DEALING WITH FUZZINESS IN ACTIVE MOBILE DATABASE SYSTEMS[1]

Yücel Saygın*, Özgür Ulusoy* and Adnan Yazıcı †

*Department of Computer Engineering and Information Science, Bilkent University, Turkey.

†Department of Computer Engineering, Middle East Technical University, Turkey.

Abstract

Current needs of industry required the development of advanced database models like active mobile database systems. An active mobile database system can be designed by incorporation of triggering rules into a mobile computing environment in which the users are able to access a collection of database services using mobile and non-mobile computers at any location. Fuzzy concepts are adapted to the field of databases in order to deal with ambiguous, uncertain data. Fuzziness comes into picture in active mobile databases especially with spatial queries on moving objects. Incorporating fuzziness into rules would also improve the effectiveness of active mobile databases as it provides much flexibility in defining rules for the supported application. In this paper we present some methods to adapt the concepts developed for fuzzy systems to active mobile databases.

Key words: Active databases, mobile databases, rule execution, fuzzy databases, fuzzy triggers, fuzzy rule execution.

1 Introduction

Conventional data models developed so far are not adequate for the storage, retrieval, and processing of ambiguous, uncertain data that we come across very frequently in the real world [TAS92]. Fuzzy concepts are incorporated to the field of databases in order to support queries closer to the natural language and to model data which is inherently fuzzy. A trend in *fuzzy databases* is to extend the relational model to incorporate fuzzy concepts [Pet96].

Conventional passive databases execute queries or transactions only when explicitly requested to do so by a user or an application program. In contrast, an *active database management system* allows users to specify actions to be executed when specific events are signaled [Day88]. In order for a conventional database management system to react to certain events, it should be incorporated with *rules*. A general rule consists of an *event* that triggers the rule, a *condition* describing a given situation, and an *action* to be performed if the condition is satisfied. These types of rules are called *Event-Condition-Action(ECA)* rules.

Recent advances in computer hardware technology made it possible the production of small size computers like notebooks and palmtops which can be carried around by users. These portable

[1]This research is supported by the Research Council of Turkey (TÜBİTAK) under grant number EEEAG-246 and the NATO Collaborative Research Grant CRG 960648.

computers can also be equipped with wireless communication devices that enable users to access global data servers while traveling. A considerable amount of research has recently been conducted in *mobile database systems* area with the aim of providing efficient access to data on both stationary data servers and mobile computers. The main topics investigated are the management of location dependent data [DHB97], handling frequent disconnections [AGK+96] of mobile computers, wireless data broadcasting, energy efficient data access [IB94], transaction processing [Chr93], [EJB95], and querying in mobile environments [IB92].

Active features can be used to support different transaction models and efficient commit protocols in mobile database systems. By building rule sets, the management of advanced and long-lived transactions can be greatly simplified. Rules can also be used to handle the queries which are executed periodically. An *active mobile database management system (AMDBMS)* can be designed by incorporation of rules into a mobile database environment. We use in this paper an active mobile database platform to explain how fuzzy features can be integrated to active mobile database systems. We adapt a battlefield environment to illustrate how the proposed approaches can be made use of in a real application.

AMDBMSs is an area where fuzzy data is unavoidable as in many complex systems. Especially in the field of spatial queries on moving objects, fuzziness is very apparent since it is not feasible to track the positions of continuously moving objects. To the best of our knowledge, no research results have appeared in the literature on the incorporation of fuzziness in mobile database systems. Mobility introduces uncertainty in the location of moving objects. Condition part of the rules that are associated with AMDBMSs may include queries on the locations of moving objects. Such queries lead to the requirement of the incorporation of *fuzzy rules.* Fuzzy ECA rules differ from the conventional active database rules in that, they consist of fuzzy events and fuzzy conditions. Fuzzy events are uncertain events like "on a slight movement of an object" or "on a considerable change in the location of a moving object". Fuzzy conditions might include fuzzy queries like "retrieve all the objects which are close to a specific object belonging to the enemy".

Based on the discussion provided above, we can say that the concepts in mobile, active, and fuzzy databases can all be merged in a common platform to construct a powerful system enabling mobility of data and computers while supporting active and fuzzy features.

The primary contributions of our work are:

- to incorporate fuzziness into rule execution via fuzzy coupling modes and scenarios,
- to explain how fuzzy primitive events can be combined to form fuzzy composite events,
- to show how fuzzy concepts can be used for rule scheduling,
- and finally to investigate the possibility of supporting more flexible spatial queries on moving objects by incorporation of fuzziness.

In the next section, an introduction to fuzzy concepts and fuzzy databases is provided. Section 3 presents a mobile database system model that is incorporated with rules and fuzzy queries. In Section 4, a description of the current work on fuzzy triggers is provided together with our contributions. A discussion on fuzzy spatial queries in active mobile environments is provided in Section 5. Finally in Section 6, conclusions and future work are discussed.

2 Overview of Fuzzy Systems and Databases

Uncertain nature of queries and inherently imprecise data has necessitated the development of *fuzzy databases*. The relevant fuzzy concepts regarding fuzzy databases are discussed in the following.

2.1 Fuzzy Sets and Fuzzy Logic

The theory of fuzzy sets was introduced by Zadeh [Zad65]. For a crisp set (an ordinary set that we are familiar with) S, which is a subset of the universal set U, for any element $e \in U$, either $e \in S$ or $e \notin S$ where for a fuzzy set there is a degree of membership in the range $[0,1]$ for each element belonging the the universal set. Crisp set theory is a special case of the fuzzy set theory where the membership degrees of any element belonging to the universal set is either 0 or 1. A fuzzy set is characterized by its membership function. This membership function, gives us the degree of membership of each element in the universal set to the fuzzy set. Membership function of a fuzzy set F on the universal set U is generally denoted by μ_F and maps each element $x \in U$ to a real number in the range $[0,1]$, i.e.,

$\mu_F(x) : U \rightarrow [0,1]$.

The fuzzy set theory is best understood with real life examples. Assume that we have a universal set U for all the ages a human being can have. We can define a fuzzy set *young* denoted by Y on U, and assign a membership function μ_Y to Y. A sample membership function can be defined as:

$$\mu_Y(x) = \left\{ \begin{array}{ll} 0, & x < 10, \\ \frac{x}{10} - 1, & 10 \leq x < 20, \\ 1, & 20 \leq x < 30, \\ \frac{-x}{10} + 4, & 40 \leq x. \end{array} \right\} \tag{1}$$

Membership function μ_Y is shown graphically in Figure 1. According to that, a person with age 15 is young with a membership degree of 0.5. Calculation of the membership functions of the union, intersection, and difference of two fuzzy sets is explained in [KF88].

Fuzzy logic can be viewed as an application area of fuzzy set theory[KCY97]. We may define the degree of truth of the fuzzy proposition "x is a member of A" as the membership degree of x in A. This can be generalized to arbitrary propositions, like P :"x is F" where $x \in A$ and F is a linguistic expression such as, low, high, old, young. The degree of truth of P can be interpreted as

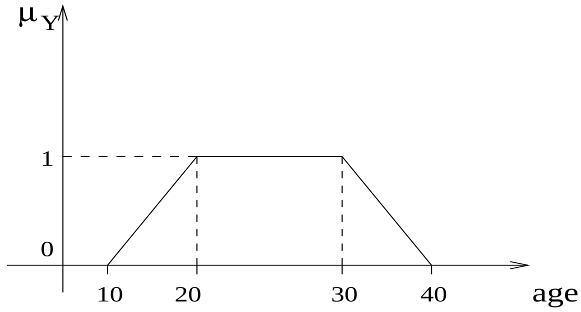

Figure 1: Membership function of the fuzzy set *young*.

the membership degree $\mu_A(x)$ where A is characterized by the linguistic expression F [KCY97]. So, using fuzzy logic, we can reason about the degree of truth of imprecise propositions. Fuzzy logic allows the use of [KF88]:

- fuzzy predicates like *old, expensive, high*,
- fuzzy quantifiers like *many, few, usually*,
- fuzzy truth values like *very true, mostly false*,
- and fuzzy modifiers like *almost, likely, extremely.*

Some examples of imprecise propositions are "John is tall is true", or "Marry is short is fairly false".

Fuzzy inference rules are the basic building blocks of a fuzzy controller(Mamdani type of control which is the most popular fuzzy control approach). In this approach, fuzzy control is performed in 4 steps [KCY97]:

1. Fuzzification
2. Fuzzy inferencing
3. Calculation of the overall conclusion
4. Defuzzification

At predefined times, the measured values of input variables are received by the controller and in the first step, the matching rules are determined. In the second step, an inference is performed by each rule that is selected. In the third step the overall conclusion is calculated, and finally in the last step, the overall conclusion is defuzzified, i.e., converted to a real value.

2.2 Fuzzy Databases

The ordinary relational database model introduced by Codd [Cod70] does not handle imprecise, inexact data well. The data that it handles is either precise or only one value, i.e., NULL, represents all possible types of imprecision such as ‘unknown’, ’not-applicable’, etc. (many types of such imprecision are cited in [ANS75]). Being incapable of handling imprecise data, this model cannot model the real world precisely.

Several extensions have been brought to relational model to capture the imprecise parts of the real world. Buckles et al. examine and compare them in their paper [BP85]. In general, three approaches are presented. The approaches mainly differ in the method they use.

The first approach uses fuzzy membership values. In this approach, a relation scheme includes a fuzzy membership attribute in addition to its normal attributes. The fuzzy membership attribute may define the membership degree of the tuple to its relation instance [GSS83], or it may determine strength of the dependency between two attributes [Bal83].

The second approach of representing imprecise data is through possibility distributions that indicate the information about the actual value of an attribute [DPR91]. Zadeh explains in his paper [Zad78] how a possibility distribution can be used in conjunction with fuzzy sets.

The third approach is the similarity-based approach. Similarity-based fuzzy relational model is not an extension to the original relational model [Cod70], but a generalization of it. It generalizes the relational model in two aspects, the allowance of a set of values for an attribute rather than only atomic values, and the replacement of identity concept with a conformance concept. For both aspects, the similarity relation is utilized. The level of similarity among the values are defined by the explicitly-defined similarity relation for the domain of the attribute values. Thus, the fuzziness of the data is well-defined in terms of its domain’s similarity relation.

3 An Active Mobile Database System

There is a wide spectrum of applications of active mobile database management systems (AMDBMSs) from military to health and insurance. One such application in military is the management and control of vehicles in a battlefield environment [Buk97], [MBM96]. In health, an active mobile computing system can be designed to reach the patients’ previous records in the hospital from the moving ambulances [MB96].

A typical architecture for mobile computing systems which is inspired from [IB94] is depicted in Figure 2. In this architecture, there is a fixed network of Mobile Support Stations (MSSs). Mobile Hosts (MHs) are the computers which are portable and capable of mobile communication. Each MH is associated with a MSS and MHs are connected to MSSs via wireless links. An MSS is generally fixed and it provides MHs with a wireless interface inside a prespecified area called a cell. Location management of transactions submitted to MHs is performed by MSSs. Transaction management can be performed by MSSs and/or MHs depending on the particular system.

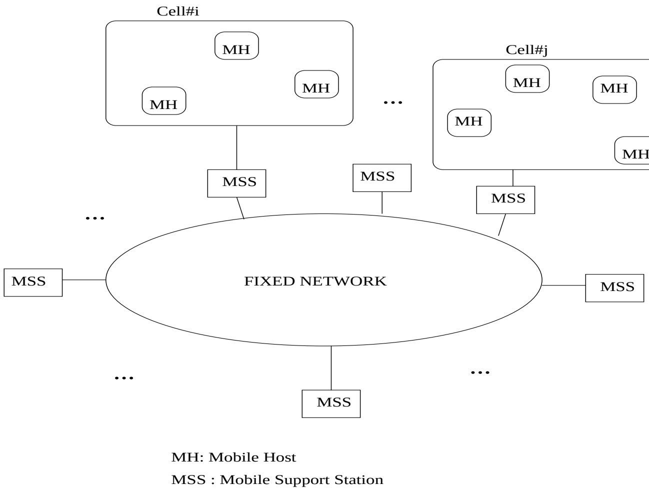

Figure 2: A general architecture of a mobile computing system

As an example application, a battlefield environment can be coordinated using a system based on the architecture provided in Figure 2 where the vehicles on land and aircrafts are moving objects which are also capable of issuing queries, i.e., they are MHs. In the fixed network there is a database management system supported with rules like:

- **event:** obj_1 is *very close* to obj_2
  condition: obj_1 belongs to enemy and obj_2 belongs to the alliances
  action: fire an alarm and inform obj_2

- **event:** send missile
  condition: there are objects *close* to the target that belong to the alliances
  action: move away those objects

- **event:** SOS signal
  condition: object sending the signal belongs to the alliances and it is not *very far*
  action: send an available team which is *close* to it

- **event:** obj_1 is *very close* to obj_2
  condition: obj_1 belongs to enemy and obj_2 belongs to the alliances
  action: send an available team which is *close* to it

The first rule has a fuzzy event that contains a fuzzy term *very close*. The second rule has a fuzzy condition that checks some properties of objects that are *very close* to a specific location, the third rule has a fuzzy condition and a fuzzy action, and finally the last rule contains a fuzzy event, a fuzzy condition and an action containing a fuzzy term.

Such kind of rules can be written by the people who are familiar with the war scenarios and the situations that may occur in a war. An obvious property of the rules listed above is that they are close to natural language, and therefore very easy to write for the experts of war scenarios who are not much familiar with data management issues. These rules involve fuzzy queries on the database and some of them have fuzzy events.

4 Fuzzy Rules in Active Mobile Database Systems

Although incorporating fuzziness to active databases introduces much flexibility, not much attention has been paid so far to this issue. To the best of our knowledge, only a research group in VTT (Finland) has worked on fuzzy triggers [BW96, BW97, BKPW97, WB98]. In [BKPW97], a Condition-Action(CA) fuzzy trigger is proposed which means that fuzziness is introduced to the CA part of an ECA rule. In a later work [BW97], the concept of CA trigger is extended to a fuzzy ECA rule by introducing the notion of fuzzy events. A CA fuzzy trigger consists of a fuzzy predicate (i.e., a predicate that has linguistic hedges) on the database as its condition, and a fuzzy action which is an overall conclusion obtained after evaluating fuzzy conditions. Wolski et al. compiled their previous work on fuzzy ECA rules and based their contributions on a sound theoretical background in [WB98]. A rule with a fuzzy condition and a crisp (i.e., not fuzzy) action is called a C-fuzzy trigger. The C-fuzzy trigger model is based on linguistic hedges. Max-min inference method is applied to the rule set to determine the truth value of the fuzzy predicates. In fuzzy ECA rules, an event may fire a set of rules. Fuzzy events are defined as fuzzy sets and use linguistic hedges like *high, low,* and *strong* [BW97]. Formally a primitive fuzzy event is represented as a tuple $< e_c, e_f >$ where e_c is a crisp event, and e_f is a fuzzy event predicate. When a crisp event is signaled (such as a database update), the current value v produced upon the operation causing the crisp event is fed into the membership function of e_f. The output of the membership function is called the event match factor, and the fuzzy event is signaled only if the event match factor is greater than zero [BW97]. Upon the signaling of the fuzzy event, the corresponding rules are fired and their conditions (which are fuzzy predicates on the database) are checked. The action of a rule may be started to execute depending on the result of condition evaluation.

The methods discussed in [BW97] and [BKPW97] introduce, what we call, *intra-rule* fuzziness to active databases, i.e., they try to incorporate fuzziness into the event and condition of a rule. We have a more global approach which we call *inter-rule fuzziness*, meaning that we deal with the rules belonging to particular fuzzy sets, together with the coupling modes and scheduling of rules. Our approach divides the whole set of rules in the system into subsets (not necessarily disjoint). Each of those subsets are actually fuzzy sets and represent a particular scenario, like *emergency*, or *normal*. Rules that belong to a scenario with a degree of membership are calculated via the membership function of that scenario. The rest of this section is devoted to the detailed discussion of our inter-rule fuzziness approach.

4.1 Fuzzy Events

The Event component of an ECA rule is the first place to look for in order to introduce fuzziness. Events can be centralized or distributed. Distributed events and distributed event detection is explained in [KL98]. In this work we will concentrate on centralized events. There exists a considerable amount of work on categorizing events and event composition. Different events and their categorization together with composite events are explained in [GGD94], [CM91], and [Buc94]. Among these references, the most comprehensive event set and composition semantics are provided in [Buc94]. Primitive events are categorized in [Buc94] as:

- method execution events,
- state transition events,
- temporal events,
- transaction and flow-control events, and
- abstract events.

Method execution events are raised when the specified methods are executed. Firing of such an event can be done before or after the method execution depending on the event specification. These kinds of events are applicable to object oriented systems. Assume that we have a missile object M which has a method $fire(target)$ that causes the missile to be sent to the specified target. When the method $M \rightarrow fire(target)$ is executed, rules whose event is "firing of a missile" are executed. State transition events are signaled when the corresponding state changes occur in the database, for example location updates of moving objects. Temporal events are either absolute or relative. An absolute temporal event is something like, "at 13:45", and a relative temporal event is like "5 seconds before the firing of a missile". Transaction and flow-control events are related with the beginning, commit and abort of transactions. Finally, abstract events are defined by user and therefore signaled explicitly by the user. Abstract events are useful when event signaling is disabled and events should be explicitly issued by the user to fire some rules [Buc94].

Method execution events, state transition events, and temporal events are important from the point of fuzzy rule execution, since there is a high level of potential for incorporating fuzzy concepts into those kinds of events.

Method execution events which are applicable only for object oriented systems can be fuzzified (i.e., converted to fuzzy events) by incorporating them with membership degrees. This can be done by utilizing the membership degrees of the attributes that the method uses. If the underlying database is a Fuzzy Object Oriented Database Management System as in [YG98], then the attributes of objects are viewed as fuzzy sets and each attribute has a degree of membership to the object it belongs to. So the membership degree of a method m can be calculated as:

$$\mu_m = \frac{\sum_{i=1}^{i=n} \mu_{a_i}}{n}$$

where μ_{a_i} is the membership degree of attribute a_i that is being used by method m, assuming that there are n attributes, namely $\{a_1, a_2, ..., a_n\}$ used by method m. If the method does not use any attributes, then its membership value is taken to be 1.

The membership degree for method m is used by the fuzzy method execution event on m in determining the rules to be fired as we will explain in Section 4.2.

Temporal events are widely used in many active database systems and can be applied to critical jobs in real time systems. Fuzzy concepts can be incorporated to temporal events by adding fuzzy modifiers to exact time values. For example, instead of the absolute temporal event, "at 13:43", we may have a fuzzy absolute temporal event like, "at about 13:43" which is more flexible. Relative temporal events can also be modified in order to convert them to fuzzy relative temporal events. For example a relative temporal event like "10 seconds after the commit" can be modified as "a short time after the commit" where "short time" is a fuzzy term. It is actually better to use fuzzy temporal events since determining the exact times in advance may not be feasible in some cases. Calculating the membership degrees of fuzzy temporal events can be done using the membership functions of the fuzzy terms and the concept of fuzzy numbers which is explained in more detail in [KF88]. Membership degree of crisp events is taken as one.

Primitive events can be combined to form composite events. Composition of primitive events can be done with different event constructors, like conjunction, disjunction, closure, sequence, history, and negation [Buc94, GGD94, CM91]. Disjunction of two events E_1 and E_2 is raised when one of E_1 or E_2 is raised. Conjunction of two events E_1 and E_2 is raised when both E_1 and E_2 have occurred, regardless of the order of occurrence. Sequence is similar to conjunction but the order of occurrence of the events is important with sequence. Closure constructor is used when multiple occurrences of the same event in a period of time (such as, during the execution of a transaction) is considered together as a composite event. History event constructor is a more restricted case of the closure event constructor where the number of occurrences of the same event is specified. Negation of an event can also be considered as a composite event and it is raised when the negated event has not occurred in a specified period of time. Events composed by multiple event constructors are composite events as well, which can be represented by a tree of composite events where the primitive events are at the leaves and constructors are the internal nodes.

Fuzzy composite events can be constructed by combining crisp primitive events listed above and fuzzy primitive events (i.e., fuzzy temporal, fuzzy state change, and fuzzy method execution events). The membership values of fuzzy composite events can be calculated depending on the semantics of the event constructors. In case of the conjunction event constructor, the event with minimum membership degree among the component events is selected, and its membership degree determines the membership degree of the composite event. When disjunction is used as the event constructor, then the maximum membership value among the membership degrees of the component events determines the membership degree of the composite event. In case of negation, the membership degree, μ_n, of the composite event is calculated as,

$$\mu_n = 1 - \mu_e$$

where μ_e is the membership degree of the event being negated.

Computation of the membership degrees of composite events constructed by history and closure is done by using the following formula:

$$\mu_c = \frac{\sum_{i=1}^{i=n} \mu_{e_i}}{n}$$

where μ_c is the membership degree of the composite event, μ_{e_i} is the membership degree of the i^{th} occurrence of event e, and n is the number of occurrences of event e. Here we should note that different occurrences of the same event may result in different membership degrees depending on the crisp parameter of the event. Membership degrees of the composite events formed by the sequence constructor are computed similar to that of the conjunction constructor.

We define the strength of a primitive or fuzzy event as the membership degree of the corresponding event parameter. For example, an event like "obj_1 is close to obj_2" can have different strengths depending on how close obj_1 is to obj_2 in a particular situation. Closer the objects, stronger is the fuzzy event.

Complexity of composite event structures may cause some problems in event detection. Let's consider two events E_1 and E_2 combined by the conjunction constructor to form a composite event E and three events occur in the sequence, e_1, $e_1\prime$, e_2 where e_1 and $e_1\prime$ are two instances of the same event, E_1, and e_2 is an instance of E_2. In this case we may take either (e_1, e_2) or $(e_1\prime, e_2)$ as the instance of the composite event E. Determining which instance to use in the composition is a problem. Our solution to this problem would be to choose the instance which has the highest membership degree, that way increasing the strength of the composite event. This method associates priorities with the events in some sense according to their membership degrees.

4.2 Inter-rule Fuzziness Via Scenarios

There may exist a finite set of events that can be signaled in an AMDBMS. We partition the whole event set E into event groups called scenarios (not necessarily disjoint). The idea of scenarios comes from the need to group rules into sets corresponding to different situations. Formally:

Definition 4.1 *Let R be the set of all the rules in a system, then a scenario S_k is a subset of R, i.e., $S_k \subseteq R$. The scenarios in the system are not necessarily disjoint.*

There can be only one active scenario at a time. Switching among scenarios is performed by rules as well. Consider the battlefield application we discussed in Section 3, where there can be *emergency* situations as well as *normal* situations. An emergency situation corresponds to the events which may have serious effects like a serious damage and should urgently be handled whereas a normal situation corresponds to the events with a low level of importance. Switching from a normal scenario to an emergency scenario is performed by rules which detect emergency situations. Each rule may be subscribed to more than one scenario. If a rule is not subscribed to a scenario, then it is called an *idle rule*. Each scenario, S_k, behaves like a fuzzy set, i.e., it has a membership function, μ_{S_k}, that maps the rules to a real number in the range $[0, 1]$. Events belonging to a scenario are fuzzy events as described in Section 4.1. Event signaling is done by considering the membership degree of the event parameter in the fuzzy event. The fuzzy event structure described in [BW97] is utilized where a primitive event is a tuple, $e :< e_c, e_f >$, consisting of a crisp part e_c which is the crisp parameter coming from the system and a fuzzy part e_f which denotes the fuzzy event set.

Definition 4.2 *Let, $r :< e, c, a >$ denote a rule r with event e, condition c, and action a. The strength of an event $e :< e_c, e_f >$ for the rule r in scenario S is defined as :*

strength(e,r) $= \mu_S(r) * \mu_{e_f}(value(e_c))$

where $value(e_c)$ is the value of the crisp event detected, μ_{e_f} is the membership function of the fuzzy event e_f, and μ_S is the membership function of scenario S.

Each rule has a firing threshold which is used to decide if a rule will be fired or not. In order to decide whether a rule r will be fired in response to the signaling of a fuzzy event e, the strength of event e for rule r is calculated and result is compared with the threshold value for rule r. If the result is greater than or equal to the threshold value, then the rule is fired. Threshold values of rules can be changed dynamically to tune to particular scenarios.

Assume that in our battlefield application, we have *emergency* and *normal* scenarios which are considered to be fuzzy sets with membership functions $\mu_{emergency}$ and μ_{normal}. Each rule belongs to one or two of the scenarios with a membership degree. Assume that the current scenario is *emergency*. Consider the following rule denoted with r_{alarm} :

event: obj_1 is very close to obj_2
condition: obj_1 belongs to enemy and obj_2 belongs to the alliances
action: fire an alarm and inform obj_2

which belongs to the *emergency* scenario with a membership degree, $\mu_{r_{alarm}} = 0.9$. Assume that its event is signaled, and the distance between obj_1 and obj_2 is 2 kilometers which is also the value of the crisp event, e_c. The fuzzy event, e_f is *close* and $\mu_{close}(r_{alarm}) = 0.7$. The strength of the fuzzy event for rule r_{alarm} is calculated as: $0.9 * 0.7 = 0.63$. If r_{alarm} has a threshold value 0.6 for that scenario, then it will be fired since $0.63 \geq 0.6$.

The threshold parameters and the membership functions for the fuzzy rules can be determined according to the results of a priori simulations.

4.3 Similarity Based Event Detection

Signaling of similar events upon an event detection is something very useful when the cost of missing events is very high in supported applications, like a nuclear reactor control system. Assume that an event such as *update in temperature level* is detected. Events with a high similarity degree, like *update in pressure level* should also be signaled automatically. This way, the risk of events escaping from detection is reduced. Similarity relations as defined by Zadeh [Zad70] are utilized in similarity based event detection.

Similarity relations are used for describing how similar two elements from the same domain are, as the name implies. Given two elements, the similarity relation maps these two elements into an element in the interval $[0, 1]$. The more similar two elements are, the higher the value of the mapped element. If the two elements are the same, that is, if we compare an element with itself, the mapped element is 1, the highest possible value. The similarity values for pairs of elements are stored as similarity matrices as shown in Example 4.1. An ordinary relation is considered to be a similarity relation when it satisfies the three conditions stated below.

Definition 4.3 *A similarity relation is a mapping, $s : D \times D \rightarrow [0, 1]$, such that for $x,\ y,\ z \in D$,*

$$
\begin{array}{ll}
s(x, x) = 1 & \textit{(reflexivity)}, \\
s(x, y) = s(y, x) & \textit{(symmetry)}, \\
s(x, z) \geq \max_{y \in D}(\min(s(x, y), s(y, z))) & \textit{(max-min transitivity)}.
\end{array}
$$

Example 4.1 *Let for a domain D, we have $D = \{e_1, e_2, e_3, e_4\}$. We define a relationship s for domain D, such that:*

s	e_1	e_2	e_3	e_4
e_1	1	0.8	0	0
e_2	0.8	1	0	0
e_3	0	0	1	0.7
e_4	0	0	0.7	1

Relation s satisfies the three conditions stated in definition 4.3. Thus, it is a similarity relation.

In similarity based event detection, when an event is signaled, other events which are similar to it should also be fired. This is done only in primitive event detection level. In order to facilitate this, a similarity matrix is needed as shown in Example 4.1 (where e_1,...,e_4 are the events in the system). This similarity matrix designates a similarity relation among the events. We also need similarity thresholds in order to avoid the system to continuously detect irrelevant events via similarity based event detection.

Definition 4.4 *Similarity threshold for a scenario is the minimum similarity requirement for similarity based event detection for that particular scenario.*

Value of an event, e_2 detected by similarity based event detection is calculated as: $value(e_2) = value(e_1) * s(e_1, e_2)$, where e_1 is the event that caused the signaling of e_2, and $s(e_1, e_2)$ is the similarity value between events e_1 and e_2.

An example would be helpful in explaining similarity based event detection. Assume that event e_1 is raised. Other events whose similarity to e_1 is greater than or equal to the similarity threshold for the current scenario also need to be considered. If, for example, the similarity threshold for a scenario s is 0.7, and e_1 is signaled (which belongs to s) and another event e_2 is similar to e_1 with degree 0.8, then event e_2 should also be signaled since $0.8 \geq 0.7$. But the membership value of e_2 is multiplied by its degree of similarity (in this case 0.8) in order to determine which rules are going to be fired as a result of e_2.

Conventional event detection in active databases is a special case of similarity based event detection where the similarity relation among the events is an identity relation and similarity thresholds are equal to one.

As an overall view, the whole rule set R is divided into scenarios, S_i each of which is a set of rules, where $S_i \subseteq R$. The system has a similarity matrix M which shows the degree of similarity among events in that scenario. Similarity matrix, M, which shows the similarities between events in a pairwise manner can be provided by the experts of the particular application; in our application they are military experts. Similarity matrix can be dynamically constructed and updated by the system via examining the event history. Signaling of two events consecutively in a short period of time implies that those events may be similar. As the consecutive signaling of two events is seen more frequently in the event history, the similarity of these events should be increased in the similarity matrix. This way, system learns the similarity values as the event history grows.

Grouping of rules into scenarios restricts the number of rules to be considered when an event is raised, improving the efficiency of rule execution especially in case of emergency when efficient use of resources is very important.

4.4 Fuzzy Coupling Modes

In ECA rules coupling modes between event and condition, and between condition and action determine when the condition should be executed relative to the occurrence of the event, and when the action should be executed relative to the satisfaction of the condition, respectively. There are three basic coupling modes: *immediate*, *deferred*, and *detached* (or *decoupled*) [Day88]. If the condition is specified to be evaluated in *immediate* mode, then it is executed right after the triggering operation that caused the event to be raised. If the action part is specified to be executed in immediate mode then it is executed immediately after the evaluation of the condition. In case the condition is specified to be in *deferred* mode, its evaluation is delayed until the commit point of

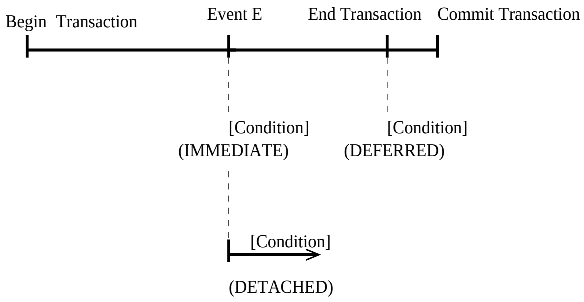

Figure 3: Basic coupling modes between event and condition.

the transaction, and similarly if the action is in deferred mode relative to the condition, again it is executed right before the transaction commits. Finally, in *detached mode*, condition is evaluated or action is executed in a separate transaction. Basic coupling modes between the event and condition are illustrated in Figure 3.

Coupling modes is a very important concept for rule execution in active database systems and should also be considered for fuzzy rule execution. In fuzzy ECA rules, the coupling modes between event and condition, and between condition and action can be determined depending on the strength of the event as defined in Section 4.2 and *credibility* of the condition respectively in case the coupling mode is not specified explicitly. We define the *credibility* of a condition as the truth value of the fuzzy predicate or the combination of the fuzzy predicates. Determination of the truth values of the fuzzy predicates is explained in [KF88]. The truth value of a simple fuzzy predicate like *"x is P"* is $\mu_P x$ where P is a linguistic variable like *young, high*, or *close*. Max-min inference method can be used to determine the truth values of complex predicates composed by using logical *and*, or logical *or* operators:

$Truth(P_1 \wedge P_2) = Min\{Truth(P_1), Truth(P_2)\}$

$Truth(P_1 \vee P_2) = Max\{Truth(P_1), Truth(P_2)\}$

A high credibility implies immediate or detached coupling mode and a low credibility implies deferred coupling mode in case the coupling modes are not specified explicitly. Each coupling mode should be assigned, what we call a *credibility threshold* which is used to determine the coupling mode between the event and condition, and condition and action. That way, implicit priorities are assigned to the condition and action depending on the strength of the corresponding event and credibility of the condition. Strength of an event signaled due to its similarity to another event is calculated as explained in Section 4.3. Assume that, in an emergency scenario, two of the events are e_1 and e_2. If the strength of e_1 for rule r_1 is 0.8, the strength of e_2 for another rule, r_2 is 0.4. Assume also that threshold values for immediate, detached and deferred coupling modes are, 0.7, 0.5, and 0.0[2], respectively. If both of e_1 and e_2 are signaled, then condition of rule r_1 will be evaluated in immediate mode where the condition of rule r_2 will be evaluated in deferred mode.

[2]A value greater then zero as a credibility threshold for deferred mode means that some rules may not be fired even in deferred mode.

The notion of credibility can also be used in scheduling of condition evaluation and action execution in case of concurrent execution of rules as will be explained in Section 4.5.

A more realistic example for the mobile battle field environment can be given by a rule with the event "obj_1 is very close to obj_2". If no coupling mode was assigned for the rule and the credibility thresholds for immediate mode is 0.9, for detached mode 0.7, and for deferred mode 0.5. If the strength of the event is 0.95 which means that when obj_1 gets very close to obj_2, then the condition should be evaluated immediately, suspending the transaction that signaled the event. But if the strength of the event is 0.6 then the evaluation of the condition can be deferred to the end of the transaction since obj_1 is not dangerously close to obj_2.

4.5 Concurrent and Sequential Fuzzy Rule Execution

An AMDBMS should support both concurrent and sequential rule execution. Sequential rule execution is necessary when a certain execution order is enforced by priorities or when the rules have a predefined sequence of execution. Sequential execution may also be supported in levels; i.e., a number of groups of rules can be executed sequentially while the rules in each group are executed concurrently. Concurrent rule execution is very important from the performance perspective of the system. Concurrency in rule execution can be achieved through either:

- inter-rule concurrency, or
- intra-rule concurrency, or
- both inter and intra-rule concurrency.

In the first case, rules are executed concurrently as if they are atomic transactions. In the second case, rules are divided into subcomponents and those subcomponents are executed concurrently. As another alternative, we may have both types of concurrency together, which is the most flexible concurrent rule execution model [SUC98].

Fuzzy rules can be executed both sequentially and concurrently. Sequential fuzzy rule execution can be done by assigning appropriate priorities to rules. Priorities for fuzzy rules are assigned according to the membership degrees of the corresponding events and conditions similar to the case of fuzzy coupling modes. As a result, the rule whose event has the highest membership degree is executed first, and the rule whose event has the next highest membership degree is executed next, and so on. This priority assignment scheme is dynamic and changes even for the same rule set at different times since the event instances which are used for calculating the membership degree of that event may not be the same for the same event at different times.

Sequential rule execution is suitable for similarity based event detection explained in Section 4.3. Priority assignment scheme that is based on the strength of events favors rule r_1 with event e_1 fired by an actual event to rule r_2 with event e_2 fired due to similarity based event detection. This is due to the fact that the strength of rule r_2 is determined by multiplying the strength of the actual event e_1 with the similarity value between e_1 and e_2 which is less than one.

Another execution method which is semi-sequential can be utilized by using the membership degrees as well. As we mentioned before, in sequential rule execution, priorities are assigned to rules according to the strength of their events (i.e., event's membership degree). According to this scheme, the rule with the strongest event is executed first and the rule with the next strongest event is executed next. This scheme can be modified by separating the condition and action of a rule considering them separately for rule execution. With respect to this scheme, the priority of the condition of a rule can be determined by the strength of the event, and the priority of the action is determined by the strength of the condition where strength of a condition is determined by the degree of its truth when it is specified as a fuzzy predicate. That way, the priority of the action of a rule is determined after the evaluation of its condition, and the execution time of the action depends on the new priority value.

Concurrent execution of fuzzy rules is similar to the concurrent execution of crisp rules which can be supported via nested transactions [Mos85]. Depending on the coupling mode between event and condition, and condition and action, a whole rule can be divided into its condition and action which can be executed concurrently. The fuzzy coupling mode determination scheme discussed in Section 4.4 can be used to decide on the coupling modes dynamically. Sequential and concurrent fuzzy rule execution can be combined by executing the fuzzy rules with a higher priority before the rules with lower priorities and executing rules with the same priority concurrently.

5 Fuzzy Spatial Queries in Active Mobile Database Systems

An important functionality of an AMDBMS is to be able to process spatial queries in an efficient manner. Fuzziness comes to picture for spatial queries since it is very hard to determine the exact positions of the mobile hosts or moving objects in general. Modeling imprecision by assigning a velocity attribute to moving objects is described in [WCD+97] where update frequency of the locations of moving objects is determined as a function of the ratio between the update cost and the cost of the imprecision in answering queries. In [IB92], querying in mobile environments is discussed where a certain degree of imprecision[3] on the locations of the mobile objects is allowed. In order to bound the imprecision, partitions are defined on the whole area in concern depending on the user profiles (statistics on the user behavior, like movement, frequency of connection from specific areas in specific times of the day, etc.). In both works mentioned, the notion of imprecision in mobile systems is discussed but none of these works make use of fuzzy concepts in order to deal with uncertainty.

Fuzzy spatial queries are the queries that include fuzzy terms in order to describe the location of the moving objects. Some sample fuzzy spatial queries are: "retrieve the positions of all the

[3] They call it "ignorance".

tanks *near* lake Van", and "find the objects which are *very close* to obj_1". The fuzzy terms in these queries are *near* and *very close*. There are many research results on the area of spatial databases, including its modeling and querying aspects (see [Gut94] for an overview). There are proposals for extending the relational query language SQL to support fuzzy queries. The area of fuzzy spatial queries is also investigated by some researchers to incorporate fuzziness into spatial queries [MP98].

In order to support fuzzy queries in an AMDBMS, we need to make use of the concepts developed by the fuzzy database researchers. This can be achieved either by building our system on top of a fuzzy DBMS, or making our system capable of processing fuzzy queries. We believe that building a system on top of a fuzzy DBMS is more advantageous in the sense that we can store fuzzy values, and issue fuzzy queries in a natural way. There exists a considerable amount of research conducted in fuzzy databases and fuzzy queries [Pet96, YG98] which can be adapted to the active mobile database research, especially in location management field as location data is inherently uncertain due to mobility and update costs.

The need for supporting fuzzy features in AMDBMSs arises from the following observations:

- It is hard to identify objects; e.g., in a battle field environment, it is very hard to determine the class of an object.
- object positions change frequently since objects are moving,
- objects' status may change; e.g., in a battle field environment the status of objects may change due to accidents, or destroyals.

With the incorporation of fuzziness into spatial queries, user would have more flexibility in writing the queries. Instead of specifying exact distance values of objects, he/she can use fuzzy terms like *close*, near, etc. Result of a fuzzy query is the superset of the corresponding crisp query which means that the user would be supplied with more options.

There exist different types of uncertainties handled in a fuzzy database system [YG98]:

- *incomplete*, that stands for range valued data,
- *null*, which represents the data that does not exist, the data is unknown, or simply not applicable,
- *fuzzy*, which is used for representing imprecise data, which is specified in descriptive terms.

Among these three types of uncertainties, *incomplete* and *fuzzy* seem to have the utmost importance for location management while *null* values can also be used. Location data can be represented either relatively or absolutely. Absolute representation of a location is provided by giving the exact coordinates of a moving object, while relative representation assumes the location data to be given relative to a fixed object. An example of relative location data is: "100 meters west of lake Van".

It is almost impossible to determine the exact position of a moving object (especially if it is an object moving very fast, like a plane), therefore we may only know the range of absolute data (by giving lower and upper bounds to the coordinates) or a range or fuzzy value for relative location data (by saying that the moving object is near a fixed object, or the relative position to a fixed object is bounded by some values). Null values *dne* (does not exist), *ni* (no information), *unk* (unknown) can also be used in AMDBMSs. Null value *dne* is used when the information about the corresponding object does not exist. Null values *ni* and *unk* are used when we do not have information about the object and when the information is unknown, respectively. We can explain the use of null values in a battlefield environment where there exist lots of aircrafts flying and vehicles moving on the ground. Some of the moving objects may even belong to the enemy (or enemies). These objects may go out of radar detection boundaries which means that their location is unknown, or they may be destroyed by weapons which means that their location does not exist. For some objects that are lost, meaning that we do not know whether they exist or not, we may place no information as their location data.

Fuzzy spatial queries may be utilized in the condition parts of ECA rules in AMDBMSs. An example rule can be constructed as:

event: a short time after the appearance of an enemy plane

condition: if there are objects whose status is *dne*

action: send the closest team for help to those locations

In this rule, a fuzzy spatial query is constructed as the condition which retrieves the objects that disappeared probably because of an enemy attack. The action part of the rule sends the closest team to the corresponding location for help.

More flexible rules can be constructed via fuzzy spatial queries. In case a rule needs to consider the vehicles around a specified area, it is very hard to determine the exact boundaries. Therefore the condition of the rule may contain a fuzzy spatial query like, "retrieve all the vehicles that are close to Lake Van".

Fuzzy rule execution methods discussed in Section 4 can be applied for rules with conditions as fuzzy spatial queries. Fuzzy spatial queries return a set of objects or tuples depending on the underlying database model. The credibility of a fuzzy spatial query, Q, can be formulated as:

$$Credibility(Q) = \frac{\sum_{i=1}^{i=n} Credibility(O_i)}{n}$$

where $Credibility(O_i)$ is the credibility of object (or tuple) O_i in the condition part of Q, and n is the number of objects returned by the query.

6 Conclusion

In this paper we have discussed a variety of issues in adapting fuzzy database concepts to an active mobile database system which incorporates active rules in a mobile computing environment. We have shown how fuzziness can be introduced to different aspects of rule execution from event detection to coupling modes. As the initial step, membership degree calculation for various types of composite events has been explained. Some interesting research issues have been raised mostly on the incorporation of membership degrees for the dynamic determination of coupling modes of rules and priority assignment. Dynamic determination of coupling modes has been done using the strengths of events and credibilities of conditions which are calculated via membership functions. Strengths of events and condition credibilities have been shown to be useful for condition and action scheduling as well. Partitioning of the rule set into scenarios has also been discussed as an example of inter-rule fuzziness. Similarity based event detection has been introduced to active mobile databases which is an important contribution from the performance perspective. Fuzzy spatial queries have been discussed briefly to show how fuzzy concepts can be utilized for supporting more flexible spatial queries in mobile computing environments.

The research conducted on the incorporation of fuzzy concepts into active and mobile databases is very new. As a future work, the concepts developed for the incorporation of fuzziness into active mobile databases can be put to practical use in a real application to measure the effectiveness of the proposed methods. Another important issue that needs further investigation is the determination of membership functions for the scenarios and threshold values for the coupling modes. All such parameters of an active mobile database system can be determined for a particular application through a performance work. Incorporation of fuzziness into distributed events can be performed as a future work. Finally, due to frequent changes in the positions and status of objects in an active mobile database environment, the issue of temporality should be considered by adapting the research results of temporal database systems area into active mobile databases.

References

[AGK+96] G. Alonso, R. Gunthor, M. Kamath, D. Agrawan, A. El Abbadi, and C. Mohan. Exotica/fmdc: A workflow management system for mobile and disconnected clients. *Distributed and Parallel Databases*, 4:229–247, 1996.

[ANS75] ANSI/X3/SPARC. Study group on database management systems: Interim report. *FDT, Bulletin of ACM SIGFIDET*, 7(2), February 1975.

[Bal83] J. F. Baldwin. Knowledge engineering using a fuzzy relational inference language. In *Proceedings of IFAC Conference on Fuzzy Information, Knowledge Representation,a dn Decision Processes*, pages 15–23, Marseille, France, 1983.

[BKPW97] Tarik Bouaziz, Janne Karvonen, Anton Pesonen, and Antoni Wolski. Design and implementation of tempo fuzzy triggers. Technical report, VTT Information Technology, 1997.

[BP85] B. P. Buckles and F. E. Petry. Uncertainty models in information and database systems. *Journal of Information Science*, 11:77–87, 1985.

[Buc94] Alejandro Buchmann. Active Object Systems. In Asuman Dogac, M. Tamer Ozsu, Alex Biliris, and Timos Sellis, editors, *Advances in Object-Oriented Database Systems*, pages 201–224. Springer-Verlag, 1994.

[Buk97] S. Morton O Bukhres. Mobile computing in military ambulatory care. In *The 10th IEEE Symposium on Computer-Based Medical Systems(CBMS'97)*, 1997.

[BW96] Tarik Bouaziz and Antoni Wolski. Incorporating fuzzy inference into database triggers. Technical report, VTT Information Technology, P.O. Box 1201, November 1996.

[BW97] Tarik Bouaziz and Antoni Wolski. Applying fuzzy events to appoxiamte reasoning in active databases. In *Proc. Sixth IEEE International Conference on Fuzzy Systems(FUZZ-IEEE'97)*, Barcelona, Catalonia, Spain, July 1997.

[Chr93] P. K. Chrysanthis. Transaction processing in a mobile computing environment. In *Proceedings of the IEEE Worskhop on Advances in Parallel and Distributed Systems*, pages 77–82, October 1993.

[CM91] S. Chakravarthy and D. Mishra. An event specification language (snoop) for active databases and its detection. Technical report, University of Florida at Gainesville, September 1991.

[Cod70] E. R. Codd. A relational model of data for large shared data banks. *Communications of ACM*, 13(6), 1970.

[Day88] Umeshwar Dayal. Active Database Management Systems. In *Proceedings of the Third International Conference on Data and Knowledge Bases*, pages 150–169, Jerusalem, June 1988.

[DHB97] Margaret H. Dunham, Abdelsalam Helal, and Santosh Balakrishnan. A mobile transaction model that captures both the data and movement behavior. *ACM/Baltzer Journal on Special Topicsin Mobile Networks and Applications(MONET)*, 1997.

[DPR91] D. Dubois, H. Parade, and J. P. Rossazza. Vagueness, typicality, and uncertainty in class hierarchies. *International Journal of Intelligent Systems*, 6:167–183, 1991.

[EJB95] Ahmed Elmagarmid, Jin Jing, and Omran Bukresh. An efficient and reliable reservation algorithm for mobile transactions. In *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM'95)*, 1995.

[GGD94] Stella Gatziu, Andreas Geppert, and Klaus R. Dittrich. The samos active dbms prototype. Technical report, Institut fur Informatik, Universitat Zurich, October 1994.

[GSS83] C. Giardina, I. Sack, and D. Sinha. Fuzzy field relational database. Technical report, Electrical Engineering and Computer Science Department, Stevens Institute of Technology, Hoboken, N.J., October 1983.

[Gut94] Ralf Hartmut Guting. An introduction to spatial database systems. *VLDB Journal*, 3(4), October 1994.

[IB92] T. Imielski and B. R. Badrinath. Querying in highly distributed environments. In *Proceedings of the 18th VLDB Conference*, Columbia, Canada, 1992.

[IB94] Tomasz Imielinski and B. R. Badrinath. Mobile wireless computing: Challenges in datata management. *Communications of the ACM*, pages 19–27, October 1994.

[KCY97] GEorge J. Klir, Ute H. St. Clair, and Bo Yuan. *Fuzzy Set Theory Foundations and Applications*. Prentice Hall PTR, 1997.

[KF88] George J. Klir and Tina A. Folger. *Fuzzy Sets, Uncertainty and Information*. Prentice Hall, 1988.

[KL98] Arne Koschel and Peter C. Lockemann. Distributed events in active database systems: Letting the genie out of the bottle. *Data and Knowledge Engineering*, 25(1):11–28, 1998.

[MB96] S. Morton and O. Bukhres. Mobile transaction recovery in distributed medical databases. In *LASTED Eighth International Conference on Parallel and Distributed Computing and Systems*, 1996.

[MBM96] S. Morton, O. Bukhres, and M. Mossman. Mobile computing architecture for a battlefield environment. In *International Symposium on Cooperative Database Systems for Advanced Applications*, 1996.

[Mos85] E. Moss. *Nested Transactions*. M.I.T. Press, Cambridge, Mass., 1985.

[MP98] Ashley Morris and Frederic E. Petry. Design of fuzzy querying in object-oriented spatial data and geographic information systems. In *National Association of Fuzzy Information Processing Society (NAFIPS'98)*, Florida, August 1998.

[Pet96] Frederick E. Petry. *FUZZY DATABASES Principles and Applications*. Kluwer Academic Publishers, 1996.

[SUC98] Yucel Saygin, Ozgur Ulusoy, and Sharma Chakravarthy. Concurrent rule execution in active databases. *Information Systems*, 23(1), 1998.

[TAS92] Toshiro Terano, Kiyoji Asai, and Michio Sugeno. *FUZZY SYSTEMS THEORY and IT'S APPLICATIONS*. ACADEMIC PRESS, INC, 1992.

[WB98] Amtoni Wolski and Tarik Bouaziz. Fuzzy triggers: Incorporating imprecise reasoning into active databases. In *Proc, 14th Int'l Conf. on Data Engineering(ICDE'98)*, Orlando, Florida, February 1998.

[WCD+97] Ouri Wolfson, Sam Chamberlain, Son Dao, Liqin Jiang, and Gisela Mendez. Cost and imprecion in modelling the position of moving objects. In *Proc, 13th Int'l Conf. on Data Engineering(ICDE'97)*, 1997.

[YG98] Adnan Yazici and Roy George. *Fuzzy Database Modeling(to be published)*. Springer-Verlag, 1998.

[Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[Zad70] L. A. Zadeh. Similarity relations and fuzzy orderings. *Information Sciences*, 3(2), 1970.

[Zad78] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1), 1978.