Proc. Sixth IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'97),
July 1-5, 1997, Barcelona, Catalonia, Spain.

Applying Fuzzy Events to Approximate Reasoning in Active Databases

Tarik Bouaziz‡ Antoni Wolski
VTT Information Technology
Technical Research Centre of Finland (VTT)
P.O. Box 1201, FIN-02044 VTT, Finland
{Tarik.Bouaziz, Antoni.Wolski}@vtt.fi

Abstract

The ever-increasing amounts of data stored in databases make it more and more difficult to get information summarized and provided in a timely manner to the user. In order to achieve this goal, a fuzzy trigger model is proposed. The model is based on the concept of a fuzzy event. Fuzzy events are integrated with fuzzy condition-actions using a new membership function modification technique called squeezing. In addition, this paper describes an application of fuzzy trigger for alarm treatment in a real-life industrial drive control system. The developed model can be applied to many other application domains, where combining fuzziness with active behavior is needed.

1. Introduction

The ever-increasing amounts of data stored in databases, referred to as data explosion, creates a need to transform these data to useful information. On the other hand, the decision process using this information depends on whether they are provided in a timely manner to the user. The work presented in this paper has been motivated by the needs mentioned above. It was originally driven by the requirements of a real application being a paper machine drive control system [PBW96]. A paper machine is equipped with tens of high-power electric motors. Process measurements data is stored in a database which is fed by sensors. End-users wanted a mechanism to satisfy two different needs. First, they wanted to be warned when the database reaches states of interest. In addition, they wanted the database states to be pictured in a fuzzy manner which is close to their thinking. The purpose was to prevent failures of the drive system. Thus, merging fuzzy logic features and active database capabilities was just a natural step to take.

‡ The work has been carried out within the ERCIM Fellowship programme (ERCIM Human Capital and Mobility Programme).

Applying fuzzy logic to databases has been an active research area since the 80's [DEB89, IS89, Pet96]. The most important issues which have been addressed are i) the enhancement of existing data models for representing uncertain and/or imprecise data (fuzzy data), ii) the extension of current database languages to handle fuzzy queries, and iii) the use of fuzzy inference to deduce answers to questions in fuzzy expert database systems [Zem89, LWL89].

Active databases, which incorporate Event-Condition-Action rules into the conventional (passive) databases, have been investigated by many researchers over the past decade [PDW+93, WC96b]. They provide the capability to react to database (and possibly external) stimuli, called *events*, without user intervention.

To our best knowledge, no attention has been paid until now on integrating imprecision and/or uncertainty within database triggers. Similar work has been addressed in the context of fuzzy expert systems [Zad84, Zad89]. Zadeh noted that since most experts' knowledge is fuzzy, most of its facts and rules are fuzzy. Existing expert systems, he noted, ignore the fuzziness of such information. 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. Fuzzy modifiers and quantifiers can be used in the antecedent and/or consequent of a rule. Fuzzy expert systems are based on approximate reasoning [Zad75, GKB+84]. Approximate reasoning deals with inference under imprecision and/or uncertainty in which the underlying logic is approximate rather than exact.

The main idea behind our work is similar in spirit to the one considered in the context of fuzzy expert systems[1]. The incorporation of imprecision and/or uncertainty within database triggers may be applied to the event, the condition or the action components of an active

[1] Note that several active database features have been adapted from production rule systems [BD83].

rule or all together. The contribution of this paper is twofold:

- The presentation of a fuzzy trigger model which identifies different levels of integration of fuzzy concepts within triggers (active rules). We pursue further the model proposed in [BW96] whereby approximate reasoning is integrated within triggers. In this paper, we introduce the concept of a fuzzy event and we investigate a corresponding rule execution model.
- The application of fuzzy triggers to alarm synthesis in a real-life drive control system. In the control room of a paper mill factory, the situation can be chaotic if several alarms are launched within a short time interval. Fuzzy triggers aggregate these alarms in such a way that a summarized alarm is launched. This approach enables the end user to make appropriate decisions.

This paper is organized as follows: Section 2 illustrates the main concepts of active databases. Section 3 presents a model of fuzzy triggers proposed in the TEMPO project. Section 4 provides an example of fuzzy triggers concerning an industrial application. Then, we conclude in section 5.

2. Active databases

Research on at least two fields have influenced work on active databases: artificial intelligence and database systems. Production rule systems [BD83] have been extended so that they could work with large number of rules and facts stored in a database [Han89]. On the other hand, active capabilities in database systems appeared in the early 70's in CODASYL data manipulation language [Cod73]. This language includes a mechanism for automatic procedure invocation in response to specified database operations. This was originally introduced as a mechanism for expressing and enforcing integrity constraints on the database. Research in active databases exploded (see [WC96a, p. 303-324] for a reference list) after the introduction of the event-condition-action (ECA rule) abstraction in the HiPAC project [DBB+88].

An active database system is a database system which detects situations of interest, evaluates the condition when they occur, and if the condition is true, then executes an action in a timely manner. In contrast, a conventional passive database system only executes queries or transactions explicitly submitted by a user or an application program. Let us consider the following example: when the quantity in stock of some item falls below a threshold, then a reordering activity may be initiated. This behavior could be implemented over a passive database system in two ways, neither of which is satisfactory. First, every program that updates the inventory database could check the constraint and invoke the reordering operation if necessary. The main disadvantage of this approach is that the modification or the deletion of this constraint requires finding and modifying the relevant code in every program. The second approach is to add a special application program that periodically polls the database to check for the constraints. However, polling too often can be inefficient and, if done infrequently, may result in delayed responses to critical situations. With an active database system, the desired behavior is expressed by rules that are defined and stored in the database. This has the benefit that the rules can be shared by many application programs, and the database system can optimize their implementation [Day95].

Research in active databases has been addressing the needs of a wide range of application domains that are not readily met by traditional database management systems such as network management, air traffic control, program stock trading, workflow management, etc.

An example of a special-purpose active database system for industrial process management applications is the RapidBase system developed in the RAPID project [WKP96].

3. Fuzzy triggers

We proposed basic forms of fuzzy triggers in [BW96]. In the simplest one, called a *C-fuzzy trigger*, a fuzzy rule set is encapsulated in a crisp-valued function which can be used in an evaluation of a regular Boolean-valued predicate of the ECA trigger condition part. A prototype active database server with C-fuzzy triggers was also implemented [BKPW97]. C-fuzzy triggers provide a convenient way of introducing the fuzzy inference mechanism to traditional active database systems but they are limited in expressive power. Namely, no actions neither events may be participants to fuzzy rule specifications.

A more powerful model, called a *CA-fuzzy trigger*, is also proposed in [BW96] whereby approximate reasoning is integrated within the condition-action component of a trigger. Its main characteristics are presented in this section. Then, the concept of a fuzzy event is introduced and its association with a CA-fuzzy trigger is established, leading to an *ECA-fuzzy trigger*. ECA-fuzzy triggers aim is integrating different types of fuzziness into a coherent framework. An ECA-fuzzy trigger (or simply fuzzy trigger) can be symbolically represented as E(CA)*, meaning that an event fires the evaluation of *n* ($n \geq 1$) fuzzy *if-then* rules having a fuzzy antecedent and a fuzzy action consequent.

3.1. CA-fuzzy triggers

CA-fuzzy triggers incorporate fuzziness into the condition-action components of a trigger. The condition component of a regular trigger is a crisp predicate on the database state. We have proposed to extend it with fuzzy predicates which provide a higher level of abstraction than crisp predicates. Fuzzy predicates are based on terms like *young*, *rich*, *high*, *tall*, etc. which allow the representation of vagueness. Similarly, a fuzzy action, defined by a linguistic term, can be fired partially and the decision to execute an action is based on an approximate reasoning process which aggregates the overlapping of elements of fuzzy actions. In addition, whereas the cause-and-effect relationship between the condition and the action part of a regular trigger is a matter of yes/no, it is a matter of degree in CA-fuzzy triggers. Besides incorporating approximate reasoning within triggers, the interpolative reasoning (based on the Max-Min inference method) mechanism could be useful to reduce the proliferation of triggers, and thus to improve the performance of the system. While addressing the approximate reasoning in databases, it is important to note that the inference engine operates on the whole database.

The cause-and-effect relationship between the database state and 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 a fuzzy predicate FP_i is constructed from fuzzy propositions q_x *X IS|ARE* a_x connected by fuzzy operators *and*, *or* and *not*. *X* is a linguistic variable representing a database value fuzzyfied using the term a_x (and the corresponding membership function) in the term set A_x, $a_x \in A_x$. The optional fuzzy quantifier $q_x \in Q_x$ (Q_x is a set of quantifier terms of *X*) may be also used. FA_i is a fuzzy action proposition represented as: Z *is* z_i where *Z* is a linguistic variable having a set of linguistic terms $\{z_1, z_2, ... z_n\}$. There also exists a set of concrete actions $A = \{a_1, a_2, ... a_n\}$ of which each a_i (i=1,2,...n) may be implemented, in a real system, as a distinct procedure, i.e. a database command (or sequence thereof) or an external 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 a_i. 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. 1, the fuzzy actions: *zero*, *low*, *medium* and *high* are associated with the concrete actions $Action_1$, $Action_2$, $Action_3$ and $Action_4$, respectively.

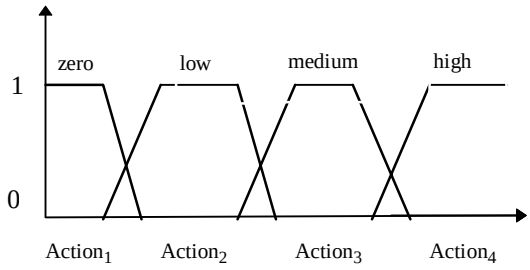

Figure 1. Example of membership functions of fuzzy actions.

The execution model of CA-fuzzy triggers involves detecting of an event, performing the interpolative reasoning process and executing an action in which the Center-of-gravity of the interpolative reasoning result has the highest membership degree.

3.2. Fuzzy events

Recently, efforts have been made to extend the expressive power of event specification languages [GD93, CM94]. However, none of these languages has addressed the specification of imprecise and/or uncertain events which are inherent i) to the incomplete states of knowledge of a particular user to describe *precisely* a situation of interest and ii) to the application domains involving imprecision and/or uncertainty such as in industrial applications in which the use of fuzzy linguistic terms convey more useful information than crisp values would do [Men95].

Fuzzy events, defined as fuzzy sets, allow the description of situations which *are not known precisely*, but known approximately. They allow users to express situations of interest using linguistic terms such as *high*, *low*, *strong*, etc. For example, let us consider the following situation which states that "*when a motor reaches a temperature of 100 °C*, then alarm the operator". A corresponding crisp event is "when updating the temperature of a motor to 100 °C", which means that the measurements 95°C, 99°C, 101°C or 103°C are just ignored. However, these values could be meaningful in practice. The example situation may be expressed as a fuzzy event "when updating the temperature of a motor to about-100°C" and is defined by the membership function $\mu_{about\text{-}100}(x)$ (Fig. 2) which is interpreted as follows: the degree of possibility that the temperature takes *t* as a value is

equal to $\mu_{about\text{-}100}(t)$. Thus, fuzzy events play a role of possibility distributions [DP88].

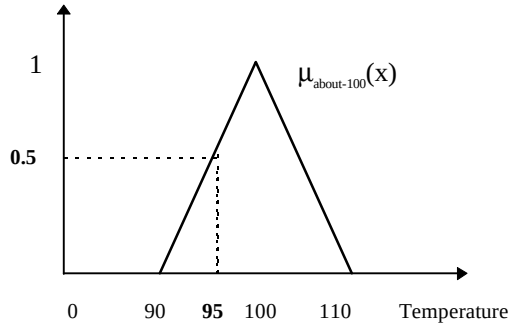

Figure 2. Membership function of a fuzzy event

We define a primitive fuzzy event as a pair $<e_c, e_f>$ where e_c is a crisp primitive event (e.g., database operations such as INSERT or UPDATE in SQL) and e_f is a fuzzy event condition (predicate) defined over the set of event parameters. For example, if the crisp event is the UPDATE operation, the updated value is considered an event parameter. Note that the concept of events qualified by crisp conditions enables to describe composite events as well as primitive events as proposed for example in ODE [GJS92]. In order to capture both the crisp event occurrence and the fuzzy event predicate, an appropriate syntax have to be proposed. For example by extending the SQL event clause [SQL3], the event discussed above may be defined as follows:

```
AFTER UPDATE OF temp MotorTemperature
ON motor
IS about-100
```

Where *temp* is an attribute of the relation *motor* fuzzyfied by the linguistic term *about-100* (Fig. 2). This term is defined within a set of terms, MotorTemperature, which is called a *linguistic type* (see Section 4).

Let us consider a fuzzy event defined by the linguistic term *Event-Term* and *v* as a current value of an update event occurrence. Event signaling is based on the computation of an *event match factor*. An event match factor is equal to the membership degree of v in the fuzzy set *Event-Term*, i.e. $\mu_{Event\text{-}Term}(v)$. The event is signaled if the event match factor is greater than zero. For example, let us consider a crisp database operation "when updating a temperature of a current motor to 95". The event match factor between the current value and the fuzzy event, which is $\mu_{About\text{-}100}(95)$, is equal to 0.5 (Fig. 2.), meaning that the event will be signaled.

3.3. ECA-fuzzy trigger execution model

The CA-fuzzy trigger model proposed in subsection 3.1 incorporates crisp events. In order to combine fuzzy events presented above with the CA-fuzzy model we propose a new membership function modification method called *squeezing*. The result of the interpolative process is based on the database state. However, many applications need to, as part of their semantics, initiate appropriate actions depending on the strength of events.

Intuitively, the idea of squeezing is to modify the membership function (typically, of an interpolative inference result) to diminish its effect. Contrary to the *concentration* [Men95], which scales down the range values of the function, squeezing converts the domain values of the function relation. We shall define squeezing in a general case first.

Let $\mu_{Z^*}: X \to [0,1]$ be a membership function defined over a range of real numbers $X = \{x_i | x_i \in [0, x_{\max}]\}$. Let $s \in [0,1]$ be a *squeeze factor*. The modified membership function $\mu^s_{Z^*}$ is a result of squeezing of μ_{Z^*} by a factor of *s* if it is evaluated as follows (• = multiplication):

$$\mu^S_{z*}(x \bullet s) = \mu_{z*}(x) \quad \textit{for each} \quad x \in X$$

For example, the following figure depicts the membership function of the possible fuzzy inference result, squeezed by *s = 0.5*, if the terms of Figure 1 are used.

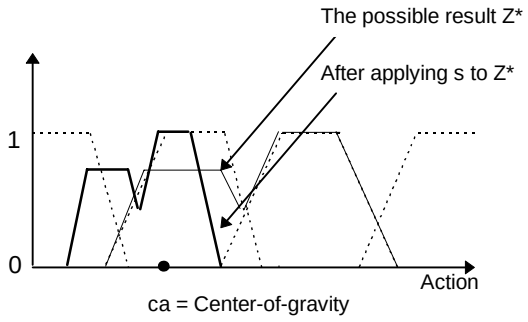

Figure 3. Membership function of a squeezed result

For squeezing to have intended semantics, the following conditions have to be held:

(1) The numerical domain X of the membership function being subject to squeezing should have zero as a minimum value.

(2) The terms of the linguistic variable being evaluated (e.g. a result variable of a rule set) are defined over X in a way correlating the center of gravity of each function with the notion of "significance" of the

corresponding term: the greater is the center of gravity, the more significant is the term.

We propose to apply squeezing to associate fuzzy events with fuzzy CA rules by squeezing the result of the fuzzy CA rule set with the event match factor. When the above conditions are met, the event match factor of 1 (perfect match) would leave the result unmodified. The lower is the event match factor the more the result function is squeezed towards zero and the resulting center of gravity moves towards zero too. When the result is mapped to concrete actions, this would result in moving towards less significant actions. In the extreme case of match factor of almost zero, the resulting concrete action may be the least significant one.

The following algorithmic steps characterize the execution model of ECA-fuzzy triggers:

(1) Event signaling: The signaling step refers to the appearance of a crisp event occurrence caused for example by an update operation. The event is signaled if the degree of match between the event occurrence and the fuzzy event is greater than zero.

(2) Interpolative reasoning: When an event is signaled, the interpolative reasoning process, using CA rules, is performed and the fuzzy result membership function is produced.

(3) Result modification: The result modification step applies the strength of events to the result of the interpolative process. The membership function of the result Z^* is squeezed as described above.

(4) Action(s) selection and execution: Selection of the concrete action corresponding to the modified result for execution. A fuzzy action whose membership function yields the highest value for the given Center-of-gravity of the modified result is selected. A multiple invocation is also possible when the Center-of-gravity maps with more than one fuzzy action.

4. Example: how to generate an overheating alarm in a drive system

The following example is based on the case study provided by ABB Industry Oy, a manufacturer of complex drive systems for industrial installations. An example of such an installation is a paper machine equipped with tens of high-power electric motors running at different speeds and loads. The problem we are illustrating here is how to generate a synthetic alarm information about motor overheating in a system like that. A serious overheating of a single motor may be a cause for an alarm to the same extent as a moderate overheating of a number of motors. It is required to have alarms reported at different intensity levels (say from 1 to 4) depending on the severity of the situation. The dynamics of the system has also to be taken into account, i.e. the fact of the rising temperature implies a higher alarm level than that of the decreasing temperature. We shall show how this complex task can be achieved by way of just *one* ECA-fuzzy trigger. The syntax of the prototype language RQL/F (RapidBase Query Language / Fuzzy) is used in this example [PBW96].

Assume that all the relevant motor measurement data are stored in a single (possibly temporal) table having the following schema:

```
motor( motorId, temp, deltaTemp )
```

where `temp` is the measured values of the motor's temperature and the derived column `deltaTemp` represents the difference between the current and the previous temperature reading, normalized by dividing by the previous reading.

In order for membership function definitions to be reusable in a database, the concept of a linguistic type is introduced. A linguistic type embodies a set of compatible terms to be applied to various linguistic variables. The following linguistic type are used to represent fuzzyfied values of temperature:

```
CREATE LINGUISTIC TYPE Temperature INTEGER(
  normal     TRAPEZOIDAL (0, 0, 120, 140),
  hot        TRAPEZOIDAL (120, 140, 300, 300),
  very_hot   TRAPEZOIDAL (145,160, 300, 300)
  )
```

The next one is a general linguistic type to deal with values whose interesting value range is [-1, 1]:

```
CREATE LINGUISTIC TYPE NegativeToPositive FLOAT(
negative       TRAPEZOIDAL (-1, -1, -0.4, -0.2),
big_negative   TRAPEZOIDAL (-1, -1, -0.8, -0.6),
small_negative TRAPEZOIDAL(-0.8, -0.6, -0.4,-0.2),
zero           TRAPEZOIDAL(-0.4, -0.2, 0.2, 0.4),
small_positive TRAPEZOIDAL (0.2, 0.4, 0.6, 0.8),
big_positive   TRAPEZOIDAL (0.6, 0.8, 1, 1),
positive       TRAPEZOIDAL(0.2, 0.4, 1, 1)
)
```

The last linguistic type will be used to represent the alarm severity level:

```
CREATE LINGUISTIC TYPE AlarmSeverity FLOAT(
      zero    TRAPEZOIDAL (0, 0, 0.5, 1.0),
      low     TRAPEZOIDAL (0.5, 1.0, 1.5, 2.0),
      medium  TRAPEZOIDAL (1.5, 2.0, 2.5, 3.0),
      high    TRAPEZOIDAL (2.5, 3.0, 4.0, 4.0)
)
```

Fuzzy quantifiers are also classified as types (the domain of a quantifier type traduces the percentage of items):

```
CREATE QUANTIFIER TYPE Amounts (
      few     TRAPEZOIDAL (0, 0, 20, 30),
      some    TRAPEZOIDAL (20, 30, 60, 70),
      most    TRAPEZOIDAL (60, 70, 100, 100)
)
```

Quantifiers may be applied to value sets yielding fuzzy quantified sets. The first value set is a set[2] of motor temperatures:

```
CREATE VALUE SET motorTemperatures OF
      ( SELECT temp FROM motor)
```

and the second one is a set of temperature deltas:

```
CREATE VALUE SET motorTempDeltas OF
      ( SELECT deltaTemp FROM motor )
```

Let us assume there are four different alarm notification actions of which at most one is to be invoked. An *action set* is defined whereby terms of the linguistic type AlarmSeverity are mapped to concrete actions:

```
CREATE ACTION SET Alarms OF AlarmSeverity (
      zero    NotifyZeroAlarm@AlarmServer,
      low     NotifyLowAlarm@AlarmServer,
      medium  NotifyMediumAlarm@AlarmServer,
      high    NotifyHighAlarm@AlarmServer
)
```

In the trigger definition, a condition-action rule set is included following the WHEN keyword. The INPUT clause is used to define the linguistic types and quantifier types of the input variables, and the OUTPUT clause is used to specify the action set(s). Both for input and output variables, optional alias specifications (following the keywords AS) are possible. Aliases enable us to use the most intuitive words in the rule formulation:

```
CREATE FUZZY TRIGGER GeneralOverheatingTrigger
AFTER UPDATE OF temp Temperature ON motor IS hot
INPUT
       motorTemperatures Temperature
       QUANTIFIED WITH Amounts AS motors,
       motorTempDeltas NegativeToPositive
       QUANTIFIED WITH Amounts AS deltas,
OUTPUT Alarms AS AlarmNotification
WHEN (
       IF some motors ARE hot
          AND most deltas ARE big_positive
               THEN AlarmNotification is low,
       IF some motors ARE very_hot
          AND some deltas ARE big_positive
               THEN AlarmNotification is low,
       IF some motors ARE very_hot
          AND most deltas ARE big_positive
               THEN AlarmNotification is medium,
       IF most motors ARE hot
          AND some deltas ARE big_positive
               THEN AlarmNotification is medium,
       IF most motors ARE hot
          AND most deltas ARE big_positive
               THEN AlarmNotification is high,
       IF most motors ARE very_hot
               THEN AlarmNotification is high)
UNIQUE ACTION
```

The `UNIQUE ACTION` clause specifies the unique action invocation policy described above. Note also, that the ECA-fuzzy trigger above does the job of three regular triggers, as one of three different actions may be invoked.

Let us assume the fuzzy set shown in Fig. 3 as the possible result Z*. Generating an alarm is based on the interpolative result and on the coming events. Let us consider two consecutive updates of the column `temp` with the following values: v_1=125 and v_2=135, respectively. The membership degrees of these values in the fuzzy event set `hot` are 0.25 and 0.75, respectively. The purpose is to generate an alarm based on the following semantics: the greater the $\mu_{hot}(t)$ is, the stronger is the reaction; the smaller $\mu_{hot}(t)$ is, the weaker the reaction. The strongest reaction is obtained when $\mu_{hot}(t)$ is equal to one. The squeeze factors are respectively equal to s_1=0.25 and to s_2=0.75. Applying these factors to the result Z* is shown Fig. 4. Since the Center-of-gravity of $\mu_{Z^*}^{s1}$ is different from the one of $\mu_{Z^*}^{s2}$, the resulting concrete actions are also different.

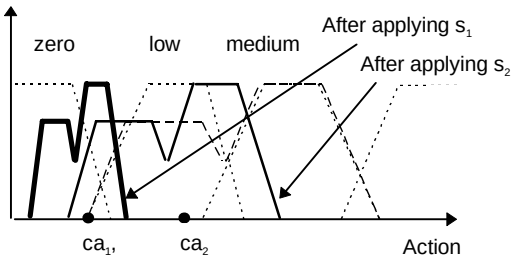

Figure 4. Squeezing of the fuzzy result action

5. Conclusions

In this paper, we have proposed an approach based on fuzzy triggers to cope with the data explosion problem. We introduce the concept of a fuzzy event representing imprecise and/or uncertain situation of interest. Then, a fuzzy trigger execution model, dealing with fuzzy events and fuzzy condition-action, is proposed. The model combines fuzzy logic features with active database capabilities to provide a high-level view of data stored in a database. A new technique for modifying fuzzy inference result, called squeezing, is proposed. It provides the cause-and-effect relationship between the fuzzy event and the fuzzy condition-action part of a trigger. An application of fuzzy triggers to alarm treatment is then presented.

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

References

[BD83] Buchanan B.G. and Duda R.O. *Principles of Rules-Based Expert Systems*. In Advances in Computers,

[2] Strictly speaking, a multiset which is defined to be set-like, but with duplicates permitted [SQL3].

1983, Vol. 22, pp. 163-216.

[BW96] Bouaziz T. and Wolski A. *Incorporating Fuzzy Inference into Database Triggers*. Research Report No TTE1-2-96, VTT Information Technology, Espoo, Finland, November 1996. Also at ftp://ftp.vtt.fi/pub/projects/rapid/f-infer-triggers.ps

[BKPW97] Bouaziz T., Karvonen J., Pesonen, A. and Wolski A. *Design and Implementation of TEMPO Fuzzy Triggers*. Research Report No TTE1-2-97, VTT Information Technology, Finland, March 1997. Also at ftp://ftp.vtt.fi/pub/projects/rapid/tempo-design.ps

[CM94] Chakravarthy S. and Mishra D. *Snoop: An Expressive Event Specification Language for Active Databases*. In Data & Knowledge Engineering , Vol. 14, 1994, pp. 1-26.

[Cod73] CODASYL Data Description Language Committee. CODASYL Data Description Language Journal Development, NBS Handbook 113, June 1973.

[Day95] Dayal U. *Ten Years of Activity in Active Database Systems. What Have We Accomplished?* Invited talk, Workshop on Active and Real-Time Database Systems (ARTDB'95), Skövde, Sweden, 1995.

[DBB+88] Dayal U., Blaustein B.T., Buchmann A.P. et al. *The HIPAC Project: Combining Active Databases and Timing Constraints*. In SIGMOD Record, 1988, Vol. 17, No 1, pp. 51-70.

[DEB89] Data Engineering Bulletin-Special Issue on Imprecision in Databases, Vol. 12, No. 2, June 1989.

[DP88] Dubois D. and Prade H. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1988.

[GD93] Gatziu**,** S. and Dittrich, K.R. *Events in an Active Object-Oriented Database System.* In Proceedings of the 1st International Workshop on Rules in Database Systems (RIDS) , Edinburg, Scotland, August 1993, pp. 23-39.

[GJS92] Gehani, N.H. & Jagadish, H.V. and Shmueli, O. *Event Specification in an Active Object-Oriented database*. In ACM SIGMOD Conference on Management of Data, San Diego, California, June 1992, pp. 81-90.

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

[Han89] E.N. Hanson. *An Initial Report on the Design of Ariel: A DBMS with an Integrated Production System.* In SIGMOD Record, Vol. 8, No. 3, Sept. 1989, pp. 12-19.

[IS89] Information Systems *(Special Issue on Fuzzy Databases).* Vol. 14, No. 6, 1989.

[KL96] George J. Klir and Yuan B. (Eds.). *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh* In Advances In Fuzzy Systems-Applications and Theory, Vol. 6, 1996, 821 pages.

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

[Men95] Mendel J. M. *Fuzzy Logic Systems for Engineering: A Tutorial*. In Proc. of the IEEE, Special Issues on Engineering Applications of Fuzzy Logic, Vol. 83, No. 3, March 1995, pp. 345 - 377.

[PBW96] Pesonen, A., Bouaziz, T., and Wolski, A. *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] Paton, N.W., Diaz, O., Williams, M.H., Campin, J., Dinn, A., and Jaim, A. *Dimension of Active Behavior*. In Proceedings of the 1st Int. Workshop on Rules in Database Systems, Edinburg (Scotland), August 1993, pp. 40-57.

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

[SQL3] Working Draft Database Language SQL3, J. Melton (ed.), August 1994, ANSI X3H2-94-329, ISO DBL:RIO-004.

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

[WC96b] Widom J. and, Ceri S. *Introduction to Active Database Systems*. In [WC96a], pp. 1-41.

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

[Zad75] Zadeh L.A. *Fuzzy Logic and Approximate Reasoning*. In Synthese, 30, 1975, pp. 407-428. (also in [KL96]).

[Zad84] Zadeh L.A. *The Role of Fuzzy Logic in the Management of Uncertainty in Expert Systems*. In [GKB+84], pp. 3-31 (also in [KL96]).

[Zad89] Zadeh L.A. *Knowledge Representation in Fuzzy Logic*. In IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1, 1989, pp. 89-100.

[Zem89] Zemankova M. *FILIP: a Fuzzy Intelligent Information System with Learning Capabilities*. In [IS89], pp. 473-486.