Logic and Computational Complexity for Boolean Information Retrieval

Manolis Koubarakis, Spiros Skiadopoulos, and Christos Tryfonopoulos

Abstract—We study the complexity of query satisfiability and entailment for the Boolean Information Retrieval models $\mathcal{WP}$ and $\mathcal{AWP}$ using techniques from propositional logic and computational complexity. $\mathcal{WP}$ and $\mathcal{AWP}$ can be used to represent and query textual information under the Boolean model using the concept of attribute with values of type text, the concept of word, and word proximity constraints. Variations of $\mathcal{WP}$ and $\mathcal{AWP}$ are in use in most deployed digital libraries using the Boolean model, text extenders for relational database systems (e.g., Oracle 10g), search engines, and P2P systems for information retrieval and filtering.

Index Terms—Boolean information retrieval, computational complexity, data models, query languages, satisfiability, entailment, proximity.

1 Introduction

We study two well-known data models of Information Retrieval (IR) [2] and digital libraries [9], [10], [8], which we have called $\mathcal{WP}$ and $\mathcal{AWP}$ in [21], [19], [30], [29], [28], [20]. Data model $\mathcal{WP}$ is based on *free text* and its query language is based on the Boolean model for *word patterns*. Word patterns are formulas that enable the expression of constraints on the existence, nonexistence, or proximity of words in a text document. Data model $\mathcal{AWP}$ extends $\mathcal{WP}$ with *named attributes* with free text as values. The query language of $\mathcal{AWP}$ is also a simple extension of the query language of $\mathcal{WP}$ so that attributes are included.

Models such as $\mathcal{WP}$ that are based on word patterns were introduced in the early days of IR and have been implemented in many digital library systems in wide use today [2]. Word patterns are also used in 1) all current search engines, 2) advanced IR models such as the model of proximal nodes [22] which allows proximity operators between arbitrary structural components of a document (e.g., paragraphs or sections), and 3) recent full-text extensions to XML-based languages e.g., TeXQuery [1].

The model $\mathcal{AWP}$ has been used recently in our systems DIAS, P2P-DIET, DHTrie, and LibraRing [17], [19], [30], [29], [28]. DIAS [19] is a distributed alert service for digital libraries which utilizes a P2P architecture and protocols similar to that of the event dissemination system SIENA [7]. DIAS uses $\mathcal{WP}$ and $\mathcal{AWP}$ as an expressive data model and query language for textual information. P2P-DIET [17] is the ancestor of DIAS and uses $\mathcal{AWP}$ as a metadata model for describing and querying digital resources. An extension of model $\mathcal{AWP}$, called $\mathcal{AWPS}$, that introduces a similarity operator based on the IR vector space model, is used in the P2P systems DHTrie [29] and LibraRing [28] that are built on top of distributed hash tables [3].

In the database literature, word patterns have been studied by Chang and colleagues in the context of integrating heterogeneous digital libraries [9], [10], [8]. The model $\mathcal{AWP}$ is essentially the model of [8] but with a slightly different class of word patterns.

Even though many deployed systems are using $\mathcal{WP}$ and $\mathcal{AWP}$ and many papers have appeared on their variations, only [9], [10], [8], [21], [19] have studied in depth the *logical* foundations of these data models. As we have previously discussed in [21], we would like to develop information retrieval and filtering systems in a *principled* and *formal* way. With this motivation and the architectures of [19], [17], [30], [29], [28] in mind, we have posed the following requirements for models and languages to be used in information retrieval and filtering systems [21]:

1. *Expressivity*. The languages for documents and queries must be rich enough to satisfy the demands of information consumers and capabilities of information providers.
2. *Formality*. The syntax and semantics of the proposed models and languages must be defined formally.
3. *Computational efficiency*. The following problems should be defined formally and algorithms must be provided for their efficient solution (keeping in mind that there will be a trade-off with the expressivity requirement):
   a. The *satisfiability* problem: Deciding whether a query can be satisfied by any document at all.
   b. The *satisfaction* problem: Deciding whether a document satisfies a query.
   c. The *filtering* problem: Given a collection of queries Q and an incoming document d, find all queries $q \in Q$ that satisfy d.

- M. Koubarakis is with the Department of Informatics and Telecommunication, National and Kapodistrian University of Athens, Panepistimiopolis, Ilisia, Athens 15784 Greece. E-mail: koubarak@di.uoa.gr.
- S. Skiadopoulos is with the Department of Computer Science and Technology, University of Peloponnese, Karaiskaki Street, 22100, Tripoli, Greece. E-mail: spiros@uop.gr.
- C. Tryfonopoulos is with the Department of Electronic and Computer Engineering, Technical University of Crete, 73100 Chania, Crete, Greece. E-mail: trifon@intelligence.tuc.gr.

Manuscript received 18 Oct. 2005; revised 3 Apr. 2006; accepted 15 June 2006; published online 18 Oct. 2006.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0484-1005.

d. The *entailment* problem: Deciding whether a query is more or less "general" than another.

In previous work, we have defined formally the models $\mathcal{WP}$ and $\mathcal{AWP}$ [19] and presented efficient centralized and distributed algorithms for the filtering problem [30], [29]. In this paper, we continue our formal work in this area and concentrate on *model-theoretic questions* for the logics of $\mathcal{WP}$ and $\mathcal{AWP}$ that have been ignored in previous papers. We study the model theory of $\mathcal{WP}$ and $\mathcal{AWP}$ and especially-questions related to satisfiability and entailment. We show that the satisfiability problem for queries in $\mathcal{WP}$ and $\mathcal{AWP}$ is $\mathcal{NP}$-complete and the entailment problem is co$\mathcal{NP}$-complete. We also discuss cases where these problems can be solved in polynomial time. Our results are original and complement the studies of [8], [21] where no such complexity questions were posed.

The rest of the paper is organized as follows: In the next section, we present the models $\mathcal{WP}$ and $\mathcal{AWP}$. Sections 3 and 4 presents our complexity results on satisfiability and entailment. Then, Section 5 discusses related work. The last section concludes the paper and discusses our plans for future work.

2 The Models $\mathcal{WP}$ and $\mathcal{AWP}$

Let us start by presenting the data model $\mathcal{WP}$ and its query language. $\mathcal{WP}$ has been inspired by [10]. It assumes that textual information is in the form of *free text* and can be queried by *word patterns* (hence, the acronym for the model).

We assume the existence of a finite *alphabet* $\boldsymbol{\Sigma}$. A *word* is a finite nonempty sequence of letters from $\boldsymbol{\Sigma}$. We also assume the existence of a (finite or infinite) set of words called the *vocabulary* and denoted by $\mathcal{V}$. A *text value* s of length n over vocabulary $\mathcal{V}$ is a total function $s : \{1, 2, \ldots, n\} \rightarrow \mathcal{V}$. In other words, a text value s is a finite sequence of words from the assumed vocabulary and $s(i)$ gives the ith element of s. $|s|$ will denote the length of text value s (i.e., its number of words).

We now give the definition of word pattern. We assume the existence of a set of *(distance) intervals*

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

Let $\boldsymbol{i}$ be an interval in $\mathcal{I}$. We will denote the left-endpoint (respectively, right-endpoint) of $\boldsymbol{i}$ by $\inf(\boldsymbol{i})$ (respectively, $sup(\boldsymbol{i})$).

Definition 1. *Let $\mathcal{V}$ be a vocabulary. A* word pattern *over vocabulary $\mathcal{V}$ is a formula in any of the following forms:*

1. *w, where w is a word of $\mathcal{V}$.*
2. *$w_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} w_n$, where $w_1, \ldots, w_n$ are words of $\mathcal{V}$ and $\boldsymbol{i}_1, \ldots, \boldsymbol{i}_{n-1}$ are intervals of $\mathcal{I}$.*
3. *$\neg\phi$, $\phi_1 \vee \phi_2$, or $\phi_1 \wedge \phi_2$, where ϕ, ϕ_1, and ϕ_2 are word patterns.*

Example 1. The following are word patterns:

$$constraint \wedge ((optimization \vee programming)$$
$$\neg algorithms \wedge ((complexity \prec_{[1,5]} satisfaction) \vee (complexity \prec_{[1,8]} filtering)).$$

Operator $\prec_{\boldsymbol{i}}$ is called a *proximity operator* and is a generalization of the traditional IR operators kW and kN [10]. Proximity operators are used to capture the concepts of *order* and *distance* between words in a text document. They can be used to construct formulas of $\mathcal{WP}$ that we will call *proximity word patterns* (Case 2 of Definition 1). The proximity word pattern $w_1 \prec_{[l,u]} w_2$ stands for "word w_1 is *before* w_2 and is separated by w_2 by *at least* l and *at most* u *words*." The interpretation of proximity word patterns with more than one operator $\prec_{\boldsymbol{i}}$ is similar.

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 $\mathcal{WP}$, the expression $wp_1 \ kN \ wp_2$ can be approximated by $wp_1 \prec_{[0,k]} wp_2 \vee wp_2 \prec_{[0,k]} wp_1$. Chang et al. [10] gives an example (page 23) that demonstrates why these two expressions are not equivalent given the meaning of operator kN. The example involves qa text value and word patterns with overlapping positions in that text value hence the difference.

The development of proximity word patterns in [9], [10], [8] 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. $\mathcal{WP}$ opts for an approach which is more in the spirit of Boolean logic, allows negation and carefully distinguishes word patterns with and without proximity operators. 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, an atomic word pattern (i.e., a word or a proximity word pattern) allows us to distinguish between text values: those that satisfy it, and those that do not. Boolean operators are then given their standard semantics.

In addition to the above operators, $\mathcal{WP}$ allows the expression of simple order constraints between words using operators $\prec_{[0,\infty]}$. 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 [22].

Definition 2. *A word pattern will be called* positive *if it does not contain negation. A word pattern will be called* proximity-free *if it does not contain formulas of the form $w_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} w_n$. A word pattern will be called* conjunctive *if it does not contain disjunction.*

Example 2. The following are positive word patterns:

$$satisfiability$$
$$local \wedge search \wedge algorithms,$$
$$information \wedge (retrieval \vee dissemination),$$
$$logic \prec_{[0,1]} computational \prec_{[0,0]} complexity.$$

The first three are proximity-free word patterns. The first, second, and fourth word pattern is conjunctive.

Definition 3. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$, and wp a 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 a proximity word pattern of the form $w_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} w_n$, then $s \models wp$ iff there exist $p_1, \ldots, p_n \in \{1, \ldots, |s|\}$ such that, for all $j = 2, \ldots, n$ we have $s(p_j) = w_j$ and $p_j - p_{j-1} - 1 \in \boldsymbol{i}_{j-1}$.*
3. *If wp is of the form $\neg wp_1, wp_1 \wedge wp_2, wp_1 \vee wp_2$ or (wp_1), then $s \models wp$ is defined exactly as satisfaction for Boolean logic.*

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

Example 3. The word patterns of Examples 1 and 2 are satisfiable. Word patterns

$$\neg programming \wedge (constraint \prec_{[0,0]} programming),$$
$$(constraint \prec_{[0,0]} programming) \wedge \neg(constraint \prec_{[0,2]} programming)$$

are unsatisfiable.

Definition 4. *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 $wp_1 \models wp_2$ and $wp_2 \models wp_1$, then wp_1 and wp_2 are called* equivalent *(denoted by $wp_1 \equiv wp_2$).*

Example 4. Word pattern $constraint \wedge programming$ entails word pattern $constraint$. Word pattern

$$optimization \wedge (constraint \prec_{[0,0]} programming)$$

entails $constraint \prec_{[0,10]} programming$.

Finally, word patterns

$$constraint \prec_{[0,4]} programming,$$
$$constraint \wedge (constraint \prec_{[0,4]} programming)$$

are equivalent.

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

Let us close this section by pointing out that proximity word patterns have been considered as atomic formulas of $\mathcal{WP}$ (Definition 1) because, in general, negation cannot be moved inside a proximity word pattern as in the case of Boolean operators. The interested reader can be persuaded by trying to do this for the following formula:

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

If we restrict our attention to proximity formulas with a single proximity operator, this restriction can easily be lifted. For example, the word pattern

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

is equivalent to the following:

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

Let us now use the machinery of $\mathcal{WP}$ to define data model $\mathcal{AWP}$. The new concept of $\mathcal{AWP}$ is the concept of *attribute* with value free text (in the acronym $\mathcal{AWP}$, the letter $\mathcal{A}$ stands for "attribute").

We assume the existence of a countably infinite set of attributes $\mathbf{U}$ called the *attribute universe*. 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. 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 5. The following is a document over schema $(\{AUTHOR, TITLE, ABSTRACT\}, \mathcal{V})$:

$$\{(AUTHOR, ''John\ Brown''),$$
$$(TITLE, ''Local\ search\ and\ constraint\ programming''),$$
$$(ABSTRACT, ''In\ this\ paper\ we\ show\ \ldots'')\}.$$

The syntax of the query language of $\mathcal{AWP}$ is given by the following recursive definition.

Definition 5. *A* query *over schema $(\mathcal{A}, \mathcal{V})$ is a formula in any of the following forms:*

1. *$A \sqsupseteq wp$, where $A \in \mathcal{A}$ and wp is a word pattern over $\mathcal{V}$ (this is 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. *$\neg\phi$, $\phi_1 \vee \phi_2$, $\phi_1 \wedge \phi_2$, where ϕ, ϕ_1, and ϕ_2 are queries.*

Example 6. The following is a query over the schema shown in Example 5:

$$AUTHOR \sqsupseteq Brown \wedge$$
$$TITLE \sqsupseteq search \wedge (constraint \prec_{[0,0]} programming).$$

Definition 6. *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 $A \sqsupseteq wp$, 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 $d \not\models \phi_1$. Similarly, for wedge and $\vee$.*

Example 7. The query of Example 6 is satisfied by the document of Example 5.

Proposition 2. *Let A be an attribute and wp_1, wp_2 be word patterns. Then, the following equivalences hold:*

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

Definition 7. *A query is called* atomic *if it is of the form $A = t$ where t is a text value, or $A \sqsupseteq wp$ where wp is a word or a proximity word pattern. A query is called* conjunctive *if it does not contain disjunction.*

Example 8. The following queries are atomic:

$$AUTHOR = \text{"}James\ Brown,\text{"}$$
$$TITLE \sqsupseteq search,$$
$$ABSTRACT \sqsupseteq constraint \prec_{[0,0]} programming.$$

Proposition 3. *Every query is equivalent to a Boolean combination of atomic queries.*

Proof. Use the first three equivalences of Proposition 2 repeatedly. □

3 Satisfiability and Entailment in $\mathcal{WP}$

An instance of the satisfiability problem for proximity-free word patterns can be considered as an instance of the satisfiability problem for Boolean logic (SAT) and vice versa (by interchanging the roles of words and Boolean variables). Thus, we have to consider any complications that might arise due to proximity word patterns only.

In what follows, we will need the binary operation of *concatenation* of two text values.

Definition 8. *Let s_1 and s_2 be text values over vocabulary $\mathcal{V}$. Then, the* concatenation *of s_1 and s_2 is a new text value denoted by s_1s_2 and defined by the following:*

1. $|s_1s_2| = |s_1| + |s_2|$
2. $$s_1s_2(x) = \begin{cases} s_1(x) & \text{for all } x \in \{1,\ldots,|s_1|\} \\ s_2(x - |s_1|) & \text{for all } x \in \{|s_1|+1,\ldots, \\ & |s_2| + |s_1|\}. \end{cases}$$

We will also need the concept of the *empty text value* which is denoted by ϵ and has the property $|\epsilon| = 0$. The following properties of concatenation are easily seen:

1. $(s_1s_2)s_3 = s_1(s_2s_3)$, for all text values s_1, s_2, and s_3.
2. $s\epsilon = \epsilon s = s$ for every text value s.

The associativity of concatenation allows us to write concatenations of more than two text values without using parentheses.

The following variant of the concept of satisfaction captures the notion of a set of positions in a text value containing *exactly* the words that contribute to the satisfaction of a *positive proximity-free* word pattern. This variant is used in Lemma 1 and in Proposition 4.

Definition 9. *Let $\mathcal{V}$ be a vocabulary, s a text value over $\mathcal{V}$, wp a positive 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 $s \models_P wp_1$.*

We also need the following notation: Let P be a subset of the set of natural numbers $\mathbb{N}$, and $x \in \mathbb{N}$. We will use the notation $P + x$ to denote the set of natural numbers $\{p + x : p \in P\}$.

Lemma 1. *Let s and s' be text values, wp be a positive proximity-free word pattern, and $P \subseteq \{1,\ldots,|s|\}$. If $s \models_P wp$, then $ss' \models_P wp$ and $s's \models_{P+|s'|} wp$.*

Positive proximity-free word patterns are satisfiable as we show below.

Proposition 4. *If wp is a positive proximity-free word pattern, then wp is satisfiable. In fact, there exists a text value s_0 such that*

1. *$|s_0| \le |wp| \cdot ops(wp)$, where $ops(wp)$ is the number of operators of wp (or 1 if wp has no operators).*
2. *Every word of s_0 is a word of wp.*
3. *$s_0 \models_{\{1,\ldots,|s_0|\}} wp$.*

Proof. The proof is by induction on the structure of wp.

Base case: Let wp be a word $w \in \mathcal{V}$. In this case, wp is satisfiable because we can form a text value s_0 such that $s_0 \models_{\{1\}} w$, where $|s_0| = 1$ and $s_0(1) = w$. The conclusion of the lemma is now obviously satisfied.

Inductive step: Let wp be a positive proximity-free word pattern of the form $wp_1 \wedge wp_2$, and assume that the inductive hypothesis holds for wp_1 and wp_2. Then, we can form text values s_0^1 and s_0^2 such that $s_0^1 \models_{\{1,\ldots,|s_0^1|\}} wp_1$ and $s_0^2 \models_{\{1,\ldots,|s_0^2|\}} wp_2$. Then, from Lemma 1, we have

$$s_0^1 s_0^2 \models_{\{1,\ldots,|s_0^1|\}} wp_1$$

and

$$s_0^1 s_0^2 \models_{\{1,\ldots,|s_0^2|\}+|s_0^1|} wp_2.$$

Finally, from Definition 9, we have

$$s_0^1 s_0^2 \models_{\{1,\ldots,|s_0^1|,|s_0^1|+1,\ \ldots,|s_0^1|+|s_0^2|\}} wp_1 \wedge wp_2$$

as required. It is also easy to see that

$$\begin{gathered} |s_0^1 s_0^2| = |s_0^1| + |s_0^2| \ \le \\ |wp_1| \cdot ops(wp_1) + |wp_2| \cdot ops(wp_2) < \\ [ops(wp_1) + ops(wp_2)] \cdot |wp| < ops(wp) \cdot |wp|. \end{gathered}$$

The $\vee$ case is done similarly. □

Obviously, proximity word patterns are also satisfiable.

Proposition 5. *Let wp be a proximity word pattern of the form $w_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} w_n$. Then, wp is satisfied by the text value $s = w_1z_1 \cdots z_{n-1}w_n$, where z_l, $l = 1,\ldots,n-1$ are text values of the following form. If $inf(\boldsymbol{i}_l) > 0$ then z_l is formed by $inf(\boldsymbol{i}_l)$ successive occurrences of the special word # which is* *not contained in wp. Otherwise, if $inf(\boldsymbol{i}_l)$, then z_l is the empty text value ϵ.*

Moreover, any text value satisfying a proximity word pattern is of a very special form.

Proposition 6. *Let wp be a proximity word pattern of the form $w_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} w_n$. If $s \models wp$, then s is of the form.*

$$s = \underbrace{? \cdots ?}_{i_0 \text{ times}} w_1 \underbrace{? \cdots ?}_{i_1 \text{ times}} w_2 \cdots w_{n-1} \underbrace{? \cdots ?}_{i_{n-1} \text{ times}} w_n \underbrace{? \cdots ?}_{i_n \text{ times}},$$

where $0 \leq i_0$, $i_1 \in \boldsymbol{i}_1, \ldots, i_{n-1} \in \boldsymbol{i}_{n-1}$, $0 \leq i_n$, and each occurrence of the symbol ? represents an arbitrary (and not necessarily the same) word.

Example 9. Let us consider the proximity word pattern

$$wp = constraint \prec_{[0,0]} programming \prec_{[0,\infty]} methods.$$

It is easy to verify that text value "many applications use constraint programming algorithms and methods to solve interesting problems" 1) is of the form set by Proposition 6 and 2) satisfies word pattern wp.

Finally, we show that any positive word pattern is satisfiable.

Proposition 7. *If wp is a positive word pattern, then wp is satisfiable.*

Proof. We will construct a text value t such that $t \models wp$. If wp contains m proximity word patterns $\phi_1, \ldots, \phi_m$, text value t is of the form $s_0 s_1 \cdots s_m$ where:

- s_0 is a sequence formed by the juxtaposition of all words appearing in wp in any order, and
- for every $j = 1, \ldots, m$, s_j is a text value, formed as in Proposition 5, such that $s_j \models \phi_j$. □

Lemma 2. *Let wp_1 and wp_2 be proximity word patterns of the following form:*

$$wp_1 = a_1 \prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{n-1}} a_n \quad \text{and}$$
$$wp_2 = b_1 \prec_{\boldsymbol{j}_1} \cdots \prec_{\boldsymbol{j}_{m-1}} b_m.$$

Word pattern wp_1 entails wp_2 iff the following conditions hold:

Condition 1. *Word pattern wp_2 is equal to*

$$a_{p_1} \prec_{\boldsymbol{j}_1} \ldots \prec_{\boldsymbol{j}_{m-1}} a_{p_m},$$

where $1 \leq p_1 < \cdots < p_m \leq n$.

Condition 2. *For every $v = 1, \ldots, m-1$, we have:*

$$inf(\boldsymbol{j}_v) \leq inf(\boldsymbol{i}_{p_v}) + \cdots + inf(\boldsymbol{i}_{p_{v+1}}) + p_{v+1} - p_{v-1}$$
$$sup(\boldsymbol{j}_v) \text{ is } \begin{cases} \geq & \begin{pmatrix} sup(\boldsymbol{i}_{p_v}) + \cdots + \\ sup(\boldsymbol{i}_{p_{v+1}}) + \\ p_{v+1} - p_{v-1} \end{pmatrix} \begin{array}{l} \text{if all } sup(\boldsymbol{i}_{p_v}), \ldots, \\ sup(\boldsymbol{i}_{p_{v+1}}) \text{ are diffe-} \\ \text{rent than } \infty \end{array} \\ \infty & \text{otherwise.} \end{cases}$$

Proof. The "if" case is obvious. For the "only if" part, let us assume that $wp_1 \models wp_2$ holds. We will prove that wp_2 is of the form set by the lemma. The proof is in three steps.

Step 1 (Condition 1). We will first prove that the words of wp_2 are a subset of the words in wp_1, i.e.,

$$\{b_1, \ldots, b_m\} \subseteq \{a_1, \ldots, a_n\}.$$

By contradiction, let us assume that there exists a word b_v, $1 \leq v \leq m$, of wp_2 such that $b_v \notin \{a_1, \ldots, a_n\}$. Let us now consider text value τ defined as:

$$\tau = a_1 \underbrace{\# \cdots \#}_{i_1 \text{ times}} a_2 \cdots a_{n-1} \underbrace{\# \cdots \#}_{i_{n-1} \text{ times}} a_n, \tag{1}$$

where # is a special word which is not contained in wp_1 and wp_2 and $i_1 \in \boldsymbol{i}_1, \ldots, i_n \in \boldsymbol{i}_n$. It is easy to verify that τ satisfies wp_1 but, since τ does not include word b_v, it does not satisfies wp_2. Thus, we have $wp_1 \not\models wp_2$ which contradicts our initial assumption.

Step 2 (Condition 1). We will now prove that the words of wp_1 that appear in wp_2 actually appear in the same order as they do in wp_1, i.e., word pattern $wp_2 = a_{p_1} \prec_{\boldsymbol{j}_1} \cdots \prec_{\boldsymbol{j}_{m-1}} a_{p_m}$, where $1 \leq p_1 < \cdots < p_m \leq n$. By contradiction, let us assume that there exist two distinct words $b_v = a_{p_v}$ and $b_{v'} = a_{p_{v'}}$, $1 \leq v < v' \leq m$, of wp_2 such that $p_v \geq p_{v'}$. In other words,

$$\begin{aligned} wp_1 = a_1 &\prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{p_{v'}-1}} \\ &a_{p_{v'}} \prec_{\boldsymbol{i}_{p_{v'}}} \cdots \prec_{\boldsymbol{i}_{p_{v-1}}} \\ &a_{p_v} \prec_{\boldsymbol{i}_{p_v}} \cdots \prec_{\boldsymbol{i}_{n-1}} a_n, \end{aligned}$$

$$\begin{aligned} wp_2 = a_{p_1} &\prec_{\boldsymbol{j}_1} \cdots \prec_{\boldsymbol{j}_{v-1}} \\ &a_{p_v} \prec_{\boldsymbol{j}_v} \cdots \prec_{\boldsymbol{j}_{v'-1}} \\ &a_{p_{v'}} \prec_{\boldsymbol{j}_{v'}} \cdots \prec_{\boldsymbol{j}_{m-1}} a_{p_m}. \end{aligned}$$

It is easy to verify that text value τ (defined in (1)) satisfies wp_1 but it does not satisfies wp_2; a contradiction.

Step 3 (Condition 2). Finally, we will prove that for every $v = 1, \ldots, m-1$, we have:

$$inf(\boldsymbol{j}_v) \leq inf(\boldsymbol{i}_{p_v}) + \cdots + inf(\boldsymbol{i}_{p_{v+1}}) + p_{v+1} - p_{v-1}$$
$$sup(\boldsymbol{j}_v) \text{ is } \begin{cases} \geq & \begin{pmatrix} sup(\boldsymbol{i}_{p_v}) + \cdots + \\ sup(\boldsymbol{i}_{p_{v+1}}) + \\ p_{v+1} - p_{v-1} \end{pmatrix} \begin{array}{l} \text{if all } sup(\boldsymbol{i}_{p_v}), \ldots, \\ sup(\boldsymbol{i}_{p_{v+1}}) \text{ are diffe-} \\ \text{rent than } \infty \end{array} \\ \infty & \text{otherwise.} \end{cases}$$

By contradiction, let us assume that there exists a subformula $a_{p_v} \prec_{\boldsymbol{j}_v} a_{p_{v+1}}$ of wp_2 such that

$$inf(\boldsymbol{j}_v) > \in f(\boldsymbol{i}_{p_v}) + \cdots + inf(\boldsymbol{i}_{p_{v+1}}) + p_{v+1} - p_v - 1. \tag{2}$$

From Step 2, word patterns wp_1 and wp_2 are of the following form:

$$\begin{aligned} wp_1 = a_1 &\prec_{\boldsymbol{i}_1} \cdots \prec_{\boldsymbol{i}_{p_v-1}} \\ &a_{p_v} \prec_{\boldsymbol{i}_{p_v}} \cdots \prec_{\boldsymbol{i}_{p_{v-1}-1}} \\ &a_{p_{v+1}} \prec_{\boldsymbol{i}_{p_v}} \cdots \prec_{\boldsymbol{i}_{n-1}} a_n, \end{aligned}$$

$$\begin{aligned} wp_2 = a_{p_1} &\prec_{\boldsymbol{j}_1} \cdots \prec_{\boldsymbol{j}_{v-1}} \\ &a_{p_v} \prec_{\boldsymbol{j}_v} \\ &a_{p_{v+1}} \prec_{\boldsymbol{j}_{v+1}} \cdots \prec_{\boldsymbol{j}_{m-1}} a_{p_m}. \end{aligned}$$

Let us now construct a text value τ' defined as:

$$\begin{aligned}\tau' = a_1 & \underbrace{\# \cdots \#}_{i_1 \text{ times}} a_2 \cdots \\ a_{p_v} & \underbrace{\# \cdots \#}_{i_{p_v} \text{ times}} a_{p_{v+1}} \cdots \\ a_{p_{v+1}-1} & \underbrace{\# \cdots \#}_{i_{p_{v+1}-1} \text{ times}} a_{p_{v+1}} \cdots \\ a_{n-1} & \underbrace{\# \cdots \#}_{i_{n-1} \text{ times}} a_n,\end{aligned} \tag{3}$$

where # is a special word which is not contained in wp_1 and wp_2, and for every s, $1 \leq s \leq n-1$, $i_s = inf(\boldsymbol{i}_s)$ holds. It is easy to verify that τ' satisfies wp_1. Notice that between words a_{p_v} and $a_{p_{v+1}}$ in τ' there are exactly $inf(\boldsymbol{i}_{p_v}) + \cdots + inf(\boldsymbol{i}_{p_{v+1}}) + p_{v+1} - p_v - 1$ words. Therefore, since (2) holds, τ' does not satisfy the subformula $a_{p_v} \prec_{\boldsymbol{j}_v} a_{p_{v+1}}$ of wp_2 and, thus, it does not satisfy wp_2. Thus, we have $wp_1 \not\models wp_2$ which contradicts our initial assumption.

The proof involving $sup(\boldsymbol{j}_v)$ is similar. It differs only in the way we construct text value τ' (3) and specifically in the values of $i_1, \ldots, i_{n-1}$. We now require that $i_1 \in \boldsymbol{i}_1, \ldots, i_{n-1} \in \boldsymbol{i}_{n-1}$ and for every s, $p_v \leq s \leq p_{v+1}$, we define:

$$i_s = \begin{cases} sup(\boldsymbol{i}_s) & \text{if } sup(\boldsymbol{i}_s) \text{ is different} \\ & \text{than } \infty \\ sup(\boldsymbol{j}_v) + 1 & \text{otherwise.} \end{cases}$$

□

Proposition 8. *Let wp_1 and wp_2 be proximity word patterns with n and m words, respectively. Deciding whether $wp_1 \models wp_2$ can be done in $O(n+m)$ time.*

Let $SAT(\mathcal{WP})$ denote the satisfiability problem for formulas of $\mathcal{WP}$. The following two propositions show that the problems SAT and $SAT(\mathcal{WP})$ are equivalent under polynomial time reductions.

Proposition 9. *SAT is polynomially reducible to $SAT(\mathcal{WP})$.*

Proof. Trivial by considering propositional variables to be words. □

Proposition 10. *$SAT(\mathcal{WP})$ is polynomially reducible to SAT.*

Proof. Let ϕ be a formula of $\mathcal{WP}$. We transform ϕ into an instance ϕ' of SAT as follows: We start with ϕ' being ϕ (words of ϕ play the role of propositional variables in ϕ'). Then, we substitute each proximity word pattern wp of ϕ' by a brand new propositional variable v_{wp}. Finally, we conjoin to ϕ' the following formulas:

- $v_{wp} \Longrightarrow w$, for each proximity word pattern wp and word w of wp.
- $v_{wp_1} \Longrightarrow v_{wp_2}$, for each pair of proximity word patterns wp_1, wp_2 such that $wp_1 \models wp_2$.

The above steps can be done in polynomial time because entailment of proximity word patterns can be done in polynomial time (Proposition 8). It is also easy to see that ϕ is a satisfiable formula of $\mathcal{WP}$ iff ϕ' is a satisfiable formula of Boolean logic. Then, the result holds. □

Propositions 9 and 10 have the following corollary.

Corollary 1. *Deciding whether a word pattern is satisfiable is a $\mathcal{NP}$-complete problem. Deciding whether a word pattern entails another is a co$\mathcal{NP}$-complete problem.*

Let us close this section by pointing out that satisfiability and entailment of conjunctive word patterns can be done in PTIME.

Proposition 11. *The satisfiability and entailment problems for conjunctive word patterns can be solved in polynomial time.*

Proof. This is easy to see given Proposition 8. □

4 Satisfiability and Entailment in $\mathcal{AWP}$

Let $SAT(\mathcal{AWP})$ denote the satisfiability problem for queries of $\mathcal{AWP}$. The following two propositions show that the problems SAT and $SAT(\mathcal{AWP})$ are equivalent under polynomial time reductions.

Proposition 12. *SAT is polynomially reducible to $SAT(\mathcal{AWP})$.*

Proof. Let ϕ be an instance of SAT (i.e., a Boolean formula). For every propositional variable p in ϕ introduce an attribute A_p. Then, substitute every occurrence of p in ϕ by $A_p = "true"$ to arrive at an instance ψ of $SAT(\mathcal{AWP})$. Obviously, ϕ is satisfiable iff ψ is satisfiable. □

Proposition 13. *$SAT(\mathcal{AWP})$ is polynomially reducible to SAT.*

Proof. Let ϕ be a query of $\mathcal{AWP}$. Using Proposition 2, ϕ can easily be transformed into a formula θ which is a Boolean combination of atomic queries. This transformation can be done in time linear in the size of the formula.

The next step is to substitute in θ atomic formulas $A = s$ and $A \sqsupseteq wp$ (where wp is a word or a proximity word pattern) by propositional variables $p_{A=s}$ and $p_{A \sqsupseteq wp}$, respectively, to obtain formula θ'. Finally, the following formulas are conjoined to θ' to obtain ψ:

1. If $A = s_1$ and $A = s_2$ are conjuncts of θ' and $s_1 \neq s_2$, then conjoin $p_{A=s_1} \equiv \neg p_{A=s_2}$.
2. If $A = s$ and $A \sqsupseteq wp$ are conjuncts of θ' and $s \models wp$, then conjoin $p_{A=s} \Longrightarrow p_{A \sqsupseteq wp}$.
3. If $A = s$ and $A \sqsupseteq wp$ are conjuncts of θ' and $s \not\models wp$, then conjoin $p_{A=s} \Longrightarrow \neg p_{A \sqsupseteq wp}$.
4. If $A \sqsupseteq wp_1$ and $A \sqsupseteq wp_2$ are conjuncts of θ' and $wp_1 \models wp_2$, then conjoin $p_{A \sqsupseteq wp_1} \Longrightarrow p_{A \sqsupseteq wp_2}$.

The above step can be done in polynomial time because satisfaction and entailment of word patterns in θ can be done in polynomial time. The result for satisfaction is obvious and the result for entailment is from Proposition 8. It is also easy to see that ϕ is a satisfiable query iff ψ is a satisfiable formula of Boolean logic. Then, the result holds. □

Propositions 12 and 13 have the following corollary.

Corollary 2. *Deciding whether a query of $\mathcal{AWP}$ is satisfiable is a $\mathcal{NP}$-complete problem. Deciding whether a query of $\mathcal{AWP}$ entails another is a co-$\mathcal{NP}$-complete problem.*

The following proposition shows that, as in the case of $\mathcal{WP}$, satisfiability and entailment of conjunctive queries in $\mathcal{AWP}$ can be done in PTIME. This is good news given that conjunctive $\mathcal{AWP}$ queries are typically utilized in implementations such as [19], [17], [28].

Proposition 14. *The satisfiability and entailment problems for conjunctive $\mathcal{AWP}$ queries can be solved in polynomial time.*

To obtain a more accurate picture of the tractable versus intractable classes of queries in $\mathcal{AWP}$ one can profitably utilize such results from the propositional satisfiability literature. For example, it is easy to see now that each tractable class C of SAT formulas has a corresponding class C' of tractable formulas of $\mathcal{WP}$ or $\mathcal{AWP}$ if the 2-variable propositional formulas used in the proofs of Propositions 10 and 13 belong to C (e.g., this holds for C being the class of propositional formulas with at most two variables using the tractability of 2-SAT).

5 Related Work

In this section, we discuss related research. Since formal analysis based on logic and complexity as done in this paper is not common in Information Retrieval research, this section briefly surveys other data models (and systems) related to the ones studied in this paper.

5.1 $\mathcal{WP}$

To the best of our knowledge, the papers by Chang and colleagues [9], [10], [8] and the present paper are the only comprehensive formal treatments of proximity word patterns in the literature.

Search engines use models similar to $\mathcal{WP}$ and $\mathcal{AWP}$. The most common support for word patterns in search engines includes the ability to combine words using the Boolean operators $\wedge$, $\vee$, and $\neg$. However, search engines support a version of negation in the form of binary operator $AND\text{-}NOT$ which is essentially set difference, and therefore *safe* in the database sense of the term [26]. 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. Note also that the previous work of [10] has *not* considered negation in its word pattern language but has considered negation in the query language which supports attributes (the one that corresponds to our model $\mathcal{AWP}$).

Proximity operators are a useful extension of the concept of "phrase search" used in current search engines. Limited forms of proximity operators have been offered in the past by various search engines of the pre-Google era (e.g., Altavista had an operator $NEAR$ which meant word-distance 10, Lycos had an operator $NEAR$ which meant word-distance 25, and Infoseek used to have a more sophisticated facility). Google supports proximity by the use of operator "$*$" which, when used between two keywords, specifies a minimum distance of one word between them (multiple occurences of $*$ can also be used to specify a larger minimum distance). The search engine Exalead[1] has an operator NEAR which returns documents that contain given keywords in a vicinity of a fixed number of words, but no ordering of words is supported.

The need to change their index structures and the high computational cost of proximity search, is probably the reason why current search engines limit proximity support to less general operators compared to those used in models $\mathcal{WP}$ and $\mathcal{AWP}$.

Proximity operators have also been implemented in other systems such as freeWAIS [23] and INQUERY [5]. There are also advanced IR models such as the model of proximal nodes [22] with proximity operators between arbitrary structural components of a document (e.g., paragraphs or sections). Data models and query languages for full-text extensions to XML, e.g., TeXQuery [1] is the most recent area of research where proximity operators have been used.

Proximity word patterns can also be viewed as a particular kind of *order constraints* in the sense of constraint networks [14] and databases [25]. There are many papers that discuss algorithms and complexity of various kinds of order constraints, e.g., gap-order constraints [24] or temporal constraints [18], [18]. The algorithms and complexity results regarding $\mathcal{WP}$ can also be viewed as a contribution to this research area.

5.2 $\mathcal{AWP}$

The data model $\mathcal{AWP}$ discussed in Section 2 complements recent proposals for representing and querying textual information in publish/subscribe systems [7], [6] by using linguistically motivated concepts such as *word* and traditional IR operators (instead of strings and operators such as string containment [7], [6]). The methodology and techniques of this paper can be used to study the complexity of satisfiability and entailment for the subscription query language of [6] and we expect the complexity results to be similar.

In [21], [19], we have extended the model $\mathcal{AWP}$ by introducing a "similarity" operator based on the IR vector space model [2]. The similarity concept of this model, called $\mathcal{AWPS}$ (where $\mathcal{S}$ stands for similarity), has in the past been used in database systems with IR influences (e.g., WHIRL [13]) and, more recently, in XML-based query languages, e.g., ELIXIR [12], XIRQL [16], and XXL [27].

6 Outlook

We have studied the model theory of $\mathcal{WP}$ and $\mathcal{AWP}$ and especially questions related to satisfiability and entailment. We showed that the satisfiability problem for queries in $\mathcal{WP}$ and $\mathcal{AWP}$ is $\mathcal{NP}$-complete and the entailment problem is co-$\mathcal{NP}$-complete. We also discussed cases where these problems can be solved in polynomial time.

We would like to use the lessons learned in this paper to study the complexity of query evaluation in RDBMS with text functionalities, combinations of RDBMS and IR systems [11], and proposals for full-text extensions to XML [1]. This recent paper [4] is a good example of such a study where the authors consider the concept of strings in various query languages.

1. Exalead (http://www.exalead.com/) is a search engine developed in France. We mention it here because Exalead is involved in the Quaero project launched in Europe in the summer of 2005 as the European response to Google.

ACKNOWLEDGMENTS

This work was performed while Manolis Koubarakis was with the Technical University of Crete.

REFERENCES

[1] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram, "TeXQuery: A Full-Text Search Extension to Query," *Proc. 13th Int'l World Wide Web Conf.*, pp. 583-594, 2004.
[2] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Addison Wesley, 1999.
[3] H. Balakrishnan, M.F. Kaashoek, D.R. Karger, R. Morris, and I. Stoica, "Looking Up Data in P2P Systems," *Comm. ACM,* vol. 46, no. 2, pp. 43-48, 2003.
[4] M. Benedikt, L. Libkin, T. Schwentick, and L. Segoufin, "Definable Relations and First-Order Query Languages over Strings," *J. ACM,* vol. 50, no. 5, pp. 694-751, 2003.
[5] J. Callan, W. Croft, and S. Harding, "The INQUERY Retrieval System," *Proc. Third Int'l Conf. Database and Expert Systems Applications,* pp. 78-83, 1992.
[6] A. Campailla, S. Chaki, E. Clarke, S. Jha, and H. Veith, "Efficient Filtering in Publish Subscribe Systems Using Binary Decision Diagrams," *Proc. 23rd Int'l Conf. Software Eng. (ICSE '01),* pp. 443-452, May 2001.
[7] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, "Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service," *Proc. 19th ACM Symp. Principles of Distributed Computing (PODC '00),* pp. 219-227, 2000.
[8] K.C.-C. Chang, "Query and Data Mapping across Heterogeneous Information Sources," PhD thesis, Stanford Univ., Jan. 2001.
[9] K.C.-C. Chang, H. Garcia-Molina, and A. Paepcke, "Boolean Query Mapping across Heterogeneous Information Sources," *IEEE Trans. Knowledge and Data Eng.,* vol. 8, no. 4, pp. 515-521, 1996.
[10] K.C.-C. Chang, H. Garcia-Molina, and A. Paepcke, "Predicate Rewriting for Translating Boolean Queries in a Heterogeneous Information System," *ACM Trans. Information Systems,* vol. 17, no. 1, pp. 1-39, 1999.
[11] S. Chaudhuri, R. Ramakrishnan, and G. Weikum, "Integrating DB and IR Technologies: What is the Sound of One Hand Clapping?" *Proc. Second Biennial Conf. Innovative Data Systems Research,* pp. 1-12, 2005.
[12] T. Chinenyanga and N. Kushmerick, "Expressive Retrieval from XML Documents," *Proc. ACM SIGIR '01,* Sept. 2001.
[13] W.W. Cohen and "WHIRL: A Word-Based Information Representation Language," *Artificial Intelligence,* vol. 118, nos. 1-2, pp. 163-196, 2000.
[14] R. Dechter, *Constraint Processing.* Morgan Kaufmann, 2003.
[15] R. Dechter, I. Meiri, and J. Pearl, "Temporal Constraint Networks," *Artificial Intelligence,* special volume on knowledge representation, vol. 49, nos. 1-3, pp. 61-95, 1991.
[16] N. Fuhr and K. Großjohann, "XIRQL: An XML Query Language Based on Information Retrieval Concepts," *ACM Trans. Information Systems,* vol. 22, no. 2, pp. 313-356, Apr. 2004.
[17] S. Idreos, C. Tryfonopoulos, M. Koubarakis, and Y. Drougas, "Query Processing in Super-Peer Networks with Languages Based on Information Retrieval: the P2P-DIET Approach," *Proc. Int'l Workshop Peer-to-Peer Computing and Databases (P2P&DB),* Mar. 2004.
[18] M. Koubarakis, "The Complexity of Query Evaluation in Indefinite Temporal Constraint Databases," *Theoretical Computer Science,* L.V.S. Lakshmanan, ed., special issue on uncertainty in databases and deductive systems, vol. 171, pp. 25-60, Jan. 1997.
[19] M. Koubarakis, T. Koutris, C. Tryfonopoulos, and P. Raftopoulou, "Information Alert in Distributed Digital Libraries: The Models, Languages, and Architecture of DIAS," *Proc. Sixth European Conf. Research and Advanced Technology for Digital Libraries (ECDL),* pp. 527-542, Sept. 2002.
[20] M. Koubarakis, C. Tryfonopoulos, S. Idreos, and Y. Drougas, "Selective Information Dissemination in P2P Networks: Problems and Solutions," *SIGMOD Record,* special issue on peer-to-peer data management, vol. 32, no. 3, pp. 71-76, 2003.
[21] M. Koubarakis, C. Tryfonopoulos, P. Raftopoulou, and T. Koutris, "Data Models and Languages for Agent-Based Textual Information Dissemination," *Proc. Sixth Int'l Workshop Cooperative Information Agents (CIA),* pp. 179-193, Sept. 2002.
[22] G. Navarro and R. Baeza-Yates, "Proximal Nodes: A Model to Query Document Databases by Content and Structure," *ACM Trans. Information Systems,* vol. 15, no. 4, pp. 400-435, 1997.
[23] U. Pfeifer, N. Fuhr, and T. Huynh, "Searching Structured Documents with the Enhanced Retrieval Functionality of Free-WAIS-sf and SFgate," *Computer Networks and ISDN Systems,* vol. 27, no. 6, pp. 1027-1036, 1995.
[24] P. Revesz, "A Closed Form Evaluation for Datalog Queries with Integer (Gap)-Order Constraints," *Theoretical Computer Science,* vol. 116, no. 1, pp. 117-149, 1993.
[25] P. Revesz, *Introduction to Constraint Databases.* Springer, 2002.
[26] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases.* Addison Wesley, 1995.
[27] A. Theobald and G. Weikum, "Adding Relevance to XML," *WebDB (Selected Papers),* pp. 105-124, 2000.
[28] C. Tryfonopoulos, S. Idreos, and M. Koubarakis, "LibraRing: An Architecture for Distributed Digital Libraries Based on DHTs," *Proc. Ninth European Conf. Research and Advanced Technology for Digital Libraries (ECDL),* pp. 25-36, Sept. 2005.
[29] C. Tryfonopoulos, S. Idreos, and M. Koubarakis, "Publish/Subscribe Functionality in IR Environments Using Structured Overlay Networks," *Proc. 28th Ann. Int'l ACM SIGIR Conf.,* pp. 322-329, Aug. 2005.
[30] C. Tryfonopoulos, M. Koubarakis, and Y. Drougas, "Filtering Algorithms for Information Retrieval Models with Named Attributes and Proximity Operators," *Proc. 27th Ann. Int'l ACM SIGIR Conf.,* pp. 313-320, July 2004.

Manolis Koubarakis received a degree in mathematics from the University of Crete, the MSc degree in computer science from the University of Toronto, and the PhD degree in computer science from the National Technical University of Athens. He joined the Department of Informatics and Telecommunications, National and Kapodistrian University of Athens in October 2005. Before coming to Athens, he held positions in the Department of Electronic and Computer Engineering, Technical University of Crete, where he was an assistant and associate professor and director of the Intelligent Systems Laboratory (www.intelligence.tuc.gr), at UMIST, Manchester, where he was a lecturer and at Imperial College, London, as a research associate. Professor Koubarakis has published papers in the areas of database and knowledge-base systems, constraint programming, intelligent agents, semantic Web, and peer-to-peer computing. More information is available at www.di.uoa.gr/~koubarak.

Spiros Skiadopoulos received the diploma the National Technical University of Athens and the MSc degree from UMIST, and the PhD degree from the National Technical University of Athens. He is an assistant professor at the University of Peloponnese. His research interests include spatial and temporal databases, constraint databases, query evaluation and optimization, and constraint reasoning and optimization. He has published more than 25 papers in international refereed journals and conferences.

Christos Tryfonopoulos received the BSc degree in computer science from the University of Crete in 2000 and the MSc degree in computer engineering from the Technical University of Crete in 2002. He is currently pursuing a PhD degree at the Technical University of Crete. His research interests include information retrieval and filtering over wide-area networks, P2P and Grid computing, publish/subscribe systems, and multiagent systems. He has published more than 20 research papers in journals, international conferences, and workshops. His work has been cited by more than 45 research papers. He has received two scholarships from the Greek Ministry of Education and a best student paper award at ECDL 2005. He has also worked as a research assistant in European IST FET projects DIET and Evergrow. More details on his research work can be found at http://www.intelligence.tuc.gr/~trifon.