Textual Information Dissemination in Distributed Event-Based Systems*

Manolis Koubarakis
Dept. of Electronic and Computer Engineering
Technical University of Crete
Chania, Crete, Greece
manolis@ced.tuc.gr
www.ced.tuc.gr/~manolis

Abstract

We define formally the data models $\mathcal{WP}$ and $\mathcal{AWP}$ especially designed for the dissemination of textual information in distributed event-based systems. We also define the problems of satisfiability, satisfaction, filtering and entailment, and point out that these problems are fundamental for the deployment of models like the ones presented here in distributed event-based systems.

1 Introduction

Today's wide-area networks are characterized by the asynchrony, heterogeneity and loose coupling of components thus making *distributed event dissemination systems* an important building block for many applications [6]. As a result, event dissemination systems have recently captured the attention of various research communities including researchers from databases [2, 18, 12], agent systems [11], digital libraries [13], distributed computing [5, 4] and others.

As an example of an application scenario, let us consider a university with 4 geographically distributed campuses (Arts, Sciences, Engineering and Medicine) and a local digital library in each campus. Each campus library has a *middle-agent* [11] acting as an intermediary to its local collection of digital copies of articles appearing in various journals in the research areas of the local faculty. Each middle-agent might in fact integrate information from various sources e.g., the middle-agent from the Engineering library might integrate information about documents *published* in the digital libraries of IEEE and ACM using *wrappers* to transform the information into a common global schema. Users *subscribe* to this information by posting *profiles* (or *long-standing queries*) to middle-agents at their local libraries to receive information about interesting documents.[1] It is the job of the middle-agent to match efficiently incoming *events* (corresponding to newly-published documents) with existing profiles and inform interested subscribers.

In a distributed scenario such as this one, a middle-agent can also forward profiles to other middle-agents in the same network, so that relevant documents are pushed to users of the library system, independently of which middle-agent these users are connected to. For example, a user at the Engineering library might want to receive information about new articles if their titles contain the phrase "query optimisation". The middle-agent of the Engineering library can forward this profile to the middle-agent of the Sciences library, just in case documents satisfying this profile are published to this middle-agent. In a more advanced system, middle-agents could also *advertise* the contents of their digital libraries to other middle agents so that only profiles matching their advertisements are forwarded to them [15, 5].

The above distributed event dissemination scenario appears in many application areas beyond digital libraries (e.g., financial trading or e-commerce where events and profiles might refer to prices, news dissemination where events and profiles might refer to textual information and so on). In this paper we are especially concerned with the distributed dissemination of events consisting of *textual information*.

The contributions of this paper are the following. We first present some requirements for the models/languages used to specify events in a distributed event-based system. This "wish-list" has motivated most of our recent work in this area. Then we develop formally the data models $\mathcal{WP}$ and $\mathcal{AWP}$ and their query languages for textual information dissemination in distributed event-based systems. Data

*To be presented at the International Workshop on Distributed Event-Based systems (DEBS'02), July 2-3, 2002, Vienna, Austria.

[1] The terms query and profile will be used interchangeably in this paper. We are in a dissemination setting where queries are long-standing ones.

model $\mathcal{WP}$ is based on free text and its query language is based on the *boolean model with proximity operators* [3, 7, 8]. Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values. Its query language is an extension of the query language of data model $\mathcal{WP}$.

In work in progress, we built on the present formal foundation and study the computational complexity of the problems of matching, filtering, satisfiability and entailment in models $\mathcal{WP}$ and $\mathcal{AWP}$.

The rest of the paper is organised as follows. Section 2 presents our requirements on models and languages for distributed event dissemination systems. Section 3 presents data model $\mathcal{WP}$ based on free text and its sophisticated query language. Section 4 presents the semantics of model $\mathcal{WP}$. Then Section 5 builds on this foundation and develops the same machinery for data model $\mathcal{AWP}$. Finally, Section 6 gives our conclusions.

2 Models and Languages for Distributed Event Dissemination

Before we embark on the technical contributions of this paper, we would like to present a number of requirements for models and languages in distributed event dissemination systems. Our requirements have been motivated by a desire to develop useful event-based systems but in a *principled* and *formal* way:

1. *Expressivity.* The language for events and profiles must be rich enough to satisfy the demands of many applications.

2. *Formality.* The syntax and semantics of the language must be defined formally.

3. *Computational efficiency*. The following problems should be defined formally and algorithms must be provided for their efficient solution:

   (a) The *satisfiability problem*: Deciding whether a profile can be satisfied by any event at all.

   (b) The *satisfaction* (or *matching*) *problem*: Deciding whether an event satisfies (or matches) a profile.

   (c) The *filtering problem*: Given a database of profiles db and an event e, find all profiles $q \in db$ that match e.

   (d) The *entailment* or *subsumption problem*: Deciding whether a profile is more or less "general" than another.

Previous research has shown that the above formal perspective is shared by other researchers in this area. Virtually all previous papers presenting sophisticated filtering algorithms have defined the syntax and semantics of their languages carefully [2, 18, 12]. The developers of SIENA in particular have carefully defined the syntax and semantics of their language for events and profiles, and have formalised the notions of satisfaction and entailment (called "covers" in [5]). The same has been done for a much more expressive language in [4] where only satisfaction and filtering have been considered. To the best of our knowledge, no work has been done so far on addressing the satisfiability problem.

In the rest of this paper, we concentrate on dissemination of events consisting of textual information and define the models $\mathcal{WP}$ and $\mathcal{AWP}$ that are suitable for this job.

3 Text Values and Word Patterns

In this section we present the data model $\mathcal{WP}$ and its query language. $\mathcal{WP}$ assumes that textual information is in the form of *free text* and can be queried by *word patterns* (hence the acronym for the model). The basic concepts of $\mathcal{WP}$ are subsequently used in Section 5 to define the data model $\mathcal{AWP}$ and its query language.

We assume the existence of a finite *alphabet* $\mathbf{\Sigma}$. A *word* is a finite non-empty sequence of letters from $\mathbf{\Sigma}$. We also assume the existence of an infinite set of words called the *vocabulary* and denoted by $\mathcal{V}$.

Definition 1 *A* text value s *of length* n *over vocabulary* $\mathcal{V}$ *is a total function* $s : \{1, 2, \ldots, n\} \to \mathcal{V}$.

In other words, a text value s is a finite sequence of words from the assumed vocabulary and $s(i)$ gives the i-th element of s. Text values can be used to represent finite-length strings consisting of words separated by blanks. The length of a text value s (i.e., its number of words) will be denoted by $|s|$.

Example 1 *In all the examples of this paper, our vocabulary will be the vocabulary of the English language and will be denoted by* $\mathcal{E}$*. The string*

$$my\ holiday\ in\ Milos$$

can be represented by a text value s *of length 4 over vocabulary* $\mathcal{E}$ *with* $s(1) = my,\ s(2) = holiday$ *etc. The text value "*$in\ Milos$*" is included in* s*.*

We now give the definition of word-pattern. The definition is given recursively in three stages.

Definition 2 *Let* $\mathcal{V}$ *be a vocabulary. A* proximity-free word pattern *over vocabulary* $\mathcal{V}$ *is an expression generated by the grammar*

$$WP \to \mathbf{w} \mid \neg WP \mid WP \wedge WP \mid WP \vee WP \mid (WP)$$

where terminal **w** *represents a word of* $\mathcal{V}$*. A proximity-free word pattern will be called* positive *if it does not contain the negation operator.*

Example 2 *The following are proximity-free word patterns that might appear in queries of a user of a news dissemination system interested in articles on holidays:*

$$holiday \wedge hotel \wedge beach,$$

$$Athens \wedge hotel \wedge \neg Hilton, \;\; holiday \wedge (beach \vee mountains)$$

Word patterns made of words and the Boolean operators $\wedge, \vee$ and $\neg$ should be understood as in traditional IR systems and modern search engines. These systems typically have a version of negation in the form of binary operator $AND\text{-}NOT$ which is essentially set difference thus safe (in the database sense of the term [1]). For example, a search engine query $wp_1 \; AND\text{-}NOT \; wp_2$ will return the set of documents that satisfy wp_1 *minus* these that satisfy wp_2. In our information dissemination setting, there is no problem considering an "unsafe" version of negation since word patterns are checked for satisfaction against a single incoming document. Note that the previous work of [8] has *not* considered negation in its word pattern language (but has considered negation in the query language which supports attributes; see Section 5).

We now introduce a new class of word patterns that allows us to capture the concepts of *order* and *distance* between words in a text document. We will assume the existence of a set of *(distance) intervals* $\mathcal{I}$ defined as follows:

$$\mathcal{I} = \{[l, u] : \; l, u \in \mathbb{N}, l \geq 0 \text{ and } l \leq u\} \cup$$
$$\{[l, \infty) : \; l \in \mathbb{N} \text{ and } l \geq 0\}$$

The symbols $\in$ and $\subseteq$ will be used to denote membership and inclusion in an interval as usual.

The following definition uses intervals to impose lower and upper bounds on distances between word patterns.

Definition 3 *Let* $\mathcal{V}$ *be a vocabulary. A* proximity word pattern *over vocabulary* $\mathcal{V}$ *is an expression*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-1}} wp_n$$

where $wp_1, wp_2, \ldots, wp_n$ *are positive proximity-free word patterns over* $\mathcal{V}$ *and* $i_1, i_2, \ldots, i_{n-1}$ *are intervals from the set* $\mathcal{I}$*. The symbols* $\prec_i$ *where* $i \in \mathcal{I}$ *are called* proximity operators*. The number of proximity-free word patterns in a proximity word pattern (i.e.,* n *above) is called its* size.

Example 3 *The following are proximity word patterns:*

$$Holiday \prec_{[0,0]} Inn, \;\; Mini \prec_{[0,0]} Palace \prec_{[0,0]} Hotel,$$

$$luxurious \prec_{[0,3]} (hotel \vee apartment), \;\; hotel \prec_{[0,\infty)} view$$

$$holiday \prec_{[0,10]} beach \prec_{[0,10]} (clean \wedge sandy)$$

The proximity word pattern $wp_1 \prec_{[l,u]} wp_2$ stands for "word pattern wp_1 is *before* wp_2 and is separated by wp_2 by *at least* l and *at most* u *words*". In the above example $luxurious \prec_{[0,3]} hotel$ denotes that the word "hotel" appears after word "luxurious" and at a distance of at least 0 and at most 3 words. The word pattern $Holiday \prec_{[0,0]} Inn$ denotes that the word "Holiday" appears exactly before word "Inn" so this is a way to encode the string "Holiday Inn". We can also have arbitrarily long sequences of proximity operators with similar meaning (see the examples above). Note that proximity-free subformulas in proximity word-patterns can be more complex than just simple words (but negation is *not* allowed; this restriction will be explained below). This makes proximity-word patterns a very expressive notation.

Traditional IR systems have proximity operators kW and kN where k is a natural number. The proximity word pattern $wp_1 \; kW \; wp_2$ stands for "word pattern wp_1 is before wp_2 and is separated by wp_2 by at most k words". In our work this can be captured by $wp_1 \prec_{[0,k]} wp_2$. The operator kN is used to denote distance of at most k words where the order of the involved patterns does not matter. In our framework the expression $wp_1 \; kN \; wp_2$ can be approximated by $wp_1 \prec_{[0,k]} wp_2 \vee wp_2 \prec_{[0,k]} wp_1$ as in [8].[2] In addition to the above operators, our framework allows the expression of simple order constraints between words using operators $\prec_{[0,\infty]}$. This feature does not appear to be useful immediately, but it is required if we want to be able to express some word patterns that are logically entailed (see Section 4). Order constraints of the form $\prec_{[0,\infty]}$ between various text structures are also present in more advanced text model proposals such as the model of proximal nodes of [16].

Definition 4 *Let* $\mathcal{V}$ *be a vocabulary. A* word pattern *over vocabulary* $\mathcal{V}$ *is an expression generated by the grammar*

$$WP \rightarrow PFWP \mid PWP \mid WP \wedge WP \mid WP \vee WP \mid (WP)$$

where non-terminals $PFWP$ *and* PWP *represent proximity-free and proximity word patterns respectively.*

A word pattern will be called positive *if its proximity-free subformulas are positive.*

Example 4 *The following are word patterns of the most general kind we allow:*

$$holiday \wedge (hotel \prec_{[0,10]} (cheap \wedge clean)),$$

$$holiday \wedge (luxurious \prec_{[0,0]} hotel) \wedge \neg Hilton,$$

$$holiday \wedge (luxurious \prec_{[0,0]} hotel) \wedge (Holiday \prec_{[0,0]} Inn),$$

[2] [8] (page 23) gives an example that demonstrates why these two expressions are not equivalent given the meaning of operator kN. The example involves a text value and word patterns with overlapping positions in that text value hence the difference.

$$Vienna \wedge ((Dolce \prec_{[0,0]} Vita \prec_{[0,0]} Hotel)$$
$$\vee (Mini \prec_{[0,0]} Palace \prec_{[0,0]} Hotel))$$

We have here completed the definition of the concept of word pattern. We now turn to defining its semantics.

4 Semantics

We now give meaning to the expressions that define word patterns. First, we define what it means for a text value to satisfy a proximity-free word pattern.

Definition 5 *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$ and wp a proximity-free word pattern over $\mathcal{V}$. The concept of s* satisfying *wp (denoted by $s \models wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models wp$ iff there exists $p \in \{1, \ldots, |s|\}$ and $s(p) = wp$.*
2. *If wp is of the form $\neg wp_1$ then $s \models wp$ iff $s \not\models wp_1$.*
3. *If wp is of the form $wp_1 \wedge wp_2$ then $s \models wp$ iff $s \models wp_1$ and $s \models wp_2$.*
4. *If wp is of the form $wp_1 \vee wp_2$ then $s \models wp$ iff $s \models wp_1$ or $s \models wp_2$.*
5. *If wp is of the form (wp_1) then $s \models wp$ iff $s \models wp_1$.*

The above definition mirrors the definition of satisfaction for Boolean logic [17]. This will allow us to draw on a lot of related results in the rest of this paper.

Example 5 *Let s be the following text value:*

During our holiday in Milos we stayed

in a luxurious hotel by the beach

Then $s \models holiday \wedge Milos$.

The following definition captures the notion of a set of positions in a text value containing exactly the words that contribute to the satisfaction of a proximity-free word pattern. This notion is then used to define satisfaction of proximity word patterns.

Definition 6 *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$, wp a proximity-free word pattern over $\mathcal{V}$, and P a subset of $\{1, \ldots, |s|\}$. The concept of s* satisfying *wp* with set of positions *P (denoted by $s \models_P wp$) is defined as follows:*

1. *If wp is a word of $\mathcal{V}$ then $s \models_P wp$ iff there exists $x \in \{1, \ldots, |s|\}$ such that $P = \{x\}$ and $s(x) = wp$.*
2. *If wp is of the form $wp_1 \wedge wp_2$ then $s \models_P wp$ iff there exist sets of positions $P_1, P_2 \subseteq \{1, \ldots, |s|\}$ such that $s \models_{P_1} wp_1$, $s \models_{P_2} wp_2$ and $P = P_1 \cup P_2$.*
3. *If wp is of the form $wp_1 \vee wp_2$ then $s \models_P wp$ iff $s \models_P wp_1$ or $s \models_P wp_2$.*
4. *If wp is of the form (wp_1) then $s \models_P wp$ iff $s \models_P wp_1$.*

Now we define what it means for a text value to satisfy a proximity word pattern.

Definition 7 *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$ and wp a proximity word pattern over $\mathcal{V}$ of the form*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-1}} wp_n.$$

Then $s \models wp$ iff there exist sets $P_1, P_2, \ldots, P_n \subseteq \{1, \ldots, |s|\}$ such that $s \models_{P_j} wp_j$ and $min(P_j) - max(P_{j-1}) - 1 \in i_{j-1}$ for all $j = 2, \ldots, n$ (the operators max and min have the obvious meaning).

Example 6 *The text value of Example 5 satisfies the following word patterns:*

$$luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach$$

$$luxurious \prec_{[0,0]} (hotel \vee apartment) \prec_{[0,5]} beach,$$

$$(holiday \wedge Milos) \prec_{[0,10]} luxurious \prec_{[0,0]} hotel$$

The sets of positions required by the definition are for the first and second word pattern $\{10\}$, $\{11\}$ and $\{14\}$, and for the third one $\{3, 5\}$, $\{10\}$ and $\{11\}$.

If the structure of wp falls under the four cases of our most general definition (Definition 4), satisfaction is similarly defined in a recursive way as in Definition 5 (for Cases 1, 3 and 4) and Definition 7 (for Case 2).

Example 7 *The text value of Example 5 satisfies word pattern $holiday \wedge (luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach)$.*

We have here completed the definition of the concept of satisfaction of a word pattern by a text value. Let us now compare our development with the approach of Chang and colleagues presented in papers [7, 8, 9]. To the best of our knowledge, these papers contain the only comprehensive treatment of proximity word patterns that exists in the literature. The development of proximity word patterns in [7, 8, 9] follows closely the IR tradition i.e., operators kW and kN (already mentioned above) are used together with the boolean operators AND and OR. These operators can be intermixed in arbitrary ways (e.g., $((w_1\ AND\ (w_2\ (8W)\ w_3))\ (10W)\ w_4)$ where w_1, w_2, w_3, w_4 are words is a legal expression), and the result of their evaluation on document databases is defined in an algebraic way (page 9 of [8]). We have opted for an approach which is more in the spirit of Boolean logic, allows negation and carefully distinguishes word patterns with and without proximity operators. Thus operators cannot be mixed in arbitrary ways but only as prescribed by Definition 4.

5 An Attribute-Based Data Model and Query Language

Now that we have studied the data model $\mathcal{WP}$ in great detail, we are ready to define our second data model and query language. Data model $\mathcal{AWP}$ is based on *attributes* or *fields* with finite-length strings as values (in the acronym $\mathcal{AWP}$, the letter $\mathcal{A}$ stands for "attribute"). Strings will be understood as sequences of words as formalised by the model $\mathcal{WP}$ presented earlier. Attributes can be used to encode textual information such as author, title, date, body of text and so on. $\mathcal{AWP}$ is restrictive since it offers a rather flat view of a text document, but it has wide applicability as we will show below.

We start our formal development by defining the concepts of document schema and document. Because, we are considering only textual information we use the concepts of *document* as a substitute of the concept of *event*. Throughout the rest of this paper we assume the existence of a countably infinite set of attributes $\mathbf{U}$ called the *attribute universe*.

Definition 8 *A* document schema *$\mathcal{D}$ is a pair $(\mathcal{A}, \mathcal{V})$ where $\mathcal{A}$ is a subset of the attribute universe* $\mathbf{U}$ *and $\mathcal{V}$ is a vocabulary.*

Example 8 *An example of a document schema for a news dissemination application is*

$$\mathcal{D} = (\{SENDER,\ EMAIL,\ BODY\}, \mathcal{E}).$$

Definition 9 *Let $\mathcal{D}$ be a document schema. A* document *d over schema $(\mathcal{A}, \mathcal{V})$ is a set of attribute-value pairs (A, s) where $A \in \mathcal{A}$, s is a text value over $\mathcal{V}$, and there is at most one pair (A, s) for each attribute $A \in \mathcal{A}$.*

Example 9 *The following is a document over the schema of Example 8:*

$$\{\ (SENDER, \text{“}John\ Brown\text{”}),$$
$$(EMAIL, \text{“}jbrown@yahoo.com\text{”}),$$
$$(BODY, \text{“}During\ our\ holiday\ in\ Milos\ ...\text{”})\ \}$$

Before we proceed, it is instructive to compare $\mathcal{AWP}$ with other models available in the literature. In previous work on event dissemination systems, textual information has been treated as *strings* and atomic formulas of the form $A\ op\ s$ have been used in defining events or profiles (in this formula A is an attribute, s is a string and *op* is an operator like equality, substring, prefix and postfix) [5, 4]. The models $\mathcal{WP}$ and $\mathcal{AWP}$ are complementary to the string-based approach since they are based on the linguistic concept of *word*. This can make $\mathcal{WP}$ and $\mathcal{AWP}$ more useful in some applications (e.g., alert systems for digital libraries or other commercial systems where similar models are supported already for *retrieval*).

The syntax of our query language is given by the following recursive definition.

Definition 10 *Let $\mathcal{D} = (\mathcal{A}, \mathcal{V})$ be a document schema. A* query *over $\mathcal{D}$ is a formula in any of the following forms:*

1. *$A \sqsupset wp$ where $A \in \mathcal{A}$ and wp is a* positive *word pattern over $\mathcal{V}$. The formula $A \sqsupset wp$ can be read as "A contains word pattern wp".*
2. *$A = s$ where $A \in \mathcal{A}$ and s is a text value over $\mathcal{V}$.*
3. *$\neg\phi$ where ϕ is a query containing no proximity word patterns.*
4. *$\phi_1 \vee \phi_2$ where ϕ_1 and ϕ_2 are queries.*
5. *$\phi_1 \wedge \phi_2$ where ϕ_1 and ϕ_2 are queries.*

Example 10 *The following are queries over the schema of Example 8:*

$$SENDER \sqsupset (John \prec_{[0,2]} Smith),$$
$$\neg SENDER = \text{“}John\ Smith\text{”} \wedge$$
$$(BODY \sqsupset (Milos \wedge (hotel \prec_{[0,5]} beach)))$$

5.1 Semantics

Let us now define the semantics of the above query language in our dissemination setting. We start by defining when a document satisfies a query.

Definition 11 *Let $\mathcal{D}$ be a document schema, d a document over $\mathcal{D}$ and ϕ a query over $\mathcal{D}$. The concept of document d* satisfying *query ϕ (denoted by $d \models \phi$) is defined as follows:*

1. *If ϕ is of the form $A \sqsupset wp$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$ and $s \models wp$.*
2. *If ϕ is of the form $A = s$ then $d \models \phi$ iff there exists a pair $(A, s) \in d$.*
3. *If ϕ is of the form $\neg\phi_1$ then $d \models \phi$ iff $d \not\models \phi_1$.*
4. *If ϕ is of the form $\phi_1 \wedge \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ and $d \models \phi_2$.*
5. *If ϕ is of the form $\phi_1 \vee \phi_2$ then $d \models \phi$ iff $d \models \phi_1$ or $d \models \phi_2$.*

Example 11 *The first query of Example 10 is not satisfied by the document of Example 9 while the second one is satisfied.*

6 Conclusions

In this paper we presented the models $\mathcal{WP}$ and $\mathcal{AWP}$ for textual information dissemination. We laid down the logical foundations of these models, we formalized two fundamental problems arising in information dissemination environments utilizing them, and studied the computational complexity of these problems.

Our current work concentrates on studying the computational complexity of satisfaction, filtering, satisfiability and entailment for models $\mathcal{WP}$ and $\mathcal{AWP}$. These problems have been formally defined in [14] and some related theoretical analysis was provided. Our current work also concentrates on designing sophisticated algorithms for the PTIME cases of the above problems. Finally, we are also extending the query language of model $\mathcal{AWP}$ to include a *similarity* operator like the one of [10] (which is based on the TF-IDF weighting scheme). This will allow users to create some very interesting profiles (for example, notify me when a paper appears in the journal "ACM Transactions on Computer Systems" and it is similar to this one).

Acknowledgements

This work was carried out as part of the DIET (Decentralised Information Ecosystems Technologies) project (IST-1999-10088), within the Universal Information Ecosystems initiative of the Information Society Technology Programme of the European Union. I would like to thank the other participants in the DIET project, from Departmento de Teoria de Senal y Comunicaciones, Universidad Carlos III de Madrid, the Intelligent Systems Laboratory, BTexact Technologies and the Intelligent and Simulation Systems Department, DFKI, for their comments and contributions. Special thanks go to Dimitris Plexousakis, Spiros Skiadopoulos, and my graduate students for reading previous versions of this paper and providing useful comments on it.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] M. Altinel and M. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[4] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith. Efficent filtering in publish-subscribe systems using binary decision diagrams. In *Proceedings of the 23rd International Conference on Software Engineering*, Toronto, Ontario, Canada, 2001.

[5] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proceedings of the 19th ACM Symposium on Principles of Distributed Computing (PODC'2000)*, pages 219–227, 2000.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[7] C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Boolean Query Mapping across Heterogeneous Information Sources. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):515–521, 1996.

[8] C.-C. K. Chang, H. Garcia-Molina, and A. Paepcke. Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System. *ACM Transactions on Information Systems*, 17(1):1–39, 1999.

[9] K. C.-C. Chang. *Query and Data Mapping Across Heterogeneous Information Sources*. PhD thesis, Stanford University, January 2001.

[10] W. W. Cohen. WHIRL: A word-based information representation language. *Artificial Intelligence*, 118(1-2):163–196, 2000.

[11] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of IJCAI-97*, 1997.

[12] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. In *Proceedings of ACM SIGMOD-2001*, 2001.

[13] D. Faensen, L. Faulstich, H. Schweppe, A. Hinze, and A. Steidinger. Hermes – A Notification Service for Digital Libraries. In *Proceedings of the Joint ACM/IEEE Conference on Digital Libraries (JCDL'01), Roanoke, Virginia, USA*, 2001.

[14] M. Koubarakis. Boolean Queries with Proximity Operators for Information Dissemination. Proceedings of the workshop on Foundations of Models and Languages for Information Integration (FMII-2001), Viterbo, Italy , 16-18 September, 2001. In LNCS (forthcoming).

[15] D. R. Kuokka and L. P. Harada. Issues and extensions for information matchmaking protocols. *International Journal of Cooperative Information Systems*, 5(2-3):251–274, 1996.

[16] G. Navarro and R. A. Baeza-Yates. Proximal Nodes: A Model to Query Document Databases by Content and Structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[17] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[18] J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Proceedings of COOPIS-2000*, 2000.