Boolean Queries with Proximity Operators for Information Dissemination*

Manolis Koubarakis

Dept. of Electronic and Computer Engineering
Technical University of Crete
73100 Chania, Crete, Greece
manolis@ced.tuc.gr, www.ced.tuc.gr/~manolis

Abstract. We study two data models and query languages especially designed for the dissemination of textual information over wide-area networks. Our data models are based on the well-known Boolean model with proximity operators inspired from traditional Information Retrieval research. The main focus of this paper is the logical foundations of these models and their comparison with similar models in the literature. In our information dissemination setting, we also formalise some fundamental problems such as the problem of matching incoming documents with submitted user queries, and the problems of query satisfiability and entailment.

1 Introduction

The selective dissemination of information to interested users is a problem arising frequently in today's information society. This problem has recently received the attention of various research communities including researchers from databases [14], agent systems [11], digital libraries [13], distributed computing [6] 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* (i.e., *continuous queries*) to middle-agents at their local libraries to receive information about interesting documents. It is the job of the middle-agent to match incoming documents to queries efficiently and forward matching documents to subscribers.

* Presented at the workshop on Foundations of Models and Languages for Information Integration (FMII-2001), Viterbo, Italy , 16-18 September, 2001. To appear in LNCS (forthcoming).

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 [18, 6].

The above information integration and dissemination scenario appears in many application areas beyond digital libraries (e.g., e-commerce, news dissemination and so on). In this paper we are concerned only with the information dissemination dimension. The importance of information dissemination (information push vs. information pull) has been argued very eloquently in [14]. Indexing user profiles for efficient matching with incoming documents in a centralised setting has been studied in SIFT [28, 29], XFilter [3] and Le Subscribe [25, 12]. The issues arising in distributed information dissemination have also been studied in SIFT [27, 29], but most notably in event dissemination systems such as SIENA [6] and others.

In this paper, we concentrate on the foundations of models and languages for *textual* information dissemination. From previous work on related data modeling efforts, our work is most closely related to work by Chang and colleagues [8–10] since we have adopted a variation of their document data model and query language. From the area of information dissemination, our work is most closely related to the event dissemination system SIENA [6] because we share an interest in the problems of query subsumption, query forwarding, matching queries with source advertisements and so on. However, we use a data model and query language different than the ones used in SIENA because we assume that sources publish textual documents and not events.

The contributions of this paper are the following:

1. We develop formally two data models and query languages for systems of textual information dissemination. The first data model is based on free text and its query language is based on the *boolean model with proximity operators* [4, 8, 9]. The second data model is based on *attributes* or *fields* with finite-length strings as values. Its query language is an extension of the query language of the first data model.
   The usefulness of very similar data models and query languages for the *retrieval and integration* of textual information, as it is commonly found in traditional Information Retrieval (IR) systems and modern digital libraries, has already been shown in [8–10]. Our work modifies and extends some of the data modeling concepts of [8–10], develops a solid formal basis for the extensions, and demonstrates their usefulness to textual information dissemination applications.

2. For each of the two data models we develop, we define formally three problems that are fundamental in all information dissemination systems: the problem of matching an incoming document to a user query, the problem of deciding whether a given query can be matched by any document at all, and the problem of deciding whether a user query is entailed (conversely, subsumed) by another. In work in progress which is not reported in this paper, we built on this foundation and study the computational complexity of these problems.

The rest of the paper is organised as follows. Section 2 presents the data model based on free text and its sophisticated query language. Sections 3 and 4 lay the logical foundations of this model, and formalise the three fundamental problems mentioned above, as they arise when this data model and query language is used for textual information dissemination. Then Section 5 builds on this foundation and develops the same machinery for our attribute-based data model and its query language. Finally, Section 6 discusses future work and concludes.

2 Text Values and Word Patterns

In this section we present our first data model and query language for textual information dissemination. The data model is based on free text which is captured formally by the concept of text value. Our query language is based on the Boolean model with proximity operators. Queries in this model are formalised using the concept of word pattern [9]. The two basic concepts of this section (text values and word patterns) are subsequently used in Section 5 to define the attribute-based data model and 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 in any of the following forms:*

1. *w where w is a word in the vocabulary $\mathcal{V}$.*
2. *$\neg wp$ where wp is a proximity-free word pattern.*
3. *$wp_1 \wedge wp_2$ where wp_1, wp_2 are proximity-free word patterns.*
4. *$wp_1 \vee wp_2$ where wp_1, wp_2 are proximity-free word patterns.*
5. *(wp) where wp is a proximity-free word pattern.*

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:

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

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

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 [2]). 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 [9] 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:

$$Holiday \prec_{[0,0]} Inn, \quad The \prec_{[0,0]} Mini \prec_{[0,0]} Palace \prec_{[0,0]} Hotel,$$
$$luxurious \prec_{[0,3]} hotel, \quad luxurious \prec_{[0,3]} (hotel \vee apartment),$$
$$holiday \prec_{[0,10]} beach \prec_{[0,10]} (clean \wedge sandy), \quad hotel \prec_{[0,\infty)} view$$

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 before 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 [9].[1] 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 3). 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 [23].

Note that proximity operators are very useful and have been popular with sophisticated users of search engines. However, only limited forms of them are in use in current popular search engines (e.g., Altavista's operator $NEAR$ means word-distance 10, Lycos' operator $NEAR$ means word-distance 25, Infoseek used to have a more sophisticated facility but not any more[2]). Proximity operators have also been implemented in other systems such as freeWAIS[3] [26] and INQUERY [5]. There are also advanced IR models such as the model of proximal nodes [23] with proximity operators between arbitrary structural components of a document (e.g., paragraphs or sections). Query languages for XML will probably be the next place to see proximity operators (the recent work [30] contains such an example).

[1] [9] (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.

[2] www.searchlores.org

[3] http://ls6-www.informatik.uni-dortmund.de/ir/projects/freeWAIS-sf/

Definition 4. *Let $\mathcal{V}$ be a vocabulary. A* word pattern *over vocabulary $\mathcal{V}$ is an expression in any of the following categories:*

1. *a proximity-free word pattern over $\mathcal{V}$*
2. *a proximity word pattern over $\mathcal{V}$*
3. *$wp_1 \wedge wp_2$ where wp_1, wp_2 are word patterns.*
4. *$wp_1 \vee wp_2$ where wp_1, wp_2 are word patterns.*
5. *(wp) where wp is a word pattern.*

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:

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

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

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

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

$$holiday \wedge (luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach)$$

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

3 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 $s \models_P wp$) 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 [24]. 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

During our holiday in Milos we stayed in a luxurious hotel by the beach

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

During our holiday in Milos we stayed in a luxurious hotel by the beach

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 [8–10]. 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 [8–10] 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 [9]). 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. This leads to a simpler language because cumbersome (and not especially useful) constructions such as the above are avoided. In the spirit of Boolean logic, a word pattern allows us to distinguish between text values: these that satisfy it, and these that do not. The presence of a word w in some word pattern means "word w is in the text value that satisfies it". Boolean operators are then given the usual semantics as in Boolean logic. Proximity operators are given semantics separately as we have done above. Negation is not allowed to appear inside proximity formulas because it does not seem to offer us any useful capabilities in terms of expressing interesting queries.

We now continue our development with the concept of satisfiable word patterns.

Definition 8. *A word pattern wp is called* satisfiable *if there is a text value s that satisfies it. Otherwise it is called* unsatisfiable.

Example 8. The following word patterns are satisfiable:

$$holiday \wedge hotel,\ holiday \vee vacation,\ Holiday \prec_{[0,0]} Inn$$

The following word patterns are unsatisfiable:

$$holiday \wedge \neg holiday,\ (nice \prec_{[0,0]} holiday) \wedge \neg holiday$$

Definition 9. *Let wp_1 and wp_2 be word patterns. We will say that wp_1* entails *wp_2 (denoted by $wp_1 \models wp_2$) iff for every text value s such that $s \models wp_1$, we have $s \models wp_2$. If $wp_1 \models wp_2$, we also say that wp_2* subsumes *wp_1. If $wp_1 \models wp_2$ and $wp_2 \models wp_1$ then wp_1 and wp_2 are called* equivalent *(denoted by $wp_1 \equiv wp_2$).*

Example 9. The word pattern $holiday \wedge hotel$ entails $holiday$. The word pattern $holiday \wedge (luxurious \prec_{[0,0]} hotel \prec_{[0,5]} beach)$ entails the word pattern $holiday \wedge (hotel \prec_{[0,10]} beach) \wedge luxurious$. The word pattern $vacation$ subsumes the word pattern $sun \wedge vacation \wedge beach$. Similarly, $vacation \vee cruise$ subsumes $cruise$. The word pattern $vacation \wedge (beach \vee mountains)$ is equivalent to $(vacation \wedge beach) \vee (vacation \wedge mountains)$.

The notions of entailment, subsumption and equivalence defined above naturally correspond to standard notions of Boolean logic. If we consider word patterns to be queries, we also have a natural correspondence with the same notions of relational database theory [7, 17]. In the case of proximity-free word patterns the correspondence with Boolean logic is exact, thus the only new feature in our enquiry is the presence of proximity word patterns. Subsumption and minimal subsuming queries involving proximity word patterns have also been studied in [9]. Our work differs from [9] in the following ways: (i) we deal with proximity in a slightly different way (our proximity operators are more general and we do not mix proximity and boolean operators in arbitrary ways), and (ii) the problem in [9] is one of integration thus capabilities of data sources have to be taken into account in relevant definitions. However, many of the results of [9] are important to us and appropriate credits will be given in the rest of the paper.

Let us close this section by discussing why negation is not allowed to apply to proximity word patterns (Definition 4). If negation is allowed, we should not expect that the resulting formula could always be rewritten equivalently into one where negation is moved somehow inwards as in the case of Boolean logic. The interested reader can be persuaded by trying to do this for the following formula (which illegal in our model):

$$\neg(luxurious \prec_{[0,3]} hotel \prec_{[0,3]} beach)$$

If we restrict our attention to proximity formulas with a single proximity operator, this restriction can easily be lifted. The word pattern $\neg(luxurious \prec_{[0,3]} hotel)$ is equivalent to

$$\neg luxurious \vee \neg hotel \vee (hotel \prec_{[0,\infty]} luxurious) \vee (luxurious \prec_{[4,\infty]} hotel).$$

4 Some Properties of Satisfaction and Entailment

In this section we present some results that are easy to show and give some important facts about the semantic notions introduced in Section 3. These results are fundamental for any system of information dissemination based on the data model we develop here.

Proposition 1. *Let wp_1 and wp_2 be word patterns and wp_2 is proximity-free. $wp_1 \models wp_2$ iff $wp_1 \wedge \neg wp_2$ is unsatisfiable.*

The above proposition gives the usual relation between entailment and unsatisfiability of Boolean logic as it should be stated in our framework. Note that wp_2 is required to be proximity-free so that the negation operator can be applied.

The next proposition allows us to compute proximity-free word patterns entailed by a given proximity word pattern. A similar result is stated in [9] for the case of IR operator kW.

Proposition 2. *Let $wp_0, wp_1, wp_2, \ldots, wp_n$ be proximity-free word patterns. If $\bigvee_{k=1}^{n} wp_k \models wp_0$ then*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-1}} wp_n \models wp_0$$

Example 10. From the above proposition we have that $luxurious \prec_{[0,0]} hotel$ entails $hotel$. Also, $hotel \prec_{[0,20]} (view \wedge sunset)$ entails $view \vee cliff$.

The next three propositions allows us to compute proximity word patterns entailed by a given proximity word pattern. The first one enables us to compute an entailed word pattern by eliminating the first or the last proximity-free subformula in a given proximity word pattern.

Proposition 3. *Let $wp_1, wp_2, \ldots, wp_{n-1}, wp_n$ be proximity-free word patterns, and wp the proximity word pattern*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-2}} wp_{n-1} \prec_{i_{n-1}} wp_n.$$

Then wp entails

$$wp_2 \prec_{i_2} \cdots \prec_{i_{n-2}} wp_{n-1} \prec_{i_{n-1}} wp_n$$

and

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{n-2}} wp_{n-1}.$$

Example 11. Let wp be

$$luxurious \prec_{[0,0]} hotel \prec_{[2,5]} (sandy \wedge clean) \prec_{[0,0]} beach.$$

Then wp entails

$$hotel \prec_{[2,5]} (sandy \wedge clean) \prec_{[0,0]} beach$$

and

$$luxurious \prec_{[0,0]} hotel \prec_{[2,5]} (sandy \wedge clean).$$

The next proposition enables us to compute an entailed word pattern by eliminating any proximity-free subformula other than the first or the last in a proximity word pattern. In this case, a new proximity operator must be computed for the resulting formula.

Proposition 4. *Let $wp_1, wp_2, \ldots, wp_{k-1}, wp_k, wp_{k+1}, \ldots, wp_n$ be proximity-free word patterns, and wp the proximity word pattern*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{k-2}} wp_{k-1} \prec_{i_{k-1}} wp_k \prec_{i_k} wp_{k+1} \prec_{i_{k+1}} \cdots \prec_{i_{n-1}} wp_n$$

where wp_k is a word or a conjunction of words. If $n > 1$ and $2 \leq k \leq n-1$ then wp entails

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{k-2}} wp_{k-1} \prec_{i'} wp_{k+1} \prec_{i_{k+1}} \cdots \prec_{i_{n-1}} wp_n$$

where i' is an interval defined as follows. Let i_{k-1} be $[l_{k-1}, u_{k-1}]$ and i_k be $[l_k, u_k]$. If $wp_k \equiv w$ where w is a word then $i' = [l_{k-1} + l_k + 1, u_{k-1} + u_k + 1]$. Otherwise, $wp_k \equiv w_1 \wedge \cdots \wedge w_m$ where $m > 1$ and $w_1, \ldots, w_m$ are distinct words and $i' = [l_{k-1} + l_k + m, \infty)$. Interval i' is defined similarly if the right endpoint of i_{k-1} or i_k is ∞.

Example 12. Let wp be

$$luxurious \prec_{[0,0]} hotel \prec_{[2,5]} (sandy \wedge clean) \prec_{[0,0]} beach.$$

Then wp entails

$$luxurious \prec_{[3,6]} (sandy \wedge clean) \prec_{[0,0]} beach$$

and

$$luxurious \prec_{[0,0]} hotel \prec_{[4,\infty)} beach.$$

The new proximity operator in the above word pattern is $\prec_{[4,\infty)}$ because we cannot put an upper bound on the number of words between $s(min(P_3))$ and $s(max(P_3))$ for any text value s such that $s \models_{P_3} sandy \wedge clean$ (see Definition 7).

It is interesting to point out that a result similar to the one stated in Proposition 4, is *not* included in [9] for the case of the IR operator kW although it seems to be natural. For example, the word pattern

$$luxurious \prec_{[0,0]} (small \wedge hotel) \prec_{[0,5]} beach$$

can be expressed by

$$luxurious\ (0W)\ (small \wedge hotel)\ (5W)\ beach$$

in the framework of [9], and it entails word pattern $luxurious \prec_{[2,\infty)} beach$. But this entailed word pattern cannot be expressed in the language of [9], thus the class of proximity word patterns studied there is not closed with respect to entailment.

The next proposition allows us to compute an entailed formula by weakening a proximity constraint.

Proposition 5. *Let $wp_1, wp_2, \ldots, wp_k, wp_{k+1}, \ldots, wp_n$ be proximity-free word patterns, and wp the proximity word pattern*

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{k-1}} wp_k \prec_{i_k} wp_{k+1} \prec_{i_{k+1}} \cdots \prec_{i_{n-1}} wp_n.$$

If $k \in \mathbb{N}$ and $1 \leq k \leq n-1$ then wp entails

$$wp_1 \prec_{i_1} wp_2 \prec_{i_2} \cdots \prec_{i_{k-1}} wp_k \prec_{i'} wp_{k+1} \prec_{i_{k+1}} \cdots \prec_{i_{n-1}} wp_n$$

for every interval i' such that $i \subseteq i'$.

Example 13. Let wp be

$$luxurious \prec_{[0,0]} hotel \prec_{[2,5]} (sandy \wedge clean) \prec_{[0,0]} beach.$$

Then wp entails

$$luxurious \prec_{[0,5]} hotel \prec_{[2,5]} (sandy \wedge clean) \prec_{[0,0]} beach.$$

The usual laws for equivalent Boolean expressions with connectives $\neg, \wedge$ and $\vee$ hold for the case of proximity-free word patterns (commutativity, associativity etc. – see Proposition 4.1 in [24]). The following equivalence which is easy to show states that proximity operators distribute over $\vee$. However, proximity operators do not *distribute* over $\wedge$ i.e., $(luxurious \wedge hotel) \prec_{[2,2]} wp_3$ is not equivalent to

$$(luxurious \prec_{[2,2]} beach) \wedge (hotel \prec_{[2,2]} beach)$$

because there is a text value (e.g., "*a luxurious hotel on the beach*") that satisfies the former but not the latter formula [9].

Proposition 6. *Let $wp_1, \ldots, wp_k, \ldots, wp_n$ be proximity-free word patterns, and wp the proximity word pattern*

$$wp_1 \prec_{i_1} \cdots \prec_{i_{k-1}} wp_k \prec_{i_k} \cdots \prec_{i_{n-1}} wp_n.$$

If word pattern wp_k is of the form $\phi \vee \psi$ then wp is equivalent to

$$(wp_1 \prec_{i_1} \cdots \prec_{i_{k-1}} \phi \prec_{i_k} \cdots \prec_{i_{n-1}} wp_n)$$
$$\vee$$
$$(wp_1 \prec_{i_1} \cdots \prec_{i_{k-1}} \psi \prec_{i_k} \cdots \prec_{i_{n-1}} wp_n).$$

For the case of IR operators kW and kN, the above law was recently presented in [9]. The curious reader can also see the rather archaic [21] (credited with the proximity laws in [9]) for related discussion that took place a long time ago.

Example 14. The word pattern

$$luxurious \prec_{[0,0]} (hotel \vee bungalow) \prec_{[2,10]} beach$$

is equivalent to

$(luxurious \prec_{[0,0]} hotel \prec_{[2,10]} beach) \vee (luxurious \prec_{[0,0]} bungalow \prec_{[2,10]} beach)$.

Proposition 7. *Let $wp_1, \ldots, wp_k, \ldots, wp_n$ be proximity-free word patterns, and wp the proximity word pattern*

$$wp_1 \prec_{i_1} \cdots \prec_{i_{k-1}} wp_k \prec_{i_k} \cdots \prec_{i_{n-1}} wp_n.$$

If word pattern wp_k is of the form $w_1 \wedge \cdots \wedge w_m$ where $w_1, \ldots, w_m$ are words then wp is equivalent to

$$\bigvee_{(l_1,\ldots,l_m) \text{ is a permutation of } (w_1,\ldots,w_m)} l_1 \prec_{[0,\infty)} \cdots \prec_{[0,\infty)} l_m.$$

Example 15. The word pattern

$$(clean \wedge cheap) \prec_{[0,5]} hotel \prec_{[2,5]} beach$$

is equivalent to

$$clean \prec_{[0,\infty)} cheap \prec_{[0,5]} hotel \prec_{[2,5]} beach$$
$$\vee$$
$$cheap \prec_{[0,\infty)} clean \prec_{[0,5]} hotel \prec_{[2,5]} beach.$$

4.1 Normal Forms

We now define the concepts of atomic word patterns, conjunctive/disjunctive word patterns, and the conjunctive/disjunctive normal forms.

Definition 10. *A word pattern is called* atomic *if it is a word, a negated word or a proximity word pattern* $w_1 \prec_{i_1} \cdots \prec_{i_{n-1}} w_n$ *where* $w_1, \ldots, w_n$ *are words. A word pattern is called* conjunctive *(resp.* disjunctive*) if it is a conjunction (resp. disjunction) of atomic word patterns. A word pattern is in* conjunctive normal form (CNF) *(resp.* disjunctive normal form (DNF)*) if it is a conjunction (resp. disjunction) of disjunctive (resp. conjunctive) word patterns.*

The following easy proposition can be proved by induction on the structure of a word pattern given the laws discussed above.

Proposition 8. *Every word pattern is equivalent to a word pattern in CNF and a word pattern in DNF.*

For the case of IR operators kW and kN, the above proposition has been stated in [9].

5 An Attribute-Based Data Model and Query Language

Now that we have studied text values and word patterns in great detail, we are ready to define our second data model and query language. This data model for text documents is based on *attributes* or *fields* with finite-length strings as values. Attributes are used to encode information such as author, title, date, body of text and so on. This simple data model 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. Throughout the rest of this paper we assume the existence of a countably infinite set of attributes $\mathbf{U}$ called the *attribute universe*.

Definition 11. *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 16. An example of a document schema for a news dissemination application is $\mathcal{D} = (\{SENDER,\ EMAIL,\ BODY\}, \mathcal{E})$.

Definition 12. *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 17. The following is a document over the schema of Example 16:

$\{\ (SENDER, \text{“}Manolis\ Koubarakis\text{”}),\ (EMAIL, \text{“}manolis@ced.tuc.gr\text{”}),$

$(BODY, \text{“}During\ our\ holiday\ in\ Milos\ we\ stayed\ in\ a\ wonderful\ hotel\ by\ the\ beach\text{”})\ \}$

Before we proceed, it is instructive to compare our text document data model with other models available in the literature. Simple text data models like ours have been popular with database researchers for some time due to their obvious connections to the relational data model [4, 22]. The model of this section is essentially the model of [8–10] with a different class of proximity word patterns as explained in Section 3. The database community has now moved to semi-structured XML-based models for the representation of textual information [1] and our work can naturally be extended to follow this important research avenue.

The model of this section appears to be rather simple in comparison with traditional relational or object-oriented models as they could have been used for storing textual data [22]. However our model is not as rigid as the relational or object-oriented data model so it enables a useful form of heterogeneity in the limited structure allowed. In this respect, our model is similar to recently proposed network directory models [16] although it does not contain a notion of class or hierarchy and all attributes have the same type. In fact, one could imagine extending network directory data models (such as the one in [16]) to allow for a text data type, and also extend the relevant query languages (such as the language $\mathcal{L}_0$ of [16]) to have the boolean and proximity operators discussed in Section 2.

As it has been first discussed in [9], simple attribute-value data models like ours have also been popular with commercial information retrieval services (e.g., Dialog[4] or Lexis-Nexis[5]), digital library protocols (e.g., Z39.50[6]), resource discovery systems [19], and metadata standardisation activities for digital collections (e.g., Dublin Core[7]). The results of this paper will be useful to any of these systems if an information integration and dissemination service like the one discussed in the application scenario of Section 1 is developed.

Our document model and query language is also related to the data model for news group warehouses recently proposed in [15]. This is rather reasonable given that news dissemination is one of the motivating applications of our work. The query language presented in [15] allows selection of news articles based on conditions involving $=$ and $\sqsupset$ operators like we do in our work. Also, the news-group maintenance problem as defined in [15] can be rephrased in our framework as the problem of finding which posted profiles are satisfied by a document that has just arrived.

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

Definition 13. *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".*

[4] www.dialog.com
[5] www.lexis-nexis.com
[6] http://www.niso.org/standard.html
[7] www.dublincore.org

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 18. The following are queries over the schema of Example 16:

$$SENDER \sqsupset (John \prec_{[0,2]} Smith),$$

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

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 14. *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 19. The first query of Example 18 is not satisfied by the document of Example 17 while the second one is satisfied.

We now define the concepts of query satisfiability, entailment and equivalence.

Definition 15. *Let* ϕ *be a query over schema* $\mathcal{D}$. *If there is a document* d *over* $\mathcal{D}$ *such that* $d \models \phi$ *then* ϕ *is called* satisfiable *otherwise it is called* unsatisfiable.

Example 20. The queries

$$SENDER = \text{“}Jones\text{”} \wedge SENDER = \text{“}King\text{”},$$

$$SENDER \sqsupset Jones \wedge \neg SENDER \sqsupset Jones,$$

$$SENDER = \text{“}Michael\ Jones\text{”} \wedge \neg SENDER \sqsupset Jones$$

are unsatisfiable.

Definition 16. *Let* ϕ_1 *and* ϕ_2 *be queries over schema* $\mathcal{D}$. *Then* ϕ_1 entails ϕ_2 *(denoted by* $\phi_1 \models \phi_2$*) iff every document* d *over* $\mathcal{D}$ *that satisfies* ϕ_1 *also satisfies* ϕ_2. *If* ϕ_1 *entails* ϕ_2 *then we also say that* ϕ_2 subsumes ϕ_1. *We will say that* ϕ_1 is equivalent *to* ϕ_2 *(denoted by* $\phi_1 \equiv \phi_2$*) iff* ϕ_1 *entails* ϕ_2 *and vice versa.*

Example 21. The query

$$(BODY \sqsupset Milos) \wedge (BODY \sqsupset Milos \wedge (hotel \prec_{[0,3]} beach)) \wedge \neg SENDER = \text{“}John\ Smith\text{”}$$

is equivalent to

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

The latter query entails (equivalently: is subsumed by) $BODY \sqsupset Milos$.

We now define the concepts of atomic queries, conjunctive/disjunctive queries, and conjunctive/disjunctive normal forms.

Definition 17. *A query is called* atomic *if it is in one of the following forms:* $A = s, \neg A = s, A \sqsupset w, \neg A \sqsupset w$ *or* $A \sqsupset wp$ *where* s *is a text value,* w *is a word and* wp *is an atomic proximity word pattern. A query is called* conjunctive *(resp.* disjunctive*) if it is a conjunction (resp. disjunction) of atomic queries. A query is in* conjunctive normal form (CNF) *(resp.* disjunctive normal form (DNF)*) if it is a conjunction (resp. disjunction) of disjunctive (resp. conjunctive) queries.*

The following proposition is easy to see.

Proposition 9. *Let* A *be an attribute and* wp_1, wp_2 *be word patterns. Then the following equivalences hold:*

1. $A \sqsupset (wp_1 \wedge wp_2) \equiv (A \sqsupset wp_1) \wedge (A \sqsupset wp_2)$
2. $A \sqsupset (wp_1 \vee wp_2) \equiv (A \sqsupset wp_1) \vee (A \sqsupset wp_2)$
3. $\neg(A \sqsupset (wp_1 \wedge wp_2)) \equiv (\neg A \sqsupset wp_1) \vee (\neg A \sqsupset wp_2)$
4. $\neg(A \sqsupset (wp_1 \vee wp_2)) \equiv (\neg A \sqsupset wp_1) \wedge (\neg A \sqsupset wp_2)$

In Cases 3 and 4, wp_1 *and* wp_2 *are assumed not to contain proximity operators.*

From Proposition 8 and 9, we now have the following result which closes this paper.

Proposition 10. *Every query is equivalent to a query in DNF and a query in CNF.*

6 Conclusions and Outlook

In this paper we presented two data models and query languages for textual information dissemination. Then we concentrated on laying their logical foundations and formalizing some important problems arising in any information dissemination system based on these two models. Our current work in this area concentrates on the following questions:

- What is the computational complexity of the problem of matching a document with a user query, and the problems of deciding query satisfiability and entailment?
- What are sophisticated indexing algorithms for the efficient selection of user profiles matching an incoming document? How can these algorithms be implemented and used efficiently in the context of the distributed information dissemination architecture designed for the DIET project [20].

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, my students in Chania and the reviewers and participants of FMII-2001 for reading previous versions of this paper and providing useful comments on it.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann Publishers, 2000.
2. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
3. M. Altinel and M.J. Franklin. Efficient filtering of XML documents for selective dissemination of information. In *Proceedings of the 26th VLDB Conference*, 2000.
4. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.
5. James P. Callan, W. Bruce Croft, and Stephen M. Harding. The INQUERY retrieval system. In *Proceedings of the International Conference Database and Expert Systems Applications (DEXA'92)*, pages 78–83, 1992.
6. 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.
7. A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proceedings of the Ninth Annual ACM Symposium on Theory of Computing*, pages 77–90, 1977.
8. 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.
9. 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.
10. K. C.-C. Chang. *Query and Data Mapping Across Heterogeneous Information Sources*. PhD thesis, Stanford University, January 2001.
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. J. Franklin and S. B. Zdonik. "Data In Your Face": Push Technology in Perspective. In *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 516–519, 1998.
15. H. Gupta and D. Srivastava. The data warehouse of newsgroups. In *Proceedings of the 7th International Conference on Database Theory (ICDT '99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 1999.
16. H. V. Jagadish, L. V. S. Lakshmanan, T. Milo, D. Srivastava, and D. Vista. Querying network directories. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 133–144, 1999.
17. Anthony Klug. On Conjunctive Queries Containing Inequalities. *Journal of ACM*, 35(1):146–160, 1988.
18. 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.
19. Shih-Hao Li and Peter B. Danzig. Boolean similarity measures for resource discovery. *TKDE*, 9(6):863–876, 1997.
20. P. Marrow, M. Koubarakis, R.H. van Lengen, F. Valverde-Albacete, E. Bonsma, J. Cid-Suerio, A.R. Figueiras-Vidal, A. Gallardo-Antolin, C. Hoile, T. Koutris, H. Molina-Bulla, A. Navia-Vazquez, P. Raftopoulou, N. Skarmeas, C. Tryfonopoulos, F. Wang, and C. Xiruhaki. Agents in Decentralised Information Ecosystems: The DIET Approach. In *Symposium on Information Agents for E-Commerce, AISB'01 Convention*, University of York, United Kingdom, March 2001.
21. P. C. Mitchell. A note about the proximity operators in information retieval. In *Proceedings of SIGIR'73*, pages 177–179, 1973.
22. E. Moss, editor. *Data Engineering – Special Issue on Integrating Text Retrieval and Databases*, volume 19, March 1996.
23. 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.
24. C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
25. J. Pereira, F. Fabret, F. Llirbat, and D. Shasha. Efficient matching for web-based publish/subscribe systems. In *Proceedings of COOPIS-2000*, 2000.
26. U. Pfeifer, N. Fuhr, and T. Huynh. Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and SFgate. In *Proceedings of the 3rd International World-Wide Web Conference*, pages 1027–1036, 1995.
27. T.W. Yan and H. Garcia-Molina. Distributed selective dissemination of information. In *Proceedings of the 3rd International Conference on Parallel and Distributed Information Systems (PDIS)*, pages 89–98, 1994.
28. T.W. Yan and H. Garcia-Molina. Index structures for selective dissemination of information under the boolean model. *ACM Transactions on Database Systems*, 19(2):332–364, 1994.
29. T.W. Yan and H. Garcia-Molina. The SIFT information dissemination system. *ACM Transactions on Database Systems*, 24(4):529–565, 1999.
30. C. Zhang, J. Naughton, D. De Witt, Q. Luo, and G. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In *Proceedings ACM SIGMOD International Conference on Management of Data*, 2001.