Interconnection Semantics for Keyword Search in XML

Sara Cohen*
Technion—Israel Institute of Technology
Haifa 32000, Israel
sarac@ie.technion.ac.il

Yaron Kanza
University of Toronto
Toronto, Canada
yaron@cs.toronto.edu

Benny Kimelfeld†
Yehoshua Sagiv†
The Hebrew University
Jerusalem 91904, Israel
{bennyk,sagiv}@cs.huji.ac.il

ABSTRACT

A framework for describing semantic relationships among nodes in XML documents is presented. In contrast to earlier work, the XML documents may have ID references (i.e., they correspond to graphs and not just trees). A specific *interconnection semantics* in this framework can be defined explicitly or derived automatically. The main advantage of interconnection semantics is the ability to pose queries on XML data in the style of keyword search. Several methods for automatically deriving interconnection semantics are presented. The complexity of the evaluation and the satisfiability problems under the derived semantics is analyzed. For many important cases, the complexity is tractable and hence, the proposed interconnection semantics can be efficiently applied to real-world XML documents.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.2 [**Database Management**]: Languages

General Terms

Algorithms

Keywords

Interconnection semantics, keyword search, XML

1. INTRODUCTION

XML facilitates the incorporation of semantic considerations into information-retrieval. The goal of this paper is to investigate how the labels (i.e., tags) of XML documents can be used for determining whether occurrences of keywords are semantically related. We are motivated by the need to develop techniques for keyword search that hide from the user the complexity of the underneath document and, yet, take into account the semantic structure when ranking results. For example, if the keyword *Harris* is labeled with `Employee` and the keyword *Jones* is labeled with `Manager`, one might want to determine whether Jones is the manager of Harris. In most systems that incorporate keyword search into relational or XML data (e.g., [1, 3, 8, 9, 10]), the sole criterion is proximity (e.g., Jones is deemed the manager of Harris if these keywords appear in a small subtree of the given XML document). In [5], it is argued that in a tree document, the keywords are semantically related if they appear in a uniquely-labeled subtree of the document. This approach is extended in [4] by incorporating information-retrieval techniques. In [17], the work of [4, 5] is improved by introducing an approach that avoids some cases of incorrect results.

We generalize and improve all of the above work in two ways. First, we consider XML documents that form arbitrary graphs, due to ID references. Earlier work [4, 5, 17] considered only trees, which are easier to deal with, since a tree document has only one minimal subtree that contains a given set of nodes. Second, we propose several different semantics to overcome the problem of wrong answers. Our approach to solving this problem takes the schema into account and, consequently, can handle gracefully missing information. In comparison, the solution of [17] ignores the schema and applies only to tree documents.

We investigate the problem of semantic relationships in XML documents from several perspectives. First, we propose a framework that allows either users or creators of XML documents to define explicitly *interconnection semantics* for the purpose of specifying how objects (i.e., elements) are semantically related. Second, we explore various types of interconnection semantics that can be derived automatically. Third, we analyze the complexity of two problems. One is the evaluation problem, that is, enumerating all the answers to a given query. The second is the satisfiability problem, namely, testing whether some given objects are semantically related. Algorithms for the latter problem can improve keyword search by incorporating into ranking techniques a test of whether occurrences of keywords are semantically related.

Our complexity analysis considers two measures: data complexity and query-and-data complexity. Since queries are usually of a small size, data complexity is commonly used. However, analyzing the complexity under the assumption that the query has an unbounded size may lead to better, more efficient algorithms; in particular, algorithms that

*The work of this author was supported by the Israel Science Foundation (Grant No. 1032/05).
†The work of these authors was supported by the Israel Science Foundation (Grant No. 96/01).

CIKM'05, October 31–November 5, 2005, Bremen, Germany.

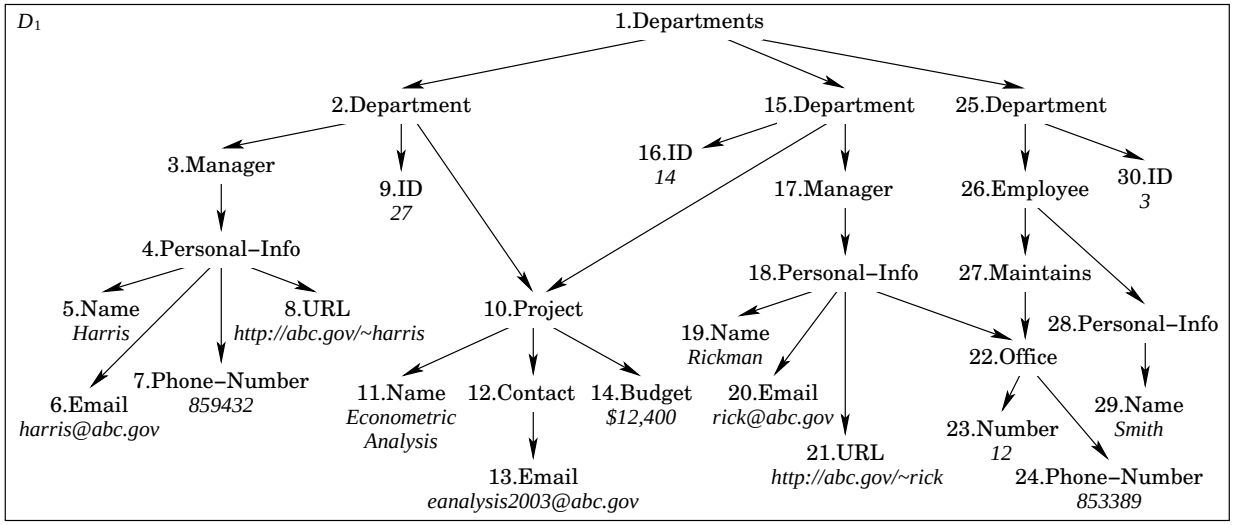

Figure 1: A document D_1

generate quickly the first few answers—an important feature in keyword search.

The main contributions of this paper are as follows. First, we give a wide spectrum of interconnection semantics that apply to XML documents with ID references. In practical terms, these interconnection semantics differ in their efficiency and recall-precision tradeoff. Second, we provide an exhaustive complexity analysis, including algorithms for the tractable cases. We envision that these algorithms can be combined with information-retrieval techniques to yield powerful systems for keyword search in XML documents.

2. PRELIMINARIES

2.1 Graphs and Trees

A *graph* is a pair $G = (V, E)$, where V is a set of nodes and $E \subseteq V \times V$ is a set of edges. A graph is either *directed* or *undirected.* We always denote an edge as (v_1, v_2). If the edge is directed, then (v_1, v_2) is an ordered pair; otherwise, it is an unordered pair (i.e., (v_1, v_2) is the same as (v_2, v_1)).

A *rooted graph* is a directed graph that has a designated node r, called the *root,* such that every node v is reachable from r by a directed path.

We use two types of *trees.* A *rooted tree* is a rooted graph, such that for every node v, there is a unique directed path from the root to v. An *undirected tree* is a connected undirected graph without cycles (and without a root). Note that when we say "tree," we simultaneously refer to both types.

$G' = (V', E')$ is a *subgraph* of $G = (V, E)$, denoted $G' \subseteq G$, if $V' \subseteq V$ and $E' \subseteq E \cap (V' \times V')$. A rooted subgraph G' of a rooted graph G need not have the same root as G. A *rooted subtree* is a special case of a rooted subgraph. We also consider *undirected subtrees* of rooted graphs by ignoring the directions of the edges.

If G_1 and G_2 are subgraphs of G, we use $G_1 \cup G_2$ to denote the subgraph that consists of all the nodes and edges of either G_1 or G_2.

A rooted tree (respectively, undirected tree) $T = (V, E)$ is *reduced* with respect to (abbr., w.r.t.) a subset $U \subseteq V$ if it has no proper rooted subtree (respectively, undirected subtree) that includes all the nodes of U.

2.2 O-Graphs and L-Graphs

An *o-graph* (and an *o-tree*) has *objects* as nodes. An object has an *object identifier* (abbr. oid) and is assigned a *label* and, possibly, a *value.* Figure 1 shows a rooted o-graph D_1, where integers are used for oid's, each node has a label, and all the leaves have values. If o is an object, then $l(o)$ denotes the label of o. Similarly, if O is a set of objects, then $l(O)$ denotes the set $\{l(o) | o \in O\}$. An o-tree is *uniquely labeled* if distinct objects do not have the same label.

An *l-graph* (and an *l-tree*) has labels as nodes. Figure 2 shows a rooted l-graph S_1, two rooted l-trees C_1 and C_2 and an undirected l-tree C_3. The l-trees C_1 and C_2 are reduced w.r.t. the set of labels $\{\texttt{Email}, \texttt{Name}\}$, and the l-tree C_3 is reduced w.r.t. $\{\texttt{Employee}, \texttt{Name}\}$ (and also w.r.t. some other sets of labels, e.g., $\{\texttt{Employee}, \texttt{Name}, \texttt{Maintains}\}$). Note that a label occurs (at most) once in an l-graph, but may have multiple occurrences in an o-graph.

A rooted o-graph D *conforms* to a rooted l-graph S if the following two conditions hold:

- $l(r)$ is the root of S, where r is the root of D, and
- If (o_1, o_2) is an edge of D, then $(l(o_1), l(o_2))$ is an edge of S.

For example, the rooted o-graph D_1 of Figure 1 conforms to the rooted l-graph S_1 of Figure 2.

A rooted o-tree T is *isomorphic* to a rooted l-tree C if T is uniquely labeled, the labels of T are exactly the nodes in C, and T conforms to C. For example, the rooted subtree of D_1 (Figure 1) that comprises objects 4, 5 and 6 is isomorphic to the rooted l-tree C_1 (Figure 2).

For undirected trees, the condition about the roots is removed from the definition of isomorphism. That is, an undirected o-tree T is *isomorphic* to an undirected l-tree C if T is uniquely labeled, the labels of T are exactly the nodes in C, and for every edge (o_1, o_2) of T, the edge $(l(o_1), l(o_2))$ is in C.

2.3 Documents and Schemas

A *document* is a rooted o-graph and a *schema* is a rooted l-graph. Figure 1 shows a document D_1 and Figure 2 shows a schema S_1.

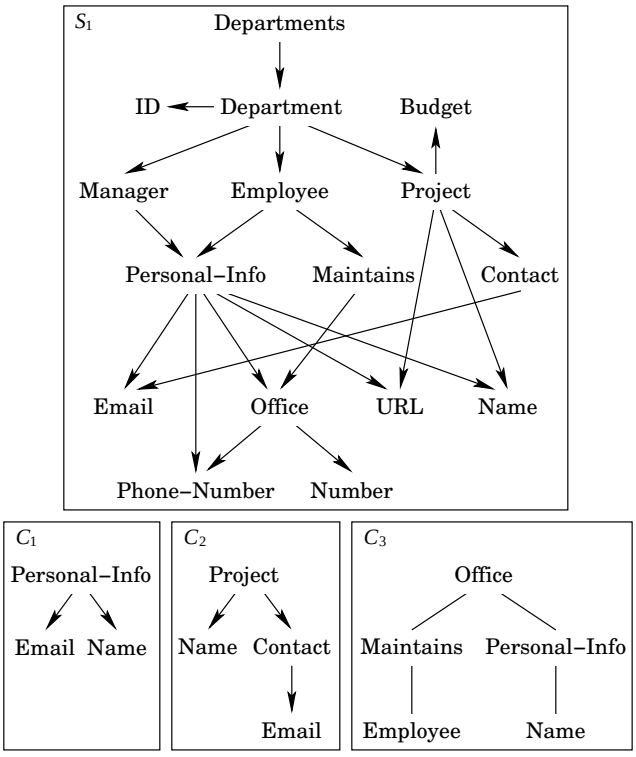

Figure 2: A schema S_1, rooted l-trees C_1 and C_2, and an undirected l-tree C_3

A document D may have a user-supplied schema. Alternatively, we can use the *derived schema* of D, denoted S_D, that consists of the following. The root $l(r)$, where r is the root of D, the labels of D as nodes, and all the edges $(l(o_1), l(o_2))$, where (o_1, o_2) is an edge of D. Note that S_D is the minimal schema that D conforms to; that is, if D conforms to S, then S_D is a rooted subgraph of S and both have the same root. As an example, the derived schema for D_1 (Figure 1) is obtained from the schema S_1 (Figure 2) by removing the edge from the label `Project` to the label `URL` (since D_1 has no edge from a node labeled with `Project` to a node labeled with `URL`).

In the remainder of this paper, when we consider a document D with a schema S, we implicitly assume that D conforms to S.

3. INTERCONNECTION SEMANTICS

3.1 Patterns

Informally, an interconnection semantics is a collection $\mathcal{P}$ of queries. We refer to these queries as *patterns* and they will be formally defined shortly. A set of nodes O, in a given document D, is interconnected if it is in the result of applying some pattern of $\mathcal{P}$ to D. We follow the principle that patterns of $\mathcal{P}$ should represent only the basic semantic relationships that exist in D, as opposed to relationships that can be obtained from the basic ones by composition, additional processing, etc. In the parlance of relational databases, it means two things. First, a pattern of $\mathcal{P}$ should not have self joins, e.g., it should not represent the manager of the manager of an employee. Second, a pattern of $\mathcal{P}$ should not join a pair of attributes in two distinct ways, e.g., the pattern could join an employee and a manager by using either the relationship between employees and project managers or the relationship between employees and departments managers, but not by using both relationships. Translating these requirements to documents means that a subgraph of a document D represents a basic semantic relationship if it is a uniquely labeled subtree of D.

Formally, a *rooted* (respectively, *undirected*) *pattern* is a pair $p = (L, C)$, where L is a set of labels and C is a rooted (respectively, undirected) l-tree that is reduced w.r.t. L. Note that L includes (at least) all the leaves of C.

Intuitively, a pattern describes when objects having the labels of L are semantically related. For example, the rooted patterns $(\{\texttt{Name,Email}\}, C_1)$ and $(\{\texttt{Name,Email}\}, C_2)$, where C_1 and C_2 are shown in Figure 2, describe when two objects labeled with `Name` and `Email` are interconnected in the document D_1 (shown in Figure 1).

Formally, let O be a set of objects appearing in a document D. A rooted (respectively, undirected) pattern $p = (L, C)$ *interconnects* O in D, denoted $p \models_D O$, if $l(O) = L$ and D has a rooted (respectively, undirected) subtree T, such that T includes all the objects of O and is isomorphic to C. Note that a pattern can only interconnect a set of uniquely-labeled objects (and, furthermore, T must be uniquely labeled).

As an example, consider again D_1, C_1 and C_2. The rooted pattern $(\{\texttt{Name,Email}\}, C_1)$ interconnects the pair of objects $(5, 6)$ and also the pair $(19, 20)$, while the rooted pattern $(\{\texttt{Name,Email}\}, C_2)$ interconnects the pair $(11, 13)$.

Some relationships cannot be captured by any rooted pattern. For example, in document D_1, object 19 and object 26 might be deemed meaningfully related, since they are the name of a manager and the employee that maintains the office of that manager, respectively. Yet, there is no uniquely-labeled rooted subtree of D_1 that contains both objects 19 and 26; therefore, these two objects are not interconnected by any rooted pattern. Objects 19 and 26, however, are interconnected by the undirected pattern $(\{\texttt{Name,Employee}\}, C_3)$, where C_3 is the undirected l-tree shown in Figure 2.

An *interconnection semantics* is a set $\mathcal{P}$ of patterns. We say that $\mathcal{P}$ is *rooted* (respectively, *undirected*) if all its patterns are rooted (respectively, undirected). In principle, an interconnection semantics can also be *mixed,* i.e., have both rooted and undirected patterns.

Let D be a document, O be a subset of the objects of D, and $\mathcal{P}$ be an interconnection semantics. We say that O is *$\mathcal{P}$-interconnected*, denoted $\mathcal{P} \models_D O$, if $\mathcal{P}$ contains a pattern p, such that $p \models_D O$.

3.2 Queries and Their Complexity

In our framework, a *query* is a set L of labels. Given a document D and an interconnection semantics $\mathcal{P}$, an *answer* to a query L is a set O of objects, such that $l(O) = L$ and $\mathcal{P} \models_D O$. The *evaluation* problem is essentially an *enumeration* problem, that is, all the answers to a given query have to be generated successively. The *satisfiability* problem is that of deciding whether a given set O of objects is interconnected, i.e., whether $\mathcal{P} \models_D O$.

Usually, efficiency of query evaluation is measured in terms of *data complexity* [20]; that is, the query is assumed to be of a fixed size. The measure of *query-and-data complexity* [20] means that the query (as well as the document) has an un-

bounded size. Consequently, the number of answers could be exponential in the input size and therefore, the running time should take into consideration both the input size and the output size.

Three complexity classes for enumeration problems are proposed in [12]. *Polynomial total time* means that the running time is polynomial in the combined size of the input and the output. *Incremental polynomial time* means that the time needed to generate the ith answer, after the first $i-1$ answers have already been produced, is polynomial in the combined size of the input and the first $i-1$ answers. The most efficient notion is enumeration with *polynomial delay*, that is, the running time between the generation of two consecutive answers is polynomial in the input size.

In this paper, we analyze the complexity of both the evaluation problem and the satisfiability problem under the two measures mentioned above. Data complexity means that the set O, in the satisfiability problem, and the query L, in the evaluation problem, are of a fixed size. Query-and-data complexity means that the sizes of O and L are unbounded. Note that under data complexity, if the satisfiability problem is in polynomial time, then so is the evaluation problem. Also note that the evaluation problem cannot be in polynomial total time if the *non-emptiness* problem is intractable, where the latter is the problem of deciding whether a given query has at least one answer.

If the interconnection semantics $\mathcal{P}$ is given explicitly, then it is part of the input. Since patterns are essentially projections of acyclic joins, the following proposition follows from the work of [21].

PROPOSITION 3.1. *Let $\mathcal{P}$ be an explicit interconnection semantics that is part of the input. The following results hold under query-and-data complexity. The satisfiability problem is in polynomial time. The evaluation problem is in polynomial delay if $\mathcal{P}$ has (at most) one pattern for L; otherwise, it is in incremental polynomial time.*

4. DERIVED SEMANTICS

Expressing explicitly all the patterns of an interconnection semantics is not always convenient or practical. As an alternative, we present several interconnection semantics that can be derived automatically from the schema S of the given document D. These semantics may depend on the specific schema S that is used for D. Note that S can be any schema that D conforms to, including the derived schema S_D.

There are two approaches for solving the satisfiability and the evaluation problems when the interconnection semantics is derived from the schema. One is to generate all the patterns that are relevant to the set of labels at hand, thereby obtaining an explicit semantics. This approach is discussed in Section 5. The second approach is to find algorithms that can accept, as part of the input, the schema S rather than the derived semantics itself. In this section, we define the derived interconnection semantics and explore the second approach. Specifically, we give algorithms for the tractable cases. Lower bounds for the intractable cases are summarized in Section 7.

4.1 The Semantics $\mathcal{P}^r_{\mathsf{all}}(S)$ and $\mathcal{P}^u_{\mathsf{all}}(S)$

Given a document D conforming to a schema S, the interconnection semantics $\mathcal{P}^r_{\mathsf{all}}(S)$ comprises all rooted patterns (L, C), such that L is a set of labels appearing in S and C is a rooted subtree of S that is reduced w.r.t. L. It is easy to show that a set O of objects is $\mathcal{P}^r_{\mathsf{all}}(S)$-interconnected in D if and only if D has a uniquely-labeled rooted subtree T that contains O. Thus, determining interconnectivity does not depend on the specific schema that is used for D; that is, $\mathcal{P}^r_{\mathsf{all}}(S) \models_D O$ if and only if $\mathcal{P}^r_{\mathsf{all}}(S_D) \models_D O$. Hence, by a slight abuse of notation, we may sometimes write $\mathcal{P}^r_{\mathsf{all}}$ instead of $\mathcal{P}^r_{\mathsf{all}}(S)$.

$\mathcal{P}^r_{\mathsf{all}}$-ARRAY$(D, k)$

```
1: let l_1, ..., l_m be the labels of S_D, sorted
   in any topological order
2: let o_1, ..., o_n be the objects of D, sorted
   in the order implied on their labels
3: for all O ⊆ {o_1, ..., o_n} s.t. |O| ≤ k do
4:    if |O| = 1 then I[O] := 0
5:    else I[O] := ∞
6: for i := 2, ..., n do
7:    for all O ⊆ {o_1, ..., o_{i-1}} such that
      1 ≤ |O| < k do
8:       O' := O ∪ {o_i}
9:       if O' is uniquely labeled then
10:         for all parents o of o_i in D do
11:            I[O'] = min(I[O'], I[O ∪ {o}] + 1)
12: return I
```

Figure 3: Enumerating results under the semantics $\mathcal{P}^r_{\mathsf{all}}$

The interconnection semantics $\mathcal{P}^u_{\mathsf{all}}(S)$ is the set of all undirected patterns (L, C), such that L is a set of labels appearing in S and C is an undirected subtree of S that is reduced w.r.t. L. If $\mathcal{P}^r_{\mathsf{all}}(S) \models_D O$, then $\mathcal{P}^u_{\mathsf{all}}(S) \models_D O$. The advantage of $\mathcal{P}^u_{\mathsf{all}}(S)$ is the ability to interconnect objects even when they are not part of a rooted hierarchy, as shown earlier for document D_1 using the undirected pattern $(\{\texttt{Name,Employee}\}, C_3)$.

We now show that the satisfiability problem for $\mathcal{P}^r_{\mathsf{all}}(S_D)$ is in polynomial time under data complexity if S_D is acyclic. The algorithm of Figure 3 uses dynamic programming in order to decide whether $\mathcal{P}^r_{\mathsf{all}}(S_D) \models_D O$, for all sets O of objects in a given document D, such that $|O| \leq k$. The output is given by the array I. Upon termination, the value of $I[O]$ is the size of a minimal rooted subtree $C \subseteq S_D$, such that $(l(O), C) \models_D O$; if $\mathcal{P}^r_{\mathsf{all}}(S_D) \not\models_D O$, then $I[O] = \infty$. Note that the values of $I[O]$ could be used for ranking the results based on proximity, since a smaller value means that the objects of O are included in a smaller subtree of D. Furthermore, the algorithm could be extended to take into account an arbitrary weight function on the edges, thereby enabling a wider range of ranking functions.

In Lines 3–5, $I[O]$ is initialized either to ∞, if $1 < |O| \leq k$, or to 0, if $|O| = 1$. In Lines 6–11, the objects of D are traversed according to a topological order on their labels, as implied by S_D. Let $o_1, \ldots, o_n$ be the objects of D, sorted in that order. When object o_i is visited, we consider all $I[O \cup \{o_i\}]$, where $O \subseteq \{o_1, \ldots, o_{i-1}\}$ and $|O| < k$. If $O \cup \{o_i\}$ has a repeated label, then $I[O \cup \{o_i\}]$ is left unchanged. Otherwise, $I[O \cup \{o_i\}]$ is assigned the minimal value among all $I[O \cup \{o\}] + 1$, where o is some parent of o_i in D. Note that all the parents of an object precede that object in the order $o_1, \ldots, o_n$. Hence, for every parent o of o_i, the value of $I[O \cup \{o\}]$ is already determined when o_i is visited.

THEOREM 4.1. *Let D be a document with an acyclic S_D, and let O be a set of objects of D, such that $|O| \leq k$. If $\mathcal{P}^r_{\mathsf{all}}(S_D) \models_D O$, then the final value of $I[O]$ is the size of a minimal rooted subtree of D that contains O and is uniquely-labeled; otherwise, $I[O] = \infty$. The running time of $\mathcal{P}^r_{\mathsf{all}}$-ARRAY$(D, k)$ is $\mathcal{O}(mn^{k-1})$, where n and m are the number of objects and edges of D, respectively.*

COROLLARY 4.2. *For the semantics $\mathcal{P}^r_{\mathsf{all}}(S)$, the satisfiability and the evaluation problems are solvable in polynomial time, under data complexity, if S_D is acyclic.*

4.2 The Semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$

The semantics $\mathcal{P}^r_{\mathsf{all}}$ and $\mathcal{P}^u_{\mathsf{all}}$ occasionally interconnect objects that are rather weakly related to each other. For example, in document D_1 (Figure 1), object 11 (the name of a project) and object 8 (the URL of a manager) are $\mathcal{P}^r_{\mathsf{all}}(S_1)$-interconnected, where S_1 is given in Figure 2.

The above problem can be avoided by adopting the common assumption (e.g., [2]) that meaningfully-related objects must be close to each other. Thus, for a given schema S, we define the notion of *minimal patterns* as follows. A rooted (respectively, undirected) pattern $p = (L, C)$ is *minimal* w.r.t. S if C is a minimal-size rooted (respectively, undirected) subtree of S that includes all the labels of L. Note that since C is minimal, it is also reduced w.r.t. L.

For example, consider S_1, C_1 and C_2 from Figure 2. The pattern $(\{\texttt{Name,Email}\}, C_1)$ is minimal w.r.t. S_1, but the pattern $(\{\texttt{Name,Email}\}, C_2)$ is not.

Formally, given a schema S, the interconnection semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ is the set of all rooted patterns that are minimal w.r.t. S. Note that defining $\mathcal{P}^r_{\mathsf{min}}(S)$ in terms of a schema S is not a limitation, because the derived schema can always be used if no schema is explicitly given.

The semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ gracefully handles missing information. Consider, for example, document D_1 (Figure 1) and schema S_1 (Figure 2). The fact that S_1 has an edge from `Project` to `URL` implies that the URL of object 10 is missing from D_1. This is realized by the semantics $\mathcal{P}^r_{\mathsf{min}}(S_1)$ that does not interconnect object 10 with any object labeled with `URL`, because the only minimal rooted pattern for the set of labels $\{\texttt{Project,URL}\}$ is $(\{\texttt{Project,URL}\}, C)$, where C is the l-tree comprising the single edge from `Project` to `URL`.

The interconnection semantics $\mathcal{P}^u_{\mathsf{min}}(S)$ is defined similarly to $\mathcal{P}^r_{\mathsf{min}}(S)$; that is, $\mathcal{P}^u_{\mathsf{min}}(S)$ is the set of all undirected patterns that are minimal w.r.t. S. The semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$ are incomparable; that is, for some documents and schemas, there are sets of objects that are $\mathcal{P}^r_{\mathsf{min}}(S)$-interconnected but not $\mathcal{P}^u_{\mathsf{min}}(S)$-interconnected and vice-versa. Unlike $\mathcal{P}^r_{\mathsf{all}}(S)$ and $\mathcal{P}^u_{\mathsf{all}}(S)$, both $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$ have a tractable data complexity (even if S is cyclic).

Testing $\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$ is done by computing the sizes n_1 and n_2 of the minimal rooted subtree of D that contains O and the minimal rooted subtree of S that contains $l(O)$, respectively. It is easy to show that $\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$ if and only if $n_1 = n_2$. The numbers n_1 and n_2 can be computed by adapting a known algorithm [6] for finding Steiner trees to directed graphs. Testing $\mathcal{P}^u_{\mathsf{min}}(S) \models_D O$ is similar.

THEOREM 4.3. *$\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$ and $\mathcal{P}^u_{\mathsf{min}}(S) \models_D O$ can be tested in $\mathcal{O}\big(n_s m_s + n_d m_d + (n_s + n_d)3^2 2^{|O|} + (n_s + n_d)3^{|O|}\big)$ time, where n_s and m_s are the number of labels and edges of S, respectively, and n_d and m_d are the number of objects and edges of D, respectively.*

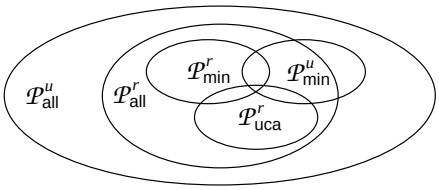

Figure 4: A Venn diagram for the derived semantics

COROLLARY 4.4. *For the semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$, the satisfiability and the evaluation problems are solvable in polynomial time under data complexity.*

4.3 The Semantics $\mathcal{P}^r_{\mathsf{uca}}(S)$

The semantics $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$ may occasionally miss meaningfully related objects just because they are slightly too far away from each other. For example, consider document D_1 (Figure 1), schema S_1 and the rooted l-trees C_1 and C_2 (Figure 2). The semantics $\mathcal{P}^r_{\mathsf{min}}(S_1)$ does not include the pair $(11, 13)$ as an answer for the set of labels $\{\texttt{Name,Email}\}$, because the pattern $(\{\texttt{Name,Email}\}, C_2)$ is not minimal w.r.t. S_1. Dealing with this problem requires a different notion of minimal rooted subtrees—one that allows to define both $(\{\texttt{Name,Email}\}, C_1)$ and $(\{\texttt{Name,Email}\}, C_2)$ as minimal patterns. This is done as follows.

Consider an acyclic schema S and a set of labels L. A node l of S is a *common ancestor* of L in S if every label of L is reachable from l via a directed path in S. A rooted pattern $p = (L, C)$, where C is a rooted subtree of S, is *structurally minimal* w.r.t. S if the following holds. C does not have any node, other than the root, that is a common ancestor of L in S. For example, the rooted patterns $(\{\texttt{Name,Email}\}, C_1)$ and $(\{\texttt{Name,Email}\}, C_2)$ are structurally minimal w.r.t. the schema S_1 (Figure 2); in fact, these are the only two rooted patterns for the set of labels $\{\texttt{Name,Email}\}$ that are structurally minimal w.r.t. S_1.

Given an acyclic schema S, the interconnection semantics $\mathcal{P}^r_{\mathsf{uca}}(S)$ (where `uca` stands for *unique common ancestor*) is the set of all rooted patterns that are structurally minimal w.r.t. S. Note that this semantics is only defined for acyclic schemas, since it may yield rather strange results when the schema is cyclic. Clearly, the interconnection semantics $\mathcal{P}^r_{\mathsf{uca}}$ does not have an analogous undirected semantics.

For document D_1 (Figure 1) and schema S_1 (Figure 2), the answers to the query $\{\texttt{Name,Email}\}$, under the semantics $\mathcal{P}^r_{\mathsf{uca}}(S_1)$, are the pairs $(5, 6)$, $(19, 20)$ and $(11, 13)$.

To show that $\mathcal{P}^r_{\mathsf{uca}}$ has a polynomial data complexity, we reduce $\mathcal{P}^r_{\mathsf{uca}}$-interconnectivity to $\mathcal{P}^r_{\mathsf{all}}$-interconnectivity as follows. Let D be a document that conforms to an acyclic schema S. Given an instance $\mathcal{P}^r_{\mathsf{uca}}(S) \models_D O$ of the satisfiability problem, let A be the set of all common ancestors of $l(O)$ in S. For a label $l \in A$, document D_l is obtained from D by deleting all objects o, such that $l(o) \in A$ and $l(o) \neq l$; in addition, a new root may have to be added to D_l. It can be shown that $\mathcal{P}^r_{\mathsf{uca}}(S) \models_D O$ if and only if there is an $l \in A$, such that $\mathcal{P}^r_{\mathsf{all}} \models_{D_l} O$. The following is therefore a consequence of Theorem 4.1.

COROLLARY 4.5. *Let S be an acyclic schema. For the semantics $\mathcal{P}^r_{\mathsf{uca}}(S)$, the satisfiability and the evaluation problems are solvable in polynomial time under data complexity.*

```
SteinerTrees(G, U)
 1: if no rooted subtree of G contains U then
 2:    return ∅
 3: for all minimal topologies τ of U do
 4:    MaterializeTopology(G, τ)
MaterializeTopology(G, τ)
 1: if τ contains only one node v then
 2:    return ({v}, ∅)
 3: arbitrarily choose a leaf v and an edge (u, v) of τ
 4: T1 := MaterializeTopology(G, τ − v)
 5: T2 := MinimalPaths(G, u, v)
 6: T := ∅
 7: for all T1 ∈ T1 do
 8:    for all T2 ∈ T2 do
 9:       T := T ∪ {T1 ∪ T2}
10: return T
```

Figure 5: Enumerating Steiner trees

4.4 Comparing the Derived Semantics

Figure 4 shows when an interconnection under one semantics is implied by an interconnection under another semantics. Note that the semantics $\mathcal{P}^r_{\mathsf{min}}(S)$, $\mathcal{P}^u_{\mathsf{min}}(S)$ and $\mathcal{P}^r_{\mathsf{uca}}(S)$ are pairwise incomparable. For tree documents, $\mathcal{P}^r_{\mathsf{min}}(S)$-interconnectivity is implied by $\mathcal{P}^u_{\mathsf{min}}(S)$-interconnectivity. In a case study we conducted, $\mathcal{P}^r_{\mathsf{uca}}(S)$-interconnectivity was usually implied by $\mathcal{P}^r_{\mathsf{min}}(S)$-interconnectivity.

A special case is when the schema (but not necessarily the document) is a tree. Given a tree schema S and any subset L of the labels of S, there is exactly one rooted subtree of S that is reduced w.r.t. L. Moreover, that rooted subtree coincides with the only undirected subtree of S that is reduced w.r.t. L. Therefore, for all the interconnection semantics considered earlier in this section, the same set of patterns is derived from S. This observation and Proposition 3.1 lead to the following.

Corollary 4.6. *Let D be a document with a tree schema S. For a set O of an unbounded size, the following are equivalent and can be tested in polynomial time: $\mathcal{P}^r_{\mathsf{all}}(S) \models_D O$, $\mathcal{P}^u_{\mathsf{all}}(S) \models_D O$, $\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$, $\mathcal{P}^u_{\mathsf{min}}(S) \models_D O$ and $\mathcal{P}^r_{\mathsf{uca}}(S) \models_D O$. Moreover, under query-and-data complexity, the evaluation problem is solvable with polynomial delay.*

5. ENUMERATING PATTERNS

We conducted a case study of several well-known XML documents and their DTDs (e.g., RSS, DBLP, Mondial and Shakespeare Plays) and concluded that the number of patterns, for a given set of labels, is usually not large (although in some cases this number could be very large). This observation has led us to an alternative approach to solving the satisfiability and the evaluation problems. In this approach, we first generate the patterns that correspond to the labels of the given query (or the given objects) and then employ Proposition 3.1. To make this approach practical, we need efficient algorithms for enumerating the relevant patterns.

In general, we are given a document D that conforms to a schema S and a set of labels L, where L is either a query or comprises the labels of a given set of objects. The goal is to generate all the patterns of the form (L, C), such that (L, C) is in a given derived interconnection semantics $\mathcal{P}$.

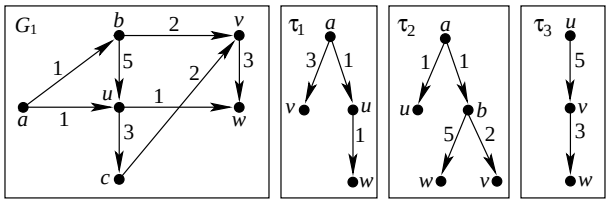

Figure 6: A graph G_1 and three topologies τ_1, τ_2 and τ_3

We start with the semantics $\mathcal{P}^r_{\mathsf{min}}$. The problem at hand is to enumerate all the minimal-size rooted subtrees of S that contain L; these subtrees are known as the *Steiner trees* of L. Traditionally, the Steiner-tree problem [11] is to find one such tree. We show how to enumerate all Steiner trees of L in polynomial total time, assuming that L has a fixed size.

The input is a weighted, directed graph $G = (V, E, w)$ and a subset $U \subseteq V$, where U is of a fixed size (note that w is a positive weight function defined on E). First, we define a key notion as follows. A *topology* of U in G is a weighted, rooted tree $\tau = (V_\tau, E_\tau, w_\tau)$, such that *(1)* $U \subseteq V_\tau \subseteq V$, *(2)* every node $v \in V_\tau \setminus U$ has at least two outgoing edges in τ, *and (3)* for all edges $e = (u, v) \in E_\tau$, the weight $w_\tau(e)$ is equal to the weight of a minimal path of G from u to v; if there is no such path, then $w_\tau(e) = \infty$. The topology τ of U is *minimal* if its weight is the minimum among all topologies of U. For example, Figure 6 depicts a graph G_1 and three topologies of $\{u, v, w\}$. The topology τ_1 is minimal.

It can be shown that a topology of U has at most $2|U| - 1$ nodes. Therefore, all the topologies of U in G can be found in polynomial time, since U has a fixed size. The algorithm SteinerTrees(G, U) of Figure 5 enumerates the Steiner trees of U in G by considering each minimal topology τ of U and replacing every edge $e = (u, v)$ of τ with some minimal-weight path of G from u to v. The replacements are done by calling MaterializeTopology(G, τ). Note that, in Line 4, MaterializeTopology calls itself recursively, on a smaller topology, i.e., a leaf v is removed from τ. For every materialized topology T_1 that is returned by the recursive call, Lines 7–9 perform all possible replacements of the edge (u, v) with some minimal path T_2. The result of each replacement is a Steiner tree $T_1 \cup T_2$, comprising the nodes and edges of T_1 and T_2, and it is added to $\mathcal{T}$. Due to its simplicity, we omit the description of the algorithm MinimalPaths(G, u, v) for enumerating minimal-weight paths.

Theorem 5.1. *The algorithm* SteinerTrees(G, U) *enumerates all Steiner trees of U in G in polynomial total time.*

The running time can be improved to polynomial delay by using coroutines, as done in [16]. Essentially, the same algorithm applies also to undirected graphs. Thus, for both $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$, all patterns (L, C), for a given L, can be enumerated with polynomial delay.

Patterns can be enumerated with polynomial delay also in the case of $\mathcal{P}^r_{\mathsf{all}}(S)$, $\mathcal{P}^u_{\mathsf{all}}(S)$ and $\mathcal{P}^r_{\mathsf{uca}}(S)$ by using enumeration algorithms described in [16] and, in the case of $\mathcal{P}^r_{\mathsf{uca}}(S)$, combining an algorithm of [16] with the reduction of $\mathcal{P}^r_{\mathsf{uca}}(S)$ to $\mathcal{P}^r_{\mathsf{all}}(S)$ that is described in Section 4.3. Note that the algorithms for enumerating patterns of $\mathcal{P}^r_{\mathsf{all}}(S)$, $\mathcal{P}^u_{\mathsf{all}}(S)$ and $\mathcal{P}^r_{\mathsf{uca}}(S)$ do not assume that L has a fixed size (in contrast to the enumeration algorithms for $\mathcal{P}^r_{\mathsf{min}}(S)$ and $\mathcal{P}^u_{\mathsf{min}}(S)$). There are also algorithms that enumerate patterns of $\mathcal{P}^r_{\mathsf{all}}(S)$, $\mathcal{P}^u_{\mathsf{all}}(S)$ and $\mathcal{P}^r_{\mathsf{uca}}(S)$ in increasing weight and with polynomial

delay, provided that the size of L is fixed. Therefore, if ranking is based on proximity, one can use these algorithms in order to generate query results in a ranked order. See [15] for more details, including the notion of enumerating patterns in an *approximate* order.

In summary, patterns can be enumerated quickly if there are not too many of them. In any case, since the enumeration is with polynomial delay (in the schema size), the stream of enumerated patterns has a fast flow and the patterns can be used to compute query answers as soon as they are generated. Another advantage is the ability to translate patterns to an XML query language, such as XQuery.

6. UNIVERSAL INTERCONNECTIONS

Interconnections semantics have been applied thus far in an existential manner; that is, one evidence for interconnection is sufficient. It is also possible to apply interconnection semantics universally by requiring that there will be an evidence of interconnection in every *context* implied by the document. For example, in document D_1 (Figure 1), objects 5 and 13 are $\mathcal{P}^r_{\mathsf{all}}$-interconnected, because there is a rooted subtree of D_1 (with object 2 as its root) that is uniquely labeled and reduced w.r.t. the set of objects $\{5, 13\}$. There is also a second rooted subtree (with object 1 as its root) that is reduced w.r.t. $\{5, 13\}$. In the context of the second subtree, however, there is no evidence that objects 5 and 13 are $\mathcal{P}^r_{\mathsf{all}}$-interconnected, because that subtree has two objects, 2 and 15, that are labeled with `Department`.

A universal application of an interconnection semantics $\mathcal{P}$ depends on how we define the possible contexts of a set O of objects in a document D. Formally, $\mathit{contexts}_D(O)$ is a nonempty set of subtrees of D that are reduced w.r.t. O (it is natural to consider only reduced subtrees as possible contexts, since they have no redundant parts as far as the set O is concerned). The exact definition of $\mathit{contexts}_D(O)$ may depend on $\mathcal{P}$. Intuitively, O is universally interconnected if every subtree in $\mathit{contexts}_D(O)$ is an evidence of interconnection. Formally, we say that O is *universally $\mathcal{P}$-interconnected* in D, denoted $\mathcal{P} \models_{\forall,D} O$, if for every subtree $T \in \mathit{contexts}_D(O)$, there is a pattern $(l(O), C) \in \mathcal{P}$, such that T is isomorphic to C. Note that this definition cannot be satisfied vacuously, since $\mathit{contexts}_D(O)$ is nonempty.

In this section, we discuss universal interconnectivity under the semantics $\mathcal{P}^r_{\mathsf{all}}$ and $\mathcal{P}^u_{\mathsf{all}}$, and we use the following definition of $\mathit{contexts}_D(O)$. For a rooted (respectively, undirected) interconnection semantics $\mathcal{P}$, we define $\mathit{contexts}_D(O)$ to be the set of all rooted (respectively, undirected) subtrees of D that are reduced w.r.t. O. This definition implies that O is universally $\mathcal{P}^r_{\mathsf{all}}$-interconnected (respectively, $\mathcal{P}^u_{\mathsf{all}}$-interconnected) in D if and only if there is no rooted (respectively, undirected) subtree T of D, such that T is reduced w.r.t. O but not uniquely labeled. Hence, $\mathcal{P}^r_{\mathsf{all}}$, $\mathcal{P}^u_{\mathsf{all}}$ and their universal versions are all equivalent on tree documents (but not on documents conforming to tree schemas).

Consider document D_1 of Figure 1 and let L be the set of labels {`Name`,`Email`}. The set $\{(5, 6), (19, 20), (11, 13)\}$ comprises all pairs of objects for L that are universally $\mathcal{P}^r_{\mathsf{all}}$-interconnected. This set also comprises all pairs of objects for L that are universally $\mathcal{P}^u_{\mathsf{all}}$-interconnected in D_1. As another example, objects 15 (a department) and object 10 (a project) are not universally $\mathcal{P}^r_{\mathsf{all}}$-interconnected, because the project belongs to more than one department.

The following important property is instrumental in developing efficient algorithms for testing universal interconnectivity. A set of objects O is universally $\mathcal{P}^u_{\mathsf{all}}$-interconnected if and only if every pair of objects from O is universally $\mathcal{P}^u_{\mathsf{all}}$-interconnected. Interestingly, a similar result for universal $\mathcal{P}^r_{\mathsf{all}}$-interconnectivity holds only if the document is acyclic. The next theorem shows that for the interconnection semantics $\mathcal{P}^r_{\mathsf{all}}$ and $\mathcal{P}^u_{\mathsf{all}}$, universal interconnectivity is more tractable than interconnectivity.

TSEnumeration$(D, L, \mathcal{P})$

1: let τ be the minimal topology of L in S_D
2: let $\hat{V}$ be the set of all objects in D with labels from τ
3: $\hat{E} := \{(o_1, o_2) \mid o_1, o_2 \in \hat{V},\ (l(o_1), l(o_2))$ is an edge in τ and $\mathcal{P} \models_{\forall,D} \{o_1, o_2\}\}$
4: let $\hat{G}$ be the o-graph $(\hat{V}, \hat{E})$
5: let l_τ be the root of τ
6: add to $\hat{G}$ a new object r that has a new label
7: add to $\hat{G}$ edges from r to all objects labeled with l_τ
8: let $\hat{D}$ be the subgraph of $\hat{G}$ that is induced by all the objects that are reachable from the root r
9: enumerate all sets O s.t. $l(O) = L$ and $(L, \tau) \models_{\hat{D}} O$

Figure 7: Enumerating results under universal semantics

THEOREM 6.1. *The satisfiability problem is in polynomial time under query-and-data complexity in the following two cases. First, the semantics is universal $\mathcal{P}^u_{\mathsf{all}}$. Second, the semantics is universal $\mathcal{P}^r_{\mathsf{all}}$ and the document is acyclic.*

Obviously, the proof uses the above property. However, it also needs algorithms for testing universal interconnectivity under $\mathcal{P}^r_{\mathsf{all}}$ and $\mathcal{P}^u_{\mathsf{all}}$ when O has only two objects. The details of these algorithms are rather intricate and follow from a close connection between our problems and the subgraph homeomorphism problem [7, 13, 19]. Interestingly, if S_D is acyclic and D has n objects and m edges, then all the pairs of objects that are universally $\mathcal{P}^r_{\mathsf{all}}$-interconnected can be found in $\mathcal{O}(nm)$-time. Thus, after this preprocessing, $\mathcal{P}^r_{\mathsf{all}} \models_{\forall,D} O$ can be tested in $\mathcal{O}(|O|^2)$ time.

For documents that conform to tree schemas and for the universal version of either $\mathcal{P}^r_{\mathsf{all}}$ or $\mathcal{P}^u_{\mathsf{all}}$, all the answers to a query L can be enumerated with polynomial delay. The algorithm TSEnumeration$(D, L, \mathcal{P})$ of Figure 7 enumerates all subsets O of the objects of D, such that $l(O) = L$ and $\mathcal{P} \models_{\forall,D} O$, where P is either $\mathcal{P}^r_{\mathsf{all}}$ or $\mathcal{P}^u_{\mathsf{all}}$. It is assumed that the derived schema S_D is a tree. Furthermore, the labels of L appear in S_D (otherwise, the result is empty).

The algorithm is a reduction to the evaluation problem under ordinary (i.e., existential) $\mathcal{P}$-interconnectivity. The first step of the algorithm is to create the minimal topology τ of L in S_D (based on the definition of a topology in Section 5). Note that since S_D is a tree, the minimal topology is unique and can be found efficiently. Next, $\hat{G}$ is the o-graph consisting of all objects o of D, such that the label of o appears in τ. A pair of objects o_1 and o_2 is connected by an edge in $\hat{G}$ if there is an edge between their labels in τ and $\{o_1, o_2\}$ is universally $\mathcal{P}$-interconnected in D. The document $\hat{D}$ is obtained from $\hat{G}$ by adding a new root r with a new label l_r. There are edges from r to all the objects that have the root of τ as their label. If $\hat{D}$ has objects that are not reachable from the new root, then they are removed. Line 9 enumerates all subsets O of the objects of

problem	gen.	D is acyc.	D is a tree	S is acyc.	S is a tree
The set of objects O has a fixed size					
$\mathcal{P}^r_{\mathsf{all}}(S) \models_D O$	NPc	NPc	P	P	P
$\mathcal{P}^u_{\mathsf{all}}(S) \models_D O$	NPc	NPc	P	NPc	P
$\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$	P	P	P	P	P
$\mathcal{P}^u_{\mathsf{min}}(S) \models_D O$	P	P	P	P	P
$\mathcal{P}^r_{\mathsf{uca}}(S) \models_D O$	N/A	N/A	N/A	P	P
$\mathcal{P}^r_{\mathsf{all}}(S) \models_{\forall,D} O$	coNPc	P	P	P	P
$\mathcal{P}^u_{\mathsf{all}}(S) \models_{\forall,D} O$	P	P	P	P	P
The set of objects O has an unbounded size					
$\mathcal{P}^r_{\mathsf{all}}(S) \models_D O$	NPc	NPc	P	NPc	P
$\mathcal{P}^u_{\mathsf{all}}(S) \models_D O$	NPc	NPc	P	NPc	P
$\mathcal{P}^r_{\mathsf{min}}(S) \models_D O$	Π^{p}_2	Π^{p}_2	coNPc	Π^{p}_2	P
$\mathcal{P}^u_{\mathsf{min}}(S) \models_D O$	Π^{p}_2	Π^{p}_2	coNPc	Π^{p}_2	P
$\mathcal{P}^r_{\mathsf{uca}}(S) \models_D O$	N/A	N/A	N/A	NPc	P
$\mathcal{P}^r_{\mathsf{all}}(S) \models_{\forall,D} O$	coNPc	P	P	P	P
$\mathcal{P}^u_{\mathsf{all}}(S) \models_{\forall,D} O$	P	P	P	P	P

Table 1: Complexity of the satisfiability problem

$\hat{D}$, such that $l(O) = L$ and O is interconnected according to the pattern (L, τ).

In fact, the algorithm is correct not just for $\mathcal{P}^r_{\mathsf{all}}$ and $\mathcal{P}^u_{\mathsf{all}}$, but also for other interconnection semantics, provided that some conditions are satisfied. Consequently, we get the following theorem.

THEOREM 6.2. *Consider a query L and a document D, such that S_D is a tree. Suppose that $\mathcal{P}$ is either a rooted or an undirected interconnection semantics that contains a pattern (L, C), such that $C \subseteq S_D$. All the answers to the query L, under universal $\mathcal{P}$-interconnectivity, can be enumerated with polynomial delay.*

7. COMPLEXITY

Table 1 summarizes the complexity of the satisfiability problem for the different types of derived interconnection semantics. The input consists of a schema S, a document D that conforms to S and a subset O of the objects of D. The top part of Table 1 gives the data complexity and the bottom part—the query-and-data complexity. Note that the problems that are in Π^{p}_2 are also both NP-hard and coNP-hard (and in Σ^{p}_2). Proofs of the intractability results are given in [14]. The tractable cases were discussed earlier.

For the derived interconnection semantics considered in this paper, the data complexity of the evaluation problem is similar to that of the corresponding satisfiability problem. That is, when the satisfiability problem is in polynomial time, then so is the corresponding evaluation problem. When the satisfiability problem is intractable, then so is the corresponding non-emptiness problem, and both problems are in the same complexity class.

If the schema is a tree, the query-and-data complexity of the evaluation problem is in polynomial delay. However, if the schema is not a tree, then the evaluation problem is intractable, under query-and-data complexity, for all the types of interconnection semantics.

8. CONCLUSION AND RELATED WORK

We have presented a framework for deciding when objects of an XML document are semantically related. Our framework includes a wide spectrum of interconnection semantics that apply to XML documents with ID references. Earlier work [4, 5, 17] considered only tree documents. Another advantage of our approach is using the schema for deriving interconnection semantics. Note that the derived schema can be used if none is given explicitly. The solution proposed in [17] for avoiding incorrect answers in tree documents is similar to our $\mathcal{P}^r_{\mathsf{uca}}$ semantics. However, in [17] only the document is taken into account and hence, their approach cannot detect missing information. The work of [18] on automatically inferring the structure of documents can also be modeled in our framework, under the reasonable assumption that a subtree of a document indicates a meaningful relationship only if it is uniquely labeled. Future work includes experimentation with our semantics and algorithms, and incorporation of information-retrieval techniques.

9. REFERENCES

[1] S. Agrawal, S. Chaudhuri, and G. Das. DBXplorer: A system for keyword-based search over relational databases. In *ICDE*, pages 5–16, 2002.

[2] M. Barg and R. Wong. Structural proximity searching for large collections of semi-structured data. In *CIKM*, pages 175–182, 2001.

[3] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *ICDE*, pages 431–440, 2002.

[4] S. Cohen, Y. Kanza, J. Mamou, and Y. Sagiv. XSEarch: a semantic search engine for XML. In *VLDB*, pages 45–56, 2003.

[5] S. Cohen, Y. Kanza, and Y. Sagiv. Generating relations from XML documents. In *ICDT*, pages 285–299, 2003.

[6] S. Dreyfus and R. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.

[7] S. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:11–121, 1980.

[8] R. Goldman, N. Shivakumar, S. Venkatasubramanian, and H. Garcia-Molina. Proximity search in databases. In *VLDB*, pages 26–37, 1998.

[9] V. Hristidis and Y. Papakonstantinou. DISCOVER: Keyword search in relational databases. In *VLDB*, pages 670–681, 2002.

[10] V. Hristidis, Y. Papakonstantinou, and A. Balmin. Keyword proximity search on XML graphs. In *ICDE*, pages 367–378, 2003.

[11] F. Hwang, D. Richards, and P. Winter. *The Steiner Tree Problem*, volume 53 of *Annals of Discrete Mathematics*. North-Holland, 1992.

[12] D. Johnson, M. Yannakakis, and C. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27:119–123, March 1988.

[13] R. Karp. On the complexity of combinatorial problems. *Networks*, 5:44–68, 1975.

[14] B. Kimelfeld. Interconnection semantics for XML. Master's thesis, The Hebrew University of Jerusalem, 2004. Available at the author's home page (`http://www.cs.huji.ac.il/~bennyk`).

[15] B. Kimelfeld and Y. Sagiv. Efficient engines for keyword proximity search. In *WebDB*, pages 67–72, 2005.

[16] B. Kimelfeld and Y. Sagiv. Efficiently enumerating results of keyword search. In *DBPL*, 2005.

[17] Y. Li, C. Yu, and H. V. Jagadish. Schema-free XQuery. In *VLDB*, 2004.

[18] A. Rajaraman and J. Ullman. Querying websites using compact skeletons. In *PODS*, pages 16–27, 2001.

[19] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995.

[20] M. Y. Vardi. The complexity of relational query languages. In *STOC*, pages 137–146, 1982.

[21] M. Yannakakis. Algorithms for acyclic database schemas. In *VLDB*, pages 82–94, 1981.