzzy trigger model. Subsequently, actions are expressed also in fuzzy terms, and the corresponding CA-fuzzy trigger model is proposed. Examples are provided to illustrate how fuzzy triggers can be applied to a real-life drive control system in an industrial installation.

1. Introduction

There has been considerable interest in active database rules (called *triggers* in commercial applications, and in the SQL3 proposal [SQL3]) over the last decade (see [WC96a], p. 303-324, for a reference list). Active databases provide the capability to react to database (and possibly external) stimuli, called *events*, without user intervention. In recent years, there has been efforts towards integrating active databases with new technologies such as temporal and real-time databases [BH95, RSS+96]. We are investigating another issue which is incorporating fuzzy logic into active databases. The objective is to apply fuzzy techniques to the evaluation of the trigger condition which has been traditionally based on a Boolean predicate.

Fuzzy logic [Zad65] deals with statements that can be true to a certain degree: the values are between 1 (completely true) and 0 (completely false). Thus, fuzzy logic (FL) provides a systematic basis for representing imprecision and/or uncertainty. Another objective of FL is to mimic the ability of the human mind to effectively employ modes of reasoning that are *approximate* rather than exact. Nowadays, applications of fuzzy logic are found in many fields [MJ94], including automatic control, artificial intelligence [ZK84, LWL89], databases [DEB89, Pet96], pattern recognition, decision analysis, etc.

To our best knowledge, no effort has been made to bring imprecision and uncertainty to database triggers. Similar work has been addressed in the context of fuzzy expert systems [Zad84, Zad89]. Zadeh noted that since most expert's knowledge is fuzzy, most of its facts and rules are fuzzy. Fuzzy expert systems allow fuzziness of antecedents and/or consequents in the rules of the form "if X is A then Y is B" where "X is A" and "Y is B" are fuzzy propositions. They also utilize approximate reasoning [Zad89, GKB+84] which deals with inference under imprecision and/or uncertainty.

The main idea behind our work is similar in spirit to the one considered in the context of fuzzy expert systems. This work was originally driven by the requirements of a real application domain which was a paper machine drive management system. In order to avoid failures of the drive system, end-users wanted a single mechanism which two requirements: (1) an ability to notify users about process data states requiring operator attention, and (2) a possibility to express the conditions to be detected in a simple and intuitive way capturing imprecise utterances like "the temperature rise is strong".

The requirements were typical for industrial process management applications characterized by ever-increasing amounts of data stored in process databases. Faced with the lack of precise process models, users demand to be able to apply imprecise measures to the management of information flood.

The paper presents two main contributions:

† Currently with Bell Sygma, Inc., 700 rue de la Gauchetiere Ouest, Montreal, Quebec, H3B 4L1 Canada.

- We propose to embed fuzzy rules within a Boolean-valued trigger condition. Model of such a trigger, called a *C-fuzzy trigger*, is presented in Section 3.
- We introduce the concept of a fuzzy action and propose a reasoning mechanism with the purpose of selecting a real action. The corresponding *CA-fuzzy trigger* model is proposed in Section 4.

Both proposed trigger models operate in a traditional environment of a crisp database. They enable to embed applications semantics in a database in an effective and intuitive way, as was shown in the case study [PBW96].

We discuss pro's and con's of the approach in Section 5. Open questions and research plans are presented in Section 6.

2. Basic concepts of fuzzy inference

This section introduces basic concepts of fuzzy sets and fuzzy inference [KY96], required to define fuzzy triggers. Informally, a *fuzzy set* is a set with imprecise boundaries in which the transition from membership to non-membership is gradual rather than crisp. In this way, a fuzzy set F in a universe of discourse U is characterized by a *membership function* μ_F, which associates each element u ∈ U with a grade of membership $\mu_F(u) \in [0, 1]$ in the fuzzy set F. Note that a classical set A in U is a special case of a fuzzy set with all membership values $\mu_A(u) \in \{0,1\}$.

1.1 Linguistic variables

The basic concept underlying fuzzy logic is that of *linguistic variable*, which is a variable whose values are words rather than numbers. A linguistic variable is characterized by a quintuple *(x, T(x), U, G, M)* in which *x* is the name of the linguistic variable; *T(x)* is the term set of *x*, that is, the set of names of linguistic values (terms) of *x* defined on *U*; *G* is a syntactic rule for generating the names of values of *x*; and *M* is a semantic rule for associating with each value its meaning.

Example 1.

Let us consider the linguistic variable *Temperature*. Its term set *T(Temperature)* could be *T (Temperature) = {low, normal, hot}* where each term is characterized by a fuzzy set in a universe of discourse *U= [0, 300]*. We might interpret "low" as "a temperature below 100°C," "normal" as "a temperature close to 120°C," and "hot" as "a temperature above about 130°C". These terms can be characterized as fuzzy sets whose membership functions are formulated below and shown in Fig. 1. For example, if the current temperature is 90°C then the *membership degree* to the fuzzy subset *low* is equal to 0.6.

μ_{low} = Trapezoidal(0, 0, 80, 100)
μ_{normal} = Trapezoidal(90, 120, 120, 140)
μ_{hot} = Trapezoidal(130, 160, 300, 300)

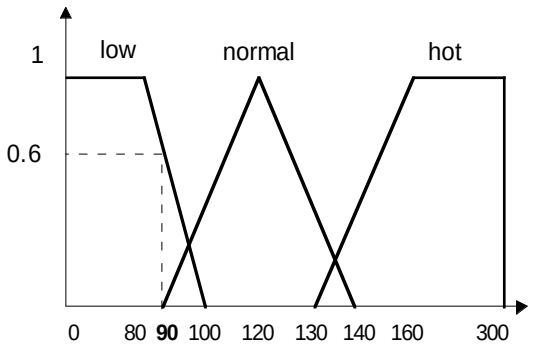

Figure 1. Membership functions of the linguistic variable Temperature.

1.2 Fuzzy inference

A *fuzzy implication* is viewed as describing a fuzzy relation between fuzzy sets forming the implication [MJ94]. A *fuzzy rule*, such as "if X is A then Y is B" is a fuzzy implication which has a membership function $\mu_{A\rightarrow B}(x, y) \in [0, 1]$. Note that $\mu_{A\rightarrow B}(x, y)$ measures the degree of truth of the implication relation between x and y. The *if* part of an implication is called the *antecedent (premise)*, whereas the *then* part is called the *consequent*. Using the *Mamdani's* (*minimum*) *implication*, the membership function of the fuzzy implication is defined as:

$$\mu_{A\rightarrow B}(x, y) = min[\mu_A(x), \mu_B(y)]$$

It is easy to see this is not a correct extension of a traditional propositional logic implication, because 0→0 yields zero. However, this interpretation of the fuzzy implication is more useful for some applications).

In fuzzy logic, Modus Ponens is extended to *Generalized Modus Ponens* in the following manner: given the input "X is A*" and the fuzzy rule "if X is A then Y is B" then the consequence is "Y is B*". The membership function of the conclusion, the fuzzy set B*, is defined as follows [Zad84, Zad89]:

$$\mu_{B*}(y) = max_{x\in A*}[\mu_{A*}(x) \wedge \mu_{A\rightarrow B}(x, y)] \qquad (1)$$

Generalized modus ponens has been adapted and used widely in control applications; the mechanism is called *interpolative reasoning*. This mechanism is needed for applications for which the input-output relationship is described by a collection of fuzzy if-then rules. A fuzzy logic system, using the interpolative reasoning, is characterized by the following algorithmical steps:

(1) Fuzzification:

The process of converting a crisp input data, $x' = x_0 \in U$, to a fuzzy set A*, is called *fuzzification*. It maps the inputs into their membership functions and truth values, these mappings are then fed into the rules. The most widely used fuzzifier is a *fuzzy singleton* defined by:

$\mu_{A^*}(x) = 1 \quad if\ x = x',\ \forall x \in U$

$\mu_{A^*}(x) = 0 \quad if\ x \neq x'$

The fuzzy input set A* only contains a crisp element x'. In this case, the formula (1) becomes a fuzzy implication:

$$\mu_{B^*}(y) = 1 \wedge \mu_{A \to B}(x', y) = \mu_{A \to B}(x', y) \qquad (2)$$

Let us now consider a rule base (where X, Y and Z are linguistic variables defined on the universe of discourse U, V and W respectively):

R_i: *if X is* A_i *and Y is* B_i *then Z is* C_i $i = 1..n$

and given the input crisp fact (x_0, y_0), the goal is to determine the output "Z is C*".

(2) Interpolative reasoning (fuzzy inference):

The most commonly used fuzzy inference method is the so-called *Max-Min inference method*. The process for obtaining the fuzzy output using the Max-Min inference method consists of the following steps:

- *Finding the firing level of each of the rules*: The truth value for the premise of each rule is computed, and applied to the conclusion part of each rule. The membership functions defined on the input variables are applied to their actual values to determine the degree of truth for each rule premise. The degree of truth for a rule's premise is sometimes referred as its *alpha*. It is computed as follows:

  $\alpha_i = \mu_{Ai\ and\ Bi}(x_0, y_0) = min(\mu_{Ai}(x_0), \mu_{Bi}(y_0))$

  If a rule's premise has nonzero degree of truth (i.e. when the input matches partially the premise of the rule) then the rule is *fired*.

- *Inferencing:* The second step is to find the output, C^*_i, of each of the rules:

  $\mu_{C^*i}(w) = \mu_{(Ai\ and\ Bi) \to Ci}(x_0, y_0, w)\ \forall\ w \in W$

  In the *Min inferencing*, which uses the Mamdani's implication rule, the implication is interpreted as a fuzzy *and* operator:

  $\mu_{C^*i}(w) = \mu_{Ai\ and\ Bi}(x_0, y_0)$ and $\mu_{Ci}(w) = min(\mu_{Ai\ and\ Bi}(x_0, y_0), \mu_{Ci}(w))$

- *Composition*: All fuzzy subsets assigned to each output variable are combined together to form a single fuzzy subset for each output variable. The purpose is to aggregate all the individual rule outputs to obtain the overall system output. In the *Max composition*, the combined output fuzzy subset C* is constructed by taking the maximum over all of the fuzzy subsets assigned to the output variable by the inference rule:

  $\mu_{C^*}(w) = max(\mu_{C^*1}(w), \mu_{C^*2}(w), \ldots, \mu_{C^*n}(w))\ \forall w \in W$

(3) Defuzzification:

The result of the fuzzy inference system is a fuzzy set. The defuzzification step produces a representative crisp value as the final output of the system. There are several defuzzification methods [Men95]. The most commonly used is the *Centroid (Center-of-gravity) defuzzifier* which provides a crisp value based on the center-of-gravity of the result (the output fuzzy set graph).

3. C-fuzzy triggers

In this section we propose a trigger model incorporating approximate reasoning (fuzzy inference) in the process of the evaluation of the condition part of an ECA trigger. We are calling such triggers *C-fuzzy triggers* (or Condition-fuzzy ECA triggers). The execution model of C-fuzzy triggers can be easily implemented using existing fuzzy inference tools.

3.1 Incorporating fuzzy inference

In order to utilize the expressive power of fuzzy rules and to apply fuzzy inference to the trigger condition part, we propose a special function called the *rule set function* (RSF):

$$RSF : \left\{ R \times S_i \times S_{i-1} \right\} \to D$$

where *R* is a set of fuzzy rules (a rule set), each of which is in the form:

if FP then FC

where *FP* is a fuzzy antecedent (predicate) and *FC* is a fuzzy consequent. *FP* and *FC* are constructed from fuzzy propositions:

X IS a_x

where *X* is a linguistic variable representing a database value, fuzzified using the term a_x (and the corresponding membership function) in the term set A_x, $a_x \in A_x$. Fuzzy antecedents can be connected by fuzzy operators *and*, *or* and *not*. The consequent linguistic variable should be the same one occurring in all the fuzzy rules in *R*.

S_i and S_{i-1} correspond to the current and previous database states. The reference to these states is possible using respectively the keywords *new* (default) and *old* in a concrete syntax. The range of RSF, *D*, is a domain (universe of discourse) of the output linguistic variable. Thus, RSF yields a crips value which can be used in a regular comparison predicate evaluating, in turn, to *true* or *false*.

3.2 Execution model

The execution model determines how triggers behave at run-time. We will not elaborate further on the behavioral dimensions of ECA triggers [PDW+93] such as coupling modes, conflict resolution, etc. We consider a simple ECA trigger execution model (as shown in [WC96b], p. 17). In its simple form, the *rule processing algorithm*, which characterizes the execution model, repeatedly executes three consecutive calculations performed when an event occurs:

(1) detecting an event and finding a relevant trigger,

(2) evaluating the condition and

(3) executing the action if the condition is true.

Our approach of incorporating fuzzy inference into triggers requires only the modification of the second calculation steps of the above rule processing algorithm. The condition may induce one or more rule set function calls. Each RSF is evaluated in the following way:

(1) *Fuzzification:* the linguistic variables in the antecedent part of the rules are evaluated, i.e., the corresponding source data are fuzzified.

(2) *Inference*: the Max-Min inference method is applied to the rule set, producing a fuzzy conclusion (a fuzzy set).

(3) *Defuzzification:* the conclusion is defuzzified using the Center-of-gravity method to yield the crisp function value which is then applied to the comparison predicate.

Example 2.

This example shows how to generate overheating alarms in a drive system. The main purpose is to watch the temperature behavior of electric motors. The temperature of a motor is expected to behave according to the motor thermal model which defines the allowable temperature as a function of power. Assume that all the relevant motor measurement series are stored in a single table having the following schema:

motor(TS, motorId, temp, deviation)

where TS is the measurement timestamp, temp is the measured value of the motor's temperature, and deviation denotes the deviation from the thermal model.

There are several steps to be followed when defining C-fuzzy triggers: the definition of the linguistic types, the rule set functions and the C-fuzzy triggers themselves. The above entities are treated as first class database objects (i.e. they can be created and removed similarly to tables). The examples below are formulated using the syntax of the language RQL/F of the TEMPO Server prototype implemented in at VTT Information Technology. RQL/F is based on the SQL language, and, specifically, the trigger syntax is based on the SQL3 proposal.

Definition of linguistic types

We begin by defining the input and output linguistic variables of the fuzzy rules. The first input variable is called tempCh and it represents the temperature change between the current and the previous temperature. The second input variable is called deviation and it reflects the deviation according to the motor thermal model. The domain of the linguistic variables is defined using *linguistic types*. To be able to represent the above linguistic variable, we define a linguistic type called TempDiff as follows:

```
create linguistic type TempDiff float (
  downStrong  trapezoid(-100, -100, -60, -50),
  downModerate  trapezoid(-60, -50, -30, -20),
  neutral  trapezoid(-30, -20, 20, 30),
  upModerate  trapezoid(20, 30, 50, 60),
  upStrong  trapezoid(50, 60, 100, 100) )
```

There is a single output variable which represents the severity level of alarms. This variable is of linguistic type Severity, defined as follows:

```
create linguistic type Severity float (
  t_none trapezoid(0, 0, 0.5, 1),
  t_low trapezoid(0.5, 1, 1.5, 2),
  t_medium trapezoid(1.5, 2, 2.5, 3),
  t_high trapezoid(2.5, 3, 4, 4) )
```

Definition of the rule set

Once the fuzzy types and their fuzzy terms are defined, we are able to define the rule set composed of fuzzy rules. Fuzzy rules are a series of "if-then" statements and they traduce the occurrence of alarming conditions. Let us assume the default alarm value is t_none which is considered an output when none of the rules is fired. The rule set may be defined as follows:

```
create rule set TemperatureAlarm
(tempCh TempDiff, dev TempDiff)
Severity DEFAULT t_none(
IF tempCh IS neutral AND dev IS upModerate
   THEN t_low,
IF tempCh IS neutral AND dev IS upStrong
   THEN t_medium,
IF tempCh IS upModerate AND dev IS upModerate
   THEN t_medium,
…)
```

The rule set TemperatureAlarm traduces that the alarm level is a combined effect of the temperature deviation and the temperature change. In order to simplify the syntax, the output variable name does not appear explicitly in the rules (only its type is specified as Severity, in the header)

Definition of C-fuzzy triggers

In the last step, we define the triggers. We include, in the condition part of the triggers, a function call to the rule set TemperatureAlarm. Three fuzzy triggers corresponding, respectively, to a low, medium and high alarm, may be defined to test the returning defuzzified value of the rule set call. One of them is shown below. The trigger is fired by each insertion of a measurement value.

```
create trigger Trig_alarm_high
INSERT ON motor
WHEN (TemperatureAlarm(
  NEW.temp-OLD.temp, deviation) > 3)
(HighTempAlarm@TempAlarms)
```

The keywords NEW and OLD represent a special semantics of RQL/F database tables which have temporal characteristics: after an INSERT, NEW denotes the inserted row and

OLD denotes the row inserted previously, in the same time series. (In the standard SQL, the OLD/NEW semantics is available with UPDATE only.)

If the WHEN predicate yields true, the action called HighTempAlarm@TempAlarms is executed. The action naming convention of RQL/F enables to call upon both internal and external (i.e. executed outside the server process) actions, depending on the application needs.

3.3 On the expressiveness of C-fuzzy triggers

For the purpose of the presentation clarity, the above example was limited with respect to the expressive power of an RSF: only one database table was used, and the rules dealt with distinct values only. In general, the expressiveness of C-fuzzy triggers is dependent on the following considerations:

- ***Domain of the rule set function:*** In many industrial applications, it is important to have the capability of reasoning based on the history of the database in order to understand the trend of the process control. The rule set function RSF, defined in section 3.1, can be extended as follows:

$$RSF : \left\{ R \times S_i \times S_{i-1} \times S_{i-2} \times ... \times S_{i-n} \right\} \rightarrow D$$

where S_i corresponds to the current database state and S_{i-j} corresponds to the j^{th} previous database state. Such an RSF may be implemented in a temporal database. In this case, an example of a rule antecedent could be "IF a motor has been too hot for more then five minutes …".

- ***Fuzzy quantifiers:*** Regular triggers use two quantifiers: *universal* and *existential*. Fuzzy triggers are based on fuzzy rules and may utilize a wide variety of fuzzy quantifiers exemplified by *few*, *several*, *usually*, *most*, *about ten*, etc. A linguistically quantified proposition may be written as "Q X's ARE A" which means Q elements of a set X are satisfying the fuzzy predicate A. For example, the quantified proposition "most motors are hot", where *hot* is a linguistic term, uses the fuzzy quantifier *most*. Fuzzy quantifiers are able to range over the database. Thus, they extend the fuzzy antecedent (predicate) of a fuzzy rule to address the whole database state.
- ***Approximate reasoning on a fuzzy database:*** Fuzzy databases provide the capability of storing imprecise or vague data. This capability enables the user to have a summarized view of the data. For example, it is more useful for the user to know that a motor is hot rather than to obtain a crisp value. In the context of fuzzy databases, the inference process does not need to fuzzify the values (i.e. the fuzzification phase is ignored). Furthermore, the result of the inference (a fuzzy set) can be stored and can be a source of other events.

3.4 Advantages and limitations of C-fuzzy triggers

We summarize the section about C-fuzzy triggers by highlighting the advantages and limitations of using them. The main advantages of adopting C-fuzzy triggers are:

- ***Ease of implementation:*** the design of C-fuzzy triggers focuses on a seamless integration of fuzzy inference within database triggers. The TEMPO Server prototype implementation has demonstrated that an existing active database server can be easily extended with fuzzy logic features, since existing fuzzy inference tools can be used.
- ***Expressiveness:*** C-fuzzy triggers, based on fuzzy rules, enable to easily capture the expert knowledge which is imprecise, incomplete or vague. This makes C-fuzzy triggers a suitable model for knowledge representation.
- ***Usability:*** Fuzzy triggers enable to shorten the application development time. A case study [PBW96] reinforced this claim. Our industrial partner[1] gave high marks to the possibility to define fuzzy database rules, using a high-level language, in a totally dynamic way.

One limitation of C-fuzzy triggers is *early defuzzification:* the defuzzification phase takes place when the crisp function output is evaluated. The defuzzification of results could imply some difficulties to define the boundaries of the comparison predicates. In the CA-fuzzy trigger this deficiency is removed.

3.5 Implementation notes

C-fuzzy triggers have been implemented, at VTT Information Technology, in the TEMPO Server which is a prototype active database system using fuzzy triggers. The TEMPO Server is an extension of the active time series database system RapidBase [WKP96] utilizing a temporal-relational model and a language based on SQL.

The C-fuzzy trigger functionality of the TEMPO Server was derived from a case study [PBW96] where requirements and specific problems of a real industrial application were analyzed, and the syntactical shape of the C-fuzzy trigger was proposed. The implementation of the TEMPO Server is described in a more detail in [BPKW97]. A demonstration application using the TEMPO Server can be downloaded from a web site[2]

4. CA-fuzzy triggers

So far we have considered C-fuzzy triggers which extend the conventional condition part of regular triggers by including fuzzy rules in a rule set function. The next step is to introduce *fuzzy actions* and to integrate them with the condition part more tightly. Such a trigger, called a *CA-*

[1] ABB Industry Oy, a manufacturer of complex drive systems for industrial installations.

[2] http://www.tte.vtt.fi/tte/projects/tempo/

fuzzy trigger, may be built by embedding action specifications in fuzzy rules. The main benefits of CA-fuzzy triggers are:

- ***Reducing the proliferation of triggers***: It is a widely acknowledged fact that the proliferation of triggers affects the performance of the system as well as it makes the process of developing applications in active databases very difficult [WC96b]. With CA-fuzzy triggers, less triggers have to be defined in the system.
- ***Providing another dimension to the causality in active databases***: Because a decision making process is inherently a process of combining different sub-decisions, extending the individual-based inference of triggers is very important. Indeed, the cause-and-effect between the condition and the action part of a regular trigger (or a C-fuzzy trigger) is a matter of yes/no (an action is executed iff the condition matches exactly), whereas it is a matter of degree in CA-fuzzy triggers.

A CA-fuzzy trigger can be symbolically represented as E(CA)*, meaning that an event fires the evaluation of *n* (n≥1) fuzzy *if-then* rules having a fuzzy antecedent and a fuzzy action consequent.

1.1 Condition-Action model

In this model, the cause-effect relationship between the database state and the concrete (implemented) actions is expressed as a fuzzy rule set in the form:

if FP_1 *then* FA_1

if FP_2 *then* FA_2

…

if FP_n *then* FA_n

where FP_i is a fuzzy predicate in the form proposed for C-fuzzy triggers and FA_i is a fuzzy action proposition represented as:

Z *is* z_i

Where Z is a fuzzy action linguistic variable such that $T(Z) = \{z_1, z_2, \ldots z_n\}$. There also exists a set of concrete actions $A = \{a_1, a_2, \ldots a_n\}$ of which each a_i (i = 1, 2, ... n) may be implemented, in a real system, as a distinct procedure. The sets Z and A are said to be *mapped* to each other if z_i is associated with a_i, for each i = 1, 2, ... n..

Thus, a linguistic term z_i denotes a fuzzy action and it is uniquely associated with a concrete action. All the membership functions of Z are defined over the same arbitrary domain where the relationships among the fuzzy actions are captured.

For example, in Fig. 2, the fuzzy actions: *zero*, *low*, *medium* and *high* are associated with (mapped to) the concrete actions $Action_1$, $Action_2$, $Action_3$ and $Action_4$, respectively.

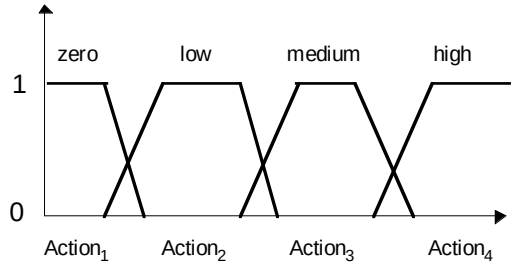

Figure 2. Example of membership functions of fuzzy actions.

1.2 Execution model

CA-fuzzy triggers induce the modification of the calculation steps of both the condition and the action parts of the rule processing algorithm. The CA-fuzzy execution model is as follows:

(1) Event signaling:

An event is detected and associated with a trigger,

(2) Fuzzification:

The linguistic variables in the antecedent part of the rules are evaluated, i.e., the corresponding source data are fuzzified.

(3) Inference:

The Max-Min inference method is performed and a result Z^* formed by the fusion of the rule results is produced. In Fig. 3, a possible result Z^* = *low-medium* is shown if the terms of the figure 2 are used in the fuzzy action proposition.

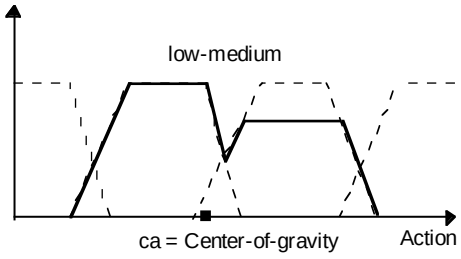

Figure 3. Example of a fuzzy result action.

(4) Action selection and execution:

The fuzzy result is then interpreted by invoking concrete actions, using one of possible strategies, such as:

- *Unique invocation* maps the fuzzy result to exactly one fuzzy action. The policy of selecting an action is to

  (a) defuzzify the result, yielding the crisp action value, *ca.*

(b) select the fuzzy action whose membership function yields the highest value, for the given *ca*, i.e., z_i such that $\mu_{zi}(ca) = \max(\mu_{z1}(ca), \mu_{z2}(ca), ...\mu_{zn}(ca))$.

(c) select the corresponding concrete action for execution.

For example, in Fig.3, the crisp value *ca* is mapped to the fuzzy action *low* and, subsequently, to the concrete action $Action_2$.

- *Multiple invocation* maps the fuzzy result to zero, one or more concrete actions. A possible mapping may involve checking the membership degrees of fuzzy subsets, in the inference result, and activating the corresponding actions if the membership degree exceeds a given threshold. In a general case, multiple invocation requires further study.

Example 3.

Let us consider Example 2 presented in Section 3, which deals with alarm treatment in a drive control system. We will show that the trigger specification can be further simplified using a CA-fuzzy trigger. Let us assume there are three different alarm notification actions of which at most one is to be invoked. The linguistic type and rule set definitions of Example 2 are used. The terms of the type Severity become fuzzy actions by virtue of the action mapping, as shown in the following trigger definition example:

```
create fuzzy trigger  OverheatingTrigger
  INSERT ON motor
  INFER TemperatureAlarm(
    NEW.temp-OLD.temp, deviation )
  UNIQUE ACTION(
    t_low AS (LowTempAlarm@ TempAlarms),
    t_medium  AS (MediumTempAlarm@TempAlarms),
    t_high AS (HighTempAlarm@TempAlarms) )
```

The above example of a CA-fuzzy trigger does the job of three C-fuzzy triggers, as one of three different actions may be invoked. The UNIQUE ACTION clause specifies the unique action invocation policy described above. A new expressive power of a trigger is achieved by a major departure from the traditional trigger syntax: the condition and action parts are replaced with the rule set (inference) part andthe action mapping part, respectively.

5. Discussion

Fuzzy triggers allow for the use of fuzzy inference to evaluate decisions when situations of interest occur. In this section we address some common criticism of the fuzzy approach and discuss possible application domains.

Critics of the fuzzy approach often ask who is going to assign a membership function of a linguistic term. This is a problem of knowledge acquisition whereby a human expertise is required. In the most common way, the user defines the shape of a membership function. There exists however a (semi-)automatic approach to generate membership functions. It is based on the use of soft computing techniques such as neural networks and genetic algorithms [Tuk91].

The proposed fuzzy trigger models can be applied to many application domains, where combining fuzziness with active behavior is needed. We believe that regular triggers and fuzzy triggers can be used in a complementary way. Thus, the part of the application which is inherently fuzzy will be easily expressed using fuzzy triggers.

Fuzzy triggers are needed particularly in the case of process management and automation systems where proprietary (non-database) solutions are still in use. The acceptance of the new database technology by the industry is best promoted by providing mechanisms which meet the requirements. Process databases are useful because they provide an efficient means of storing and querying process measurement data. Mechanisms such as fuzzy triggers are useful because they allow to analyze intelligently large volumes of stored data. In this respect, we believe that fuzzy triggers will make databases even more attractive to developers of industrial applications.

An attentive reader noticed that we have dealt mostly with applications requiring generation of intelligent event notifications. One may ask: What about the traditional role of triggers in the area of maintaining database consistency? In a brief study [BW97b] we have shown that fuzzy triggers can be used for that purpose, too.

6. Future work

The CA-fuzzy trigger model and fuzzy quantifiers are being implemented in the SENSE project[3]. More study will be also performed on the needs of the industrial user community in order to achieve the most convenient interface format of the proposed mechanisms.

There remain several other issues which require further investigations, such as a general model of fuzzy events capturing event composition. In [BW97a], we have introduced the concept of fuzzy event and proposed a trigger model dealing with it. Another important issue to study is the inter-relationship between the proposed fuzzy concepts and other behavioral dimensions of active database systems [PDW+93] like coupling modes, termination, etc.

7. Conclusions

Various applications, for example in industrial systems, require fuzzy concepts to capture the relevant semantics. To do this, we propose fuzzy database triggers (fuzzy active database rules) which aim at combining two important areas: fuzzy reasoning and active databases. In this paper, we first extend the basic semantics of event-condition-action rules with fuzzy rules and fuzzy inference. The corresponding C-fuzzy trigger model was implemented and tested in an example application. A more advanced model, the CA-fuzzy trigger model, is also proposed, where fuzzy

[3] http://www.tte.vtt.fi/tte/projects/sense/

actions are introduced and integrated with the inference process. CA-fuzzy triggers reduce the proliferation of triggers by providing a possibility to capture several actions in a single trigger definition. By way of examples utilizing concrete syntax, we showed that the proposed models may result in intuitive and user-oriented interfaces.

References

[BH95] M. Berndtsson, J. Hansson (eds.): *Active and Real-Time Database Systems (ARTDB-95)*, Proc. First Int'l Workshop on Active and Real-Time Database Systems, Skövde, Sweden, 9-11 June 1995.

[BW97a] T. Bouaziz and A. Wolski, "Applying Fuzzy Events to Approximate Reasoning in Active Databases", *Proc. Sixth IEEE Int'l Conference on Fuzzy Systems (FUZZ-IEEE'97)*, July 1-5, 1997, Barcelona, Catalonia, Spain, pp. 729–735, also at: ftp://ftp.vtt.fi/pub/projects/rapid/f-event-triggers.ps.

[BW97b] T. Bouaziz and A. Wolski, "Maintaining Soft Constraints using Fuzzy Triggers", position paper, *Workshop on Imprecision/Uncertainty in Databases*, Spain, July 5-6, 1997, (Jointly with FUZZ-IEEE'97 conference).

[BPKW97] T. Bouaziz, J. Karvonen, A. Pesonen, and A. Wolski, "Design and Implementation of TEMPO Fuzzy Triggers", *Proc. Eighth Int'l conference on Database and Expert Systems Applications (DEXA'97)*, Sept. 1-5, 1997, Toulouse, France, pp. 91–100, also at: ftp://ftp.vtt.fi/pub/projects/rapid/tempo-design.ps.

[DEB89] *Data Engineering Bulletin*, Special Issue on Imprecision in Databases, 12(2), June 1989.

[GKB+84] M.M. Gupta, A. Kandel, W. Bandler, and J. Kiszka (eds.), *Approximate Reasoning in Expert Systems*. Elsevier Science Publishers, North-Holland, 1984.

[IS89] *Information Systems*, 14(6), 1989.

[KY96] J. George Klir and B. Yuan (eds.), "Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh", *Advances In Fuzzy Systems-Applications and Theory*, Vol. 6, 1996.

[LWL89] K.S. Leung, M.H. Wong and W. Lam, "A Fuzzy Expert Database System", *Data & Knowledge Engineering*, Vol. 4, 1989, pp. 287–304.

[Men95] J.M. Mendel, "Fuzzy Logic Systems for Engineering: A Tutorial", *Proc. of the IEEE*, Special Issues on Engineering Applications of Fuzzy Logic, 83(3), March 1995, pp. 345–377.

[MJ94] T. Munakata and Y. Jani, "Fuzzy Systems: An Overview", *CACM*, 37(3), March 1994, pp. 69–76.

[PBW96] A. Pesonen, T. Bouaziz, and A. Wolski, "Case Study: Applying Fuzzy Triggers to a Drive Control System", *Research Report* No. J-6/96, VTT Information Technology, Espoo, Finland, August 1996.

[PDW+93] N.W. Paton, O. Diaz, M.H. Williams, J. Campin, A. Dinn, and A. Jaim, "Dimension of Active Behavior", *Proc. 1st Int'l Workshop on Rules in Database Systems*, Edinburg (Scotland), August 1993, pp. 40–57.

[Pet96] F.E. Petry, *Fuzzy Databases: Principles and Applications*, with contribution by Patrick Bosc, International Series in Intelligent Technologies, 1996, 240 pages.

[RSS+96] K. Ramamrithan, R.M. Sivasankaran, J.A. Stankovic, D.T. Towsley and M. Xiong, "Integrating Temporal, Real-Time, and Active Databases", *SIGMOD Record* 25(1), 1996, pp. 8–12.

[SQL3] ANSI X3H2-97-030/LBL:LGW-008 (Working Draft), Database Language SQL / Foundation, J. Melton (ed.), March 1997.

[Tuk91] I.B. Turksen, "Measurement of Membership Functions and their Acquisition", *Fuzzy Sets and Systems*, Vol. 40, 1991, pp. 5–38.

[WC96a] J. Widom and S. Ceri (eds.) *Active Database Systems: Triggers and Rules For Advanced Database Processing*. Morgan Kaufmann, 1996.

[WC96b] J. Widom and S. Ceri, "Introduction to Active Database Systems" in [WC96a], pp. 1–41.

[WKP96] A. Wolski, J. Karvonen and A. Puolakka, "The RAPID Case Study: Requirements for and the Design of a Fast-Response Database System", *Proc. First Workshop on Real-Time Databases (RTDB'96)*, March 7-8, Newport Beach, CA, USA, pp. 32–39, also at: ftp://ftp.vtt.fi/pub/projects/rapid/case.ps.

[Zad65] L.A. Zadeh "Fuzzy Sets", *Information Control*, 8(3), June 1965, pp. 338–353.

[Zad84] L.A. Zadeh, "The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems", In [GKB+84], pp. 3–31.

[Zad89] L.A. Zadeh, "Knowledge Representation in Fuzzy Logic", *IEEE Trans. on Knowledge and Data Engineering*, 1(1), 1989, pp. 89100.

[ZK84] M. Zemankova and A. Kandel, "Uncertainty Propagation to Expert Systems" In [GKB+84], pp. 529–548.