Modeling Uncertainties in Publish/Subscribe Systems

Haifeng Liu and Hans-Arno Jacobsen
Department of Electrical Computer Engineering
Department of Computer Science
University of Toronto
hfliu@cs.toronto.edu, jacobsen@eecg.toronto.edu

Abstract

In the publish/subscribe paradigm, information providers disseminate publications to all consumers who have expressed interest by registering subscriptions. This paradigm has found wide-spread applications, ranging from selective information dissemination to network management. However, all existing publish/subscribe systems cannot capture uncertainty inherent to the information in either subscriptions or publications. In many situations, exact knowledge of either specific subscriptions or publications is not available. Moreover, especially in selective information dissemination applications, it is often more appropriate for a user to formulate her search requests or information offers in less precise terms, rather than defining a sharp limit. To address this problem, this paper proposes a new publish/subscribe model based on possibility theory and fuzzy set theory to process uncertainties for both subscriptions and publications. Furthermore, an approximate publish/subscribe matching problem is defined and algorithms for solving it are developed and evaluated.

1 Introduction

A new data processing paradigm – publish/subscribe – is becoming increasingly popular for information dissemination applications. Publish/subscribe systems anonymously interconnect information providers with information consumers in a distributed environment. Information providers publish information in the form of publications and information consumers subscribe their interests in the form of subscriptions. The publish/subscribe system performs the matching task and ensures the timely delivery of published events to all interested subscribers. Example applications range from selective information dissemination [18], online shopping, online auctioning [17] to location-based services [6] and sensor networks [21], to just name a few.

Publish/subscribe has been well studied and many systems have been developed supporting this paradigm. Existing research prototypes, include, among others, Gryphon [3], LeSubscribe [10], and ToPSS [2, 16]; industrial strength systems include various implementations of JMS, the CORBA Notification Service, and TIBCO's Tib/Rendezvoud product.

All existing publish/subscribe systems are based on a *crisp data model*, which means that neither subscribers nor publishers can express uncertain, imprecise, or vague information in subscriptions and publications, respectively, – often naturally inherent to the application domain. In this traditional crisp model, subscriptions, are evaluated to be either true, or false, for a given publication. Here, we refer to publications and subscriptions from this model as *crisp*.

However, in many situations exact knowledge to either specify subscriptions or publications is not available. In these cases, the uncertainty about the true state of the world has to be cast into a crisp value that defines absolute limits. That means, the uncertain state is "approximated" with a definite crisp value. Moreover, for a user of an application based on the publish/subscribe paradigm it may be much simpler to describe the state of the world with uncertain, imprecise, and vague concepts, rather than to guess or assess an absolute, but possibly incorrect, value. Often, it may not even be possible to determine an absolute value, since the property described is of a gradual nature. We next illustrate this dilemma with a number of concrete application scenarios and use cases.

Selective information dissemination: In an online auction or online shopping context, information consumers may want to submit subscriptions about music CDs with a constraint on price expressed as "cheap", a constraint on style expressed as "seventies", and a constraint on melody expressed as "happy". Similarly, subscriptions referring to characteristics or moods, may refer to the "blueness" and "lightness" of objects, or ask for a "bearish" or a "bullish" mood. Also, a "fast-paced" auction may trigger an alert set by a user. On the other hand, information providers may not have exact information for all items published. In an online real-estate market, for instance, an agent may not know the exact age of an apartment, so she simply describes it as an “old” object, “close” to downtown, with a “sunny” appearance, but can not describe it with definite values.

All these constraints designate situations that cannot be crisply evaluated – that is based on sharp boundaries and true and false assessments. A given object may satisfy each constraint to a certain degree, since boundaries are imprecisely defined and not absolutely set. This results in further possibilities to trade-off a deficiency of match of one constraint against a closer match of another constraint. Moreover, it is clear that these constraints are highly context sensitive and depend on the users’ subjective perception, which needs to be expressible by a publish/subscribe model supporting the modelling of uncertainty.

Further examples can be drawn from the areas of **location-based services** and **sensor networks**, where measurement precision is traded-off against accuracy and cost. Location positioning of mobile users is only possible with a certain degree of accuracy and sensors only return measurements distributed within an error-interval around the true value. Both application areas lend themselves well to data processing based on the publish/subscribe paradigm [6, 21].

For these reasons, we think, it is of great advantage to extend the publish/subscribe paradigm and develop an approximate matching scheme that allows the expression and processing of uncertainty for both subscriptions and publications. We refer to subscriptions and publications in this extended model as *approximate*.

We identify five interesting cases according to the different combinations of subscriptions and publications expressing uncertainty. These cases are: 1. *crisp subscriptions* and *crisp publications* (traditional publish/subscribe), 2. *approximate subscriptions* and *crisp publications*, 3. *crisp subscriptions* and *approximate publications*, 4. *approximate subscriptions* and *approximate publications*, and 5. the combination of *crisp* and *approximate constraints* in subscriptions and publications. Cases 2 to 5 constitute new publish/subscribe system models not previously investigated.

All existing publish/subscribe systems are based on a crisp data model that cannot capture notions of uncertainty in either publications or subscriptions. The only exception is the A-ToPSS software demonstration – the Approximate Matching-based Toronto Publish/Subscribe System [16] – that has introduced a subscription language model that can express notions, such as “cheap”, “large”, and “close to” as predicate constraints in subscriptions. In this paper we describe the theoretical basis of A-ToPSS, develop a model that embraces all five cases above, and present a detailed experimental evaluation. The contributions of this paper are:

1. An original and highly flexible publish/subscribe system model supporting the expression of uncertainties in the subscription language model and the publication data model. This model supports all five cases described above and is fully implemented. The model allows for fine-grained adjustment to express different users’ subjective perception of the concepts modelled and tune the matching relations. A subscription can be any arbitrary boolean function; most publish/subscribe systems developed to date allow for conjunctive subscriptions only.

2. The capturing of uncertainty in subscriptions and publications raises questions regarding the matching of crisp/approximate subscriptions with crisp/approximate publications. This paper articulates the approximate publish/subscribe matching problem and develops algorithms for solving it.

3. A thorough experimental evaluation of the proposed approximate matching algorithms is presented that compares traditional, crisp publish/subscribe with approximate publish/subscribe. The comparison is based on matching performance, memory use, the number of matches, matching precision and recall.

4. A brief experimental analysis of a reduced encoding of data structures in the approximate matching algorithms. Approximate publish/subscribe trades off uncertainty of input against computational precision. The reduced representation admissible for approximate matching exploits this trade off to save storage.

The paper is organized as following. In Section 2, we will briefly introduce the necessary background material our approach is based on, namely possibility theory and fuzzy set theory. The various publish/ subscribe models supporting uncertainty are developed in Section 3. Section 4 describes our algorithms and data structures. Section 5 presents the experimental evaluation and Section 6 summarizes related work.

2 Background

A key question in our work is how to express and process uncertainty in publish/subscribe systems. A simple method to express uncertainty about an imprecisely known value is to define it as an interval. For example, the interval [50, 150] would be reasonable to represent the age of a piece of “post-modern” painting in an online auction. In a crisp system, it needs two predicates to represent this interval: $(age \geq 50)$ and $(age \leq 150)$. Moreover, this method imposes a sharp boundary to differentiate members belonging to the set of post-modern paintings from non-members. A painting which was created 49 years ago may satisfy the subscriber, but it won't be delivered to the subscriber since it is out of the domain of the interval [50,150]. To overcome this limitation, fuzzy set theory [14] and possibility theory [9] have been developed. The publish/subscribe model we are introducing is based on these theories to model uncertainty in publications and subscriptions. In this chapter we give a brief overview of the key concepts used in our work, a more detailed discussion can be found in [14, 9].

2.1 Fuzzy set theory

Sharp boundaries that differentiate between objects belonging to a set versus objects not belonging to a set can be eliminated by introducing degrees of membership. This is the approach taken by fuzzy set theory.

Definition: A fuzzy set $\tilde{M}$ on a universal set U is a set that specifies for each element x of U a degree of membership to the fuzzy set $\tilde{M}$. It is defined by a membership function (a.k.a. characteristic function),

$$\mu_{\tilde{M}} : U \to [0,1]$$

that specifies for each $x \in U$ its degree of membership $\mu_M(x)$ to the fuzzy set $\tilde{M}$.$\square$

The membership function is a generalization of the characteristic function in classic set theory. It allows to express gradual set membership. For example, we can define a possible membership function for the fuzzy set of "post-modern paintings"as shown in Figure 1, where the domain ranges over the possible ages in the given application context. We use membership functions to represent predicates in subscriptions that constraint uncertain and vague concepts, such as "price is cheap" "age is old", and "location is close to".

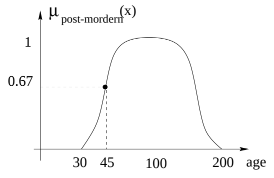

Figure 1. The membership function representing "old art piece".

There are many possible function representations to express gradual set membership. Here, we use a parametric representation as suggested by many authors [9, 14]. The membership function of a fuzzy set $\tilde{M}$ can be described with a pair of functions, defined on $\Re^+ \to [0,1]$, denoted by L and R, such that L (and also R) is monotonically increasing (and monotonically decreasing) and is upper semi-continuous ($u.s.c$). This function pair and four parameters $(\underline{m}, \overline{m}, \alpha, \beta) \in \Re^2 \cup \{+\infty, -\infty\}$ define the membership function of a fuzzy set $\tilde{M}$ as follows:

$$\mu_M(u) = \begin{cases} L(u) & \forall u \in [\underline{m} - \alpha, \underline{m}] \\ 1 & \forall u \in [\underline{m}, \overline{m}] \\ R(u) & \forall u \in [\overline{m}, \overline{m} + \beta] \end{cases}$$

A fuzzy set is characterized by its membership function, so without ambiguity we can say $\tilde{M}$ is defined by $\mu_M(u) = (\underline{m}, \overline{m}, \alpha, \beta)_{LR}(u)$. This representation can be used to model a wide range of different gradual set membership relations (e.g., bell-shaped, trapezoidal, triangular etc.)

Definition: $[\underline{m}, \overline{m}]$ is the *core* of fuzzy set $\tilde{M}$, denoted by $\dot{\mu}_M$. $\underline{m}$ and $\overline{m}$ are referred to the *lower* and *upper model values* of $\tilde{M}$, respectively. The *support* of a fuzzy set $\tilde{M}$, denoted by $S(\mu_M)$, is the domain of values where $\mu_M(u) > 0$. If $\tilde{M}$ is of bounded support, then $S(\mu_M)$=$[\underline{m}-\alpha, \overline{m}+\beta]$. α and β are called the *left-hand spread* and the *right-hand spread* (cf. Figure 1).$\square$

There are many advantages of this representation. First, it eliminates the sharp boundaries inherent to a crisp or interval-based representation. Second, it is a very general representation and it is straight forward to implement. Third, this formalization is very expressive. Finally, it is easily extended to represent crisp sets defined through crisp constraints. In this case the membership function degenerates to the characteristic function as follows:

$$\mu_{p \geq v}(x) = \begin{cases} 1 & \text{if} \quad x \in [v, \infty) \\ 0 & \text{if} \quad x \notin [v, \infty) \end{cases}$$

Operations involving two or more fuzzy sets are generally defined by a mapping T that aggregates the membership functions as follows:

$$\mu_{\text{op}(A_1,\ldots,A_n)}(x) = T(\mu_{A_1}(x), \ldots, \mu_{A_n}(x))$$

Intersection, union, and other set operations are defined in this manner. The operator T is referred to as a triangular norm (T-norm). T-norms that model set intersection must satisfy the following axioms (a generalization from classical set theory): $T(0,0) = 0, T(a,1) = T(1,a) = a$ (boundary condition), $T(a,b) \leq T(c,d)$ *if* $a \leq c$ *and* $b \leq d$ (monotonicity), $T(a,b) = T(b,a)$ (commutativity), and $T(a, T(b,c)) = T(T(a,b),c)$ (associativity).[1] Set union is defined and motivated in a similar manner. Operators that define set union are denoted as S-norms. Different S-norms and T-norms are used in the literature to represent set union and set intersection. A popular choice is to use maximum as union and minimum as intersection.

[1]The first axiom imposes the correct generalization to crisp sets. The second axiom implies that a decrease in the membership values in A or B cannot produce an increase in the membership value in A intersection B. The third axiom indicates that the operator is indifferent to the order of the fuzzy sets to be combined. Finally, the fourth axiom allows us to take the intersection of any number of sets in any order of pairwise groupings.

2.2 Possibility Theory

Possibility theory formally defines measures, which reflect users' subjective uncertainty of a given state of the world [9]. The measures express the confidence in the possibility that x is A. Possibility measures are based on possibility distributions, $\pi_A(x)$, that quantify these conditions.

A possibility measure that has values for each element in the universe of discourse can be interpreted by the membership function of a fuzzy set. For example, the antique shop has an art piece, where the age is described as post-modern:

$$\pi_{age-of-art-piece}(x) = \mu_{post-modern}(x).$$

We use two measures, referred to as *possibility measure*(Π) and *necessity measure*(N) to express the plausibility and necessity associated with each attribute in a publication. A possibility measure quantifies information about the plausibility of occurrence of the state represented by the attribute. If it is completely possible to be true then possibility is $\Pi(A) = 1$, if it is impossible then the possibility is $\Pi(A) = 0$; intermediate numbers between [0,1] are also admissible. A necessity measure is introduced to complement the information available about the state described by the attribute. It is associated with the degree with which the occurrence of A is certain. If an event A is sure to happen without any doubt, then necessity $N(A) = 1$.

The relationship between possibility and necessity satisfies[2]:

$$N(A) = 1 - \Pi(\overline{A})$$
$$\forall A, \Pi(A) \geq N(A).$$

3 Publish/Subscribe System Model

Our objective is to model uncertainties in subscriptions and publications, and to define an approximate matching semantic for different cases of matching crisp with approximate subscriptions and publications.

3.1 Language and Data Model

Subscription language model: A subscription defines user's interests through a Boolean function over a number of crisp and approximate predicates. In the following we just refer to predicates, unless their approximate character is especially underlined. Each predicate expresses a constraint over a domain of values and is defined through an attribute, an operator, and a value triple. In the predicate, "x is $\tilde{A}$", x is the attribute name, 'is' is the operator, and $\tilde{A}$ is a fuzzy set. The fuzzy set represent a fuzzy constraint over all possible values the attribute can take on. The predicate is evaluated by applying the membership function of the fuzzy set to the attribute's value in publication. The resulting value constitutes the degree of match of the predicate. Note, this may be any value in the interval [0, 1]. Thus, the truth value (i.e., the degree of match) of each predicate, "x is $\tilde{A}$," is uniquely defined by $\mu_A(x)$. Crisp predicates can be defined in the same manner. In the crisp case, however, the membership function degenerates to the characteristic function over the set of values defined by the predicate (i.e., it yields 1 for all set members and 0 otherwise.)

Predicate matching degrees are aggregated in a subscription relation to yield a final degree of match for each subscription. We use R to represent the relation of the Boolean function over predicates defining a subscription. R represents conjunction, disjunction or any other Boolean operation connecting individual predicates. Thus, a subscription, s, is formalized as follows:

$$s(x_1, \cdots, x_m) = R(\mu_{A_1}(x_1), \cdots, \mu_{A_m}(x_m)).$$

Here, the subscription, s, consists of m predicates of the form, "x_i is $\tilde{A_i}$", where R defines the Boolean function relating all predicates in s. For example, s may be in conjunctive form:

$$s(x_1, \cdots, x_m) = x_1 \text{ is } \tilde{A_1} \wedge \cdots \wedge x_m \text{ is } \tilde{A_m}$$

or disjunctive form, or any other form. R employs standard fuzzy set operators (cf. Section 2) to define the subscription relation. No limitation is imposed by the form of s. That is s may be any Boolean function, not necessarily in normal form. Mathematically, R constitutes a function in the hyperspace defined over the Cartesian product of the domains of x_is. For a given input vector $(x_1, \cdots, x_m)$ in this hyperspace, R yields the truth value of s for this input.

As a concrete example, let us define a subscription for a student who is looking for an apartment with constraints on *price*, *size*, and *age*. The subscription that specifies these constraints looks as follows:

$$\begin{array}{lllll} S: & size & is & medium & \texttt{AND} \\ & price & is & \textit{no more than } \$450 & \texttt{AND} \\ & age & is & \textit{not very old} & \end{array}$$

The second predicate constrain the attribute price. It is defined in a crisp manner. It can be represented by:

$$\mu_{\leq 450}(x) = \begin{cases} 1 & \text{if} \quad x \leq 450; \\ 0 & \text{if} \quad x > 450; \end{cases}$$

[2]A possibility distribution is similar to a probability distribution. However, the difference between both is that there is no restriction that the sum of all possibilities on the whole universe must be equal to 1. Another difference is that probability distributions must be defined on disjoint subsets, but possibility distribution can be defined on distinct (as long as not equal) subsets. Thus, a possibility is a more general notion than a probability. [9]

The first and third predicates constitute approximate predicates. We use the following membership functions to represent the concept of "medium" and "old", respectively.

$$\mu_{medium}(x) = \begin{cases} 0 & \text{if} \quad x \leq 40; \\ \frac{x-40}{10} & \text{if} \quad 40 < x < 50; \\ 1 & \text{if} \quad 50 \leq x \leq 70; \\ 1 - \frac{x-70}{10} & \text{if} \quad 70 < x < 80; \\ 0 & \text{if} \quad x \geq 80; \end{cases}$$

$$\mu_{old}(x) = \begin{cases} 0 & \text{if} \quad x \leq 40; \\ \frac{x-40}{40} & \text{if} \quad 40 < x < 80; \\ 1 & \text{if} \quad x \geq 80; \end{cases}$$

Formally the subscription is represented by:

$$s(x_1, x_2, x_3) = min(\mu_{medium}(x_1), \mu_{\leq \$450}(x_2), 1 - \mu_{old}^2(x_3)),$$

where min is used to model a conjunct. To demonstrate some features of fuzzy set theory, we use the negation of the membership function to define the qualifier "not" and the qualifier "very" through the squaring (i.e., damping) the fuzzy set's membership function.

Publication data model: Publications describe real world artifacts or describe states of interest through a set of attribute value pairs. In our model we account for the fact that for certain attributes precisely defined values may not be available or cannot be defined. In these cases we use a possibility distribution, as defined in Section 2, to represent the attributes' approximate values. These latter attributes are also referred to as approximate attributes, whereas attributes with exactly defined values are referred to as crisp attributes. However, our model integrates both kinds of attributes and does not distinguish between them. In the attribute value pair, "(A, $\pi(x)$)", A is the attribute and π is the "value" – crisp or approximate. The possibility distribution, π, expresses that it is possible that the attribute, A, has the value, x, and quantifies this with a possibility degree, $\pi(x)$. The possibility distribution is defined by a fuzzy set that yields the possibility degree for the value x, as defined by the underlying fuzzy set's membership function. Crisp attributes, "(A, x_0)," are formalized analogously; π degenerates to a function that yields 1 for input x_0 and 0, otherwise. For short, we describe the attribute value pair, "(A, $\pi(x)$)", simply as $\pi_A(x)$. A publication is thus defined as a vector of attribute value pairs:

$$p = (\pi_{A_1}(x), \pi_{A_2}(x), \cdots, \pi_{A_n}(x))$$

For example, in an apartment that is advertised for rent as:

$$p = ((\text{size}, 60m^2), (\text{rent}, \textit{cheap})),$$

the first attribute is crisp; it defines a value for attribute size. The second attribute is approximate; it is qualified as cheap, which is a fuzzy set that defines the degree of possibility for each value of the domain of discourse (i.e., all admissible rent values) as being "cheap". More formally, this publication can be represented by a vector of attribute values as follows:

$$P = ((size, \pi_{60}), (rent, \pi_{cheap}))$$

where

$$\pi_{60}(x) = \begin{cases} 1 & \text{if} \quad x = 60; \\ 0 & \text{if} \quad x > 60 \text{ or } x < 60 \end{cases}$$

$$\pi_{cheap}(x) = \begin{cases} 1 & \text{if} \quad x \leq 1200; \\ 1 - \frac{x-1200}{300} & \text{if} \quad 1200 \leq x \leq 1500; \\ 0 & \text{if} \quad x > 1500 \end{cases}$$

3.2 Approximate Matching

In the crisp publish/subscribe model a subscription, either matches a publication, or does not match it. However, in the approximate model, either the subscription, the publication, or both may refer to concepts of uncertainty and an evaluation to, either true, or false no longer captures the true state, which, given the uncertainty involved, is somewhere in between true and false. In the approximate model each subscription is therefore assigned a degree of match for each publication processed by the system. Individual subscription can match a given publication more or less, depending on this degree of match.

With this matching semantic a much larger number of subscriptions will match than before, as all matches with degrees greater than 0 are perspective matching candidates. Users' perception of what constitutes a "good" match versus a "bad" match will certainly differ. Furthermore, a large number of slightly matching subscriptions may not be a useful idea, since the publish/subscribe system will have to process a large amount of notifications and users may be overwhelmed with notifications about publications that only marginally meet their actual interests. For these reasons, the approximate matching model introduces a number of parameters to control the tolerance of a match on a very fine-granular basis. These parameters offer great flexibility and control over the matching process, and allow to fine-tune the approximate publish/subscribe model on a single-user basis (i.e., predicate and subscription basis.) While this may seem as an overwhelming amount of parameters to set, it offers great flexibility. All parameters are initialized with default values that do not affect the matching process, such that, in the default case, all possible matches are signaled.

These parameters are the predicate thresholds θ_Π and θ_N and the subscription thresholds ω_Π and ω_N. We provide further motivation for these thresholds below. With these parameters a publication matches a subscription, if its degrees of match evaluates to values larger than these thresholds.

The general form of subscriptions and publications is as follows:

$$s^{\omega_\Pi,\omega_N}(x_1,\cdots,x_m) = R(\mu_{A_1}^{\theta_{\Pi_{A_1}},\theta_{N_{A_1}}}(x_1),\cdots,\mu_{A_1}^{\theta_{\Pi_{A_m}},\theta_{N_{A_m}}}(x_m)).$$

$$p = (\pi_{A_1}(x_1), \pi_{A_2}(x_2), \cdots, \pi_{A_n}(x_n))$$

The *approximate matching problem* can now be stated as follows. *Given a set of subscriptions S and a publication p identify all $s \in S$ such that s and p match with a degree of match greater than the thresholds defined on s.*

We define a match between a subscription and a publications as a *measure* of the *possibility* and *necessity* with which the publication satisfies the constraints expressed by a subscription. We use the pair (Π_{A_i}, N_{A_i}) to denote the evaluation of this measure. Technically speaking, the problem comes down to measuring the match between the predicate, $\mu_{A_i}(x_i)$, and the value, $\pi_{A_i}(x_i)$ for all i and for all x and aggregating the resulting values in the subscription relation R. This measure is taken by computing the intersection between μ_{A_i} and π_{A_i}. Next, we define this measurement process more formally.

Definition:: The *possibility* and *necessity* of a match between μ and π is computed as

$$\Pi = \sup_x \min(\mu(x), \pi(x))$$

$$N = \inf_x \max(\mu(x), 1 - \pi(x)),$$

respectively.□

inf is the "infimum" and sup is the supremum. For finite domains both can be replaced by the "minimum" and the "maximum" operator, respectively. However, for infinite domains the more general inf/sup operators are required, which is the reason for using them in the above equations.

Definition:: Formally, a publication, p, *matches* a subscription, s, if and only if:

$$\forall i \;\; \Pi_i \geq \theta_{\pi_{A_i}} \wedge N_i \geq \theta_{N_{A_i}} \wedge$$

$$R(\mu_{A_1}^{\theta_{\Pi_{A_1}},\theta_{N_{A_1}}}(x_1),\cdots,\mu_{A_1}^{\theta_{\Pi_{A_m}},\theta_{N_{A_m}}}(x_m)) \geq \omega_\Pi \wedge$$

$$R(\mu_{A_1}^{\theta_{\Pi_{A_1}},\theta_{N_{A_1}}}(x_1),\cdots,\mu_{A_1}^{\theta_{\Pi_{A_m}},\theta_{N_{A_m}}}(x_m)) \geq \omega_N. \quad \square$$

From the possibility and necessity computation equations, the following properties can be easily deduced.

Properties:

$$\begin{aligned}
\forall x, \quad & \quad \Pi(x) \geq N(x). && (1)\\
\Pi = 0 \quad \Leftrightarrow & \quad S(\mu) \cap S(\pi) = \emptyset && (2)\\
\Pi = 1 \quad \Leftrightarrow & \quad \exists x \in \dot{\mu} \cap \dot{\pi}. && (3)\\
N = 0 \quad \Leftrightarrow & \quad \exists x \in \overline{S(\mu)} \cap \dot{\pi} && (4)\\
N = 1 \quad \Leftrightarrow & \quad S(\pi) \subseteq \dot{\mu} && (5)
\end{aligned}$$

These properties are exploited in the algorithm to optimize its performance (cf. Section 4). The properties relate characteristics about the support and the core of the possibility distribution and fuzzy set to infer the degree of match with less computation. These properties are graphically illustrated in Figure 2(1–4). Figure 2(1) & (4) illustrate property (4) and (5). Figure 2(3) illustrates property (3). Figure 2(2) illustrates the most general case, where $0 \leq \Pi \leq 1$ and $0 \leq N \leq 1$.

The possibility measure, Π, represents the degree of match and, its dual measure, N, represents the degree of no-match (cf. discussion Section 2). From Property 1, above, it follows that the possibility, Π, is always greater or equal to the necessity, N. The subjective interpretation of this is that an optimistic subscriber would count on the leaner possibility measure, while, a pessimistic subscriber would count on the stricter necessity measure.

Finally note, that for crisp attributes, "(A, x_0)", the possibility distribution function π yields 1 for x_0 and 0, otherwise. So the intersection of π and μ can only occur at the point x_0, which is the value $\mu(x_0)$.

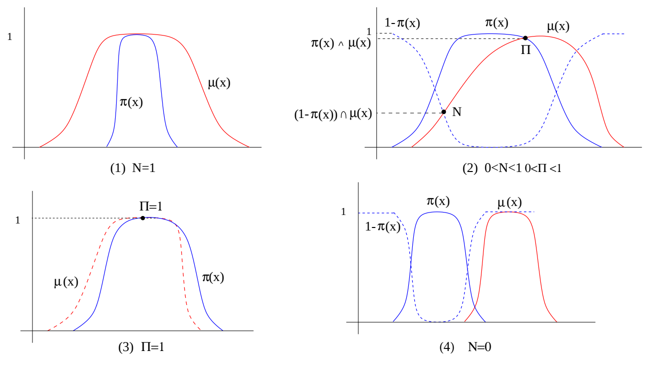

Figure 2. Cases of possibility and necessity measure

3.3 Discussion of Alternative Matching Semantic

Intuitively speaking, the ratio of the area of overlap between π and μ over the whole area of π may seem like an alternative measure to evaluate the degree of match between predicates and values. An interpretation of this ratio could be the assessment of how the domain of π can satisfy μ. However, this method is not sufficient, as there exist situation in which subscriptions match only to a small degree, but the degree of match computed by this method is 1. Consider the example in Figure 3. The domain of the fuzzy set defining the approximate attribute in publication, π, is totally contained inside μ and it is completely covered by μ. It seems that all the values of the domain of discourse would satisfy the predicate defined by μ over this domain, thus yielding a degree of match of 1. However, consider the price \$60, its membership in π is 0.1, its membership in μ is 0.5. It is still possible that the price, the publisher observes

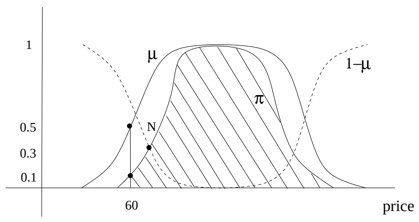

Figure 3. Degree of match defined as ratio of overlap

is \$60, though this possibility is rated as only 0.1. The subscription matches with this price with a degree of match of 0.5 (as resulting from the application of the membership function at the point 60), but not with degree 1. Therefore, it is not appropriate to define the matching degree as 1 in this situation. On the other hand, possibility and necessity measures solve this problem. It is possible that the value provided by the publication satisfies the subscription, the *possibility degree* is 1. But it is not necessarily the case; so according to the formula above, the *necessity degree* is only 0.3.

4 Data Structure and Algorithm

The matching algorithm proceeds in two stages. First predicates are matched and, second, matching subscriptions are identified. This is a similar break-down as applied in many crisp matching algorithms.

4.1 Data Structure

Predicate evaluation is based on two data structures: a hash table to index predicates according to their names and a predicate vector to store the degree of match for each predicate. Subscription evaluation is based on the list linked to each predicate to record the subscriptions that contain it (or using an association bit matrix) and a subscription vector to keep track of the degree of match of each subscription. The overall data structure is depicted in Figure 4.

In Figure 4, a_i is the attribute name. Each predicate is represented by a pair (pid, μ). pid is the predicate ID and μ is the membership function to describe user's constraint on the attribute a_i. μ is represented by a list of parameters $(\underline{m}, \overline{m}, \alpha, \beta, L_m, R_m)$. L_m and R_m are the indexes into a function family indicating which functions are used for left-hand spread and right-hand spread functions. The exact choice of these parameters depends on the real application. We use one predicate vector to store both thresholds (θ_Π, θ_N) and the matching degrees (Π, N) . A flag is used to indicate whether the numbers are thresholds or matching degrees. At first it stores the thresholds θ_Π and θ_N.

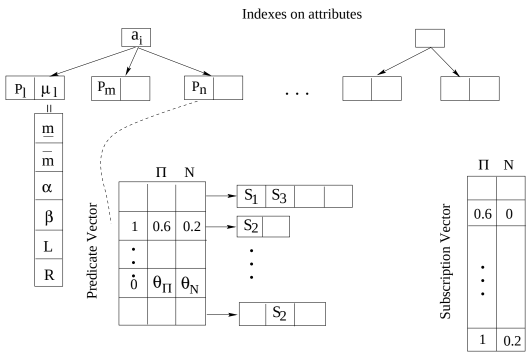

Figure 4. Data structures

Each publication is a set of pairs $(attr, \pi)$ for different attributes. π is a function showing the possibility distribution of uncertain value. Similar to μ, π is represented as $(\underline{n}, \overline{n}, \gamma, \delta, L_n, R_n)$.

4.2 Matching Algorithms

Predicate evaluation: A publication is a set of pairs of $(attr, \pi)$ where $\pi = (\underline{n}, \overline{n}, \gamma, \delta, L_n, R_n)$. The attribute-name, $attr$, is used as the hash key to locate the corresponding predicate-table. Each predicate is stored only once in the system. Each predicate is in the form $(pid, attr, \mu, \theta_\Pi, \theta_N)$, where $\mu = (\underline{m}, \overline{m}, \alpha, \beta, L_m, R_m)$. The predicate evaluation computes the possibility and necessity of match for the given input attribute, respectively. After all attributes of the given publication have been processed the matched degrees (i.e., each possibility and necessity) are used to derive matched subscriptions. Figure 5 depicts the predicate matching algorithm.

```
**Input:**
$e = \{(a_1, \pi_1)(a_2, \pi_2) \cdots (a_n, \pi_n)\}$
**Global Variables:**
$I$: set of index;
$V_p$ : predicate vector storing (Π, $N$) for each predicate;
$SatPreds$: set of satisfied predicates;
**Body:**
1. $V_p = 0$, $SatPreds = \emptyset$
2. **for** each attribute $a_i$ in $e$
        locate the corresponding index $i$ in $I$
        **for** each predicate p $(a_i, \mu_i, \theta_{\Pi_i}, \theta_{N_i})$ reached by $i$
            $V_p[p].\Pi$=sup min$(\mu_i, \pi_i)$
            $V_p[p].N$=inf max$(\mu_i, \pi_i)$
        **if** $V_p[p].\Pi > 0$
        **then** $SatPreds = SatPreds \cup \{p\}$
3. return $SatPreds$
```

Figure 5. Predicate matching algorithm

The possibility of predicate $V_p[p].\Pi$=$\sup \min(\mu_i, \pi_i)$ is computed according to the cases discussed in Section 3.2.

```
**Procedure** $\sup \min(\mu_i, \pi_i)$
**begin**
    **if** $\bar{m} + \beta \le \underline{n} - \gamma$ or $\bar{n} + \delta \le \underline{m} - \alpha$ **then** $\Pi = 0$;
    **else if** $\underline{m} \le \underline{n}$
        **if** $\bar{m} \ge \underline{n}$ **then** $\Pi = 1$
        **else** find $c$ such that $R_m(\frac{c-\bar{m}}{\beta}) = L_n(\frac{\underline{n}-c}{\gamma})$
            $\Pi = R_m(\frac{c-\bar{m}}{\beta})$
    **else**
        **if** $\underline{m} \le \bar{n}$ **then** $\Pi = 1$
        **else** find $c$ such that $R_m(\frac{c-\bar{n}}{\delta}) = L_n(\frac{\underline{m}-c}{\alpha})$
            $\Pi = R_m(\frac{c-\bar{n}}{\delta})$
**end**
```

Figure 6. Possibility Computation

Figure 6 depicts the detail of the possibility computation process. Necessity computation is similar.

Subscription evaluation: Subscriptions may be conjuncts of predicates, disjuncts of predicates, or normal forms. We use the intersection and union operations defined in Section 2 to model these operations. The algorithm we present for subscription evaluation works for either conjunctive or disjunctive subscriptions. To also process normal forms a further stage based on the truth values of subscription terms is required, which we don't present here (it is analogous to the subscription evaluation stage.) We also limit our presentation of the subscription evaluation algorithm to the use of the minimum T-norm (other norms could simply be plugged in.) The algorithm calculates the degree of match, as expressed by a possibility measure and a necessity measure for each subscription. At the end of evaluation, we will compare the possibility and necessity of each subscription with user's thresholds ω_Π and ω_N, only return user the subscriptions whose degrees are larger. This is just a little further comparison, we don't include in the algorithm depiction.

```
**input:**
$SatPreds$: output of the predicate matching stage
**global variables:**
$V_p$: predicate vector;
$V_s$: subscription vector;
$List$: array of lists that store predicate subscription association;
$SatS$: set of matching subscriptions for event $e$
**Body**
1. $V_s = 0$, $SatS = \emptyset$, $Count = 0$
2. **for** each $p \in Vp$ where $V_p[p].\Pi \ge p.\theta_\Pi$ and $V_p[p].N \ge p.\theta_N$
    **for** each $s$ in List[p]
        **if** $Count[s] = 0$ **then**
            $V_s[s].\Pi = V_p[p].\Pi$
            $V_s[s].N = V_p[p].N$
        **else** $V_s[s].\Pi = \min(V_s[s].\Pi, Vp[p].\Pi)$
            $V_s[s].N = \min(V_s[s].N, V_p[p].N)$
        $Count[s] + +$
3. **for** each $s$
    **if** $Count[s] = preds\_per\_sub[s]$
    **then** $SatS = SatS \cup \{s\}$
4. return $SatS$
```

Figure 7. Subscription evaluation

4.3 Optimizations

Improved predicate matching: The previous algorithm evaluates all predicates related to one attribute that is referenced by a given publication (i.e., iterated over each of its attributes). More specifically, at least one comparison between the two functions μ and π was required for each predicate to determine whether a match occurred. To minimize the number of comparisons, we improve our algorithm by sorting the predicates of the same attribute so that the predicate matching algorithm can stop earlier rather than evaluate all predicates.

For each attribute, the order of predicates depends on 4 parameters of their functions μ. In the representation of function μ, let $m_1 = \underline{m} - \alpha$, $m_2 = \underline{m}$, $m_3 = \bar{m}$, $m_4 = \bar{m} + \beta$. These are four critical points because they differentiate the boundaries where π has value 0 and where π has value 1 (refer to Figure 9). Obviously, we have $m_1 \le m_2 \le m_3 \le m_4$. Similarly, for function π, let $n_1 = \underline{n} - \gamma$, $n_2 = \underline{n}$, $n_3 = \bar{n}$, $n_4 = \bar{n} + \delta$, and we have $n_1 \le n_2 \le n_3 \le n_4$.

The predicate won't match the publication if its right-hand spread is to the left of the attribute function π (e.g., in Figure 8, $m_{14} \le n_1$). A match is established once the predicate "touches" the publication, i.e., μ and π intersect (e.g., μ_2 and μ_3 in Figure 8). If the predicate's left-hand spread is to the right of π (e.g., μ_4 in Figure 8, $m_{41} \ge n_4$), it will no longer match.

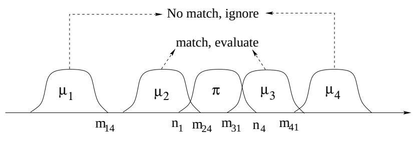

Figure 8. Examples of match and no-match between μ and π

Based on this observation, predicates with the same attribute name, are organized in the order of their μ functions from smallest to largest starting from m_1 to m_4. For example, there are two predicates, p_i and p_j, that are under the same attribute index. We first compare m_{i_1} and m_{j_1}. The predicate with the smaller m_1 is placed ahead of the other. If $m_{i_1} = m_{j_1}$ then we compare m_{i_2} with m_{j_2} and take the one with a lower value for m_2 and place it ahead of the other. If the m_2 are equal then the same comparison is done for m_3, m_4. If all the parameters are the same, then the predicate who enters the system earlier is placed ahead of the other.

For each attribute a_i of a publication, we pass the predicates whose membership functions are to the left of the π_i, only evaluate predicates that intersect with the attribute.

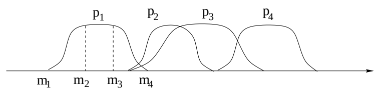

Figure 9. Examples of ordered predicates $p_1 < p_2 < p_3 < p_4$

Predicate matching stops as soon as the above rules establish further none-matches.

In the possibility computation, the sequencing algorithm first compares n_1 (the first point of π) with the m_4 (the last point of function μ) of the predicates through the ordered predicate list until it reaches the predicate whose μ_4 is larger then n_1. Before that, all predicates are to the left of the π (as the left case in Figure 8), hence impossible to match. After $m_4 > n_1$ then we check m_1. If $m_1 > n_4$, then we can stop because from now on all predicates afterwards are to the right of π, thus impossible to match either (as the right case in Figure 8). We just need to evaluate the predicates whose $m_4 < n_1$ and $m_1 > n_4$. Figure 10 shows the detailed possibility computation.

```
procedure Improved-Possibility-Computation sup $\min(\mu, \pi)$
begin
        1. $j = 1$
        2. while $m_{1j} < n_4$ do
        begin
                while $m_4 j \le n_1$ do j++
                if $m_{3j} \le n_2$ then find $c$ such that
                        $R_m(\frac{c-\bar{m}}{\beta}) = L_n(\frac{\underline{n}-c}{\gamma})$
                        $\Pi = R_m(\frac{c-\bar{m}}{\beta})$
                else if $m_{2j} \le n_3$ then$\Pi = 1$
                else find $c$ such that
                        $R_m(\frac{c-\bar{n}}{\delta}) = L_n(\frac{\underline{m}-c}{\alpha})$
                        $\Pi = R_m(\frac{c-\bar{n}}{\delta})$
                j++,
        end
end
```

Figure 10. Improved Possibility Computation

In the necessity evaluation, the algorithm first compares n_3 (the third point of function π) with m_4 (the last point of function μ) through the ordered predicate list until it reaches the predicate whose μ_4 is larger than n_3. Before that, the complements of predicate functions μ are always intersected with the core of the π, so the necessity must be 0. After $m_4 > n_3$ we compute the necessity of each predicate until $m_1 \le n_2$ because from now on all necessities afterwards must be 0. The procedure is similar to possibility computation, we don't elaborate here.

Precision-space trade off: The approximate matching scheme trades off the processing of uncertain and vague information against precision. This suggest that a degree of match that is computed for a subscription must not be highly accurate, i.e., accurate to the n-th digit after the comma, as it is based on uncertainty anyway. We use this as motivation to experiment with different encodings for the degrees of match in our algorithm. The objective is to save space, while not sacrificing computational accuracy in our approximate matching model. We use three encodings: Float, one-byte, representing ten values, and one-byte representing 256 possible values for the degree of match. This is a straight forward encoding, with more refined schemes deferred to future work. The effects of different encodings will be evaluated in the experiments chapter.

4.4 Time-Space Analysis

Space cost: The space cost includes mainly the following parts: predicate hash table, predicate vector, subscription vector, and the association list for each predicate.

$$Space = \sum(Space_p * N_p) + 2N_p + 2N_s + N_p * N_{s_p}$$

Where $Space_p$ is the space for one approximate predicate function $\mu = [\underline{m}, \bar{m}, \alpha, \beta]_{LR}$, N_p is the number of predicates, $Space_p * N_p$ is the space to store all distinct predicates in the system, and N_s is the number of subscriptions. Each predicate and subscription is associated with two measures: possibility and necessity. Their types depend on the encoding chosen (float or char). The space cost for approximate matching is greater than crisp matching in which just one bit is used to record whether a predicate is matched or not matched. Using N_{s_p} as the average number of subscriptions associated with each predicate, the space of association lists takes $N_p * N_{s_p}$. Overall, the space cost is linear with the number of predicates and subscriptions: $Space = O(N_p + N_s)$.

Matching time: The algorithm consists of two steps. First, predicate matching, consists of the time to retrieve the attribute from the index, which is just one lookup (hash table). Then all predicates under the same attribute are evaluated. In the original algorithm, all predicates membership functions under the same attribute need to be computed to get the possibility and necessity matching against the publication possibility distribution function. Assume that the time spending to evaluate each predicate membership function associate with the attribute is t_1, and all predicates are distributed uniformly on each attribute. Then the matching time for predicate matching is

$$Time(\text{regular predicate matching}) = t_1 * \frac{N_p}{N_a} * N_{a_e}$$

where N_p is the total number of all predicates, N_a is the total number of all attributes, hence $\frac{N_p}{N_a}$ is the number of predicates associated with one attribute. N_{a_e} is the average attributes number in the event.

In the improved algorithm, we don't need to evaluate all predicates associated with one attribute because of the good organization of the predicates. Evaluation stops at the point where all other predicates won't match for sure. We define $\alpha(\in [0,1])$ as the coefficient between the number of predicates evaluated in the improved algorithm and in the original one. This gives us a matching time of:

$$Time(\text{improved predicate matching}) = \alpha * t_1 * \frac{N_p}{N_a} * N_{a_e}$$

In the subscription evaluation, we suppose the time for one lookup is t_2. In our algorithm, for each matched predicate, we need to look up at the predicate subscription association matrix to find out which subscription contains this predicate, hence the time is $t_2 * N_{s_p} * N_{p_{sat}}$ where $N_{p_{sat}}$ is the average number of matched predicates. Since the thresholds θ_Π and θ_N are used to trim off the predicates whose matching degrees are not big enough to satisfy users. We denote β as the coefficient between the number of evaluated predicates and the number of the matched predicates whose matching degrees are beyond the thresholds, then we get $N_{p_{sat}} = \beta * \frac{N_p}{N_a} * N_{a_e}$. The subscription evaluation time is

$$Time(\text{subscription evaluation}) = \beta * t_2 * N_s * \frac{N_p}{N_a} * N_{a_e}.$$

In all, the matching time cost of the sequencing algorithm is

$$Time = \alpha * t_1 * \frac{N_p}{N_a} * N_{a_e} + \beta * t_2 * N_s * \frac{N_p}{N_a} * N_{a_e}.$$

5 Experiment

In this section, we discuss the experimental performance of the algorithms presented in Section 4. The performance is evaluated from time and memory aspects to confirm the efficiency of the algorithms and compare the differences between crisp publish/subscribe model and approximate model. We also examined the tradeoff between matching precision against the space used for storage. At last, we experimented the freedom of choices of aggregation function to get subscription matching degree. Experiments are processed under various subscription and publication workloads.

5.1 Experiment Setup

We ran all experiments on a dual-CPU Pentium III workstation with 900MHz i686 CPUs 1.5 GB RAM and 2G swap operating under Linux (RedHat 7.2). We used a workload specification file to configure the workload to be simulated. In this file, we can specify the following parameters: n_S, the total number of the subscriptions; n_E, the number of publications to be processed; n_P, the number of predicates per subscription; n_A, the number of attributes per publication; and types (crisp or approximate) of subscriptions, predicates and publications. Our workload can generate both crisp and approximate subscriptions and publications according to the types defined in the specification file. To make the approximate and crisp cases comparable, we generated crisp subscriptions and publications based on approximate ones.

For subscriptions, we define three interval types of crisp predicates derived from the approximate one: *optimistic*, *pessimistic* and *middle*. They are three ways to determine the lower bound and upper bound of the interval. If m_1, m_2, m_3, m_4 are the four parameters for the representation of the approximate predicate then those three interval types are defined in Table 1.

Sub Type	Value function
approx	(m_1, m_2, m_3, m_4)
pessi	$[m_1, m_4]$
middle	$[\frac{m_1+m_2}{2}, \frac{m_3+m_4}{2}]$
optim	$[m_2, m_3]$

Pub Type	Value function
appro	(n_1, n_2, n_3, n_4)
interval	$[n_2, n_3]$
point	$\frac{n_2+n_3}{2}$

Table 1. Definitions for different subscription and publication types

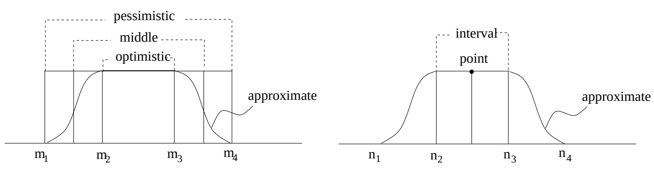

Figure 11. Examples of different subscription and publication types

Publications are generated similarly. We have two choices when generating crisp publications on the basis of approximate publications: *point* and *interval*. They referr to the types of the value for each attribute in the publication. *Point* type is defined to be consistent with the publication language data model in crisp publish/subscribe system so that they are comparable. *Interval* type serves to compare the difference between an interval representation and a fuzzy set representation for an uncertain constraint. Since we define three choices to generate interval subscriptions, we can compare the effects of different lower bound and upper bound of the interval in the subscriptions. Therefore, we only generate one interval type of publication. The definition is in Table 1. Figure 11 gives examples of different subscription and publication types.

The attribute names for predicates are drawn from a predefined set of names. The same set of attribute names is used to draw the attribute names for publications. The total number of names available is determined by n_t. For each attribute name, we provide a set of concepts (the number of those concepts is n_v) to be drawn as the uncertain constraints for attributes. A domain restricted by a lower and an upper bound, l_P and u_P, respectively is also provided to generate the possible membership functions to represent the uncertain constraint. In this experiment, we use only a trapezoidal form function, hence 4 points are selected from the domain, governed by a uniform distribution, to form the representation function.

5.2 Performance Evaluation

To evaluate performance metrics, we classify the implementations into 3 pairs according to different emphasis: 1. algorithms: regular matching vs improved matching algorithms; 2. matching result representation: *float-wise* (4 bytes) vs *bit-wise* (8 bits or 4 bits); 3. the data structure for the association between predicates and subscriptions: matrix-based vs list-based. We consider the following metrics: subscription loading time, matching time and used memory space. The matching time measurement starts just before the publication has been submitted to the system and ends right after the system responds.

Figure 12 compares the matching time across all implementations. In this experiment we use the following workload specification: $W0 = (n_t = 42, n_P = 2, n_A = 4, n_E = 10, n_v = [2..5], n_S = [10000..100000])$. No doubt, the matching time depends on the number of predicates for each attribute and the number of subscriptions that include those matched predicates, hence matching time increases with increasing the number of subscriptions n_S. In this graph, we compare the float-wise, bit-wise and improved matching implementations. The advantage of the improved predicate matching algorithm is not distinguished since the subscription evaluation step takes much more time than predicate matching. The bit-wise implementation runs more slowly than the float-wise one because it needs more computation to set the bit values. To show the benefits of the improved predicate matching algorithm, we ran the predicate matching process only and it showed that the improved one runs much faster.

In our experiment, the workload is generated randomly, thus the number of subscriptions that contain the same predicate is very small compared to the total number of subscriptions. Therefore, both the matching time and memory using the list-based approach is much less than the matrix-based approach considering the size of the list for each predicate is much smaller. The results are shown in the last two graphs of Figure 12. In the case where each predicate is contained

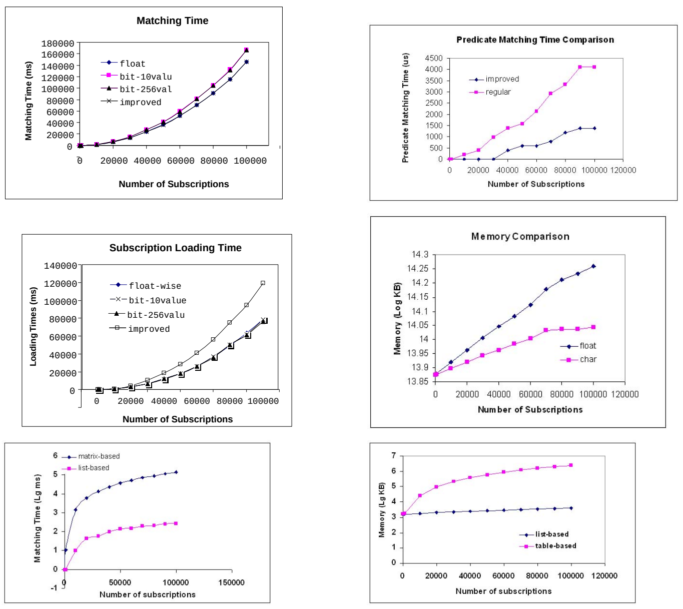

Figure 12. Matching performance(matching processing time, memory resident size and subscription loading time)

in most subscriptions, the matrix-based version should be much better because access to the table is faster.

The loading time figure compares the loading time among different algorithms. Contrary to the matching time, the improved algorithm needs more time than the other three. This is because predicates need to be inserted into a sorted list based on the 4 points of the function. This is a tradeoff between the loading time and matching time. In a real application, most subscriptions stay in the system for a long time and the matching time is more important to the user. With the high publication submission rate, it is better to process the matching quickly and respond as soon as possible.

The memory comparison figure shows memory utilization for the float-wise and bit-wise algorithms. The difference shows up only in the storage of matched result of predicates and subscriptions, so we only consider the space used here. We can see that bit-wise one uses less than the float version due to the space saved by using several bits instead of 4 bytes.

The decrease in space using bits instead of float results in a loss of precision in the matched results. A performance measure *precision* is defined as:

$$precision = \frac{\sharp\text{Correct Subscriptions Returned}}{\sharp\text{Subscriptions Returned}}$$

In publish/subscribe systems, correctness means that the matched subscriptions our system gets are exactly what users want. For example, a user wants to be notified when her subscription matches with a degree larger than 0.8. In

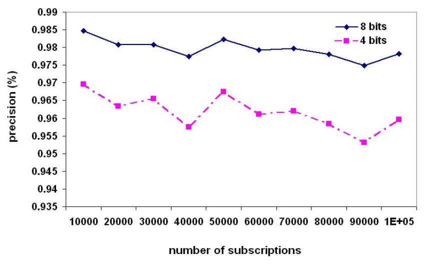

Figure 13. Precision space tradeoff

the 8 value bit-wise implementation (divide 0 to 1 into 8 parts), those matched degrees between 0.75 and 0.8 are stored in the same bit as those between 0.8 and 0.875. Users are only satisfied with the latter ones, however, all of them will be returned. Compared to the float-wise implementation which always return the correct data, the bit-wise version will also return some subscriptions that not satisfied because of the precision loss. In our context, the precision is computed by

$$precision = \frac{\sharp\text{Subscriptions float-wise Returned}}{\sharp\text{Subscriptions bit-wise Returned}}$$

Figure 13 shows the precision of 8 value bit-wise (3 bits) and 4 value bit-wise (2 bits) implementations. We are happy to see that the precision of the 3 bits version is stable around 98% and the 2 bits is stable around 96%. Considering the acceptance of users' error range in the real world, the decrease of the bits don't introduce much error.

5.3 Comparison Between Crisp and Approximate Matching

In this set of experiments we compare the crisp and approximate publish/subscribe matching model with respect to the number of matches identified under different conditions. Table 2 shows the different numbers of matches based on the evaluation of a fixed number of approximate publications over different kinds of subscriptions and different thresholds. We use α as the thresholds to assess a minimal possibility and necessity beyond which a subscription is not counted as a match (i.e., $\omega_\Pi = \omega_N = \alpha$). We can see that for one type of subscription, the number decreases with increasing α, which indicates the threshold effect of α. With the same α, the pessimistic case results in the largest number of matches and the optimistic case results in the fewest matches. The approximate case and the middle case do not exhibit much difference. This is due to the wider restriction of subscription, the greater the probability of being matched.

Table 2 then shows the numbers of matched subscriptions for different type of event when subscription type is fixed. When $\alpha = 0$, the fuzzy publication returns the most

Subscription Type	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
appro	4628	184	7
pessi	4628	804	281
middle	4438	184	39
optim	3763	47	7
Publication Type	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$
appro	4628	184	7
interval	3720	474	170
point	2960	1932	868

Table 2. Comparison of number of matches for various types of subscriptions with approximate publications and number of matches for various types of publications with approximate subscriptions, ($n_S = 70,000$, $n_E = 10$.)

subscriptions and point type returns the least. This is the same as for subscriptions. However, with the increase of α, the fuzzy event matched a very small number of subscriptions, whereas point-value event matched the most. This is because α is used as the threshold for both possibilities and necessities. Think of the intuitive meaning of possibility and necessity we defined in the model section. For the fuzzy event, the function restricting the attribute has a wider domain, thus it is more likely that the event intersects with the complementary region of subscriptions, therefore the necessities is very likely to be 0, make it more difficult to reach the α threshold. For the point-value event, it is easy for such value located in the core of the subscription function, thus more are matched with high α.

5.4 Effect of Choice of Aggregation Functions

To compute the overall degree of match for each subscription, different operations can be chosen to aggregate the degrees of match of predicates (e.g., min, weighted average etc.). For example, when students are looking for housing close to campus, they will consider, both the price and the distance. One student may worry more about the price, another student may be more indifferent and be satisfied with a balanced average, while a third student maybe more location-sensitive. In the proposed approximate matching scheme one aggregation function evaluates the degree of match of all subscriptions in the system, which maybe influenced by different thresholds. However, it is also important to understand the effect different aggregation functions have on matching effectiveness. This effectiveness is evaluated through *precision* and *recall* metrics. Three popular aggregation operations: *min*, *max* and *average* are compared. The definition of precision is given before, recall is defined as:

$$recall = \frac{\sharp\text{Correct Subscriptions Returned}}{\sharp\text{Correct Subscriptions}}.$$

The F-measure is a common metric for the evaluation of information systems. It relates precision and recall. It is computed as follows:

$$F - measure = \frac{2 * Precision * Recall}{Precision + Recall}$$

The relationship of the set of matching subscriptions using different aggregation operations is shown in Figure 14. The experiment runs by distributing user's aggregation expectation uniformly on 4 choices: min, max, average and weighted average (assign a weight to each predicate.) The correct data set should contain subscriptions whose overall degree computed according to user's expectation are larger than the threshold (ω_{Π}, ω_N). The data set we returned contain subscriptions whose overall degree computed by only one uniform function (either min, max or weighted average) is larger than the thresholds. Among the set we returned, there maybe some subscriptions whose overall degree is less than threshold if computed according to user's expectation, which is a positive error. Similarly, outside the data set we got, there maybe some subscription are not returned to user, but the overall degree is larger than threshold, which is the negative error. Figure 14 shows the comparison of the F-measures on these operations. It can be observed that all operations have high F-measures (around 95%), while the result of the average aggregation performs best.

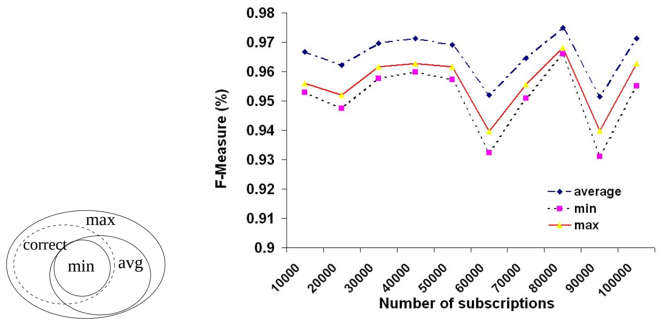

Figure 14. F-measure on aggregations

6 Related Work

Much work has been devoted to developing publish/subscribe systems and event notification services such as ELVIN, Gryphon, LeSubscribe, READY, Salamander, and SIENA. LeSubscribe [10] aims at publish/subscribe support for web-based applications. It focuses on the algorithmic efficiency supporting millions of subscriptions and high event processing rates. The supported language and data model is based on an LDAP-like semi-structured data model for expressing subscriptions and publications. Elvin [20] is a notification service that targets application integration environments and monitoring of distributed systems. ELVIN supports a more expressive subscription language including powerful string processing functions and operators on built-in data types. SIENA [7] and Gryphon [3] comprise other examples of publish/subscribe research projects that expose a very similar publication data model and subscription language model. Common to all these systems is the crisp matching semantic – either a match is established or no match is established; a gradual match, as defined in this work, expressed as a confidence, a degree of match, or a probability does not exist in any previously studied models.

A number of techniques, including, probability theory, fuzzy set theory, and a general similarity metric based approach have been applied to model uncertainty and imprecision in query and data. A full exploration would go beyond the scope of this paper. We discuss a number of representative examples. Fuhr introduced a probabilistic relational algebra in [13] to represent imprecise attribute values and integrate vague queries in database system. A similar approach, based on fuzzy set theory, is advocated by Ciaccia *et al.* [8]. Another popular approach is based on a vector space model as in [24], where the similarity between a document and a profiled is computed by means of an Euclidean distance measure. This metric gives rise to an "importance" value, a notion comparable to gradual set membership or probabilities.

The idea of using fuzzy sets in a multimedia retrieval model appears in [11, 12]. Fagin uses fuzzy sets to assign a grade of membership to each attribute of every object in a database and develops a list merging algorithm based on this rating of objects for multimedia databases. Nowadays, applications of fuzzy logic are found in many fields, including databases [4, 1, 19, 22], expert systems [15]. Wolski *et al.* [23] propose a fuzzy database trigger where fuzzy membership functions are used to model event-condition-action rules and integrate approximate reasoning into a crisp database rule evaluation mechanism. A similar idea is put forward in [5]. [22] introduce a retrieval language based on fuzzy logic and address the problem of retrieving using relevance feedback, a method do automatically adapt the representation of the underlying fuzzy set.

Although there is a large amount of related work involving representation and processing of uncertainty in databases and information management systems, none has studied the use of possibility theory and fuzzy set theory to model uncertainty in the language and data model of publish/subscribe systems. This paper is the first to use possibility theory and fuzzy set theory for expressing notions of uncertainty and vagueness in publications and subscriptions.

7 Conclusion

In this paper we propose an approximate publish/subscribe model to express uncertainties in both subscriptions and publications when exact information is not available. Fuzzy set theory and possibility theory are used to represent uncertain notions in predicates and publications. Based on this model, we define an approximate matching mechanism between publications and subscriptions. In addition, an approximate algorithm and an improved sequencing algorithm are designed to perform the matching task. There are several key properties of this approximate publish/subscribe model: 1) The language model is flexible and powerful in that it allows subscriptions and publications to be either crisp or approximate. 2) The possibility and necessity measures are expressive, these two matching degrees could be used for different optimistic and pessimistic users. 3) We have demonstrated that the algorithms we proposed can be used to process approximate matching for millions of subscriptions. 4) In our implementation, we use *min* as the conjunction operation to evaluate the overall matched degree of all the predicates within one subscription, the algorithm also works for other aggregation functions such as *product* since we use a two-step algorithm to apply the aggregation function after the predicate matching phase to pick out the matched subscriptions. 5) The algorithms are designed with respect to conjunctions of predicates, but it can be easily extended to disjunctions as long as we substitute the *min* operator with other boolean combination functions like *max*.

References

[1] Special issue on imprecision in databases. *Data Engineering Bulletin*, 12(2), 1989.

[2] G. Ashayer, H. K. Y. Leung, and H.-A. Jacobsen. Predicate matching and subscription matching in publish/subscribe systems. In *Workshop on Distributed Event-based Systems*, 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2nd-5th July 2002. IEEE Computer Society.

[3] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Storm, and D. Sturman. An efficient multicast protocal for content-based publish-subscribe systems. In *International Conference on Distributed Computing Systems*, 1999.

[4] P. Bose, M. Galibourg, and G. Hamon. Fuzzy querying with sql: Extensions and implementation aspects. *Fuzzy Sets and Systems*, 28, 1988.

[5] T. Bouaziz and A. Wolski. Applying Fuzzy Events to Approximate Reasoning in Active Databases. In *Proc. Sixth IEEE International Conference on Fuzzy Systems*, Barcelona, Catalonia, Spain, July 1997. .

[6] I. Burcea and H.-A. Jacobsen. L-ToPSS: Towards push-oriented location-based servives. Technical report, March 2003. (submitted for publication).

[7] A. Carzaniga, D. Rosenblum, and A. Wolf. Design of a scalable event notification service: Interface and architecture. In *Technical Report CU-CS-863-98*, Department of Computer Science, University of Colorado, August 1998.

[8] P. Ciaccia, D. Montesi, W. Penzo, and A. Trombetta. Fuzzy query languages for multimedia data.

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

[10] F. Fabret, H. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering algorithm and implementation for very fast publish/subscribe systems. In *ACM SIGMOD conference*, Santa Barbara, California, USA, May 2001.

[11] R. Fagin. Combining fuzzy information from multiple systems. In *Proc. ACM SIGMOND/SIGACT conf. on Princ. of Database Syst. (PODS)*, Montreal, Canada, 1996.

[12] R. Fagin. Fuzzy queries in multimedia database systems. In *Proc. ACM SIGMOND/SIGACT conf. on Princ. of Database Syst. (PODS)*, Seattle, WA, USA, 1998.

[13] N. Fuhr and T. Rolleke. A probabilistic relational algebra for integration of information retrieval and database systems. *ACM Transactions on Information Systems*, 15(1), 1997.

[14] G. J. Klir and T. A. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice Hall International Editions, 1992.

[15] K. Leung, M. Wong, and W. Lam. A fuzzy expert database system. *Data and Knowledge Engineering*, 4:287–304, 1989.

[16] H. Liu and H.-A. Jacobsen. A-topss – a publish/subscribe system supporting approximate matching. In *28 th International Conference on Very Large Data Bases*, Hong Kong, China, 2002.

[17] J. Pereira, F. Fabret, H.-A. Jacobesen, F. Llirbat, R. Preotiuc-Prieto, K. Ross, and D. Shasha. Le subscribe: Publish and subscribe on the web at extreme speed. In *SIGMOD digital library*, 2001. http://caravel.inria.fr/LeSubscribe/LeSubscribe.html.

[18] J. Pereira, F. Fabret, H.-A. Jacobesen, F. Llirbat, and D. Shasha. WebFilter: A high-throughput XML-based publish and subscribe system. In *VLDB conference*, 2002.

[19] F. Petry. Fuzzy databases: Principles and applications, with contribution by patrick bose. *International Series in Intelligent Technologies*, page 240, 1996.

[20] B. Segall and D. Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of AUUG97*, Brisbane, Australia, September 1997.

[21] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman. A taxonomy of wireless microsensor network models, 2002.

[22] O. Wolfson, A. Lelescu, and B. Xu. Approximate retrieval from multimedia databases using relevance feedback.

[23] A. Wolski and T. Bouaziz. Fuzzy Triggers: Incorporating Imprecise Reasoning into Active Databases. In *Proceedings of the 14th International Conference on Data Engineering*, pages 108–115. IEEE Computer Society Press, 1998.

[24] T. Yan and H. Molina. Index structures for information filtering under the vector space model. In *Proceedings of the International Conference on Data Engineering*, November 1993.