# A Model for Publish/Subscribe System Supporting Uncertainties

by

Haifeng Liu

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

# Abstract

A Model for Publish/Subscribe System Supporting Uncertainties

Haifeng Liu

Master of Science

Graduate Department of Computer Science

University of Toronto

2003

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 neither subscriptions nor 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 convenient 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 thesis proposes a new publish/subscribe model based on possibility theory and fuzzy set theory to process uncertain information for both subscriptions and publications. Furthermore, an approximate publish/subscribe matching problem is defined and algorithms for solving it are developed and evaluated.

# Contents

**1 Introduction** **1**

1.1 Overview . . . . . 1

1.2 Motivation . . . . . 2

1.3 Problem Definition . . . . . 4

1.4 Contributions . . . . . 6

1.5 Thesis Outline . . . . . 7

**2 Related Work** **8**

2.1 Crisp Publish/Subscribe Systems . . . . . 9

2.2 The Toronto Publish/Subscribe System Family . . . . . 10

2.3 Work Related to Uncertainty and Similarity Measure . . . . . 12

**3 Background** **14**

3.1 Fuzzy Logic Theory . . . . . 14

3.2 Set Representation . . . . . 17

3.3 Operations on fuzzy sets . . . . . 20

3.4 Possibility Theory . . . . . 22

**4 A Model for Publish/Subscribe with Uncertainties** **24**

4.1 Characteristics of Approximate Publish/Subscribe Model . . . . . 24

4.2 Subscription Language Model . . . . . 25

4.3 Publication Data Model . . . . . 27
4.4 The Approximate Matching Problem . . . . . 28
4.5 Case Studies . . . . . 34

**5 Data Structures and Algorithms** **36**
5.1 Data Structures . . . . . 36
5.2 Approximate Matching Algorithm . . . . . 38
5.2.1 Predicate matching . . . . . 38
5.2.2 Subscription evaluation . . . . . 41
5.3 Sequencing Algorithm . . . . . 41
5.4 Precision-Space Trade Off: . . . . . 45
5.5 Time and Space Analysis . . . . . 46

**6 Experiments** **48**
6.1 Experimental Framework . . . . . 48
6.2 Experimental Results . . . . . 51
6.2.1 Performance Evaluation . . . . . 51
6.2.2 Comparison between crisp model and fuzzy model . . . . . 54

**7 A-ToPSS System Implementation** **57**
7.1 Normal system operation: Toronto apartment rental market . . . . . 59
7.2 Experimental system operation . . . . . 61

**8 Conclusion and Future Work** **63**
8.1 Conclusion . . . . . 63
8.2 Future Work . . . . . 64

**Bibliography** **65**

# List of Tables


1.1 Examples of combinations of publications and subscriptions in an antique auction system. . . . . 5

3.1 Set representations of crisp predicates . . . . . 20

3.2 Possible Operations for T-norm and S-norm . . . . . 21

6.1 The Definitions for different types of subscriptions . . . . . 49

6.2 The definitions for different types of publications . . . . . 50

6.3 Loading Time comparison . . . . . 53

6.4 Memory Cost Comparison . . . . . 54

6.5 Comparison of number of matched subscriptions for various types of subscriptions, event type is fuzzy, $n_S = 70,000$, $n_E = 10$ . . . . . 55

6.6 Comparison of Number of matched for various types of publications, subscription type is fuzzy, $n_S = 70,000$, $n_E = 10$ . . . . . 55

# List of Figures


1.1 Publish/Subscribe Paradigm . . . . . 2

3.1 A possible membership function of the fuzzy set of post-modern paintings. 15

3.2 Shapes of fuzzy interval . . . . . 18

3.3 Possibility and Necessity of $x \in I$ . . . . . 22

4.1 Membership functions for predicates . . . . . 27

4.2 Possibility distributions for publication . . . . . 28

4.3 Cases of possibility measure . . . . . 31

4.4 Cases of necessity measure . . . . . 31

5.1 Data structures . . . . . 37

5.2 Matching procedure . . . . . 38

5.3 Predicate matching algorithm . . . . . 39

5.4 Possibility computation . . . . . 40

5.5 Necessity computation . . . . . 41

5.6 Subscription Evaluation . . . . . 42

5.7 The case of no match between $\pi$ and $\mu$ . . . . . 43

5.8 Improved Possibility Computation . . . . . 44

5.9 Improved Necessity Computation . . . . . 45

6.1 Matching process time . . . . . 52

6.2 Predicate Matching Time Comparison . . . . . 53

7.1 Overall Architecture of Publish/Subscribe System . . . . . 57
7.2 Demonstration Setup of Approximate Toronto Publish/Subscribe System (A-ToPSS). . . . . 58
7.3 The power user's interface for defining approximate subscriptions. . . . . 60
7.4 A–TOPSS implementation design . . . . . 61

# Chapter 1

# Introduction

## 1.1 Overview

The proliferation of pervasive computing devices, the integration of network access technologies (mobile wireless and Internet), and the large number of information providers offering content are driving the need for an information dissemination model that offers its users highly pertinent information in a demand-driven manner. This can be supported extremely well through the publish/subscribe paradigm, where content providers constitute the publishers of information, while content seekers constitute subscribers.

The publish/subscribe paradigm is an interaction model that consists of information providers, who publish events to the system, and information consumers, who subscribe specific interest in events within the system. The publish/subscribe system performs the matching task and ensures the timely notification of subscribers upon event occurrence. Figure 1.1 shows the paradigm of publish/subscribe system.

Events are published in the form of publications and users' interests are subscribed in the form of subscriptions. A publication describes the attributes of a real word artifact. A subscription defines a user's interest through a list of predicates, each predicate is a constraint on an attribute domain. The matching problem is to filter all satisfied

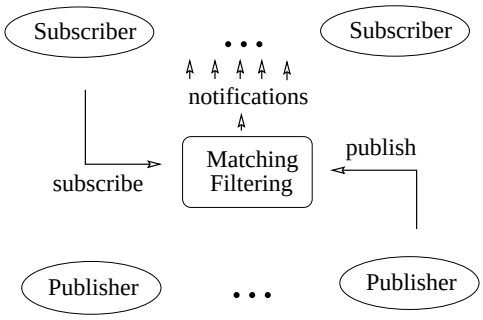


Figure 1.1: Publish/Subscribe Paradigm

subscriptions whose constraints are matched by a coming publication.

In comparison to the query procedure in a database system, events in the publish/subscribe system can be seen as data items (e.g., tuples, columns, or tables) in a relational database model and subscriptions closely resemble database queries. From this point of view, publish/subscribe systems solve a problem inverse to database query processing. Publish/subscribe system performs the evaluation of a piece of data (i.e. event) on a set of queries (i.e. subscriptions) to identify the matching queries.

## 1.2 Motivation

Publish/subscribe has been well studied and many systems have been developed supporting this paradigm. Existing research prototypes include, among others, Gryphon [1], LeSubscribe [8], and ToPSS [13]; industrial strength systems include various implementations of JMS [11], the CORBA Notification Service [3], and TIB/RV [2]. In all these systems, the information of subscriptions and publications are quantified without any uncertainty. These systems can not process vague notions such as "old" etc., but define a sharp limit to differentiate the domain of this notion from the domain that doesn't belong to. These publish/subscribe systems that do not support uncertainties are said to be based on a *crisp* data model.

However, in many situations exact knowledge to either specify subscriptions or pub-

lications is not available. In these cases, the uncertainty about the state of the world has to be cast into the crisp data model that defines absolute limits. However, for a user of the publish/subscribe system it may be much simpler to describe the state of the world with vague and uncertain concepts. That means the subscriptions and publications are expressed in an approximate manner.

In a selective information dissemination context, for instance, users may want to submit subscriptions about a book whose constraint on price is "cheap". On the other hand, information providers may not have exact information for all items published. In a second-hand market, a seller may not know the exact age of an antique vase so that she can only describe it as an "old" vase. Temperature and humidity information collected by sensors are often not fully precise, but only correct within a certain error interval around the value measured. It would be better to publish imprecise information, rather than a wrong exact value, if such publish/subscribe capabilities were possible. Also, uncertainty exists in location-based information dissemination services. Because of the movement of the mobile user and transmission delay, approximate location would be more appropriate rather than assessing a precise location of each user. To date location-based enabling technology, for instance, does not allow continuous and accurate tracking of a user's location. In many current LBS models, a user location is assumed to be provided by the user herself. If it is gathered through GPS it is only accurate to a certain degree and can only be gathered every so often (e.g., it takes around 30 seconds to get the first location quote.)

All publish/subscribe systems developed to date are based on the assumption that a match between a subscription stored in the system and an event submitted to the system is either true or false – that is, a subscription matches or does not match. This partitions the set of subscriptions stored in the Publish/Subscribe system into two disjoint subsets, the matching subscriptions and the non-matching subscriptions. A subscriber is only notified if her subscription is in the set of matching subscriptions. However, it is very

difficult to cast the uncertainties inherent in real world scenarios into a crisp data model that establishes exact limits, as we illustrated in the above scenarios.

For these reasons, we think, it is of great advantage to provide a publish/subscribe data model and a matching scheme that allow the expression and processing of uncertainties for both subscriptions and publications.

## 1.3 Problem Definition

To address the above problem, this thesis extends subscription and publication languages to incorporate the expression of uncertainties at the language level, and develop a matching mechanism to support the processing of the extended language in Publish/Subscribe systems. The extended subscriptions and publications supporting uncertainties will be called *approximate subscriptions* and *approximate publications*. Correspondingly, the mechanism to match approximate publication against approximate subscriptions is called *approximate matching* and the publish/subscribe system (or model) is called *approximate system (or model)*. In an approximate model, a subscription will match a given event with a certain degree of confidence. A degree of zero corresponds to the crisp case of not matching at all and the degree of one corresponds to the crisp case of matching, while a degree between one and zero expresses more or less confidence in the established match.

Table 1.1 gives an overview and a set of examples to illustrate different scenarios of publish/subscribe models with uncertainty. There are five interesting cases according to the different combinations of subscriptions and publications with uncertainties. These are:

1. crisp subscriptions and crisp publications (conventional publish/subscribe system);

2. approximate subscriptions and crisp publications;

3. crisp subscriptions and approximate publications;

| Case | Subscription | Publication |
|---|---|---|
| Crisp subscription<br>Crisp publication | $s : (price \leq \$1000)$<br>and $(age \geq 50)$ | $e : (price, \$800)$<br>$(age, 660)$ |
| Approximate subscription<br>Crisp publication | $s : (price\ is\ not\ expensive)$<br>and $(ageisold)$ | $e : (price, \$800)$<br>$(age, 660)$ |
| Crisp subscription<br>Approximate publication | $s : (price \leq \$1000)$<br>and $(age \geq 50)$ | $e : (price, cheap)$<br>$(age, old)$ |
| Approximate subscription<br>Approximate publication | $s : (price\ is\ not\ expensive)$<br>and $(ageisold)$ | $e : (price, cheap)$<br>$(age, old)$ |
| Miscellaneous | $s : (price\ is\ not\ expensive)$<br>and $(age \geq 50)$ | $e : (price, \$800)$<br>$(age, old)$ |

Table 1.1: Examples of combinations of publications and subscriptions in an antique auction system.

4. approximate subscriptions and approximate publications;

5. a fifth case that combines crisp and approximate constraints in subscriptions and publications (e.g., subscription: `(object = bike) & (model is new ) & (price` $\leq$ `$90)`).

Models 2 to 5 constitute new publish/subscribe system models that haven't been investigated. Problems are raised regarding the matching between crisp or approximate subscriptions and crisp or approximate publications.

All existing publish/subscribe systems are based on a crisp data model that cannot process uncertainties in either publications or subscriptions. This thesis proposes a novel publish/subscribe model that integrates all the five cases described above and we define a matching mechanism that applies to the cases involving uncertainties.

## 1.4 Contributions

The contributions of this thesis are:

1. A new publish/subscribe model supporting uncertainties in subscription and publication formalization. This new model supports all five cases described above and is fully implemented. The expression and processing of uncertain information underlying the application domain raise questions regarding the matching of crisp/approximate subscriptions with crisp/approximate publications.

2. The articulate formalization of the approximate publish/subscribe matching problem and algorithms for solving it.

3. A thorough experimental evaluation of the proposed matching scheme that compares the crisp publish/subscribe model with the approximate publish/subscribe model with respects to performance, memory usage, and number of matched subscriptions.

4. An experimental analysis of a reduced encoding of data structures in the approximate matching algorithms. Approximate publish/subscribe system trades uncertainty of input off against computational precision. The reduced representation admissible for approximate matching exploits this tradeoff.

5. A software demonstration simulating a real world application of the approximate publish/subscribe system is implemented. It integrates a web server through which users can submit their information including subscriptions and publications to the system and a filtering engine running in the background to support the matching and the notification. Furthermore, a control and monitoring panel is developed to experiment with the different effects of parameters on system metrics.

## 1.5 Thesis Outline

The thesis is organized as follows: in Chapter 2, we briefly summarize other related work. In Chapter 3, we introduce the necessary background knowledge – fuzzy set theory and possibility theory, on which we base our model that deals with uncertainties in subscriptions and publications, and matching between them. The approximate publish/subscribe model supporting uncertainty is developed in Chapter 4. In Chapter 5, we present an approximate counting algorithm for a local matching engine and an optimized sequencing algorithm; and analyze the complexity of these algorithms in terms of space and time cost. Chapter 6 evaluate experimentally the performance of our algorithm and Chapter 7 is devoted to the our implementation – the A-ToPSS system. A–TOPSS is the *Approximate Matching-based Toronto Publish/Subscribe System* research prototype that serves us as an experimentation platform to evaluate different subscription and publication language models and approximate matching algorithms. Finally, Chapter 8 concludes the thesis and discusses some directions for future work.

# Chapter 2

# Related Work

Much work has been devoted to developing publish/subscribe systems and event notification services such as Gryphon [1], LeSubscribe [8], and ToPSS[]. Industrial strength systems include various implementations of JMS [11], the CORBA Notification Service [3], and TIB/RV [2]. Common to all the current systems is the crisp matching semantic – neither subscriptions nor publications can express uncertain information and either a match is established or no match is established. A gradual match, as defined in this thesis, that is expressed as a confidence, a degree of match, or a probability, does not exist in any previously studied models.

With the respect of expressing and processing uncertainties, much related work has been done in database querying and document searching. However, none is in the application domain of publish/subscribe systems.

In this chapter, we will review some related work significant in the aspects of crisp publish/subscribe system and approaches involving uncertainty and similarity measure for data retrieval.

## 2.1 Crisp Publish/Subscribe Systems

Much research focuses on the crisp matching model of publish/subscribe system and event notification services. These systems, including LeSubscribe, Gryphon, Elvin, Siena, BDD etc., are distinguished in the subscription language and publication data model they offer and algorithms performing the matching task.

LeSubscribe [8] aims at publish/subscribe support for web-based applications. It focuses on the algorithmic efficiency in supporting millions of subscriptions and high event processing rates. The supported language and data model are based on an LDAP-like semi-structured data model for expressing subscriptions and publications. In this system, a subscription is a conjunction of predicates each of which is a triple (attribute, operator, value). The relational operators supported include $<, \leq, =, \neq, >, \geq$. This system supports push and pull style information dissemination. The matching engine of LeSubscribe falls within the class of two-step matching algorithms – predicate matching step and subscription evaluation step. In the first step, all predicates are matched against the publication; then subscriptions are evaluated in the second step based on the set of matched predicates.

Instead of two-step matching algorithms, Gryphon [4] uses a tree-based data structure to index subscriptions which leads to another category of matching algorithm. In Gryphon, all subscriptions are preprocessed into a tree where each non-leaf node is a test for one attribute and the edges derived from that node represent different results. During the matching, the coming event goes down through the branch it matches until arriving at the leaf nodes containing the matched subscriptions.

Another approach using tree-based algorithm is Binary Decision Diagrams (BDDs) [6]. In this model, each subscription is a boolean function represented by a BDD. This approach is distinguished two aspects: one is that it can support any boolean formula; the other is that overlapping subscription expressions are operated only once if variable

ordering is chosen properly.

Elvin [16] is a content based notification/messaging service that targets application integration environments and monitoring of distributed systems. ELVIN supports a more expressive subscription language which is created as strings including powerful string processing functions and operators on built-in data types covering integer, string and boolean relation. In addition to those traditional comparison operators such as $<, \leq, =, \neq, >, \geq$, Elvin supports operations as matching extended regular expression with strings.

SIENA [5](Scalable Internet Event Notification Architectures) comprises another example of a publish/subscribe event-notification service that presents a very similar publication and subscription language model. This research project is based on content-based networking service and focuses on the routing of subscriptions and publications in a distributed environment so that both services – notification selection (i.e, determining which publications match which subscriptions) and notification delivery (i.e, distributing matching notifications from publishers to subscribers) – are balanced. The advantage of this infrastructure is that it maximizes expressiveness in the selection mechanism without sacrificing scalability in the delivery mechanism.

Lastly we introduce a research project led by At&T research lab. This is READY [15], an implementation of the CORBA notification service. The specific features of READY, which are not offered by existing commercial products, include: information consumer specifications can be matched over both single and compound event patterns; quality of service (QoS) is managed providing ordering properties for event delivery.

## 2.2 The Toronto Publish/Subscribe System Family

Recently, Publish/Subscribe paradigm has gained a significant interest in the applications as selective information dissemination services and location based services. Middleware

System Research Group in University of Toronto works on a series of such paradigm called Toronto Publish/Subscribe System Family including A-ToPSS, S-ToPSS and L-ToPSS. They are Publish/Subscribe systems working in different scenarios.

A-ToPSS (Approximate Toronto Publish/Subscribe System) is a research prototype supporting approximate matching. It is based on a software demonstration [13]. In [13] only subscription language is allowed to express uncertain information for each predicate constraint, such as "cheap", "large", and "close to", so it only works for the second case in Table 1.1. Now this system has been extended to support the expression of uncertainty in both subscriptions and publications. The matching mechanism has been revised to suit all the cases in Table 1.1. This thesis is also based on this work.

S-ToPSS (Semantic Toronto Publish/Subscribe System) is a semantic-aware system where the matching between subscriptions and publications is based on the semantic of terms not on the syntax. For example, the notifications about "automobiles" may be sent to the subscribers who are interested in "vehicles". S-ToPSS uses three approaches to support semantic matching capabilities. One is the use of synonyms. The second uses a concept hierarchy which provides the relationships (specialization and generalization) between attributes and values. The third defines a set of mapping functions that allow arbitrary relationship between schema and attribute values. The added semantic capability is realized by passing the new event and subscriptions through three components where implemented the above approaches to generate a set of semantically equivalent events and subscriptions to be matched by existing algorithm.

L-ToPSS (Push-oriented Location Based Services) uses Publish/Subscribe system in the location based services. It targets to facilitate mobile users for information requests. Upon the filtering engine, L-ToPSS adds a location staging component to collect updates of users' location periodically and a location matching engine to match the location constraints of a subscriber and a publisher. Considering the limited power and input capability of mobile devices, this prototype provides the service in a push-oriented style

to correlate subscriptions with provided publications, thus offer an efficient notification to each mobile user.

## 2.3 Work Related to Uncertainty and Similarity Measure

In many application domains such as information retrieval, document searching, some work has investigated expression and processing of uncertain information. We cite a few representative examples, each of which is out of the scope of publish/subscribe research.

Ronald Fagin uses operations on fuzzy sets to combine fuzzy information from multiple database systems. In a multimedia database, each object has several attributes and a grade of membership is assigned to each attribute for measurement. To determine the top $k$ objects that have the highest overall grades, Fagin proposes an efficient algorithm ("Fagin Algorithm", or FA [9]) to merge several sorted lists based on the rating of the objects from different multimedia database systems. For some monotone aggregation functions, FA is optimal with high probability in the worst case. A more elegant and remarkably simple algorithm ("Threshold Algorithm". or TA [10]) is proposed 3 years later. TA is proved to be optimal in a much stronger sense than FA. [10] shows that TA is essentially optimal for all the monotone aggregation functions, and not just in a high-probability worst-case sense, but over every database. Unlike FA which requires large buffers whose sizes may grow unboundedly as the database size grows, TA allows early stopping, which yields, in a precise sense, an approximate version of the top $k$ answers.

Another application using the knowledge of fuzzy sets is presented in [17]. Wolski *et al.* propose a fuzzy trigger to incorporate imprecise reasoning in active database. The rules that control the event-condition action are modelled by fuzzy membership functions. This work proposes two trigger models. C-fuzzy trigger involves fuzzy inference only in the process of evaluation of the condition. If actions are also expressed in fuzzy terms

and integrated with the condition part, it leads to the CA-fuzzy trigger.

With the proliferation of web documents, an efficient information filtering system based on the similarity between the documents and users' profiles is in great demand. [18] suggest an indexed structure under a vector space model and an algorithm to compute the similarity, which is a distance function of match between a document and a profile based on the "importance" value of certain attributes. In this model, a document is identified by a set of terms represented as a multi-dimensional vector. Each term is assigned a weight as the statistical importance indication. Profiles appear as documents, consisting of a list of terms, each term with a weight. The degree of similarity between a document-profile pair is measured by applying a cosine measure which is a scalar product operation, if both documents and profiles are normalized by their lengths. Based on this similarity measurement, several index refinements are devised to improve I/O and CPU processing time.

Although there is some related work involving representation of uncertainty, none is in the domain of publish/subscribe systems. This thesis is the first to express uncertain notions in publications and subscriptions and concerns the approximate matching problem between them.

There are several approaches to model uncertainties, *i.e.*, fuzzy set theory, possibility measures, probability measures and belief measures. This thesis focus on possibility theory and fuzzy set theory and use them in approximate publish/subscribe system.

# Chapter 3

# 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 a imprecisely known value is to define it as an interval. For example, the interval [50, 150] would be reasonable to represent a piece of "post-modern" painting in an online auction. However, in 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 generated 49 years ago may satisfy the subscriber, but it won't be delivered to the subscriber since it is out of the domain of interval [50,150]. To overcome this limitation in representing uncertain concepts, fuzzy set theory [12] and possibility theory [7] have been developed. The publish/subscribe model we are introducing is based on these theories in modelling 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 [12, 7].

## 3.1 Fuzzy Logic 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 3.1** *A fuzzy set $M$ on a universal set $U$ is a set that specifies for each element $x$ of $U$ a degree of membership to the fuzzy set $M$. It is defined by a membership function (a.k.a. characteristic function),*

$$\mu_M : U \rightarrow [0, 1]$$

*that specifies for each $x \in U$ its degree of membership $\mu_M(x)$ to the fuzzy set $M$.* [1]

This membership function can be either a discrete or continuous function depending on whether the domain is countable or uncountable.

For example, we can define a possible membership function for the fuzzy set of "post-modern paintings"as shown in Figure 3.1, where the domain ranges over the possible ages in the given application context. Using this function, we can determine the membership grade of each age in this fuzzy set, which signifies the degree to which that age is post-modern. For instance, the age 30 is assigned a grade of 0, the age 45 a grade of 0.67 and the age 100 a grade of 1.

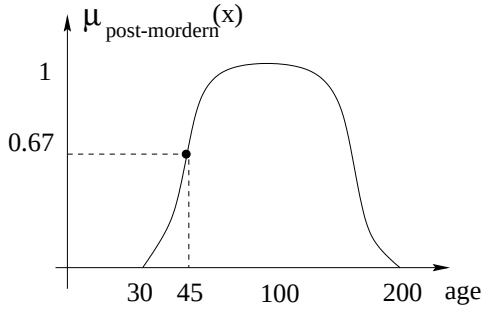


Figure 3.1: A possible membership function of the fuzzy set of post-modern paintings.

In subscriptions, each attribute is constrained by a predicate. We will use a fuzzy set to represent the predicate whose constraint includes uncertain information. Each fuzzy

[1] The membership function is a generalization of the characteristic functions of classic set theory. In classic set theory, there are only two values 0 or 1 for the characteristic function to classify the elements belonging to a set or not.

set is distinguished by its membership function, thus membership function in stead of fuzzy set will be used hereafter to represent the constraint of attribute for each predicate.

In fuzzy logic [19], the representation of fuzzy natural language is based on test-score semantics which may be viewed as a collection of elastic or fuzzy constraints. There are three elements in this semantics: variables $X_1, X_2, \cdots, X_n$ whose values are constrained; constraints $C_1, C_2, \cdots, C_n$ and membership $\mu_i$ assigned to each variable $X_i$ representing a degree to which $C_i$ is satisfied. Correspondingly, in the publish/subscribe system, variables are the attributes(e.g. price, size or height), constraints are fuzzy values (e.g. cheap, small or tall). Membership can be assigned by discrete numbers or continuous function. Using these three elements, we will introduce the concept of a canonical form. The representation of publications and subscriptions in the formal model will use the canonical form.

**Definition 3.2** *A canonical form of proposition p, abbreviated as $cf(p)$ is expressed as*

$$cf(p) = X \text{ is } F$$

*where $X = (X_1, X_2, \cdots, X_n)$ is the constrained variable and $F$ is a fuzzy notion which plays the role of an elastic constraint on $X$.*

For example, in natural language, a predicate $p$ presents

$$p : a \text{ big room},$$

the the implicit canonical form of $p$ is

$$cf(p) = Size \text{ is } big.$$

The concept of canonical form is convenient for us to unify crisp subscriptions and publications and approximate ones since the predicate operator used in traditional crisp system is omitted. Now the representation of publications and subscriptions comes down

to the representation of fuzzy constraints, where the membership function in fuzzy set theory works well.

Pertaining to the modified predicate, we have a set of rules to represent natural language modifiers. The canonical form of a predicate $p$ is

$$p : X \ is \ F.$$

The modified predicate has a form

$$p^+ : X \ is \ mF \rightarrow X \ is \ F^+$$

where $F^+$ is a modification of $F$ induced by $m$. We define:

(a) $m$ is *not*: $F^+ = 1 - F$ (negation)

(b) $m$ is *very*: $F^+ = F^2$ (concentration)

(c) $m$ is *more or less*: $F^+ = \sqrt{F}$ (diffusion)

With the canonical form of natural language and the rules for modifiers, hereafter, we will express predicates in the form of $p : X \ is \ mF$.

## 3.2 Set Representation

There are many possible choices for membership functions such as discrete function, trapezoidal form or other analytical function. In our model, we use a general parametric representation suggested by [7]. The membership function of a fuzzy set $M$ can be represented with a pair of functions denoted by $L$ and $R$ and four parameters $(\underline{m}, \overline{m}, \alpha, \beta) \in \Re^+ \cup \{+\infty, -\infty\}$. $L$ and $R$ (both are $\Re^+ \rightarrow [0, 1]$) are monotonically increasing and decreasing, respectively, and are upper semi-continuous ($u.s.c$). This pair of functions and the four parameters define the membership function of a fuzzy set $M$ as

follows:

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

A fuzzy set is characterized by its membership function, so without ambiguity we can say $M$ is defined by $\mu_M(x) = (\underline{m}, \overline{m}, \alpha, \beta)_{LR}$. The intuition of this representation is: $[\underline{m}, \overline{m}]$ defines the domain where elements definitely belong to the set where they have grade 1; $[\underline{m}-\alpha, \underline{m}]$ and $[\overline{m}, \overline{m}+\beta]$ are the left and right boundaries that show the transitions from grade 0 to grade 1. $L$, $R$ are the gradual functions of these transitions. The related terminologies are defined as follows:

**Definition 3.3** *$[\underline{m}, \overline{m}]$ is the core of the fuzzy set $M$, denoted by $\dot{\mu}_M$. $\underline{m}$ and $\overline{m}$ are referred to the lower and upper model values of $M$, respectively. The support of a fuzzy set $M$, denoted by $S(\mu_M)$, is the domain of values where $\mu_M(u) > 0$. If $M$ is of bounded support, then $S(\mu_M)$=$[\underline{m}-\alpha, \overline{m}+\beta]$. $\alpha$ and $\beta$ are called the left-hand spread and the right-hand spread, respectively.*

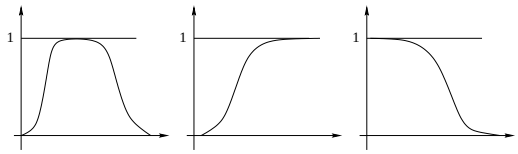


Figure 3.2: Shapes of fuzzy interval

Figure 3.2 shows three sample shapes of membership functions of fuzzy sets: "bell-shaped" curves, non-decreasing function and non-increasing function. Their parametric representations are as follows:

$$\textit{bell-shaped function: } \mu_M = (\underline{m}, \overline{m}, \alpha, \beta)_{LR}$$

$$\textit{non-decreasing function: } \mu_M = (\underline{m}, +\infty, \alpha, +\infty)_{LR}$$

$$\textit{non-increasing function: } \mu_M = (-\infty, \overline{m}, -\infty, \beta)_{LR}$$

There are many advantages of this representation. First, it keeps the nature of fuzzy set which eliminates sharp boundaries existing in an interval representation to differentiate set members from non-members. Second, this parameterization is a very general representation and is straight forward to implement. Third, this formalization is expressive. It can represent many different types of information including crisp data and uncertain data. We list some representatives in the following,

- $M$ is a set which contains only one real number $m \in \Re$. Then, by convention, $\mu_M = (m, m, 0, 0)_{LR}, \forall L, \forall R$.

- $M = [a, b]$. Then, $\mu_M = (a, b, 0, 0)_{LR}, \forall L, \forall R$.

- The membership function of $M$ is of trapezoidal form. Then $\mu_M = (\underline{m}, \overline{m}, \alpha, \beta)_{LR}$ applies, with $L(x) = R(x) = \max(0, 1 - x)$.

- $M$ is such that $\forall x \geq \underline{m}$, $\mu_M(x) = 1$. Then $\mu_M = (\underline{m}, +\infty, \alpha, +\infty)_{LR}$.

- $M$ a set contains a fuzzy number. Then $\underline{m} = \overline{m} = m$, and we write $\mu_M = (m, m, \alpha, \beta)_{LR}$. One can choose $L(x) = \max(0, 1 - x)^p$ or $\max(0, 1 - x^p)$ with $p > 0$, or $e^{-x}$, $e^{-x^2}$, etc.

Finally, this representation is easily extended to represent crisp sets defined by Boolean relational predicates, therefore it integrates the crisp data model. For example, for a crisp predicate $(p \geq v)$, the membership function degenerates to the characteristic function as follows:

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

Table 3.1 lists the representations and the domains where the function values are 1 for all cases of crisp predicates.

| crisp predicate | set representation | domain where has value 1 |
|---|---|---|
| price $\geq v$ | $\mu_M = (v, +\infty, 0, +\infty)_{LR}$ | $[v, \infty]$ |
| price $\leq v$ | $\mu_M = (-\infty, v, +\infty, 0)_{LR}$ | $[-\infty, v]$ |
| price $= v$ | $\mu_M = (v, v, 0, 0)_{LR}$ | $v$ |

Table 3.1: Set representations of crisp predicates

## 3.3 Operations on fuzzy sets

Since each predicate is represented by a fuzzy set, the relation of those predicates within one subscription is on operation on fuzzy sets. Operations R involving two or more fuzzy sets $A_1, ..., A_n$ are generally defined by a mapping $f$ that aggregates the membership functions as follows:

$$\mu_{R(A_1,...,A_n)}(x) = f(\mu_{A_1}(x), ..., \mu_{A_n}(x)).$$

Intersection, union, and other set operations are defined in this manner. The mapping $f$ can be T-norm or S-norm functions. A T-norm, which means triangular norm and is often used to model set intersection is a binary mapping $T$ satisfying the following axioms:

- boundary: $T(0,0) = 0, T(1,1) = 1, T(1,0) = T(0,1) = 0$,
- monotonicity: $T(a,b) \leq T(c,d)$ *if* $a \leq c$ *and* $b \leq d$,
- commutativity: $T(a,b) = T(b,a)$,
- associativity: $T(a, T(b,c)) = T(T(a,b), c)$.

The first axiom imposes a correct generalization to crisp sets, that is, $T$ behaves as the classical intersection with 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 the intersection of A and B. The third axiom indicates that the operator is indifferent to the

| norm | $\mu_{R(A_1,...,A_n)}(x)$ | $f(\mu_{A_1}(x), ..., \mu_{A_n}(x))$ |
|---|---|---|
| T-norm | $\mu_{A_1 \wedge A_2}(x)$ | $\min(\mu_{A_1}(x), \mu_{A_2}(x))$<br>$\mu_{A_1}(x) \cdot \mu_{A_2}(x)$<br>$\max(0, \mu_{A_1}(x) + \mu_{A_2}(x) - 1)$ |
| S-norm | $\mu_{A_1 \vee A_2}(x)$ | $\max(\mu_{A_1}(x), \mu_{A_2}(x))$<br>$\mu_{A_1}(x) + \mu_{A_2}(x) - \mu_{A_1}(x) \cdot \mu_{A_2}(x)$<br>$\min(1, \mu_{A_1}(x) + \mu_{A_2}(x))$ |

Table 3.2: Possible Operations for T-norm and S-norm

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.

Set union is defined and motivated in a similar manner. Operators that define set union are denoted as co-T-norms or S-norms. A S-norm operator is a binary mapping $S$ satisfying:

- boundary: $S(1,1) = 1, S(1,0) = S(0,1) = S(0,0) = 0$,
- monotonicity: $S(a,b) \leq S(c,d)$ *if* $a \leq c$ *and* $b \leq d$,
- commutativity: $S(a,b) = S(b,a)$,
- associativity: $S(a, S(b,c)) = S(S(a,b), c)$.

Different S-norms and T-norms are used in literature to represent set union and set intersection. In our model, we choose the popular operations where maximum is used for union and minimum for intersection. Anther choice is multiplication and addition. Table 3.2 list possible operations on fuzzy sets.

## 3.4 Possibility Theory

Possibility theory formally defines fuzzy measure, which reflect user's subjective uncertainty towards a given state of the world. Dubois [7] gives a good example to explain this notation clearly: given an imprecise variable $x$ which can be any value in $A = [a, b]$, the proposition that "$x$ belongs to the interval $I$" can naturally be qualified by two modalities "possibility" denoted by $\Pi$ and "necessity" denoted by $N$.

- If $A \cap I$ is not empty then "$x \in I$" is possibly true, but not necessarily, where $\Pi = 1, N = 0$;
- if $A \subseteq I$ then "$x \in I$" is necessarily true where $\Pi = 1, N = 1$.


Figure 3.3: Possibility and Necessity of $x \in I$

In general, possibility and necessity can be not only 0 or 1, but any value in between. Therefore, a function $\pi_A(x)$, which can be viewed as a likelihood function is used to indicate the distribution of possibility $x$ belongs to $A$. We can use the same parametric representation for membership function to represent possibility distribution function. In our work, a possibility distribution function is used to model publication information where we are not sure of the occurrence of certain values of the event. For example, the antique shop has an art piece whose age is not aware. The seller only describe it as a post-modern art. For this publication, we use a possibility distribution function $\pi_{post-modern}(x)$ to represent the possibility of different age. This function which is defined by the information provider can be the same as Figure 3.1 or a different one.

A possibility distribution is similar to a probability distribution. A degree of possibility can be viewed as an upper probability bound, a possibility distribution can also encode probability distributions with extreme values. However, it differs from probability theory because it uses a pair of dual set functions (possibility and necessity measures) instead of only one. Besides, it is not additive and makes sense on ordinal structures. A possibility is a more general notion than a probability.

# Chapter 4

# A Model for Publish/Subscribe with Uncertainties

Our objective is to model uncertainties in subscriptions and publications, and to define an approximate matching semantic for publish/subscribe systems. This chapter compares the characteristics of publish/subscribe systems that support approximate matching and crisp matching, and presents the formal definition of the approximate model for publish/subcribe systems. The approximate model is based on the concepts of fuzzy sets and possibility theory.

## 4.1 Characteristics of Approximate Publish/Subscribe Model

In the introduction chapter, we motivated the need for a publish/subscribe system that allows the expression and processing of information with uncertainty. A publish/subscribe system supporting approximate matching differs from traditional publish/subscribe system in the following aspects:

- Predicates: In the approximate model, predicates are qualified by words used in natural language, e.g., *tall, warm, cheap*. These predicates do not specify an exact quantity and the interpretations may vary with context. In contrast, the predicates under the crisp model are quantifications of some attributes. Furthermore, in all existing crisp systems, only one modifier is allowed, which is negation, *not*. In the approximate model, a variety of predicate modifiers can be used, e.g. *very, more or less, not*.

- Possibilities: In the crisp model, the coming event is supposed to happen for sure, where the data of publication always has the occurrence probability of 1. In the approximate model, we don't have this assumption. We use a grade of possibility and necessity to cast the occurrence of the publication. The grades are quantified as a number between 0 and 1.

- Truth: In the crisp publish/subscribe systems, the match result of a publication and a subscription is either true of false, no degrees in between. The approximate model, however, uses a value in the interval [0,1] to express the degree of the match between a publication and a subscription.

We will present an approximate publish/subscribe model to deal with these characteristics. In the following sections, we will introduce a structured representation for publications and subscriptions. Then we give the definition of approximate matching. At last, we discuss the approximate model in concrete cases.

## 4.2 Subscription Language Model

Subscriptions define users' interests through boolean combinations of a list of predicates. Each predicate expressed in a canonical form is a constraint over a domain of values. A predicate is represented as $(a_i, \mu_i^+)$. $a_i$ is the attribute of the predicate, $\mu_i^+$ is a modi-

fied membership function to represent fuzzy constraint on the attribute. Here, we have already involved *modifier* into the membership function. We use $R$ to represent the boolean relation of predicates within one subscription, then a subscription is formalized as

$$s = R((a_1, \mu_1^+), (a_2, \mu_2^+), \cdots, (a_m, \mu_m^+)).$$

$R$ can be any relation, i.e. *conjunction* or *disjunction*. For example, a student is looking for an apartment with constraints on price, size and age. Her subscription in natural language that specifies these constraints is as follows:

$$\begin{array}{lllll} S: & size & is & & medium \quad \texttt{AND} \\ & price & is & no\ more\ than & \$1500 \quad \texttt{AND} \\ & age & is & not\ very & old \end{array}$$

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

$$\mu_{\leq 1500}(x) = \begin{cases} 1 & \text{if} \quad x \leq 1500; \\ 0 & \text{if} \quad x > 1500; \end{cases}$$

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

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

and

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

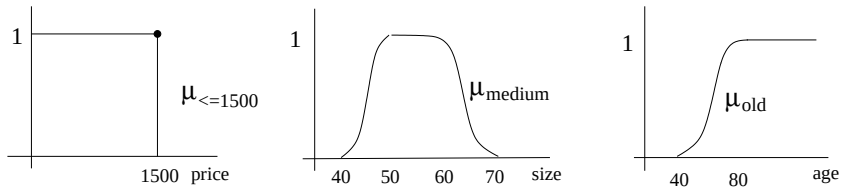


Figure 4.1: Membership functions for predicates

The three membership functions of this subscription is pictured in Figure 4.1. Applying the definitions of the modifiers "very", then "not", the third predicate is expressed as (age, $1 - \mu^2_{old}$). In this subscription, the relation of the three predicates is conjunctive. All predicates are linked together by *intersection* (i.e., *and* $\wedge$). The formalization of this subscription is

$$S = (size, \mu_{medium}) \wedge (price, \mu_{\leq \$1500}) \wedge (age, 1 - \mu^2_{old})$$

For simpleness, hereafter we only use $\mu$ for the modified membership function, not $\mu^+$ in the representation. However, we keep in mind that $\mu$ has already involved modifiers into it.

## 4.3 Publication Data Model

Publications (i.e., events) describe real world artifacts or states of interest through a set of attribute value pairs. For certain attributes, exact values may not be available. In these cases, we use a possibility distribution as defined in Chapter 3. A possibility distribution shows the possibility that the attribute has a given value. A publication is thus defined as a list of attribute function pairs as follows:

$$e = \{(a_1, \pi_1), (a_2, \pi_2), \cdots, (a_n, \pi_n)\}.$$

For example, an apartment advertised for rent, may be described with this notation as follows:

$$\text{e: (size is } 60m^2\text{) and (rent is cheap).}$$

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 possibility of each value of the domain of discourse (i.e., all admissible rent values) as being "cheap". Formally, this publication can be represented by a set of attribute function pairs as follows:

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

where

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

and

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

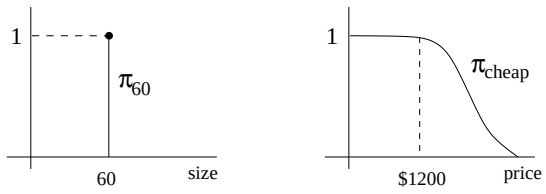


Figure 4.2: Possibility distributions for publication

## 4.4 The Approximate Matching Problem

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. The truth value, true or false, is no longer sufficient for representing the state of a match between a publication and a subscription, because the degree of a match can be between true and false. In the approximate model, a value between 0 and 1 is used to represent the degree of

match between a subscription and each publication processed by the system. Individual subscription can match a given publication more or less, depending on this degree of match.

Subscriptions and publications are represented as:

$$sub = R((a_1, \mu_1), (a_2, \mu_2), \cdots, (a_n, \mu_n))$$

$$pub = \{(a_1, \pi_1), (a_2, \pi_2), \cdots, (a_n, \pi_n)\}$$

The semantics of matching subscription with publications is to measure the possibility and necessity with which the publication satisfies the expectation expressed by a subscription. Based on possibility theory, we use a pair $(\Pi_i, N_i)$ to denote the evaluation of the possibility and necessity of how the publication satisfies predicate $i$ (i.e., the match between $\mu_i$ and $\pi_i$). This measure is done by computing the intersection between $\mu_i$ and $\pi_i$.

**Definition 4.1** *The possibility and necessity of a match between two functions $\mu$ and $\pi$ are computed by*

$$\Pi_i = \sup_{x \in D} \min(\mu_i(x), \pi_i(x))$$

*and*

$$N_i = \inf_{x \in D} \max(\mu_i(x), 1 - \pi_i(x)),$$

*respectively.*

The possibility $\Pi$ and necessity $N$ are a pair of dual fuzzy measures. A degree of possibility can be viewed as an upper probability bound, thus possibility is a "loose" measure compared to probability. Only $\Pi$ itself is not enough to illustrate the matching degree between a publication and a subscription. We need its dual measure necessity as the complementarity of possibility. Moveover, we give an interpretation from a subjective view in the approximate publish/subscribe model. Since a possibility degree is always larger than its dual necessity degree, an optimistic user will count on the loose possibility

measure and a pessimistic user, however, would count on a more strict necessity measure. Considering the example in Figure 3.3, when $A \cap I \neq \emptyset$, the possibility of $x \in I$ is 1. If the user is optimistic, he thinks that $\Pi = 1$ means $x$ is very likely in $I$, he will be satisfied with such $I$. If the user is strict, he is not satisfied with such $I$ because it is still likely $x$ is not in $I$ though $\Pi = 1$. He is satisfied only with the $I$ where $A \subseteq I$ and $N = 1$. $\Pi$ and $N$ can be considered as two bounds on the degree of match, depending on the point of view of the user.

From the possibility computation formula, it easily gets the following lemma.

**Lemma 4.1**

$$\Pi_i = 0 \Leftrightarrow S(\mu_i) \cap S(\pi_i) = \emptyset$$

$$\Pi_i = 1 \Leftrightarrow \exists x = \dot{\mu_i}(x) \cap \dot{\pi_i}(x).$$

We give an interpretation of this lemma through a list of figures in Figure 4.3. If these functions don't intersected with each other, the predicate defined by $\mu_i$ cannot be satisfied by any value of $\pi_i$, so the possibility $\Pi_i = 0$ Figure (4) shows such case. If the cores of the two functions $\mu$ and $\pi$ overlap as shown in case (2), it means the value with possibility 1 in $\pi_i$ is also the value whose membership is 1 in $\mu_i$, then it is possible that the predicate is satisfied, thus $\Pi = 1$. If the function $\pi$ that constrains the publication attribute is a point value, for example, $a = v$, then the possibility and necessity of the match are the same which is the value of predicate function $\mu$ at the point of $v$, where $\Pi = N = \mu(v)$. (3) is a general case where the possibility is the intersection of these two functions.

The explanation for the dual measure – necessity is similar as shown in Figure 4.4. The explanation for $N_i = 0$ is that there is a value with possibility 1 in $\pi_i$ located in the region defined by the complement of $\mu_i$, so there is little chance that the predicate can be matched. When the whole $\pi_i$ is included in the core of $\mu_i$, then any value of $\pi$ has a membership 1 in $\mu_i$, of course the publication will always satisfy the subscription no matter what its value is. In all, we have:

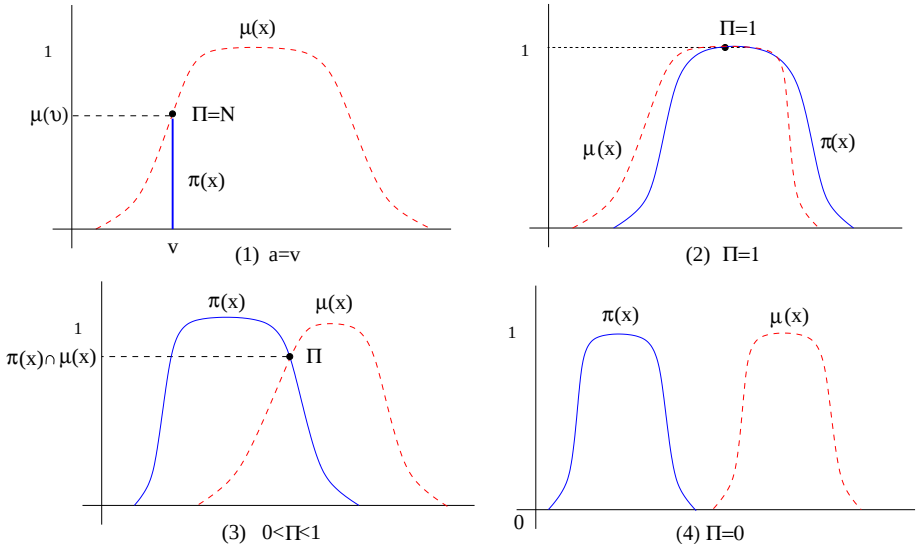


Figure 4.3: Cases of possibility measure

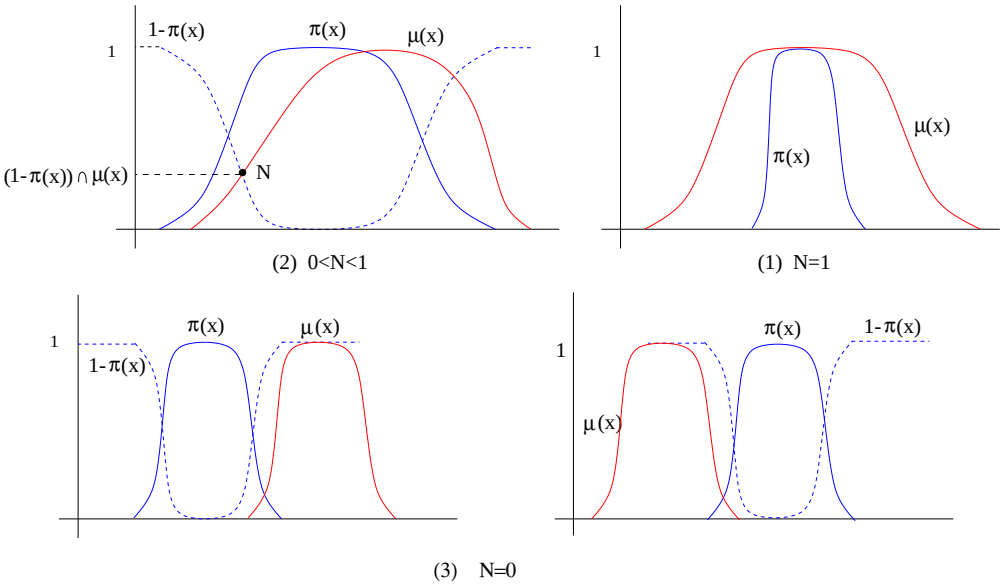


Figure 4.4: Cases of necessity measure

**Lemma 4.2**

$$N_i = 0 \Leftrightarrow \exists x \in \overline{S(\mu_i)} \cap \dot{\pi}_i$$

$$N_i = 1 \Leftrightarrow S(\pi_i) \subseteq \dot{\mu}_i$$

There is one more point need to be mentioned. The matching between a predicate and an attribute function pair of the publication does not depend on the words used to describe the fuzzy constraint, but on the function shapes that represent the meaning of those words. For example, if the two functions $\mu_{cheap}$ and $\pi_{cheap}$ don't intersect each other, publication can not match the predicate even though both have the same constraint "cheap". On the other hand, $\mu_{cheap}$ may be matched by $\pi_{expensive}$ even though one constraint is "cheap" and the other is "expensive" as long as these two functions overlap each other.

Having the possibility and necessity degree for each predicate, the overall matching degree for a subscription is evaluated using the s-norm or t-norm function according to whether the relation of predicates contained in the subscription is conjunctive or disjunctive. Usually we choose maximum operation as the t-norm function and minimum as the s-norm.

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's perception of what constitutes a "good" match versus a "bad" match will certainly differ. Furthermore, a large number of slightly matching subscriptions, i.e., with a low degree of match, may not be a useful idea, since users may be overwhelmed with the number of matched returned. For these reasons, the approximate matching model introduces two parameters to control the tolerance of a match on a per predicate basis for each subscription. They are $\theta_\Pi$ and $\theta_N$.[1] Users' constraints are matched if both

[1] $\theta_\Pi$ and $\theta_N$ define users' satisfactions on the possibility and necessity that their interests are matched (i.e., $\Pi \geq N$). Because the possibility $\Pi$ of a match is never less than the necessity $N$, the threshold $\theta_\Pi$ should be larger or equal to $\theta_N$ (i.e., $\theta_\Pi \geq \theta_N$).

possibility and necessity degree are larger than the thresholds for $\theta_\Pi$ and $\theta_N$. To control these parameters, we change to subscription of the apartment example in Section 4.2 to:

$$S : (size, \mu_{medium}, 0.8, 0) \wedge (rent, \mu_{\leq 1500}, 1, 0) \wedge (age, \mu_{new}, 1, 0.3)$$

This subscription matches a publication as long as its size predicate matches a *medium* value with a degree of more than 0.8, which is the possibility threshold. The necessity threshold is irrelevant here (it is 0). The rent should be less than $1500, and the age predicate matches a new value with a possibility threshold of 1 and a necessity threshold of 0.3.

The general representation of subscription is modified to:

$$sub = R((a_1, \mu_1, \theta_{\Pi_1}, \theta_{N_1}), \cdots, (a_n, \mu_n, \theta_{\Pi_n}, \theta_{N_n}))$$

Now we give the definition of *matching* between subscriptions and publications.

**Definition 4.2** *Given a set of subscriptions $S$ and a publication $p$, the matching problem in the approximate publish/subscribe system is to identify all $s \in S$ such that $s$ and $p$ match with a degree greater than the thresholds defined on $s$ by any subscriber.*

Based on the above discussion, we generalized the matching problem into a formula:

$$S_i(x_1, x_2, \cdots, x_n) = R(\mu_{i1}(x_1), \mu_{i2}(x_2), \cdots, \mu_{in}(x_n))$$

$$e(x_1, x_2, \cdots, x_n) = \{\pi_1(x_1), \pi_2(x_2), \cdots, \pi_n(x_n)\}$$

$$\Pi_{e \circ S_i}(x_1, x_2, \cdots, x_n) = t(\sup\min(\mu_{i1}(x_1), \pi_1(x_1)), \cdots, \sup\min(\mu_{in}(x_n), \pi_1(x_n)))$$

$$N_{e \circ S_i}(x_1, x_2, \cdots, x_n) = t(\inf\max(\mu_{i1}(x_1), \pi_1(x_1)), \cdots, \inf\max(\mu_{in}(x_n), \pi_1(x_n)))$$

We take $x_1, \cdots, x_n$ as the attributes that are concerned by subscriptions and publications, thus attribute names are omitted in the representations. $e \circ S_i$ stands for "$e$ matches $S_i$". $t$ is the operator to treat relation $R$ for overall evaluation, We can choose any one operation from Table 3.2 as $t$. For example, if the relation $R$ of the predicates is conjunctive and we choose *min* as the operation $t$, then the overall match degree of a subscription is the minimum of the degrees of predicates this subscription contains.

## 4.5 Case Studies

In this section, we discuss the approximate matching problem in four concrete cases. When matching a publication with a subscription, the substantial problem is the matching between predicates and attribute descriptions (hereafter attribute is used for the attribute description) of the publication. Having the possibility and necessity degree for each predicate, the evaluation of the match degree of a subscription is only a $t$ operation (e.g., min or max) on these degrees no matter subscriptions and publications are crisp or approximate. Therefore, we base our case studies on the categories whether predicate or attribute rather than subscription or publication is approximate or crisp.

In many fuzzy logic references, *crisp* refers to the cases where two-value logic is applied and *fuzzy*, or *approximate* in this thesis, refers to the cases where multi-value logic is applied. Under this understanding, an interval predicate is a crisp predicate. However, using the subscription and publication language model supporting uncertainties presented in this thesis, an interval predicate is a special case of the approximate predicate. Therefore, we modify the case classification according to whether the predicate or attribute has a point value or approximate one. *Point* refers to the case where the constraint on an attribute is equal to a point value. *Approximate* includes all other cases where the representation for a value is not point, but an interval, a trapezoidal function, or a "bell-shaped" function etc.

**Case 1 Point predicate and point attribute:** Both the values of predicate and attribute are points. For example, the predicate about age is "age is 70 years". The representation in traditional publish/subscribe system is "p: (age, =, 70)". The formal representation in our model is "p: (age, $\mu_{=70}, \theta_\Pi, \theta_N$)". Function $\mu_{=70}$ yield a degree 1 only at the point 70. It yields values 0 in all other places. The publication information about age is also a point as "e: (age, $y$)". Then the predicate is matched if and only if $y = 70$ with possibility $\Pi = 1$ and necessity $N = 1$. Otherwise, if $y \neq 70$, the predicate

is not matched which means $\Pi = 0, N = 0$.

**Case 2 Point predicate and approximate attribute:** If the predicate value is a point $v_p$, the attribute value of the publication has a range with possibility distributed by a function $\pi$, then the possibility of the predicate being matched is the value of function $\pi$ at point $v_p$, $\Pi = \pi(v_p)$. The necessity is always 0. This is intuitive because the attribute of the publication has a much broader value range than predicate constraint, which is only a point. Thus $\pi$ has little chance to satisfy the predicate. For example, predicate p is still "age is 70 years". Publication is "old aged", represented by "e: (age, $\pi_{old}$)", where the function $\pi_{old}$ is as in Figure 3.1. Then the possibility of $e$ matches $p$ is $\Pi = \pi_{old}(70) = 0.67$, and the necessity is $N = 0$.

**Case 3 Approximate predicate and point attribute:** Contrary to case 2, in this case the predicate uses a function $\mu$ to represent its fuzzy constraint and the attribute of the publication has a point value $v_e$. Possibility measure is similar to case 2 where $\Pi = \mu(v_e)$, but necessity measure is quite different. In this case, the necessity is the same as the possibility, $N = \mu(v_e)$, but not 0. In Figure 4.3, (1) illustrated this situation. For example, we exchange the predicate and the attribute of publication of case 2, having "p: (age, $\mu_{old}, \theta_\Pi, \theta_N$)" and "e: (age, 70)". We will get $\Pi = N = \mu_{old}(70) = 0.67$. The user will be satisfied if 0.67 is larger than both $\theta_\Pi$ and $\theta_N$.

**Case 4 Approximate predicate and approximate attribute:** This is the most general case. In the previous section, we have illustrated the matching problem between approximate predicates and approximate attributes of publications in Figure 4.3 and Figure 4.4. The possibility and necessity of match are calculated by Definition 4.1.

# Chapter 5

# Data Structures and Algorithms

This chapter proposes a basic two-phase algorithm and a data structures to solve the approximate matching problem defined in the approximate publish/subscribe model. The two-phase matching algorithm first matches all predicates against the incoming event. Subscriptions that associate with the matched predicates are then evaluated, and the matched subscriptions are identified. As an improvement to the basic algorithm, a sequencing algorithm is developed to evaluate only predicates that have a high probability of being matched, thereby minimizing the number of evaluated predicates. These algorithms are designed to support efficient operations for millions of subscriptions and high publishing rates. The time and space cost of the algorithms will be analyzed.

## 5.1 Data Structures

Each subscription contains a set of predicates. The predicates in different subscriptions may be the same if they are constraints on the same attribute and the functions representing the constraints are the same. Considering the space cost, the same predicate is stored only once. And subscription is identified by storing a list of predicate IDs that this subscription contains.

Predicate evaluation is based on two data structures: a hash table to index predi-

cates according to their attribute names and a predicate vector to store the possibility and necessity degrees of match for each predicate. Subscription evaluation is based on an association matrix to record which predicate belongs to which subscription and a subscription vector to keep track of the measures of match of each subscriptions. The overall data structure is depicted in Figure 5.1.

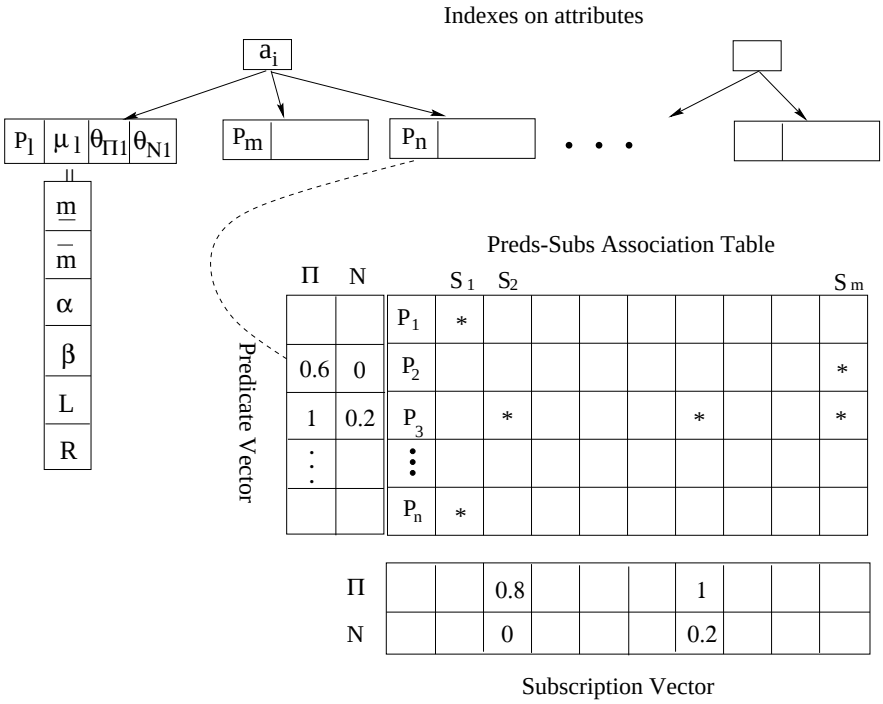


Figure 5.1: Data structures

Each predicate is a tuple with 5 elements $(pid, attr, \mu, \theta_\Pi, \theta_N)$. Obviously, *pid* is the predicate ID, *attr* indicates what attribute this predicate concerns. $\mu$ is the function to describe user's constraint on *attr*, represented by parameters $(\underline{m}, \overline{m}, \alpha, \beta, L_m, R_m)$. $[\underline{m}-\alpha, \overline{m}+\beta]$ is the support of the predicate (i.e., domain values for which the membership degree is large than 0.) $L_m$ and $R_m$ are indexes of a function family indicating which functions are used to represent the change of left-hand spread and right-hand spread. The choices of these parameters depends on concrete applications. Section 3.2 explains that this representation can model both approximate and crisp predicates.

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

## 5.2 Approximate Matching Algorithm

The matching algorithm proceeds in two stages. First predicates are matched and, second, matching subscriptions are identified. The procedure is shown in Figure 5.2.

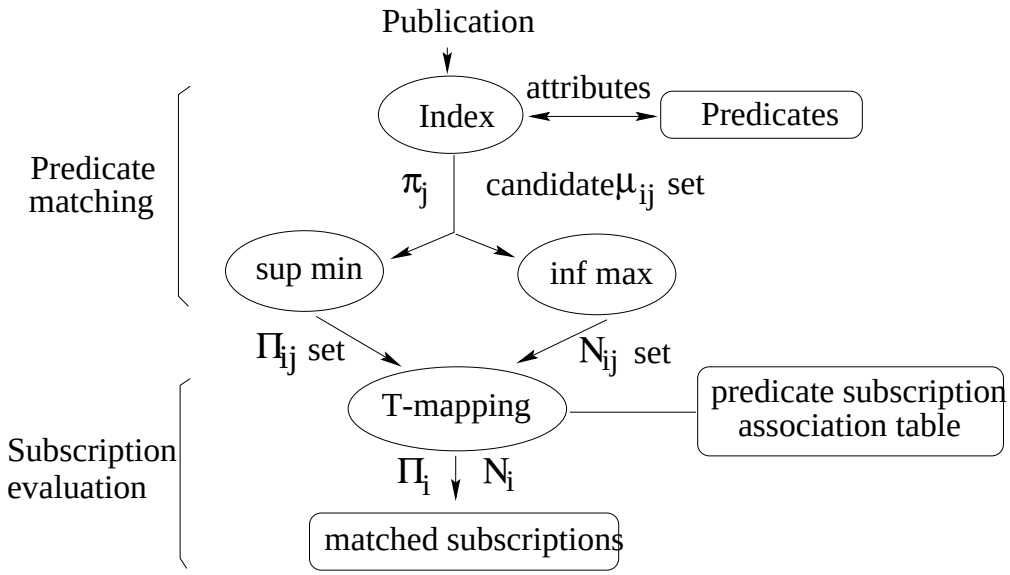


Figure 5.2: Matching procedure

### 5.2.1 Predicate matching

**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 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.

**Input:** $e = (a_1, \pi_1)(a_2, \pi_2) \cdots (a_n, \pi_n)$

**Global Variables:**

$I$: a set of index;

$Vp$ : a predicate vector storing $(\Pi, N)$ for each predicate;

$SatPreds$: a set of satisfied predicates;

**Body:**

1. $Vp = 0$, $SatPreds = \emptyset$
2. **for** each attribute $a_i$ in $e$

   locate the corresponding index $i$ in $I$

   **for** each predicate $(a_i, \mu_i, \theta_{\Pi_i}, \theta_{N_i})$ reached by $i$

   $Vp[p].\Pi$=Possibility-computation$(\mu_i, \pi_i)$

   $Vp[p].N$=Necessity-computation$(\mu_i, \pi_i)$

   **if** $Vp[p].\Pi > p.\theta_\Pi$ and $Vp[p].N \geq p.\theta_N$ **then**

   $SatPreds = SatPreds \cup \{p\}$
3. return $SatPreds$

Figure 5.3: Predicate matching algorithm

There are three cases in the computation of possibility (refer to Figure 4.3) :

$$\Pi = \sup_{x \in D} \min(\mu(x), \pi(x))$$

- $\Pi = 0$ if the supports of $\mu$ and $\pi$ are disjoint , where $[\underline{m}-\alpha, \bar{m}+\beta] \cap [\underline{n}-\gamma, \bar{n}+\delta] = \emptyset$. In this case it is impossible they are matched each other.

- $\Pi = 1$ if the cores of $\mu$ and $\pi$ are overlapped, where $[\underline{m}, \bar{m}] \cap [\underline{n}, \bar{n}] \neq \emptyset$. The possibility that publication satisfies this predicate is 1;

- otherwise, $\pi$ equals to the intersection between left-hand arm of $\mu$(or $\pi$) and right-hand arm of $\pi$(or $\mu$).

Figure 5.4 depicts the detail of the possibility computation process:

**Procedure** Possibility-computation($\mu$,$\pi$);

**begin**

**if** $\bar{m} + \beta \leq \underline{n} - \gamma$ or $\bar{n} + \delta \leq \underline{m} - \alpha$ **then** $\Pi = 0$;

**else if** $\underline{m} \leq \underline{n}$ ;

**if** $\bar{m} \geq \underline{n}$ **then** $\Pi = 1$

**else** find $c$ such that $R_m(\frac{c-\bar{m}}{\beta}) = L_n(\frac{\underline{n}-c}{\gamma})[= \Pi]$

**else**

**if** $\underline{m} \leq \bar{n}$ **then** $\Pi = 1$

**else** find $c$ such that $R_m(\frac{c-\bar{n}}{\delta}) = L_n(\frac{\underline{m}-c}{\alpha})[= \Pi]$

**end**

Figure 5.4: Possibility computation

Similarly, we discuss three cases to compute the necessity (refer to Figure 4.4)

$$N_i = \inf_{x \in D} \max(\mu_i(x), \pi_i(x))$$

- $N = 1$ if the support of $\pi$ ($[\underline{n} - \gamma, \bar{n} + \delta]$) is included in the core of $\mu$ ( $[\underline{m}, \bar{m}]$). In this case, all the possible values of publication satisfy the predicate with degree 1.

- $N = 0$ if the core of $\pi$ ($[\underline{n}, \bar{n}]$) is intersected with the supplementary domain of $\mu([\underline{m} - \alpha, \bar{m} + \beta])$. In other words, if $\bar{m} + \beta < \bar{n}$ or $\underline{m} - \alpha > \underline{n}$, the necessity is 0.

- Otherwise, it is the intersection between the left-hand arms(or right-hand arms) of $\mu$ and $\pi$.

Figure 5.5 depicts the detailed algorithm of the necessity computation (i.e., the necessity that the publication (i.e., the attribute) satisfies the predicate).

**Procedure** Necessity-computation($\mu$,$\pi$);

**begin**

**if** $\underline{m} - \alpha \geq \underline{n}$ and $\bar{m} + \beta \leq \bar{n}$ **then** $N = 1$;

**else if** $\bar{m} + \beta < \bar{n}$ or $\underline{m} - \alpha > \underline{n}$ **then** $N = 0$;

**else**

find $c_1$ and $c_2$ such that

$R_m(\frac{c_1 - \bar{m}}{\beta}) = 1 - R_n(\frac{c_1 - \bar{n}}{\delta})[= N_1]$

$L_m(\frac{\underline{m} - c_2}{\alpha}) = 1 - L_n(\frac{\underline{n} - c_2}{\gamma})[= N_2]$

$N = \min(N_1, N_2)$

**end**

Figure 5.5: Necessity computation

### 5.2.2 Subscription evaluation

Subscriptions may be conjuncts of predicates, disjuncts of predicates, or normal forms. We use a t-mapping function or s-norm function defined in chapter 3 to model the operations of these relations. 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 function (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.

## 5.3 Sequencing Algorithm

The challenge of large scale publish/subscribe system is to effectively process millions of subscriptions with high publishing rate. The previous algorithm evaluated all predicates

**input:** $SatPreds$: output of the predicate matching

**global variables:**

$Vp$: a predicate vector;

$Vs$: the subscription vector;

$P\_S\_tbl$: a predicate subscription association table

$SatS$: the set of subscriptions matches the event

**Body**

1. $Vs = 0$, $SatS = \emptyset$, $Count = 0$
2. **for** each $p \in SatPreds$
   **for** each $s$
   **if** $P\_S\_tbl[p, s] = 1$ **then**
   **if** $Count[s] = 0$ **then**
   $Vs[s].\Pi = Vp[p].\Pi$
   $Vs[s].N = Vp[p].N$
   **else** $Vs[s].\Pi = \min(Vs[s].\Pi, Vp[p].\Pi)$
   $Vs[s].N = \min(Vs[s].N, Vp[p].N)$
   $Count[s] + +$
3. **for** each $s$
   **if** $Count[s] = preds\_per\_sub[s]$
   **then** $SatS = SatS \cup \{s\}$
4. return $SatS$

Figure 5.6: Subscription Evaluation

that were referenced by a given publication (i.e., iterated over each of its attributes.) More specifically, at least one comparison between the two functions $\mu$ and $\pi$ was required for each predicate to determine whether a match occurred. To minimize the number of comparisons, we improve our algorithm by sequencing predicates of the same attribute so that the predicate matching algorithm can stop earlier rather than evaluate all predicates.

For each attribute, the sequence of predicates depends on the sequence of their functions $\mu$. In the representation of function $\pi$, let $n_1 = \underline{n} - \gamma$, $n_2 = \underline{n}$, $n_3 = \bar{n}$, $n_4 = \bar{n} + \delta$. They are four critical points because they differentiate the boundaries where has value 0 and where has value 1. Obviously, we have $n_1 \leq n_2 \leq n_3 \leq n_4$. Similarly, for function $\mu$, let $m_1 = \underline{m} - \alpha$, $m_2 = \underline{m}$, $m_3 = \bar{m}$, $m_4 = \bar{m} + \beta$, and we have $m_1 \leq m_2 \leq m_3 \leq m_4$.

The predicate won't match the publication if its right-hand spread is to the left of the attribute function $\pi$, i.e., $m_4 \leq n_1$. A match is established once the predicate "touches" the publication, i.e., $\mu$ and $\pi$ intersect. If the predicate's left-hand spread is to the right of $\pi$, i.e., $m_1 \geq n_4$, it will no longer match. Based on this observation, predicates, in the same predicate-table, are organized in the order of their $\mu$ 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 about the same attribute index. We first compare $m_{i_1}$ and $m_{j_1}$. The predicate with 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 $m_2$s are equal then the same comparison is done for $m_3$, $m_4$.

For each attribute $a_i$ of a publication, we only evaluate those predicates that intersect with the attribute. Predicate matching stops as soon as the above rules establish further none-matches.

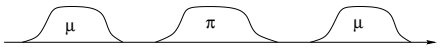


Figure 5.7: The case of no match between $\pi$ and $\mu$

In the possibility computation, let the algorithm encounter $m_4$ (the last point of function $\mu$) through those predicates until it is larger then $n_1$ (the first point of $\pi$). Before that, the predicates are all on the left of the publication pair (as the left case in Figure 5.7), hence impossible to be matched. Now $m_4 > n_1$ then we check $m_1$. If $m_1 > n_4$, then we can stop because from now on all predicates afterwards are on the right of the publication pair, thus impossible to be matched either (as the right case in Figure 5.7). We just need to evaluate the predicates whose $m_4 < n_1$ and $m_1 > n_4$. Figure 5.8 shows the detailed possibility computation.

**Procedure** Improved-Possibility-Computation;
**begin**

1. $j = 1$
2. **while** $m_{1j} < n_4$ **do**

**begin**

**while** $m_4 j \leq n_1$ **do** j++

**if** $m_{3j} \leq n_2$ **then** find $c$ such that

$R_m(\frac{c-\bar{m}}{\beta}) = L_n(\frac{\underline{n}-c}{\gamma})[= \Pi]$

**else if** $m_{2j} \leq n_3$ **then**$\Pi = 1$

**else** find $c$ such that

$R_m(\frac{c-\bar{n}}{\delta}) = L_n(\frac{\underline{m}-c}{\alpha})[= \Pi]$

j++,

**end**

**end**

Figure 5.8: Improved Possibility Computation

In necessity evaluation, let the the algorithm encounter $m_4$ (the last point of function $\mu$) through those predicates until it is larger then $n_3$. Before that, the complements of predicate functions $\mu$ are always intersected with the core of the $\pi$, so the necessity

must be 0. After $m_4 > n_3$ we compute the necessity of each predicate until $m_1 \leq n_2$ because from now on all necessities afterwards must be 0. Figure 5.9 shows the detailed algorithm.

**Procedure** Improved-Necessity-Computation;

**begin**

1. $j = 1$,
2. **while** $m_{1j} < n_2$ **do**

**begin**

**while** $m_{4j} \leq n_3$ **do** j++

**if** $m_{2j} < n_1$ and $m_{3j} > n_4$ **then** N=1

**else** find $c_1$ and $c_2$ such that

$R_m(\frac{c_1-\bar{m}}{\beta}) = 1 - R_n(\frac{c_1-\bar{n}}{\delta})[= N_1]$

$L_m(\frac{\underline{m}-c_2}{\alpha}) = 1 - L_n(\frac{\underline{n}-c_2}{\gamma})[= N_2]$

$N = \min(N_1, N_2)$

$j++$

**end**

**end**

Figure 5.9: Improved Necessity Computation

## 5.4 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 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.

## 5.5 Time and Space Analysis

We will analyze the performance properties of this algorithm in terms of space cost and matching time.

**Space cost:** The space cost includes mainly the following parts: predicate hash table, predicate vector, subscription vector, and the predicate subscription association (bit) matrix.

$$Space = \sum(Space_p * N_p) + 2N_p + 2N_s + N_p * N_s$$

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. Overall, the space cost is dominated by the predicate subscription association matrix: $Space = O(N_p * N_s)$.

**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

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

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. $N_{p_e}$ 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:

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

In the subscription evaluation, we suppose the time for one look up 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 $t_2 * N_s * N_{p_{sat}}$ where $N_{p_{sat}}$ is the average number of matched predicates. Considering the use predicate $\alpha$-cut into the subscription evaluation, which means we only select those predicates whose matched degrees are larger than their $\alpha$-cuts to find out the associated subscriptions, the time decreases much. We denote the decreasing coefficient is $\beta$, the subscription evaluation time is

$$Time(\text{subscription evaluation}) = \beta * t_2 * N_s * N_{p_{sat}}.$$

In all, the matching time cost of our algorithm is

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

# Chapter 6

# Experiments

In this chapter, we discuss the experimental performance of the algorithms presented in Chapter 5. 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. Experiments are processed under various subscription and publication workloads.

## 6.1 Experimental Framework

We ran all the experiments on a dual-CPU Pentium III workstation with 900MHz i686 CPUs and 1.5 GB RAM, 2G swap operating under Linux (RedHat 7.2). The publish/subscribe system was run as a process on this workstation waiting for subscriptions and publications to be processed. 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 generator is powerful. It 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 on the basis of approximate ones.

For subscriptions, we define three types of crisp predicates derived from the approximate one: *optimistic, pessimistic* and *middle*. Now we explain the meaning of these 3 choices in detail using a concrete example. Considering an approximate predicate of price=[20,25,35,40], these four numbers are $m_1, m_2, m_3, m_4$, respectively, as defined in Section 5.3. We want to generate a crisp predicate to be compared with this approximate one. We use an interval to represent the crisp constraint of price, i.e. $l \leq price \leq u$. *Optimistic, pessimistic* and *middle* are three ways to determine the lower bound and upper bound $[l, u]$. *Optimistic* type takes the core of the fuzzy set $[25, 35]$ as the interval domain. For *pessimistic* choice, the support of the fuzzy set $[20, 40]$ is taken as the interval domain. *Middle* method is to compromise the first two and take the middle value $[22, 32]$ as the interval domain. For the definition for all kinds of predicates, please refer to Table 6.1.

| Sub Type | approximate | pessimistic | middle | optimistic |
|---|---|---|---|---|
| Attribute constraint | $[\underline{s}, \underline{m}, \bar{m}, \bar{s}]$ | $[\underline{s}, \bar{s}]$ | $[\frac{\underline{s}+\underline{m}}{2}, \frac{\bar{m}+\bar{s}}{2}]$ | $[\underline{m}, \bar{m}]$ |

Table 6.1: The Definitions for different types of subscriptions

Publications are generated similarly. We have two choices to generate crisp publications: *point* and *interval*. They are 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. which takes the core of fuzzy set as the crisp interval bound and *point* type where the medium point in the core of the fuzzy set is taken as the point value for event attribute, which is the traditional case in previous publish/subscribe system. The point type is Since we can compare the effects of the size of the interval domain on the number of matched results from the respect of subscriptions, we only define two types for publications. Table 6.2

defines the differences between these two types of crisp publications and the approximate publications.

| Pub Type | approximate | crisp | point |
|---|---|---|---|
| Attribute constraint | $[\underline{s}, \underline{m}, \bar{m}, \bar{s}]$ | $[\underline{m}, \bar{m}]$ | $\frac{\underline{m}+\bar{m}}{2}$ |

Table 6.2: The definitions for different types of publications

Subscription is a list of predicates. Predicates are determined by an attribute name, a lingual fuzzy concept value and a function quantitatively expressing the meaning of this fuzzy value. Predicate attribute names are drawn from the predefined set of names. The same set of attribute names is used to draw attribute names for publications. The total number of names available is determined by $n_t$. For each attribute name, we provided a set of fuzzy concepts (the number of those concepts is $n_v$) to be drawn as predicate's lingual fuzzy value and a domain restricted by a lower bound and upper bound, $l_P$ and $u_P$, respectively, on which the possible functions to represent the fuzzy concept are generated. In this experiment, we only use a trapezoidal form function, hence 4 points are selected from the domain, governed by a uniform distribution, to form the representation function.

Publication is generated similarly. To make it possible that there would be some subscriptions matched publication, we use the same set of attribute names, fuzzy values and fuzzy value domains to generate publications.

We ran several experiments multiple times and did not notice a significant difference in the results. Therefore, we do not report variances in our figure, which were lower than 0.1%, for the repeated experimental runs.

## 6.2 Experimental Results

### 6.2.1 Performance Evaluation

In a series of experiments the publish/subscribe system is subjected to a large number of subscriptions and high publishing rate. These are the basic assumptions upon which we designed the matching algorithm. To evaluate performance metrics, we consider four versions of our algorithms:

1. the regular algorithm using float (4 bytes) to store all the possibilities and necessities of predicates and subscriptions;
2. the regular algorithm using char (1 byte) with 10 values to store the possibilities and necessities;
3. regular algorithm using char but with 256 values;
4. the improved sequencing algorithm.

These versions of algorithms are implemented in our publish/subscribe engine prototype. To compare performance of different versions of algorithms, we consider the following metrics: subscriptions loading time, overall system throughput and memory used. Loading time measures the time it takes for workload generator to submit all subscriptions to the system. Overall system throughput measures the matching time for event to match against all predicates and subscriptions stored in the system. The resident memory size of the publish/subscribe engine process is measured. Time measurements are taken in milliseconds and memory in KB. The matching time measurement start just before the publication has been submitted to the system process and end right after the system responds. In case that system responds to the publication submission with the notification containing the IDs of matched subscriptions, the time measurements therefore include the interprocess communication time and individual timings account for the processing of an entire batch of subscriptions or publications submitted.

Figure 6.1 compares the matching time across all algorithms. In this experiment we use the following workload specification: $W0 = (n_t = 42, n_P = 2, n_A = 4, n_E = 10, n_v = [2..5], n_S = [100, 100000])$. No doubt, the matching time depends on the size of hash table for each attribute and the number of subscriptions that contains those matched predicates, hence matching time increases with the increasing the number of subscriptions $n_S$.

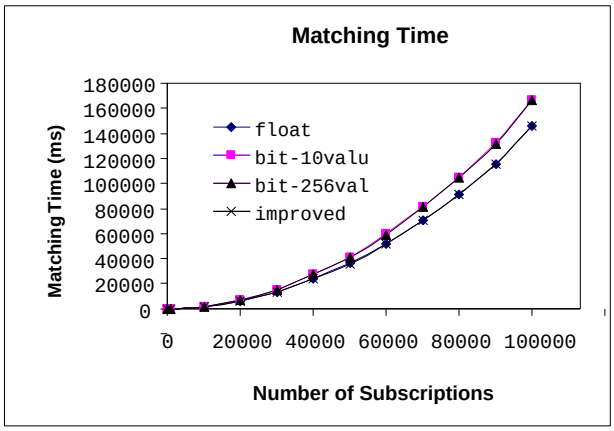


Figure 6.1: Matching process time

Because the predicates for each subscription are generated so randomly, the probability of overlap between predicates is very small. In our workload, most domains to determine the function for fuzzy value have a span of 100 and each function is determined by 4 points, so the probability of overlap between two predicates is $p_{overlap} = \frac{1}{n_t * n_v * 100^4} \simeq 10^{-10}$, which is very small. Hence the number of distinct predicates increases almost proportionally to the number of subscriptions. In the subscription matching time comparison, there is not much difference between the regular approximate matching algorithm and the improved sequencing algorithm. And the bits-wise algorithms run more slowly since it needs more computation to set the bit value. There is not much difference between the bits-256 and bits-10 algorithms is because they use the same strategy almost everywhere during the matching. To show the progress of improved predicate matching algorithm, we ran the predicate matching process only. From the Figure 6.2, we can see

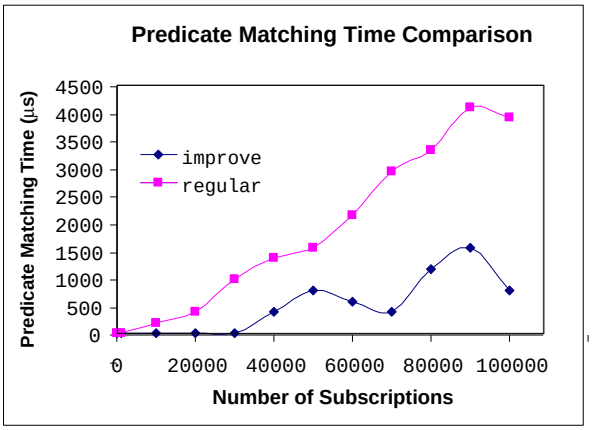


Figure 6.2: Predicate Matching Time Comparison

that the improved sequencing algorithm runs much faster.

Figure 6.3 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 we need to insert the predicate according to an sorted order based on the 4 points of function. This is a tradeoff between the loading time and matching time. In the real application, most subscriptions stayed in the system for a long time, so 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.

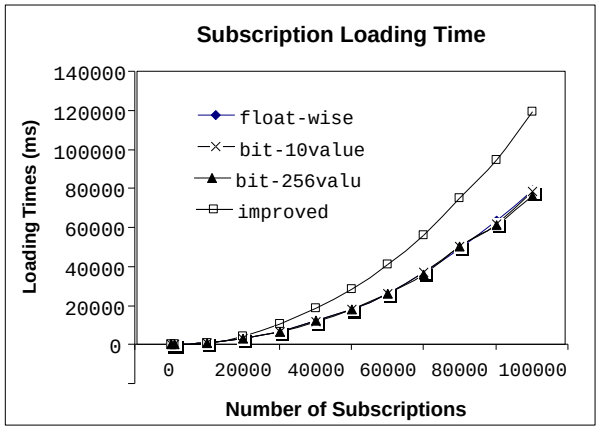


Table 6.3: Loading Time comparison

Memory comparison show memory utilization for float-wise and bits-wise algorithms. Since the subscription-predicate association table takes up the majority space, we only consider the space used to store the matched result of predicates and subscriptions. From Figure 6.4 we can see that bits-wise algorithms use less than the float version due to the space saved by using 1 byte char in stead of 4 bytes float.

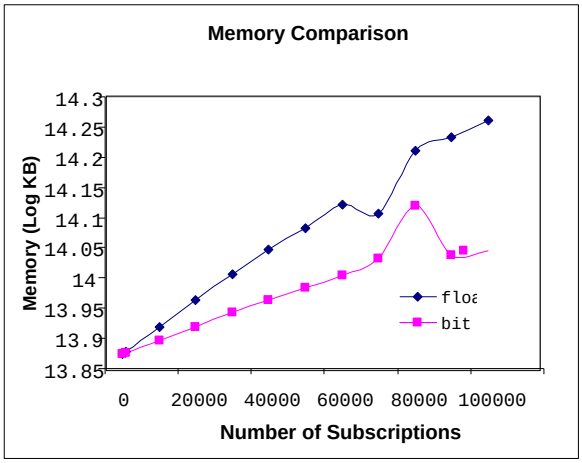


Table 6.4: Memory Cost Comparison

### 6.2.2 Comparison between crisp model and fuzzy model

As we mentioned before, there are several properties unique to our publish/subscribe model with uncertainties such as the expression of predicates, truth value and possibilities etc. In this section, we compare the characteristics between the crisp language model and our approximate model. We split the four cases of table 1 into two scenarios.

Table 6.5 shows the different numbers of matched subscriptions when a fixed event is matched against various types of subscriptions with different $\alpha$-cut. We can see that for one type of subscription, the number decreases with the increasing of $\alpha$-cut which indicates the threshold effect of $\alpha$. With the same $\alpha$, the pessimistic case results the largest number of matches, and the optimistic case results fewest matches. The approximate case and the middle case have not much difference because the wider restriction of

| Subscription Type | $\alpha = 0$ | $\alpha = 0.5$ | $\alpha = 1$ |
|---|---|---|---|
| fuzzy | 4628 | 184 | 7 |
| pessi | 4628 | 804 | 281 |
| middle | 4438 | 184 | 39 |
| optim | 3763 | 47 | 7 |

Table 6.5: Comparison of number of matched subscriptions for various types of subscriptions, event type is fuzzy, $n_S = 70,000$, $n_E = 10$

subscription, the more probability being matched.

| Publication Type | $\alpha = 0$ | $\alpha = 0.5$ | $\alpha = 1$ |
|---|---|---|---|
| fuzzy | 4628 | 184 | 7 |
| crisp | 3720 | 474 | 170 |
| point | 2960 | 1932 | 868 |

Table 6.6: Comparison of Number of matched for various types of publications, subscription type is fuzzy, $n_S = 70,000$, $n_E = 10$

Table 6.6 shows the numbers of matched subscriptions for different types of events when subscription type is fixed. When $\alpha = 0$, the fuzzy publication returns the most number and point type returns the least. This is because fuzzy event has wider domain of values, thus easier to match subscription's restriction. However, with the increase of $\alpha$, the fuzzy event matched very small number of subscriptions, point-value event matched the most in reverse. $\alpha$ is used as the threshold for both possibilities and necessities. This reversal phenomenon is reasonable if the intuitive meaning of possibility and necessity we defined in the model section is considered. 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,

more difficult to be reached above the $\alpha$. For the point-value event, it is easy for such a value to be located in the core of the subscription function, thus more are matched with high $\alpha$.

# Chapter 7

# A-ToPSS System Implementation

The main challenge in applying publish/subscribe systems to selective information dissemination lies in the design of good architecture and efficient algorithm that exhibits good scalability. At Internet-scale, such a system has to be able to process millions of subscriptions and react to thousands of events. The overall architecture of system is shown in Figure 7.1.

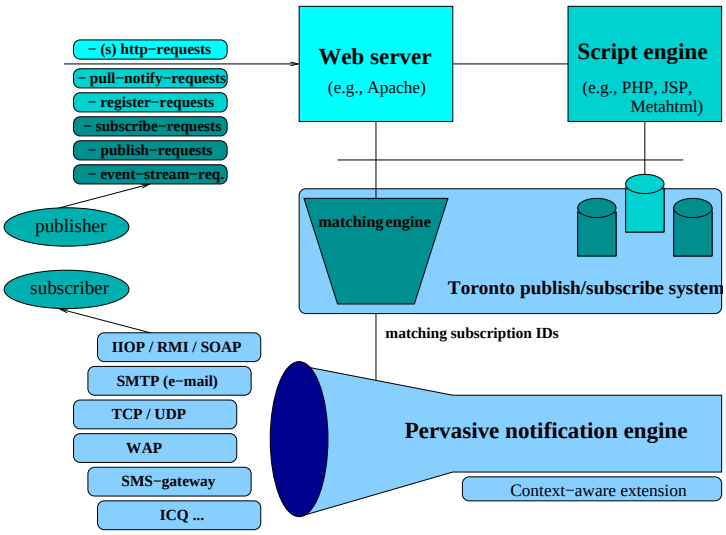


Figure 7.1: Overall Architecture of Publish/Subscribe System

The introduction of approximate matching increases the number of possible matches for a subscription, since more events are matched, but to different degrees. It is necessary

to better understand how this notion of approximate matching can help to better satisfy user's queries for information. This can only be done in an experimental manner. We have therefore built a demonstration matching engine A–TOPSS that can manage more than ten million subscriptions and process many events per second. A–TOPSS is the *Approximate Matching-based Toronto Publish/Subscribe System* research prototype that implements approximate matching scheme and serves us as experimentation platform to evaluate different subscription and publication language models and approximate matching algorithms. We have integrated this with a web-server and an application server to make the service available online. In our software demonstration we aim to exhibit the combination of all four cases from Table 1.1. Our prototype performs the matching correctly no matter whether publications or subscriptions are represented in a crisp or approximate form.

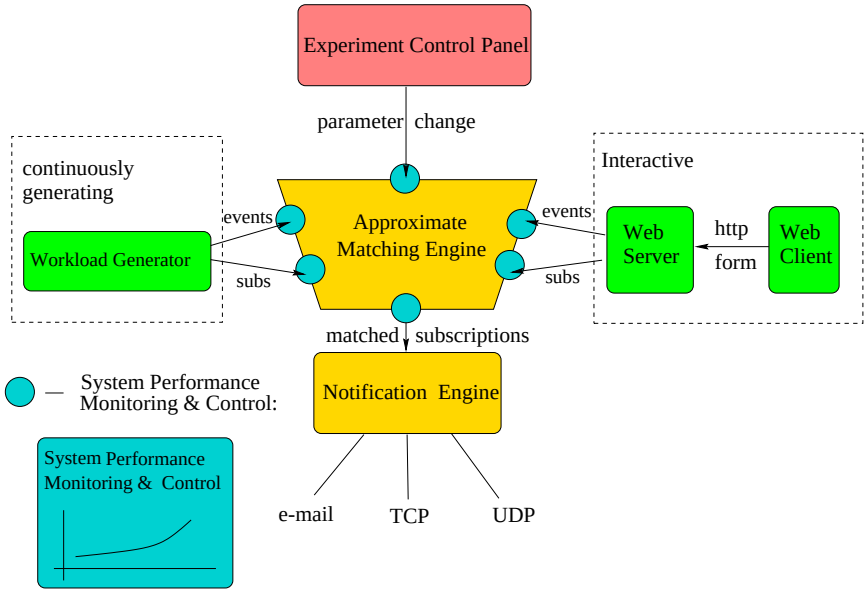


Figure 7.2: Demonstration Setup of Approximate Toronto Publish/Subscribe System (A-ToPSS).

We will demonstrate two scenarios. First, normal system operation, which constitutes an example of the apartment rental market application we have discussed and, second,

the experimental system operation to investigate the effects of different parameters to control our model. The demonstration system setup for both scenarios is depicted in Figure 7.2. All components are fully implemented.

## 7.1 Normal system operation: Toronto apartment rental market

This scenario shows the deployment of A–TOPSS as an information dissemination service for an apartment rental market web server. In the demonstration we offer features for a user to input subscriptions over the web and receive notifications as e-mail messages. In the background we simulate other users and content providers (i.e., landlords or other agencies offering apartments, for instance) through a workload generation process. Our workload generator is script-driven, it can continuously generate new subscriptions and publications at random according to the specification file. This scenario is meant to validate the real world applicability of the approximate matching model we are proposing.

The interactive user interface uses Meta-HTML web programming language. Meta-HTML is a powerful, extensible server-side programming language specifically designed for working on the World Wide Web. It resembles a hybrid of HTML and Lisp and has a huge existing function library, including support for sockets, image creation, perl, GNU plot, etc. It is extensible in both Meta-HTML and other languages (C, etc.).

A–TOPSS is a secure and powerful system. We will explain the operations of subscriptions as an example. There are two levels to define a subscription. The administrative level is "subscription type" which defines the number of predicates and the type for each predicate value – crisp or fuzzy. Subscription type is the template for user to insert subscriptions. Only system administrator or power user with high priority can operate on subscription type: create new types, edit the existing ones or delete them. Before modifying or deleting the current type, system will check whether there is any subscrip-

tion defined under this type. Subscription type only can be edited when no subscription is defined under it. The user level of operations on subscriptions are designed for the ordinary users. Subscribers can add a new subscription, edit or delete those what they defined before. When adding a new subscription, users first choose the type, then system will ask users to input the corresponding information according to the requirements specified by the subscription type. For crisp subscriptions, people need to provide attribute names, operators ($>, <, =, \neq$ etc.) and values (integer, string etc.). It is more complicated to add a new fuzzy subscriptions. Other than attribute names, user need to define how many fuzzy values each attribute contains. For example, the "price" attribute may have 3 fuzzy values including "expensive", "reasonable" and "cheap". For the representation of each fuzzy value, the demonstration will show the flexibility for users to describe predicates. A user chooses among a family of functions to represent uncertain information. Figure 7.3 shows a screen shot of the subscription entry panel of our system, where a user can view and adapt the membership functions representing her subscriptions. Operations on publications are similar, we won't elaborate here.

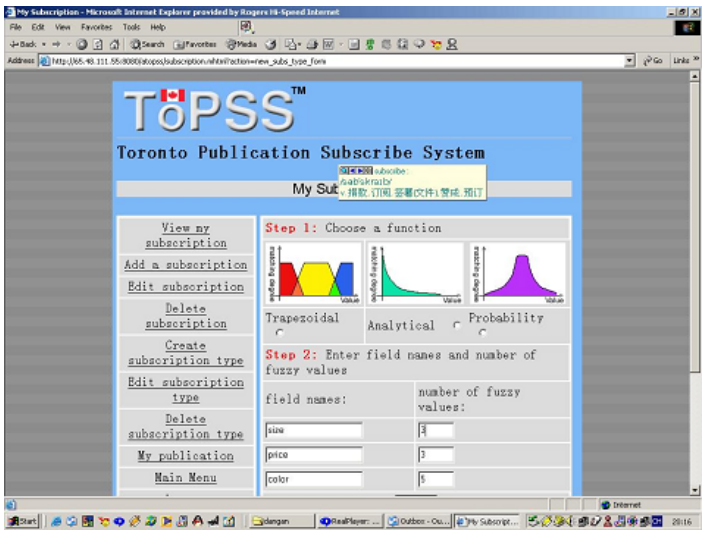


Figure 7.3: The power user's interface for defining approximate subscriptions.

In the background, a workload generator is running continuously to simulate other

information providers in the real world. This workload generator is implemented by C language. Now we have two sources of information. Subscriptions and publications from web users and the workload generator will both enter the matching engine being processed there. We use socket to communicate between our matching engine and the web server. There are two data buffers between them to accept the subscriptions and publications coming from web server. The main process running background has two threads to read the buffer at a certain rate and take out the data if it is not empty. In the real world, every subscription and publication have a period of validity or time stamp, thus we need a thread to control the deletion of subscriptions and publications. The detail implementation design is shown in Figure 7.4.

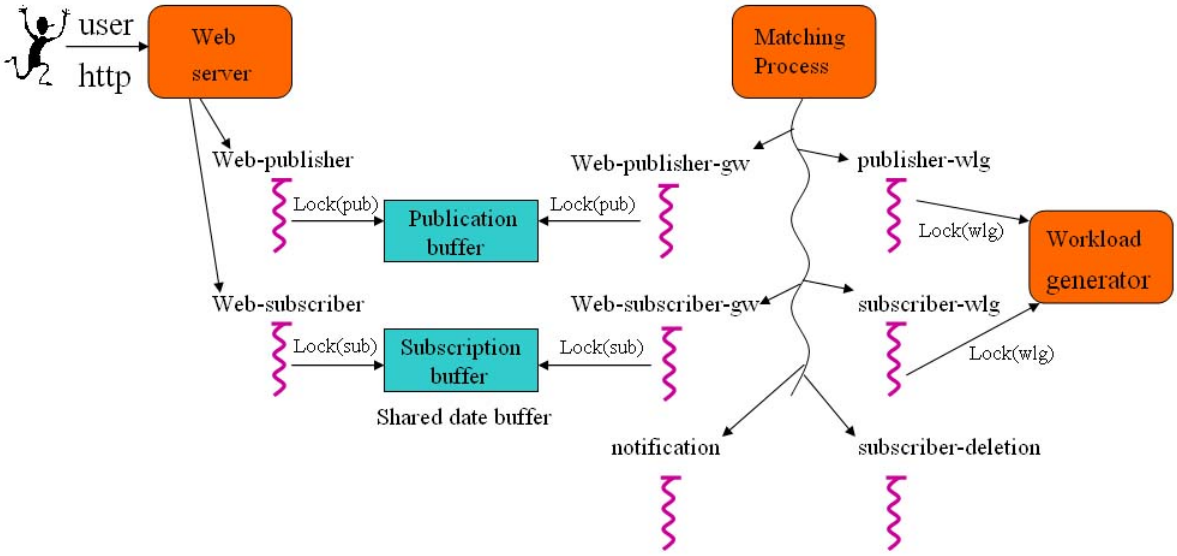


Figure 7.4: A–TOPSS implementation design

## 7.2 Experimental system operation

Other than in the crisp-model, currently favored by all existing P/S systems, the representation of approximate subscriptions involves many parameter choices. For instance, membership functions can be parameterized or substituted by different functions. The

degree of confidence beyond which subscriptions should no longer be counted as matches is another parameter which may influence system operation. There are a number of interesting effects to test experimentally.

1. What function representation should be chosen for approximate subscriptions and publications?

2. What kind of aggregation function should be chosen to combine the degrees of match of the individual predicates?

3. How should the degree of confidence which models users' tolerance in receiving partially matched results be set?

To demonstrate the effects of these parameters on the system operation, we deploy an experiment control panel (java applet) that can change these parameters, and a monitoring and control panel (java applet) that observes and displays system metrics (i.e., throughput, number of subscriptions and publications in system queues) These metrics are taken at monitoring and control points indicated in the Figure 7.2. This scenario aims at experimenting with the approximate matching model to demonstrate and explore its degrees of freedom.

All these parameters influence overall system performance. On the other hand, it is important to understand the effects these parameters have on the matching results obtained. It is also important to be able to demonstrate an increase in user satisfaction, which is difficult to quantify numerically.

# Chapter 8

# Conclusion and Future Work

## 8.1 Conclusion

In this paper we propose a model to express the uncertainty in both subscriptions and publications when the exact information is not available. Fuzzy set is used to approximate vague notions in predicate and publication. Based on our modal, we define a matching mechanism between publications and subscriptions. Moreover, an extended counting algorithm is designed to process the approximate matching problem. There are several nice properties in our model:

1) The language schema is flexible and powerful in that it allows subscriptions and publications to be either crisp or approximate.

2) The possibility and necessity measure is reasonable and expressive, these two optimistic and pessimistic matching degree could satisfy different users.

3) Our algorithm can be used to process approximate matching for millions of subscriptions and high event publishing rate.

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 since we use the two-step based algorithm to apply the aggregation function after the predicate matching phase to pick out the matched subscriptions.

5) The algorithm is designed with respect to conjunctions of predicates, but it also can be extended for disjunctions as long as we substitute the *min* operator by other boolean combination functions.

## 8.2 Future Work

In this thesis we designed an approximate matching algorithm to process information with uncertainties. The question it leaves for us is how to improve the algorithm efficiency. Can we pick out the matched predicates and subscriptions more quickly based on the properties of membership functions used to describe the vagueness? Are the users satisfactory degrees useful to trim out unnecessary data?

This thesis targets at the model and algorithm for publish/subscribe system supporting uncertainties. In terms of the implementation, the matching engine is memory-based where all the data information is stored into the memory in a specific structure to be processed. In order to apply this model into real applications, it needs to integrate this model and matching semantic into a database system. Also, a scalable algorithm for distributed brokers to route approximate information is desirable in distributed systems, other than the centralized architecture.

# Bibliography


[1] Distributed messaging systems. In *Dependable Systems and Networks*. www.research.ibm.com/gryphon.

[2] Tibco/rendezvous concepts. In *TIBCO Inc.*, October 2000.

[3] Object management group. common object request broker architecture, version 3.0. In *Object Mangement Group*, November 2002.

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

[5] A. Carzaniga, D.S. Rosenblum, and A.L. 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.

[6] A. Compailla, S. Chaki, S. Jha, and H. Veith. Efficient filtering in publish-subscribe system using binary decision diagrams. In *23rd International Conference on Software Engineering(ICSE)*, 2001.

[7] Didier Dubois and Henri Prade. *Possibility Theory: An Approach to Computerized Processing of Uncertainty*. Plenum Press, New York, 1988.

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

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

[10] R. Fagin, Lotem A., and Naor M. Optimal aggregation algorithms for middleware. In *Proc. Twentieth ACM Symposium on Principles of Database Systems*, pages 102–113, 2001.

[11] M. Happner, R. Burridge, and R. Sharma. Jave message service. October 1998.

[12] George J. Klir and Tina A. Folger. *Fuzzy Sets, Uncertainty, and Information*. Prentice Hall International Editions, 1992.

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

[14] J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Conference on Cooperative Information Systems*, 2000.

[15] The READY Project Group AT&T Labs Research. Ready a high performance event notification service. http://www.research.att.com/ ready.

[16] 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.

[17] A. Wolski and T. Bouaziz. Fuzzy triggers: Incorporating imprecise reasoning into active database. In *Proceedings of the 14th International Conference on Data Engineering*, 1998.

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

[19] L.A. Zadeh. Fuzzy logic. In *Multiple-Valued Logic of the Computer Magazine*.