

Interconnection Semantics for XML

Benny Kimelfeld

School of Engineering and Computer Science
The Hebrew University of Jerusalem

Supervised by Prof. Yehoshua Sagiv

Submitted in partial fulfillment of the requirements for the degree

MASTERS OF COMPUTER SCIENCE

May 21, 2005

Abstract

A framework for defining and automatically discovering semantic relationships among nodes in XML documents is presented. A specific *interconnection semantics* in this framework consists of a set of *patterns*. Interconnection semantics can be specified explicitly or derived automatically. Several methods to automatically derive interconnection semantics are presented. The complexity of determining when nodes are interconnected under these semantics is analyzed. For many important cases, the complexity is tractable and hence, the proposed interconnection semantics can be efficiently applied to real-world documents. In particular, for acyclically-labeled documents, determining interconnection for a bounded-size set of nodes is polynomial for most of these semantics. The inverse problem of constructing a document from a given set of objects and the interconnections that hold among those objects is also considered. It is shown that under a natural condition of unambiguity, a document that satisfies exactly the specified interconnections can be constructed efficiently, if such a document exists. If not, the set of new interconnections that are introduced by the construction is minimal.

Acknowledgments

My deepest gratitude is to my advisor, Prof. Shuky Sagiv, for his invaluable support and guidance. His patience, attention to detail, excellent direction, and great efforts have made this accomplishment possible.

I am extremely thankful to Sara Cohen and Yaron Kanza, whom I regard as both friends and advisors. This work was inspired and developed by many hours of discussions with each one of them. I am particularly grateful to Sara for teaching me, together with Prof. Sagiv, what mathematical writing is all about.

I thank the people of the Computer Science department in the Hebrew University, and particularly the members of the Database Group, for providing a supportive environment.

Finally, I deeply thank my mother and wife for their unconditional encouragement, patience and support.

Contents

1	Introduction	1
2	Preliminaries	6
2.1	Graphs and Trees	6
2.2	O-Graphs and L-Graphs	7
2.3	Documents, Schemas and Reduced Trees	8
2.4	O-Tuples and O-Relations	8
3	Patterns and Interconnection Semantics	9
3.1	Testing Interconnection and Computing Interconnection Queries	10
4	Deriving Interconnection Semantics	21
4.1	The Interconnection Semantics $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$	21
4.2	The Interconnection Semantics $\mathcal{P}_{\text{min}}^r(S)$ and $\mathcal{P}_{\text{min}}^u(S)$	29
4.3	The Semantics $\mathcal{P}_{\text{uca}}^r(S)$	35
4.4	Interconnections in Tree Schemas	37
5	Universal Interconnectivity	39
5.1	Universal $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$ interconnections	40
5.2	Universal Interconnectivity in Tree Schemas	48
6	Practical Considerations of Interconnectivity	54
6.1	A Discussion on the Complexity Results	54
6.2	Heuristics For Intractable Cases	56

7	Constructing Meaningful Documents	58
7.1	Layouts	58
7.2	Combining Data Extraction and Document Construction	62
8	NP-Complete Properties of Labeled Graphs	67
8.1	Uniquely Labeled Paths	67
8.2	Uniquely-Labeled Subgraph Isomorphisms	69
9	Conclusion	72
A	Enumeration of Reduced Subtrees	76
A.1	Rooted Subtrees	76
A.1.1	Extensions of Reduced Subtrees	76
A.1.2	Generating $\rho^r(G, V)$	78
A.2	Undirected Subtrees	80

Chapter 1

Introduction

The aim of information retrieval is to find data that are relevant to some keywords. Data extraction is different in the sense that the goal is to extract pieces of information that are related to each other. Traditionally, this is done by a query language and, in the context of XML documents, it could conceivably be accomplished by writing queries in some XML query language, e.g., XQuery. However, phrasing queries correctly is often a cumbersome task, especially for naive users. Therefore, our goal is to automate as much as possible the process of data extraction from XML documents. One way to do that is by developing a query language for specifying the data item to be extracted, without having to specify how those data items are arranged in the documents of interest. A user could specify a set of labels, e.g., `{Name,Email}`, and the result should be a relation with these labels as attributes and with tuples containing a name and an email that are meaningfully related. In the document of Figure 1.1, for example, objects (i.e., nodes) 5 and 6 contain the name and email of the same person and hence are meaningfully related. Similarly, objects 11 and 13 are also meaningfully related, since they have the name and email of the same project. As for objects 11 and 6, it would seem obvious that they are not related in any meaningful way, since object 11 is the name of a project and object 6 is the email of a manager (who manages a department that participates in the project). However, if the email of the project is missing or nonexistent, then it could be argued that giving the email of that manager as an answer is better than not giving any email at all. Thus, data extraction is similar to information retrieval in the sense that it has an inherent precision-versus-recall tradeoff.

The problem at hand is to capture formally when some objects of a given document are *interconnected*, i.e., meaningfully related. Ideally, those who create XML data should also supply some means of identifying meaningful relationships in their documents. Toward this end, we propose *interconnection semantics* as a tool for explicitly describing the meaningful relationships that exist in XML documents. Creators of XML data, however, cannot always be expected to supply interconnection semantics, since defining those semantics could be cumbersome and tedious. Thus, there will always be a need for automatic discovery of meaningful relationships among nodes of XML documents. Yet, it is unlikely that a single approach can always succeed in finding automatically those relationships, since small changes in XML documents (e.g., renaming some labels) could drastically change the meaningful relationships that are represented in those documents. Therefore, we will propose several approaches that automatically derive different interconnection semantics, with varying degrees of recall and precision.

Let O be a set of objects. The first step in automatically deriving an interconnection semantics is to determine some basic requirements that should be fulfilled if the objects of O are deemed interconnected. Just requiring the objects of O to appear in a single document renders the problem uninteresting and practically useless. We adopt the approach, enunciated in [2], that different pieces of information can be combined together as long as they do not belong to distinct entities of the same type. For example, it does not make much sense to interconnect the name of one manager with the email of another manager. Thus, a basic requirement for an interconnection among the objects of O is the existence of a uniquely-labeled rooted subtree of the given document, such that O is contained in that subtree. In Section 4.1, this approach is defined formally as the semantics $\mathcal{P}_{\text{all}}^r$. Although $\mathcal{P}_{\text{all}}^r$ is a rather simple semantics, it seems to capture exactly the sets of meaningfully-related nodes for many popular XML documents, such as the XML version of the Sigmod Record¹, Shakespeare's plays² and the Bible³.

In order to increase the precision of $\mathcal{P}_{\text{all}}^r$, we will combine it with two different notions of minimal rooted subtrees. The first notion is inspired by the common assumption (e.g., [1]) that meaningfully-related nodes must be close to each other. According to this approach,

¹<http://www.acm.org/sigmod/record/xml>

²<http://www.ibiblio.org/xml/examples/shakespeare>

³<http://www.bibletechnologies.net>

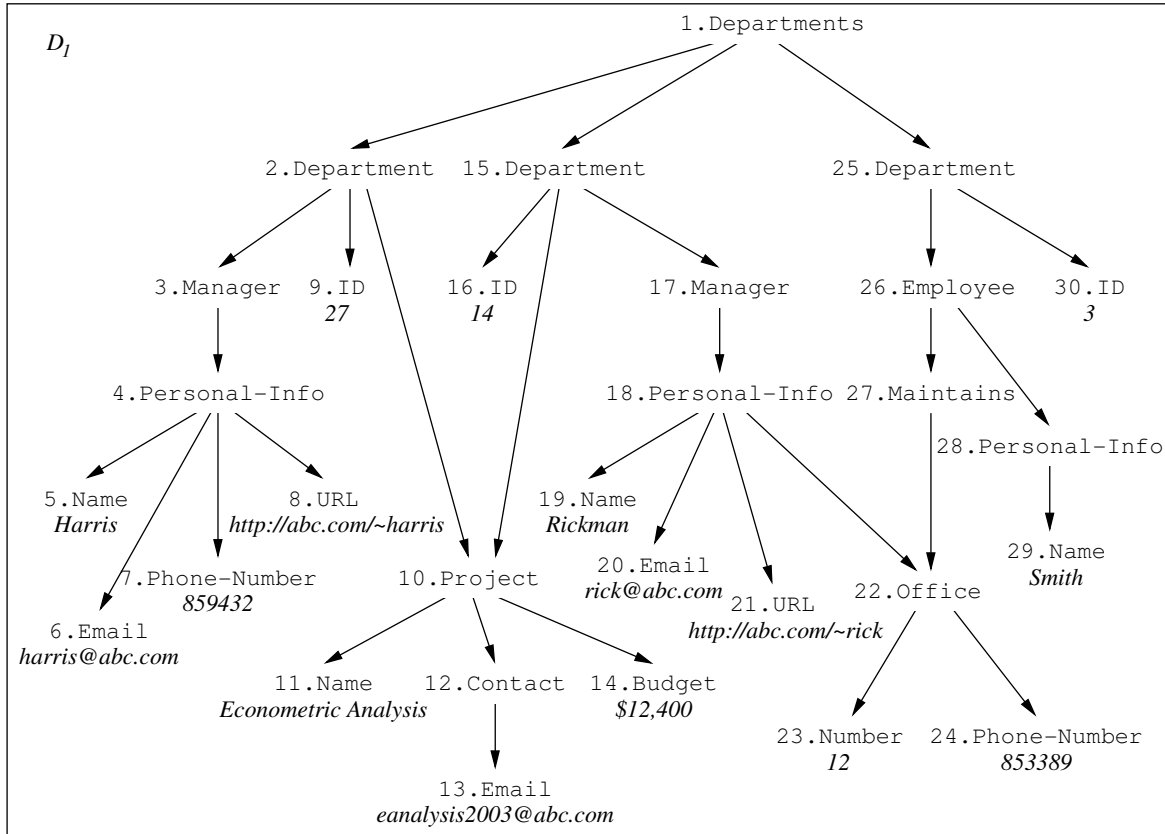


Figure 1.1: A document D_1 describing departments, employees, managers and projects.

the objects of O are deemed interconnected if they appear in a uniquely-labeled rooted subtree that has a minimal size (among all rooted subtrees that include the labels of O). For example, when looking for a name and an email that are meaningfully related, objects 5 and 6 appear in a rooted subtree that has three nodes (the third one is object 4). Thus, these two objects are deemed interconnected by the semantics that is defined formally as \mathcal{P}_{\min}^r in Section 4.2. Objects 11 and 13, on the other hand, are not deemed interconnected by that semantics, since there is no rooted subtree with three nodes that includes these two objects (the smallest rooted subtree that contains them has four nodes and its root is object 10).

The fact that objects 11 and 13 are not deemed interconnected, simply because they are slightly too far away from each other, can be remedied by a different notion of minimal interconnections. The idea is to define a notion of minimal rooted subtrees that is based

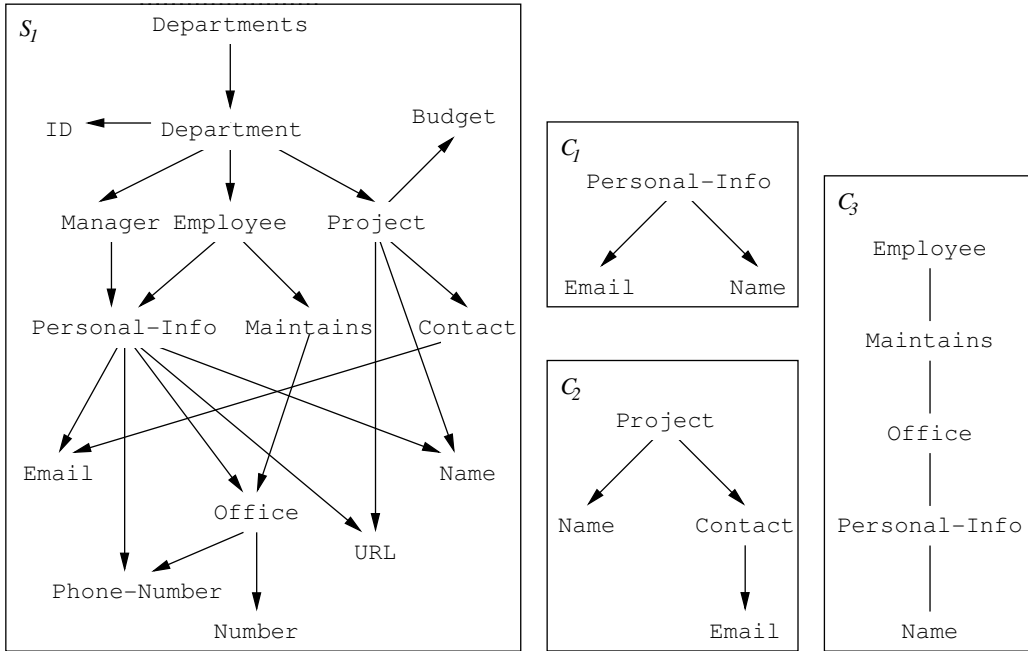


Figure 1.2: A Schema S_1 , rooted l-trees C_1 and C_2 , and an undirected l-tree C_3 .

on the hierarchy of objects in the document. According to this approach, both the rooted subtree comprising objects 4, 5 and 6, and the rooted subtree comprising objects 10, 11, 12 and 13 are minimal. But the rooted subtree that has object 2 as its root and objects 11 and 6 as its only leaves is not minimal, because it has some interior nodes (i.e., objects 4 and 10) that are (or could potentially be) roots of subtrees that include a name and an email. This third approach is defined formally as \mathcal{P}_{uca}^r in Section 4.3.

The above three semantics define different sets of meaningfully-related objects. The semantics \mathcal{P}_{all}^r usually yields *all* meaningfully-related sets of objects, but may also capture sets that are not meaningfully related. The semantics \mathcal{P}_{min}^r , on the other hand, usually yields *only*, but not necessarily *all* meaningfully-related sets of objects. The semantics \mathcal{P}_{uca}^r offers (when the schema is acyclic) the middle road between the former two semantics.

We further enhance the spectrum of interconnection semantics by introducing semantics that are based on undirected (rather than rooted) subtrees and semantics that are applied universally, rather than existentially (i.e., all subtrees that contain the given set of objects, rather than just one, must have a certain property). The interconnection semantics that we propose have different precision-versus-recall tradeoffs. We believe that each one may

be useful under different circumstances.

We also analyze the complexity of each semantics. We do so in the spirit of data complexity. In our case, the query is a set of labels (e.g., $\{\text{Name}, \text{Email}\}$) and therefore, the tractable cases are those that have a polynomial-time complexity when the size of this set is constant.

In Chapter 7, we use interconnection semantics to formalize the following problem. Given some objects and a specification of the interconnections that hold among them, construct a new document that includes the given objects, conforms to a given schema and preserves the given interconnections. We describe when such a document can be constructed, but in general it may have more interconnections than those originally specified. We give a simple algorithm and show that if a natural condition of unambiguity is satisfied, then the algorithm creates a document that has a minimal set of new interconnections. This approach for constructing XML documents can be fully automatic compared to existing methods that rely on queries that are manually generated.

The outline of this work is as follows. Chapter 2 presents some preliminary definitions. In Chapter 3, patterns and interconnection semantics are introduced, and their complexity is analyzed. Chapter 4 presents interconnection semantics that can be derived automatically, along with a discussion of their complexity. In Chapter 5, universal interconnectivity is introduced, and several important cases are studied. A discussion on the practical considerations of interconnectivity is presented in Chapter 6. In Chapter 7, the notion of constructing meaningful documents from a specification of interconnections is formalized, and the construction algorithm is presented. Chapter 8 presents some NP-completeness results that are used throughout this work. Finally, Chapter 9 surveys some related work and concludes.

Chapter 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).

$G' = (V', E')$ 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. If T is a rooted tree and v is some node in T , we use $T|_v$ to denote the subtree of T that is rooted at v (that is, $T|_v$ is the rooted subtree of T induced by all nodes that are reachable from v through a directed path). We will also consider *undirected subtrees* of rooted graphs by ignoring the directions of the edges.

2.2 O-Graphs and L-Graphs

An *o-graph* (or o-tree) has *objects* as nodes. An object has an *object identifier* (abbr. oid) and is assigned a *label* and, possibly, a *value*. Figure 1.1 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-graph is *uniquely labeled* if distinct objects do not have the same label.

An *l-graph* (or l-tree) has labels as nodes. Figure 1.2 shows a rooted l-graph S_1 , and two rooted l-trees C_1 and C_2 . Note that a label may occur just once in an l-graph, but may have multiple occurrences in an o-graph.

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

- If (o_1, o_2) is an edge of G , then $(l(o_1), l(o_2))$ is an edge of S , and
- If G and S are rooted, then the root of S is the label of the root of G .

For example, the rooted o-graph D_1 of Figure 1.1 conforms to the rooted l-graph S_1 of Figure 1.2.

A directed o-graph G is *acyclically labeled* if G conforms to some directed l-graph S , such that S is acyclic. Note that an acyclically-labeled o-graph must be acyclic, but an acyclic o-graph is not necessarily acyclically labeled.

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 1.1) that comprises objects 4, 5 and 6 is isomorphic to the rooted l-tree C_1 (Figure 1.2).

Isomorphism between undirected trees is defined similarly by removing the condition about the roots. 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, Schemas and Reduced Trees

A *document* is a rooted o-graph and a *schema* is a rooted l-graph. Figure 1.1 shows a document D_1 and Figure 1.2 shows a schema S_1 .

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 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. Also note that D is acyclically labeled if and only if S_D is acyclic. As an example, the derived schema for D_1 (Figure 1.1) is obtained from the schema S_1 (Figure 1.2) 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**). Note that D_1 is acyclically labeled since S_1 is acyclic.

A rooted tree (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 (undirected subtree) that includes all the nodes of U . Figure 1.2 shows the rooted l-trees C_1 and C_2 that are reduced w.r.t. to the set of labels $\{\mathbf{Email}, \mathbf{Name}\}$.

2.4 O-Tuples and O-Relations

We use relations that have objects as data elements and labels as attributes. An *o-tuple* is a set of the form $t = \{l_1 : o_1, \dots, l_k : o_k\}$, where l_1, \dots, l_k are distinct labels and each o_i is an object that is labeled with l_i . The set $\{o_1, \dots, o_k\}$, denoted $O[t]$, is called the *content* of t . The set $\{l_1, \dots, l_k\}$, denoted $L[t]$, is called the *signature* of t . An *o-relation* is a set of o-tuples with the same signature.

Chapter 3

Patterns and Interconnection Semantics

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

Intuitively, a rooted pattern describes when objects having the labels of L are interconnected. For example, consider the document D_1 in Figure 1.1 and the rooted l-trees C_1 and C_2 in Figure 1.2. The rooted patterns $(\{\mathbf{Name}, \mathbf{Email}\}, C_1)$ and $(\{\mathbf{Name}, \mathbf{Email}\}, C_2)$ describe when two objects labeled with **Name** and **Email** are interconnected in document D_1 . This is formally defined as follows.

Definition 3.1 (interconnected objects) *Let O be a set of objects in a document D . A rooted (undirected) pattern $p = (L, C)$ interconnects O in D , denoted $p \models_D O$, if $l(O) = L$ and D has a rooted (undirected) subtree T , such that T includes all the objects of O and T is isomorphic to C .*

Note that a pattern can only interconnect a set of uniquely-labeled objects. As an example, consider the document D_1 (Figure 1.1), and the l-trees C_1 and C_2 (Figure 1.2). The rooted pattern $(\{\mathbf{Name}, \mathbf{Email}\}, C_1)$ interconnects the pair of objects (5, 6) and also the pair (19, 20), while the rooted pattern $(\{\mathbf{Name}, \mathbf{Email}\}, C_2)$ interconnects the pair (11, 13).

Some relationships cannot be captured by any rooted pattern. For example, in document D_1 (Figure 1.1), object 19 and object 26 are 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 $(\{\text{Name}, \text{Employee}\}, C_3)$, where C_3 is the undirected l-tree shown in Figure 1.2.

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

Definition 3.2 (\mathcal{P} -interconnectivity) *Let D be a document and \mathcal{P} be an interconnection semantics. Given a subset O of the objects in D , we say that O is \mathcal{P} -interconnected, denoted $\mathcal{P} \models_D O$, if \mathcal{P} contains a pattern p , such that $p \models_D O$.*

An *interconnection query* is a query of the form “select all tuples of interconnected objects over a given set of labels.” Formally, an interconnection query is a pair of the form $q = (L, \mathcal{P})$, where \mathcal{P} is an interconnection semantics and L is a set of labels. The result of applying the query $q = (L, \mathcal{P})$ to a document D , denoted $\mathcal{R}(q, D)$, is the o-relation consisting of all o-tuples t , such that (1) $O[t]$ is a subset of the objects in D , (2) $L[t] = L$ and (3) $\mathcal{P} \models_D O[t]$.

3.1 Testing Interconnection and Computing Interconnection Queries

Consider an interconnection semantics \mathcal{P} , a set O of objects in a document D and an interconnection query $q = (L, \mathcal{P})$. Throughout this work, we consider the following computational problems.

1. Determine whether $\mathcal{P} \models_D O$.
2. Compute $\mathcal{R}(q, D)$.

If O and L are of fixed size, these problems have similar complexities. Otherwise, the size of $\mathcal{R}(q, D)$ can be exponential in the size of the input. Hence, we will analyze the

ISOMORPHISMS(G, C)

```
1: let  $G' \subseteq G$  be the graph induced by all the objects of  $G$  with labels from  $C$ 
2: for all edge  $e = (o_1, o_2)$  of  $G'$  do
3:   if  $(l(o_1), l(o_2))$  is not an edge of  $C$  then
4:     remove  $e$  from  $G'$ 
5: let  $l_1, \dots, l_m$  be the nodes of  $C$  in bottom-up order
6: for  $q = 1$  to  $m$  do
7:   let  $l'_1, \dots, l'_j$  be the children of  $l_q$  in  $C$ 
8:   for all object  $o$  in  $G'$  s.t.  $l(o) = l_q$  do
9:     if for all  $1 \leq i \leq j$ , object  $o$  has an outgoing edge to an object  $o_i$ 
       s.t.  $l(o_i) = l'_i$  and  $o_i$  is marked then
10:      mark  $o$ 
11: let  $G'' \subseteq G'$  be the o-graph induced by all the objects that are reachable
       from an object labeled with  $l_m$  through a directed path of marked objects
12: return  $G''$ 
```

Figure 3.1: An algorithm for finding all the objects and edges that participate in some subgraph of G that is isomorphic to the l-graph C .

computation of $\mathcal{R}(q, D)$ in terms of input-output complexity. That is, the running time is a function of the combined size of the input and the output.

In the above problems, the interconnection semantics \mathcal{P} will not always be given explicitly (i.e., as part of the input). This is the case, for example, when \mathcal{P} is either infinite or very large w.r.t. D . In the following chapters, we will encounter specific interconnection semantics for which these problems are intractable. However, if \mathcal{P} is given explicitly as part of the input, then both determining interconnections according to \mathcal{P} and computing interconnection queries are tractable, even if the size of either O or L is not constant. This is shown in the rest of this chapter.

Consider a directed o-graph G and a rooted l-tree C . Let $\mathcal{I}(G, C)$ be the set of all the subgraphs of G that are isomorphic to C . The *restriction* of G w.r.t. C , denoted as $G[C]$, is the subgraph of G that consists of all objects and edges of the subgraphs $H \in \mathcal{I}(G, C)$. Figure 3.1 describes the algorithm ISOMORPHISMS(G, C) for computing $G[C]$. In Lines 1–4 of this algorithm, we create the subgraph G' of G . This subgraph includes an object o if $l(o)$ is a label of C , and it includes an edge $e = (o_1, o_2)$ if $(l(o_1), l(o_2))$ is an edge of C . In Lines 6–10, we traverse the objects in G' according to a bottom-up order on their labels that is induced by the l-tree C (note that label l_m in Line 5 must be the root of C). During

this traversal, we mark an object o in G' if the labels of the marked children of o include all the children of $l(o)$ in C . The result of the algorithm is the induced subgraph G'' of G' that is constructed in Line 11 by removing objects from G' as follows. At first, we remove all the unmarked objects. Then we remove all objects o , such that o is not reachable from any object that is labeled with the root of C . The correctness of the algorithm is proved in the next lemma.

Lemma 3.3 *Let G be a directed o -graph and C be a rooted l -tree. The result of executing ISOMORPHISMS(G, C) (shown in Figure 3.1) is the graph $G[C]$.*

Proof. We will show that the output of the algorithm is the graph $G'[C]$. This proves the lemma, since $G[C] = G'[C]$ holds because G' is obtained from G (in Lines 1–4) by removing objects and edges that cannot be in any subgraph of G that is isomorphic to C . The following facts about the graph G' are true because of Lines 3–4 and since the rooted l -tree C has no cycle.

Fact 1 *If an object p is a parent of an object o in G' , then the label $l(p)$ is the parent of the label $l(o)$ in C .*

Fact 2 *The graph G' does not have any directed path between two distinct objects that have the same label.*

Our proof consists of the following three claims.

Claim 1 *An object o of G' is marked in Line 10 if and only if o is the root of a subtree T of G' that is isomorphic to the subtree $C_{|l(o)}$ (i.e., the subtree of C that is rooted at $l(o)$).*

Claim 2 *The objects of the graph G'' , which is constructed in Line 11, are the objects of the graph $G'[C]$.*

Claim 3 *The edges of the graph G'' are the edges of the graph $G'[C]$.*

We assume, by induction, that Claim 1 holds for all objects that are considered in the first $q - 1$ iterations of Line 6. Suppose that object o is considered in the q th iteration. For the “if” direction, suppose that o is the root of a subtree T of G' that is isomorphic to

$C_{|l(o)}$. Notice that the children of o in T are considered in the first $q - 1$ iterations. The fact that T is isomorphic to $C_{|l(o)}$ and the inductive hypothesis imply that each child of o in T is already marked when o is considered. Hence, o satisfies the condition of Line 9 and is marked in Line 10. For the “only if” direction, suppose that o is marked in Line 10. By the inductive hypothesis and the condition of Line 9, for each child l of $l(o)$ in C , there is a subtree T_l of G' , such that the root of T_l is a child of o in G' and T_l is isomorphic to $C_{|l}$. The required subtree T comprises o as the root and the T_l as the subtrees of the root (note that the subtrees T_l do not share labels and therefore do not share objects).

Claim 2 is proved by proving the following equivalent claim.

Claim 4 *An object o labeled with l_i is in $G'[C]$ if and only if G' has a path from an object \hat{o} labeled with l_m to the object o , such that all the objects along that path are marked.*

For the “only if” direction, consider an object o in $G'[C]$. By the definition of $G'[C]$, object o appears in a subtree T of G' , such that T is isomorphic to C . By Claim 1, all the objects of T are marked. Hence, the path from the root of T to o satisfies the requirements of the claim. The “if” direction is proved by induction on the reverse bottom-up order of Line 5. For the base case, observe that Fact 2 implies that if a path in G' starts and ends at objects with the same label l_m , then that path must consist of a single object. Hence, the base case of the induction follows from Claim 1. For the inductive step, suppose that there is a path in G' that starts at an object \hat{o} labeled with l_m , ends at an object o labeled with l_i , and includes only marked objects. Let o' be the object that precedes o on that path. Note that the label of o' precedes l_i in the reverse bottom-up order of Line 5. Hence, by the inductive hypothesis, o' is in $G'[C]$. By the definition of $G'[C]$, object o' appears in a subtree T' of G' that is isomorphic to C . Fact 1 implies that in T' , object o' has a child o'' labeled with l_i , but this child is not necessarily o . However, by Claim 1, object o is the root of a subtree T'' of G' that is isomorphic to $C_{|l_i}$. Let T be obtained from T' by replacing the subtree of T' that is rooted at o'' with T'' . Notice that T is a subtree of G' that is isomorphic to C . Hence, T shows that o is in $G'[C]$ and the inductive step is proved.

To prove Claim 3, consider at first an edge (o', o) of $G'[C]$. By Claim 2, both o' and o are in G'' . Since G'' is an induced graph of G' , it follows that (o', o) is in G'' . For the other

direction, suppose that the edge (o', o) is in G'' . By the construction of G'' in Line 11, G' has a path of marked objects from an object \hat{o} labeled with l_m to o' . That path can be extended by adding the edge (o', o) . Similarly to the inductive part of the proof of Claim 4, we can show that there is a subtree T of G' that is isomorphic to C and includes the edge (o', o) . Thus, the edge (o', o) is in $G'[C]$. \square

As a result of Lemma 3.3, we can efficiently test whether an l-tree is isomorphic to some subgraph of a directed o-graph, regardless of whether the l-tree is rooted or undirected. This is shown in the next corollary. In Section 8.2 of Chapter 8, we show that this isomorphism problem becomes NP-complete if arbitrary l-graphs are allowed instead of l-trees.

Corollary 3.4 *Let G be a directed o-graph and C be a rooted (undirected) l-tree. Testing whether C is isomorphic to some subgraph of G is in polynomial time.*

Proof. For the case where C is rooted, we apply the algorithm ISOMORPHISMS(G, C). Clearly, the running time of this algorithm is polynomial in the size of G . By Lemma 3.3, C is isomorphic to a rooted subtree of G if and only if the algorithm returns a nonempty graph.

For the case where C is undirected, we use the following reduction to the former case. First, we arbitrarily choose directions to the edges of C to form a rooted tree C' . Next, we create a directed o-graph G' from G as follows. The objects of G' are the objects of G , and G' has a directed edge (o_1, o_2) if C' has the edge $(l(o_1), l(o_2))$ and G has either the edge (o_1, o_2) or the edge (o_2, o_1) . Having created C' and G' , we apply ISOMORPHISMS(G', C'). \square

We can now show that testing whether a pattern interconnects a set of objects in a document is also tractable.

Corollary 3.5 *Let D be a document, O be a set of objects and p be a pattern. Testing $p \models_D O$ is polynomial in D .*

Proof. Let $p = (L, C)$. In order to test $p \models_D O$, we do the following. First, we verify that O is uniquely labeled and $L = l(O)$. We then remove from D all objects o , such that $l(o) \in L$ and $o \notin O$. Testing $p \models_D O$ is now equivalent to testing whether C is isomorphic to a subgraph of D and that can be efficiently tested, by Corollary 3.4. \square

```

PATTERNINTERCONNECTIONS( $D, L, p$ )
1: let  $p = (L', C)$ 
2: if  $L' \neq L$  then
3:   return  $\emptyset$ 
4:  $G := \text{ISOMORPHISMS}(C, D)$ 
5: let  $l_1, \dots, l_m$  be the nodes of  $C$  in a bottom-up order
6: for  $q = 1$  to  $m$  do
7:   let  $l'_1, \dots, l'_j$  be the children of  $l_q$  in  $C$ 
8:   for all objects  $o$  in  $G'$  s.t.  $l(o) = l_q$  do
9:     for  $i = 1$  to  $j$  do
10:      let  $o_1, \dots, o_k$  be the children of  $o$  that are labeled with  $l'_i$ 
11:       $R_i = \bigcup_{r=1}^k R(o_r)$ 
12:      if  $l(o) \notin L$  then
13:         $R(o) := R_1 \times \dots \times R_j$ 
14:      else
15:         $R(o) := \{\{l_q : o\}\} \times R_1 \times \dots \times R_j$ 
16: let  $O_m$  be the set of objects in  $G$  such that  $l(o) = l_m$ 
17: return  $\bigcup_{o \in O_m} R(o)$ 

```

Figure 3.2: An algorithm for computing all sets O of objects from D that are interconnected by the pattern p .

We will now consider the computation of interconnection queries for single-pattern semantics. The results shown here will later be extended to all semantics that are given explicitly. Consider a document D , a set L of labels and a rooted pattern $p = (L', C)$. Let q be the interconnection query $(L, \{p\})$. We use the algorithm $\text{PATTERNINTERCONNECTIONS}(D, L, p)$, described in Figure 3.2, in order to generate the o-relation $\mathcal{R}(q, D)$. If $L \neq L'$, this o-relation is empty by definition. Thus, we assume that $L = L'$. Our algorithm consists of two phases:

1. Construction of the o-graph $G = D[C]$;
2. Generation of all o-tuples t , such that
 - $L[t] = L$, and
 - $O[t]$ is a subset of the objects of some subgraph of G that is isomorphic to C .

For the second phase, we use the algorithm of [8] for generating relations that are induced by documents with “perfect compact skeletons.” We now describe this phase in details.

Consider an object o in G and a subtree C' of C . Let $\mathcal{I}(G, C', o)$ be the set of all the subgraphs of G that are isomorphic to C' and have o as their root. We define $R(G, p, o)$ to be the o -relation that consists of all o -tuples t such that

- $O[t]$ is a subset of the objects of some subgraph $H \in \mathcal{I}(G, C_{|l(o)}, o)$ (that is, some subgraph of G is rooted at o , contains $O[t]$ and is isomorphic to $C_{|l(o)}$), and
- $L[t]$ is the intersection of L with the labels of $C_{|l(o)}$.

In Lines 5–15, we traverse the objects of G according to a bottom-up order on their labels that is induced by the l-tree C (i.e., a traversal order similar to the one used in the algorithm ISOMORPHISMS). When an object o is visited, we create the o -relation $R(G, p, o)$. In the algorithm, this set is denoted $R(o)$. Observe that when $l(o)$ is a leaf of C , the label $l(o)$ must be in L and hence, according to Lines 12-15, $R(o)$ is the singleton o -relation $\{t\}$, where $t = \{l(o) : o\}$. If $l(o)$ is not a leaf, then $R(o')$ is already defined for every child o' of o . In this case, we group the o -relations $R(o')$ of the children o' of o , according to the labels of these children (Lines 9–11). The value of $R(o)$ is then set to be the Cartesian product of the groups (Lines 12–15). If $l(o) \in L$, the o -relation $\{\{l(o) : l\}\}$ also participates in the Cartesian product. The result of this algorithm is the union of all o -relations $R(o)$, such that $l(o)$ is the root of C . The correctness of the algorithm $\text{PATTERNINTERCONNECTIONS}(D, L, p)$ can be proved by a reduction to the correctness of the algorithm presented in [8], which was mentioned above. Specifically, the compact skeleton used by that algorithm can be generated from the pattern p . However, we chose to present an independent proof that is suitable to our data model. The next lemma shows the correctness of the algorithm and presents an upper bound on its complexity.

Lemma 3.6 *Let D be a document, L be a set of labels, and p be a rooted pattern. Let q be the interconnection query $(L, \{p\})$. Executing $\text{PATTERNINTERCONNECTIONS}(D, L, p)$ results in the o -relation $\mathcal{R}(q, D)$. Furthermore, this algorithm runs in time $O(|L|mM \log(nM))$, where n and m are the number of objects and edges in D , respectively, and M is the number of tuples in $\mathcal{R}(q, D)$.*

Proof. Let $p = (L', C)$. We assume that $L' = L$, otherwise, correctness of the algorithm is trivial. By Lemma 3.3, the graph G , constructed in Line 4, is the graph $D[C]$. Hence, G is

a subgraph of D , and all the subgraphs of D that are isomorphic to C are also subgraphs of G . Let the root of C be the label l_m . Let O_m be the set of all objects in G that are labeled with l_m . It holds that

$$\mathcal{R}(q, D) = \bigcup_{o \in O_m} R(G, p, o). \quad (3.1)$$

Note that the right-hand side of Equation 3.1 is the set of all tuples t , such that $L[t] = L$ and $O[t]$ is included in some subtree of G that is isomorphic to C . Hence, the equality holds by the definition of $\mathcal{R}(q, D)$. From the construction of the result in Line 16, it is sufficient to show that when this line is reached, $R(o) = R(G, p, o)$ for all objects $o \in O_m$. So, we will prove the following claim.

Claim 1 *Let o be an object in G . After object o is visited (Lines 8–14), the value of $R(o)$ is the o -relation $R(G, p, o)$.*

Assume, by induction, that Claim 1 holds for every object o considered in the first $q - 1$ iterations of Line 6. Let o be some object considered in the q th iteration. We assume that $l(o) \notin L$. The case where $l(o) \in L$ is similarly proved. Let t be an o -tuple such that $O[t]$ is a subset of the objects in G . First, assume that $t \in R(o)$. From Lines 9–12, there exist objects o_1, \dots, o_j and tuples t_1, \dots, t_j such that

- o_1, \dots, o_j are children of o in G ,
- $l(o_1), \dots, l(o_j)$ are all the children of $l(o)$ in C ,
- $t = t_1 \times \dots \times t_j$, and
- when object o is considered, $t_i \in R(o_i)$ for all $1 \leq i \leq j$.

By the induction hypothesis, G has subtrees T_1, \dots, T_j such that $T_i \in \mathcal{I}(G, C_{|o(l_i)}, o_i)$ and T_i includes the objects of t_i , for all $1 \leq i \leq j$. Consider the rooted subtree T that has o as a root and T_1, \dots, T_j as subtrees of the root. Then $T \in \mathcal{I}(G, C_{|l(o)}, o)$ and T includes the objects in $O[t]$. Hence, $t \in R(G, p, o)$, as required.

Now assume that $t \in R(G, p, o)$. Let $T \in \mathcal{I}(G, C_{|l(o)}, o)$ be a rooted subtree such that it includes all the objects in $O[t]$. Let T_1, \dots, T_j be the subtrees of o (i.e., the root) in

T . Let o_i be the root of T_i . Let t_i be the o -tuple such that $O[t_i]$ is the set of all objects in T_i that have labels from L . Then, $t = t_1 \times \cdots \times t_j$. Note that objects o_1, \dots, o_j are considered in iterations that precede the q th iteration in which o is considered. By the induction hypothesis, in the q th iteration, $t_i \in R(o_i)$ for all $1 \leq i \leq j$. Hence, from the construction of $R(o)$ in Lines 9–13, it follows that $R(o)$ contains t , as required. This proves Claim 1 and shows the correctness of the algorithm.

The running time of the algorithm is proved as follows. Let n and m be the number of objects and edges in D , respectively, and M be the number of o -tuples in $\mathcal{R}(q, D)$. Let $d_{out}(o)$ denote the out degree of an object o in D . Our proof consists of the following claims.

Claim 2 *For every object o in G and for every tuple $t \in R(G, p, o)$, there exists a tuple $\hat{t} \in \mathcal{R}(q, p)$, such that $t \subseteq \hat{t}$.*

Claim 3 *For every object o in G , $O(|L|d_{out}(o)M \log(d_{out}(o)M))$ steps are executed in order to construct $R(o)$.*

Claim 4 *The running time of the algorithm is $O(|L|mM \log(nM))$.*

To prove Claim 2, consider an o -tuple $t \in R(G, p, o)$. Let T be a subtree in $\mathcal{I}(G, C_{l(o)}, o)$ that includes the objects in $O[t]$. From Lemma 3.3, we conclude that D has a subtree T' such that T' is isomorphic to C and T' includes object o . Consider the subtree \hat{T} , that is obtained from T' by replacing the subtree rooted at o with the tree T . Let \hat{t} be the tuple such that $O[\hat{t}]$ consists of all objects in \hat{T} that have labels from L . Thus, $t \subseteq \hat{t}$. Since \hat{T} is isomorphic to C , it follows that $\hat{t} \in \mathcal{R}(q, D)$, as required.

Claim 3 is proved as follows. From Claims 1 and 2, it follows that for every object o' in G , the number of tuples in $R(o')$ is at most M . Consider an object o in G . For each child o' of o , the set $R(o')$ consists of at most M o -tuples, each of size at most $|L|$. Hence, the grouping in Lines 9-11 takes $O(|L|d_{out}(o)M \log(d_{out}(o)M))$ steps (note that sorting is needed in order to remove duplicates). The number of steps needed to construct the Cartesian product (Lines 12-15) is proportional to $|L| \cdot |R(o)|$, and hence is $O(|L|M)$.

To prove Claim 4 we use the following analysis. Let U be the set of objects in G . By Claim 3, the running time of the algorithm is bounded by

$$O\left(\sum_{o \in U} |L| d_{out}(o) M \log(d_{out}(o) M)\right).$$

We conclude by observing that

$$\sum_{o \in U} |L| d_{out}(o) M \log(d_{out}(o) M) \leq |L| M \log(nM) \sum_{o \in U} d_{out}(o) = |L| m M \log(nM),$$

which proves Claim 4. \square

By Lemma 3.6, computing $\mathcal{R}(q, D)$, where $q = (L, \{p\})$, is tractable if p is rooted. The next corollary extends this result to patterns that are either rooted or undirected.

Corollary 3.7 *Let D be a document and $q = (L, \{p\})$ be an interconnection query. Computing $\mathcal{R}(q, D)$ is in polynomial time in D and the output.*

Proof. When p is rooted, this claim is a direct result of Lemma 3.6. If p is undirected, we use a reduction similar to the one used in the proof of Corollary 3.4, as follows. Let C' be obtained from C by arbitrarily choosing directions to the edges, such that the result is a rooted tree. Let G be the subgraph of D induced by all edges (o_1, o_2) such that $(l(o_1), l(o_2))$ is an (undirected) edge in C . We redirect the edges of G so that for each edge (o_1, o_2) of G , the rooted tree C' contains the edge $(l(o_1), l(o_2))$. We create a document D' by adding to G a root with a new label and outgoing edges to all other objects. It holds that $\mathcal{R}(q, D) = \mathcal{R}(q', D')$, where $q' = (L, \{p'\})$ and $p' = (L, C')$. Since p' is rooted, $\mathcal{R}(q, D)$ can be generated efficiently, as required. \square

We conclude by showing that both testing interconnection of a set of objects and computing interconnection queries are tractable, when interconnection semantics are given explicitly.

Theorem 3.8 *Let D be a document and \mathcal{P} be a finite interconnection semantics.*

1. *For a set of objects O in D , testing $\mathcal{P} \models_D O$ is polynomial in the size of D and \mathcal{P} .*

2. For an interconnection query $q = (L, \mathcal{P})$, generating $\mathcal{R}(q, D)$ is polynomial in the size of D , \mathcal{P} and the output.

Proof. Corollaries 3.5 and 3.7 show Parts 1 and 2, respectively, for $|\mathcal{P}| = 1$. For $|\mathcal{P}| > 1$, observe that

$$\mathcal{P} \models_D (O) \Leftrightarrow \bigvee_{p \in \mathcal{P}} p \models_D O \quad (3.2)$$

and

$$\mathcal{R}(q, D) = \bigcup_{p \in \mathcal{P}} \mathcal{R}(q_p, D) \quad (3.3)$$

where q_p denotes the interconnection query $(L, \{p\})$. Hence, the algorithms we proposed for singleton semantics can be used for general semantics, while the running time increases by a factor that is proportional to $|\mathcal{P}|$. \square

Chapter 4

Deriving Interconnection Semantics

Expressing explicitly all the patterns of an interconnection semantics is not always convenient or practical. As an alternative, we will present several interconnection semantics that can be derived automatically. When an interconnection semantics \mathcal{P} is described implicitly, it may be very large or hard to compute. Therefore, it is not appropriate to measure the complexity of interconnectivity by assuming that \mathcal{P} is part of the input. Hence, the complexity analysis for the automatically-derived interconnection semantics will be in terms of the given document D and the given set of objects O . For formal reasons, sometimes the given schema S is also part of the input, although in practice the size of D is usually bigger than the size of S .

4.1 The Interconnection Semantics $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$

Intuitively, a set of objects O is $\mathcal{P}_{\text{all}}^r$ -interconnected if O is contained in a hierarchy that does not include two objects of the same type. Formally, $\mathcal{P}_{\text{all}}^r$ is the set of all rooted patterns. In principle, $\mathcal{P}_{\text{all}}^r$ has infinitely many patterns, but for a given document D , only the finite subset of patterns with labels from D is relevant. The interconnection semantics $\mathcal{P}_{\text{all}}^u$ is the set of all undirected patterns. $\mathcal{P}_{\text{all}}^r$ is more restrictive than $\mathcal{P}_{\text{all}}^u$ in the sense that if $\mathcal{P}_{\text{all}}^r \models_D O$, then $\mathcal{P}_{\text{all}}^u \models_D O$. The following observation characterizes interconnectivity

according to these semantics.

Observation 4.1 *A set of objects O is $\mathcal{P}_{\text{all}}^r$ -interconnected ($\mathcal{P}_{\text{all}}^u$ -interconnected) in a document D if and only if D has a uniquely-labeled rooted (undirected) subtree T that contains O .*

The advantage of $\mathcal{P}_{\text{all}}^u$ is the ability to interconnect objects even when they are not part of a uniquely-labeled hierarchy, as was shown earlier for document D_1 (Figure 1.1) using the undirected pattern $(\{\text{Name}, \text{Employee}\}, C_3)$ (this example was discussed on page 9).

In this section, we analyze the complexity of using $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$. For tree documents, testing interconnectivity according to each of these semantics is clearly tractable (see [2] for details). Unfortunately, for general documents, it becomes intractable. Nevertheless, we will show an efficient algorithm that tests $\mathcal{P}_{\text{all}}^r$ -interconnectivity for bounded sets of objects in acyclically-labeled documents (i.e., documents that conform to acyclic schemas). Using these results, we will study the complexity of computing interconnection queries for the semantics $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$.

The following lemma shows that testing $\mathcal{P}_{\text{all}}^r$ -interconnection is generally NP-complete.

Lemma 4.2 *Let O be a set of objects in a document D . Testing $\mathcal{P}_{\text{all}}^r \models_D O$ is NP-complete even if $|O| = 2$ and D is acyclic (but not necessarily acyclically labeled).*

Proof. Observe that an appropriate subgraph of the document is a polynomial-time-testable evidence of interconnection, hence this problem is in NP.

To prove NP-hardness of $\mathcal{P}_{\text{all}}^r \models_D O$, where $|O| = 2$ and D is acyclic, we use by a simple reduction from the following problem. Given two objects o_1 and o_2 in an acyclic o-graph G , determine whether o_2 is reachable from o_1 through a uniquely-labeled directed path in G . In Chapter 8 we show that this problem is NP-complete (Proposition 8.1). So, let o_1 and o_2 be two objects in a given acyclic o-graph G . Let D be the subgraph of G that is induced by all objects reachable from o_1 through a directed path. Then, D is a document that is rooted at object o_1 . Since G is acyclic, object o_1 has no incoming edges in D . Hence, it holds that $\mathcal{P}_{\text{all}}^r \models_D \{o_1, o_2\}$ if and only if some directed path in G , from o_1 to o_2 , is uniquely labeled. \square

The following corollary shows that computing interconnection queries that involve $\mathcal{P}_{\text{all}}^r$ is intractable as well.

Corollary 4.3 *Let D be a document and L be a set of labels. Let q be the interconnection query $(L, \mathcal{P}_{\text{all}}^r)$. Testing $\mathcal{R}(q, D) \neq \emptyset$ is NP-complete even if $|L| = 2$ and D is acyclic (but not necessarily acyclically labeled).*

Proof. For membership in NP, observe that the following is a polynomial-time-testable evidence of the non-emptiness of $\mathcal{R}(q, D)$:

- A set O of objects in D such that $l(O) = L$, and
- A rooted subtree $T \subseteq D$ such that T is uniquely labeled and T contains the objects in O .

To prove NP-hardness, we use the following reduction from the problem of testing $\mathcal{P}_{\text{all}}^r \models_D O$. Given a document D and a set O of objects in D , we define the document \hat{D} , as follows.

- \hat{D} contains all objects and edges in D ;
- For each object $o \in O$, document \hat{D} contains an additional new object \bar{o} , that has a unique label;
- For each $o \in O$, document \hat{D} has an edge from o to \bar{o} .

Note that if D is acyclic, then so is \hat{D} . Let $L = \{l(\bar{o}) \mid o \in O\}$ (i.e., the set of new labels that were added to the document). Let q be the interconnection query $(L, \mathcal{P}_{\text{all}}^r)$. Then, $\mathcal{P}_{\text{all}}^r \models_D O$ if and only if $\mathcal{R}(q, \hat{D}) \neq \emptyset$. Hence, the reduction is correct. \square

If the size of O is not fixed, testing $\mathcal{P}_{\text{all}}^r \models_D O$ is NP-complete even if D is acyclically labeled. This is shown in the next lemma.

Lemma 4.4 *Let O be a set of objects in a document D . Testing $\mathcal{P}_{\text{all}}^r \models_D O$ is NP-complete even if S_D is acyclic (but the size of O is unbounded).*

Proof. Membership of this problem in NP is shown as in the proof of Lemma 4.2. We prove NP-hardness by using a reduction from 3SAT. Let $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$ be a conjunction of clauses over the variables v_1, \dots, v_n , where each C_i a disjunction of 3 literals. We construct the following document D_F .

- The root of D_F is labeled with \mathbf{r} .

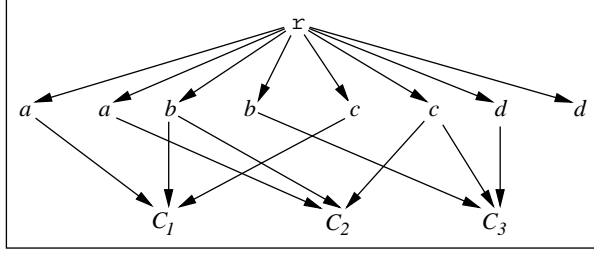


Figure 4.1: The document D_F for the formula $F = (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg b \vee \neg c \vee d)$.

- For each variable v_k , there are two objects that correspond to the literals v_k and $\neg v_k$. Both objects are labeled with v_k and have incoming edges from the root.
- For each clause C_i , there is an object o_i , such that o_i is labeled with C_i and o_i has incoming edges from the objects that correspond to the literals in C_i .

As an examples, Figure 4.1 shows the document D_F that is built for the instance $F = (a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg b \vee \neg c \vee d)$. Note that the edges of S_{D_F} are either from r to variables or from variables to clauses. Hence, D_F is acyclically labeled. Let $O = \{o_1, \dots, o_m\}$. We complete our proof by proving the following claim.

Claim 1 F is satisfiable if and only if $\mathcal{P}_{\text{all}}^r \models_{D_F} O$.

For the “if” direction, suppose that T is a subtree of D_F , such that T is reduced w.r.t. O and T is uniquely labeled. Observe that in T , every object o_i has one incoming edge, from an object that corresponds to some literal in C_i . Since T has no label repetition, T does not contain any two objects that correspond to contradictory literals (i.e., v_k and $\neg v_k$ for some k). Hence, the objects in T that correspond to literals represent a satisfying assignment for F . For the “only if” direction, suppose that F can be satisfied by simultaneously satisfying the literals l_1, \dots, l_n , where $l_k \in \{v_k, \neg v_k\}$. Consider the subgraph G of D_F that is induced by the root, the objects in O , and the objects that correspond to the literals l_1, \dots, l_n . Then, G is a rooted subgraph of D_F and G is uniquely labeled. Hence, G is an evidence for $\mathcal{P}_{\text{all}}^r \models_{D_F} O$, as required. \square

The proof of the next corollary is similar to the proof of Corollary 4.3.

Corollary 4.5 Let $q = (L, \mathcal{P}_{\text{all}}^r)$ be an interconnection query and D be a document. Testing $\mathcal{R}(q, D) \neq \emptyset$ is NP-complete even if S_D is acyclic (but the size of L is unbounded).

$\mathcal{P}_{\text{all}}^r\text{-ARRAY}(D, k)$ 1: let l_1, \dots, l_m be the labels of S_D , sorted in any topological order 2: let o_1, \dots, o_n be the objects of D , sorted by the order implied on their labels 3: for all object $o \in D$ do $I[\{o\}] := \mathbf{true}$ 4: for $i := 1, \dots, n$ do 5: for all $O \subseteq \{o_1, \dots, o_{i-1}\}$ such that $1 \leq O < k$ do 6: if ($O \cup \{o_i\}$ is uniquely labeled) and (o_i has a parent o s.t. $I[O \cup \{o\}] = \mathbf{true}$) then 7: $I[O \cup \{o_i\}] := \mathbf{true}$ 8: else $I[O \cup \{o_i\}] := \mathbf{false}$ 9: return I

Figure 4.2: A polynomial-time algorithm for computing $\mathcal{P}_{\text{all}}^r$ -interconnection for all sets of size at most k of objects in D .

Thus far, we have shown that testing $\mathcal{P}_{\text{all}}^r \models_D O$ is NP-complete, even if one of the following conditions hold.

- D is acyclically labeled;
- O is of fixed size.

However, if both conditions hold, this problem becomes tractable. Next, we present a polynomial-time computation of $\mathcal{P}_{\text{all}}^r \models_D O$ provided that the two conditions above are satisfied. Algorithm $\mathcal{P}_{\text{all}}^r\text{-ARRAY}(D, k)$ uses dynamic programming in order to determine whether $\mathcal{P}_{\text{all}}^r \models_D O$, for all sets O of objects in D such that $|O| \leq k$. The result of this algorithm is a binary array I , where the indices of I are the sets with k or fewer objects in D . The value of $I[O]$ is **true** if $\mathcal{P}_{\text{all}}^r \models_D O$, and **false** otherwise. In this algorithm, we traverse the objects of D according to a topological order implied by their labels in S_D . Let o_1, \dots, o_n be the objects of D , sorted in that order. When an object o_i is visited, we consider all object sets O such that

- $|O| < k$ and
- O is a subset of the set $\{o_1, \dots, o_{i-1}\}$ (that is, all objects in O precede o_i in the traversal order).

We set the value of $I[O \cup \{o_i\}]$ to **true** if the following two conditions hold.

1. The objects in $O \cup \{o_i\}$ are uniquely labeled;
2. $I[O \cup \{o\}] = \mathbf{true}$ for some parent o of o_i .

If either one of these conditions does not hold, we set $I[O \cup \{o_i\}]$ to **false**. For Condition 2, note that all the parents of an object precede that object in the order o_1, \dots, 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. The correctness of the algorithm follows immediately from the next lemma.

Lemma 4.6 *Let D be a document such that S_D is acyclic. Let O be a set of two or more uniquely-labeled objects in D . Suppose that $o \in O$ is an object such that $l(o)$ is last among the labels in $l(O)$, according to the topological order implied by S_D . Then, $\mathcal{P}_{\text{all}}^r \models_D O$ if and only if $\mathcal{P}_{\text{all}}^r \models_D (O \setminus \{o\} \cup \{\hat{o}\})$ for some parent \hat{o} of o .*

Proof. For the “if” direction, suppose that \hat{o} is a parent of o , and that $\mathcal{P}_{\text{all}}^r \models_D \hat{O}$, where $\hat{O} = O \setminus \{o\} \cup \{\hat{o}\}$. Let T_1 be a rooted subtree of D , such that T_1 is uniquely labeled and reduced w.r.t \hat{O} . We use the following claim.

Claim 1 *None of the objects in T_1 are labeled with $l(o)$.*

Consider the tree T_2 that is obtained by adding to T_1 the edge from \hat{o} to o . By Claim 1, it follows that the tree T_2 is uniquely labeled. Since T_2 contains the objects in O , it holds that $\mathcal{P}_{\text{all}}^r \models_D O$, as required. Thus, for the “if” direction, it is sufficient to prove Claim 1. Assume, by way of contradiction, that T_1 contains an object o' that is labeled with $l(o)$. Observe that object o' cannot be a parent of o , since S_D is acyclic. Hence, $o' \neq \hat{o}$. Furthermore, o' cannot be any of the objects in $O \setminus \{o\}$, since o is the only object in O that is labeled with $l(o)$. We conclude that $o' \notin \hat{O}$. Since T_1 is reduced w.r.t. \hat{O} , one of the objects in \hat{O} must be reachable from o' through a directed path in T_1 . This contradicts the fact that all labels in $l(\hat{O})$ precede the label l in the topological order implied by S_D .

For the “only if” direction, assume that $\mathcal{P}_{\text{all}}^r \models_D O$. Let T be a rooted subtree of D , such that T is uniquely labeled and T is reduced w.r.t. O . Since $l(o)$ is the last label of $l(O)$ in the topological order, object o cannot be the root of T . Let object \hat{o} be the parent of o in the subtree T , and \hat{O} be the set $O \setminus \{o\} \cup \{\hat{o}\}$. Then, T is a uniquely-labeled rooted subtree that contains all objects in \hat{O} . Hence, $\mathcal{P}_{\text{all}}^r \models_D \hat{O}$, as required. \square

The running time of the algorithm $\mathcal{P}_{\text{all}}^r\text{-ARRAY}(D, k)$ (shown in Figure 4.2) is considered in following proposition.

Proposition 4.7 *Let k be some fixed positive integer and D be a document such that S_D is acyclic. Algorithm $\mathcal{P}_{\text{all}}^r\text{-ARRAY}(D, k)$ runs in time $O(mn^{k-1})$, where n and m are the number of objects and edges in D , respectively.*

Proof. Let o_1, \dots, o_n be the objects of D , in the traversal order of the algorithm. Let $d_{\text{in}}(o_i)$ be the in-degree of object o_i . When object o_i is visited, the number of steps is bounded by the number of possibilities to choose a parent o of o_i and a subset $O \subseteq \{o_1, \dots, o_{i-1}\}$ such that $|O| < k$. Hence, considering k as fixed, the number of steps that are executed when visiting object o_i is

$$O \left[d_{\text{in}}(o_i) \left(\binom{i-1}{1} + \binom{i-1}{2} + \dots + \binom{i-1}{k-1} \right) \right] = O \left(d_{\text{in}}(o_i) \cdot (i-1)^{k-1} \right).$$

We conclude that total running time of the algorithm is

$$O \left(\sum_{i=1}^n d_{\text{in}}(o_i) \cdot (i-1)^{k-1} \right) = O \left(n^{k-1} \sum_{i=1}^n d_{\text{in}}(o_i) \right) = O \left(mn^{k-1} \right),$$

as required. □

Corollary 4.8 *Let D be a document and O be a set of objects in D . If S_D is acyclic and O is of fixed size, then $\mathcal{P}_{\text{all}}^r \models_D O$ can be tested in polynomial time.*

We will now consider the problem of testing $\mathcal{P}_{\text{all}}^u$ -interconnection. Unlike the directed case, testing $\mathcal{P}_{\text{all}}^u \models_D O$ is NP-complete even if $|O|$ is of fixed size and D is acyclically labeled.

Lemma 4.9 *Let O be a set of objects in a document D . Testing $\mathcal{P}_{\text{all}}^u \models_D O$ is NP-complete even if $|O| = 2$ and S_D is acyclic.*

Proof. For membership in NP, observe that an appropriate subgraph of D can be used as a polynomial-time-testable evidence for $\mathcal{P}_{\text{all}}^u$ -interconnection. To prove NP-hardness, we will use a reduction from the following problem. Given an o-graph G and two objects o_1 and o_2 in G , determine whether G contains an undirected path P between o_1 and o_2 ,

such that P is uniquely labeled. We prove that this problem is NP-complete in Chapter 8 (Proposition 8.1). Let G be a directed o-graph and let o_1 and o_2 be two objects in G . We will construct a document D that has the following properties:

- D contains all the objects of G ;
- D is acyclically labeled; *and*
- $\mathcal{P}_{\text{all}}^u \models_D \{o_1, o_2\}$ if and only if G contains an undirected path between o_1 and o_2 , that is uniquely labeled.

Our construction consists of the following steps.

1. We define D to be the o-graph that contains all the objects of G and no edges at all.
2. We arbitrarily choose a complete order on the labels of the objects in D .
3. For each edge e in G with incident objects o and o' , we add to D the edge (o, o') if label $l(o)$ precedes label $l(o')$ according to the order defined in 2.
4. We choose a new label l . For each object o in D , we add to D a new object \hat{o} that is labeled with l and an edge from \hat{o} to o .
5. We add to D a new object r , with a new label. We then add to D edges from r to each of the new objects \hat{o} , which were added in 4.

In Steps 1–3, we define D to be similar to G , except for the directions of the edges; the edges of D are directed in an acyclic manner. At that time, D may not be a legal document, since it may have no root. In Steps 4 and 5, we add to D the root r , and connect r to each object o through a third object \hat{o} . All the new objects \hat{o} have the same label l . This ensures that every undirected path between objects o_1 and o_2 in D must also be a path in G , unless this path has at least two objects that are labeled with l . Hence, $\mathcal{P}_{\text{all}}^u \models_D \{o_1, o_2\}$ is equivalent to G having an undirected path between o_1 and o_2 that is uniquely labeled. \square

The next corollary can be proved similarly to Corollary 4.3.

Corollary 4.10 *Let L be a set of labels and q be the interconnection query $(L, \mathcal{P}_{\text{all}}^u)$. Given a document D , testing $\mathcal{R}(q, D) \neq \emptyset$ is NP-complete even if $|L| = 2$ and S_D is acyclic.*

Interestingly, among the cases considered in this work, the only ones for which testing $\mathcal{P}_{\text{all}}^u$ -interconnection is not NP-hard, are ones in which $\mathcal{P}_{\text{all}}^u$ is equivalent to $\mathcal{P}_{\text{all}}^r$. This is the case, for example, when D is a tree or when S_D is a tree. The case where S_D is a tree is considered later in this chapter.

In the next theorem, we summarize the complexity results that were proved in this section.

Theorem 4.11 *Let D be a document, O be a subset of the objects in D and L be a set of labels. Let q_r and q_u be the interconnection queries $(L, \mathcal{P}_{\text{all}}^r)$ and $(L, \mathcal{P}_{\text{all}}^u)$, respectively.*

1. *If O is of constant size and S_D is acyclic, then testing $\mathcal{P}_{\text{all}}^r \models_D O$ is in polynomial time in the size of D ; otherwise, testing $\mathcal{P}_{\text{all}}^r \models_D O$ is NP-complete.*
2. *Testing non-emptiness of $\mathcal{R}(q_r, D)$ is NP-complete, even if L is of constant size or S_D is acyclic (but not both).*
3. *Testing $\mathcal{P}_{\text{all}}^u \models_D O$ is NP-complete, even if O is of constant size and S_D is acyclic.*
4. *Testing non-emptiness of $\mathcal{R}(q_u, D)$ is NP-complete, even if L is of constant size and S_D is acyclic.*

Proof. Part 1 of the theorem follows from Lemma 4.2, Lemma 4.4 and Corollary 4.8. Part 2 follows from Corollaries 4.3 and 4.5. Part 3 was shown in Lemma 4.9. Part 4 was shown in Corollary 4.10. □

4.2 The Interconnection Semantics $\mathcal{P}_{\text{min}}^r(S)$ and $\mathcal{P}_{\text{min}}^u(S)$

The semantics $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$ occasionally interconnect objects that are rather weakly related to each other. For example, in document D_1 (Figure 1.1), object 11 (the name of a project) and object 8 (the URL of a manager) are $\mathcal{P}_{\text{all}}^r$ -interconnected.

The above anomaly can be avoided by adopting the common assumption (e.g., [1]) that meaningfully-related objects must be close to each other. This is captured by the interconnection semantics $\mathcal{P}_{\text{min}}^r$ and $\mathcal{P}_{\text{min}}^u$, described as follows.

A rooted (undirected) *Steiner tree* of a set U of nodes in a graph G is a rooted (undirected) tree T that has the following properties.

1. T contains all the nodes in U ; and
2. The number of nodes in T is minimal among all rooted (undirected) subtrees of G that satisfy 1.

Note that a Steiner tree of a set U of objects must be reduced w.r.t. U . The notion of minimal patterns is defined for a given set of labels L and a given schema S as follows. A rooted (undirected) pattern $p = (L, C)$ is *minimal* w.r.t. S if C is a subtree of S and C is a rooted (undirected) Steiner tree of L in S .

For example, consider the schema S_1 and the l-trees C_1 and C_2 , depicted in Figure 1.2. The pattern $(\{\text{Name}, \text{Email}\}, C_1)$ is minimal w.r.t. S_1 , while the pattern $(\{\text{Name}, \text{Email}\}, C_2)$ is not.

Formally, given a schema S , the interconnection semantics $\mathcal{P}_{\min}^r(S)$ is the set of all rooted patterns that are minimal w.r.t. S . Note that defining $\mathcal{P}_{\min}^r(S)$ 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 $\mathcal{P}_{\min}^r(S)$ gracefully handles missing information. Consider, for example, document D_1 (Figure 1.1) and schema S_1 (Figure 1.2). The fact that S_1 has an edge from **Project** to **URL** implies that the URL of object 10 is missing from D_1 . The semantics $\mathcal{P}_{\min}^r(S_1)$ does not interconnect object 10 with any object labeled with **URL**, because the rooted pattern $(\{\text{Project}, \text{URL}\}, C)$ is minimal w.r.t. S_1 , where C is the edge from **Project** to **URL**.

The interconnection semantics $\mathcal{P}_{\min}^u(S)$ is defined similarly to $\mathcal{P}_{\min}^r(S)$; that is, $\mathcal{P}_{\min}^u(S)$ is the set of all undirected patterns that are minimal w.r.t. S . The semantics \mathcal{P}_{\min}^r and \mathcal{P}_{\min}^u are incomparable; that is, some sets of objects are \mathcal{P}_{\min}^r -interconnected but not \mathcal{P}_{\min}^u -interconnected and vice versa.

The next proposition gives the following characterization of \mathcal{P}_{\min}^r -interconnectivity (\mathcal{P}_{\min}^u -interconnectivity). A set of objects is \mathcal{P}_{\min}^r -interconnected (\mathcal{P}_{\min}^u -interconnected) if and only if it is contained in a sufficiently small rooted (connected) subgraph of the document.

Proposition 4.12 *Let D be a document that conforms to the schema S . $\mathcal{P}_{\min}^r(S) \models_D O$ ($\mathcal{P}_{\min}^u(S) \models_D O$) if and only if the size of a rooted (undirected) Steiner tree of O in D is equal to the size of a rooted (undirected) Steiner tree of $l(O)$ in S .*

Proof. We prove this proposition for the directed case. The proof of the undirected case is similar. Let $T \subseteq D$ be a rooted subtree of D that contains all object in O . Let $C \subseteq S$ be the subgraph of S induced by $l(O)$. Note that the mapping $o \rightarrow l(o)$, from the objects of T to the labels of C , defines a homomorphism between T and C . Hence, the following facts hold.

Fact 1 *If all objects in T are reachable from some object o through a directed path, then all labels in C are reachable from the label $l(o)$ through a directed path.*

Fact 2 *If T is uniquely labeled, then T and C have the same number of nodes. Otherwise, the number of objects in T is strictly larger than the number of labels in C .*

For the “if” direction of the proposition, suppose that the size of T is the size of a directed Steiner tree of $l(O)$ in S . Fact 1 implies that C has a subtree that is reduced w.r.t. $l(O)$. Hence, from Fact 2, the subtree T must be uniquely labeled. We conclude that T is isomorphic to a subtree of S that is a rooted Steiner tree of $l(O)$, and hence $\mathcal{P}_{\min}^r \models_D O$. For the “only if” direction, we observe that Facts 1 and 2 also imply that a directed Steiner tree of O in D is always at least as large as a directed Steiner tree of $l(O)$ in S . Hence, if T is isomorphic to a rooted Steiner tree of $l(O)$ in C , then T must be a rooted Steiner tree of O in D , as required. \square

As a result of Proposition 4.12, testing interconnectivity, according to either \mathcal{P}_{\min}^r or \mathcal{P}_{\min}^u , is tractable for small sets of objects. We formalize this in the following corollary.

Corollary 4.13 *Let D be a document that conforms to a schema S . If O is of constant size, then testing either $\mathcal{P}_{\min}^r(S) \models_D O$ or $\mathcal{P}_{\min}^u(S) \models_D O$ is polynomial in the size of D and S .*

Proof. By Proposition 4.12, in order to test $\mathcal{P}_{\min}^r(S) \models_D O$ ($\mathcal{P}_{\min}^u(S) \models_D O$), one has to compare the size of a rooted (undirected) Steiner tree of O in D with the size of a rooted (undirected) Steiner tree of $l(O)$ in S . In [4] it was shown that a rooted (undirected) Steiner tree can be efficiently found for a fixed number of nodes. Hence, both $\mathcal{P}_{\min}^r(S) \models_D O$ and $\mathcal{P}_{\min}^u(S) \models_D O$ can be efficiently tested when the size of O is fixed. \square

Unfortunately, both \mathcal{P}_{\min}^r -interconnection and \mathcal{P}_{\min}^u -interconnection are intractable for unbounded sets of objects. Similarly, interconnection queries, involving either \mathcal{P}_{\min}^r or \mathcal{P}_{\min}^u , are hard to compute without a boundedness assumption. This is shown in the next theorem.

Theorem 4.14 *Let D be a document that conforms to the schema S , O be a set of objects in D and L be a set of labels. Let q_r be the interconnection query $(L, \mathcal{P}_{\min}^r)$ and q_u be the interconnection query $(L, \mathcal{P}_{\min}^u)$. Testing each one of the following is both NP-hard and coNP-hard, and is in $\Sigma_2^P \cap \Pi_2^P$.*

1. $\mathcal{P}_{\min}^r(S) \models_D O$.
2. $\mathcal{P}_{\min}^u(S) \models_D O$.
3. $\mathcal{R}(q_r, D) \neq \emptyset$.
4. $\mathcal{R}(q_u, D) \neq \emptyset$.

Proof. We will prove the hardness of these problems by reductions from *Set Cover* (SCP) to all four problems and their complements. SCP is defined as follows. Given a set \mathcal{X} , a collection \mathcal{C} of subsets of \mathcal{X} , and a positive integer d , decide whether \mathcal{X} can be covered by d or fewer of the subsets of \mathcal{C} .

Our first reduction is from SCP to the four problems. Let $(\mathcal{X}, \mathcal{C}, d)$ be an instance of SCP. Let $\mathcal{X} = \{x_1, \dots, x_n\}$ and $\mathcal{C} = \{A_1, \dots, A_m\}$. We define a document D with the following properties:

- The root of D is an object r that has a unique label l_r ;
- r has a child o_i^a that is labeled with A_i , for each subset $A_i \in \mathcal{C}$;
- D has an object o_j^x that is labeled with x_j , for each element $x_j \in \mathcal{X}$; and
- D has an edge from o_i^a to o_j^x , for each $A_i \in \mathcal{C}$ and $x_j \in \mathcal{X}$ such that $x_j \in A_i$.

Note that document D is uniquely labeled and hence, D is isomorphic to S_D . Let C_1 be the l-graph S_D . Let C_2 be an l-graph that forms a path $l_1 \rightarrow \dots \rightarrow l_d$ of d unique labels. Let S be the schema that is obtained by connecting the l-graphs C_1 and C_2 as follows. There is an

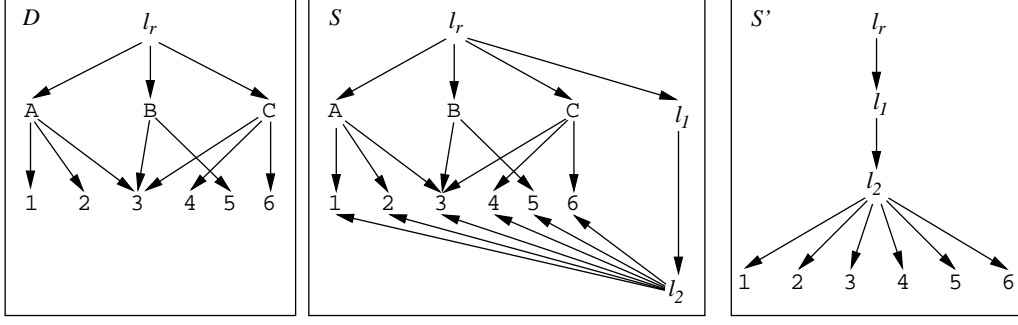


Figure 4.3: The document D and the schemas S and S' constructed for the SCP instance $(\mathcal{X}, \mathcal{C}, d)$, where $d = 2$, $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$, $\mathcal{C} = \{A, B, C\}$, $A = \{1, 2, 3\}$, $B = \{3, 5\}$ and $C = \{3, 4, 6\}$.

edge from the label l_r to the label l_1 , and edges from l_d to each of the labels x_1, \dots, x_n . Let S' be the subgraph of S induced by the set of labels $\{l_r, l_1, \dots, l_d, x_1, \dots, x_n\}$. Note that S' is a rooted tree, and hence, a schema. For example, Figure 4.3 depicts the document D , the schema S and the schema S' that are constructed for $d = 2$, $\mathcal{X} = \{1, 2, 3, 4, 5, 6\}$, and $\mathcal{C} = \{A, B, C\}$, where $A = \{1, 2, 3\}$, $B = \{3, 5\}$ and $C = \{3, 4, 6\}$. Let O be the set $\{r, o_1^x, \dots, o_n^x\}$ and L be the set $l(O)$ (i.e., $L = \{l_r, x_1, \dots, x_n\}$). Let q_r and q_u be the interconnection queries $(L, \mathcal{P}_{\min}^r(S))$ and $(L, \mathcal{P}_{\min}^u(S))$, respectively. We will show that the following hold.

Claim 1 $\mathcal{P}_{\min}^r(S) \models_D O$ if and only if \mathcal{X} can be covered by d or fewer subsets in \mathcal{C} .

Claim 2 $\mathcal{P}_{\min}^u(S) \models_D O$ if and only if \mathcal{X} can be covered by d or fewer subsets in \mathcal{C} .

Claim 3 $\mathcal{R}(q_r, D) \neq \emptyset \Leftrightarrow \mathcal{R}(q_u, D) \neq \emptyset \Leftrightarrow \mathcal{X}$ can be covered by d or fewer subsets in \mathcal{C} .

Claim 1 is proved as follows. For the “if” direction, assume that \mathcal{C} contains a cover A_{i_1}, \dots, A_{i_l} of \mathcal{X} , where $l \leq d$. Let \hat{C} be the rooted subgraph of S induced by the labels in $\{l_r, x_1, \dots, x_n, A_{i_1}, \dots, A_{i_l}\}$. Thus, \hat{C} contains the labels in L and has at most $1 + n + d$ nodes. If a directed Steiner tree of L in S has one of the labels l_1, \dots, l_d , then it must include all the labels in S' and hence, have a size of at least $1 + n + d$. Therefore, some rooted Steiner tree of L is a subgraph of C_1 . This Steiner tree is isomorphic to a subtree of D that contains the objects in O , since D is isomorphic to C_1 . This shows that $\mathcal{P}_{\min}^r(S) \models_D O$, as claimed.

For the “only if” direction, suppose that $\mathcal{P}_{\min}^r(S) \models_D O$. From Proposition 4.12, it follows that D has a rooted subtree T that contains all objects of O and has the size of a rooted Steiner tree of L in S . In particular, T has at most $1 + n + d$ objects, since this is the number of labels in S' . Hence, T contains at most d of the objects that have labels from \mathcal{C} , and the labels of these objects form a cover of \mathcal{X} , as required.

The proof of Claim 2 is similar to the proof of Claim 1. For Claim 3, consider the document D , the set O and the set L that were defined in the proof of Claim 1. Let q_r and q_u denote the interconnection queries $(L, \mathcal{P}_{\min}^r(S))$ and $(L, \mathcal{P}_{\min}^u(S))$, respectively. It holds that $\mathcal{P}_{\min}^r(S) \models_D O$ if and only if $\mathcal{R}(q_r, D) \neq \emptyset$, because D is uniquely labeled. Similarly, $\mathcal{P}_{\min}^u(S) \models_D O$ if and only if $\mathcal{R}(q_u, D) \neq \emptyset$.

Claims 1–3 show that our reduction is correct and hence, all four problems are NP-hard.

Next, we will show a reduction from X3C to the complement of the four problems. For this reduction, we will modify the construction shown in the proof of Claim 1. Let $(\mathcal{X}, \mathcal{C}, d')$ be an instance of X3C, and let $d = d' + 1$. Consider the schemas S and S' and the object set O that were constructed for the instance $(\mathcal{X}, \mathcal{C}, d)$ in the proof of Claim 1. We construct a document D' that is isomorphic to the schema S' . Hence, D' is a tree document that is reduced w.r.t. O , and D' has $2 + n + d'$ objects. Using arguments similar to the ones used in the proofs of Claims 1 and 2, the following can be shown.

Claim 4 $\mathcal{P}_{\min}^r(S) \models_{D'} O \Leftrightarrow \mathcal{P}_{\min}^u(S) \models_{D'} O \Leftrightarrow$ *the minimal number of sets in \mathcal{C} that are needed in order to cover \mathcal{X} is greater than d .*

This shows that testing either \mathcal{P}_{\min}^r -interconnectivity or \mathcal{P}_{\min}^u -interconnectivity is coNP-hard. The proof of coNP-hardness of $\mathcal{R}(q_r, D) \neq \emptyset$ and $\mathcal{R}(q_u, D) \neq \emptyset$ is similar to the proof of Claim 3.

Membership in Σ_2^P of all four problems is obtained by observing the following facts. Note that these facts are direct results of Proposition 4.12.

- $\mathcal{P}_{\min}^r(S) \models_D O$ ($\mathcal{P}_{\min}^u(S) \models_D O$) if and only if there exists a rooted (undirected) subtree $T \subseteq D$, such that T is reduced w.r.t. O and for all rooted (undirected) subtrees $C \subseteq S$, if C contains $l(O)$, then C is not smaller than T .
- $\mathcal{R}(q_r, D) \neq \emptyset$ ($\mathcal{R}(q_u, D) \neq \emptyset$) if and only if there exists a subset O of the objects in D , such that $l(O) = L$ and $\mathcal{P}_{\min}^r(S) \models_D O$ ($\mathcal{P}_{\min}^u(S) \models_D O$).

Similarly, the following facts show membership of our problems in Π_2^P .

- $\mathcal{P}_{\min}^r(S) \models_D O$ ($\mathcal{P}_{\min}^u(S) \models_D O$) if and only if for every rooted (undirected) subtree $C \subseteq S$ that contains $l(O)$, there exists a rooted (undirected) subtree $T \subseteq D$ that contains O and has at most the number of nodes that C has.
- $\mathcal{R}(q_r, D) \neq \emptyset$ ($\mathcal{R}(q_u, D) \neq \emptyset$) if and only if for every rooted (undirected) subtree $C \subseteq S$ that contains L , there exists a rooted (undirected) subtree $T \subseteq D$ such that T is not larger than C and for every label $l \in L$, the subtree T contains an object that is labeled with l .

□

4.3 The Semantics $\mathcal{P}_{uca}^r(S)$

The semantics $\mathcal{P}_{\min}^r(S)$ and $\mathcal{P}_{\min}^u(S)$ may occasionally miss meaningfully related objects just because they are slightly too far away from each other. For example, consider document D_1 (Figure 1.1), schema S_1 and the rooted l-trees C_1 and C_2 (Figure 1.2). The semantics $\mathcal{P}_{\min}^r(S_1)$ does not include the pair (11, 13) as an answer for the set of labels $\{\mathbf{Name}, \mathbf{Email}\}$, because the pattern $(\{\mathbf{Name}, \mathbf{Email}\}, C_2)$ 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 $(\{\mathbf{Name}, \mathbf{Email}\}, C_1)$ and $(\{\mathbf{Name}, \mathbf{Email}\}, C_2)$ 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 its root, that is a common ancestor of L in S . For example, the rooted patterns $(\{\mathbf{Name}, \mathbf{Email}\}, C_1)$ and $(\{\mathbf{Name}, \mathbf{Email}\}, C_2)$ are structurally minimal w.r.t. the schema S_1 (Figure 1.2); in fact, these are the only two rooted patterns for the set of labels $\{\mathbf{Name}, \mathbf{Email}\}$ that are structurally minimal w.r.t. S_1 .

Given an acyclic schema S , the interconnection semantics $\mathcal{P}_{uca}^r(S)$ (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 $\mathcal{P}_{\text{uca}}^r$ does not have an analogous undirected semantics.

For document D_1 (Figure 1.1) and schema S_1 (Figure 1.2), the set of $\mathcal{P}_{\text{uca}}^r(S_1)$ -interconnected pairs of objects for $\{\text{Name}, \text{Email}\}$ consists of (5, 6), (19, 20) and (11, 13).

The complexity of using $\mathcal{P}_{\text{uca}}^r$ is considered in the next theorem.

Theorem 4.15 *Let D be a document that conforms to an acyclic schema S . Let O be a subset of the objects in D , L be a set of labels and q be the interconnection query $(L, \mathcal{P}_{\text{uca}}^r(S))$.*

1. *Testing $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ is NP-complete.*
2. *If O is of constant size, then testing $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ is in polynomial time in the size of D and S ;*
3. *Testing non-emptiness of $\mathcal{R}(q, D)$ is NP-complete.*

Proof. Claim 1 is proved as follows. Testing $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ is in NP since, given a subtree T of D , one can efficiently test whether the following hold:

- T is reduced w.r.t. O ; and
- Among the objects in T , only the root of T has a label that is a common ancestor of $l(O)$ in S .

To prove NP-hardness, we use the reduction from 3SAT that was used in the proof of Lemma 4.4. Let F be a conjunction of 3-variable clauses (i.e., an instance of 3SAT). Consider the document D_F and the object set O that were constructed for F in the reduction of the proof of Lemma 4.4. We may assume that none of the variables in F is common to all clauses, since satisfiability of F can be efficiently tested if there is such a variable. Under this assumption, the root of S is the only common ancestor of $l(O)$ in S . Therefore, $\mathcal{P}_{\text{all}}^r \models_{D_F} O$ if and only if $\mathcal{P}_{\text{uca}}^r(S) \models_{D_F} O$. Hence, testing $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ is NP-hard.

To prove Claim 2, we use a reduction to a tractable case of testing $\mathcal{P}_{\text{all}}^r$ -interconnection, as follows. Let L_a be the set of labels that are common ancestors of $l(O)$ in S . Let l_1 and l_2 be two distinct new labels. For each label $l \in L_a$, we create the document D_l that is obtained from D as follows.

- We remove from D all objects o such that $l(o) \in L_a \setminus \{l\}$;
- We add to D a new object r that is labeled with l_1 ;
- For each object o in D , we add to D a new object \hat{o} that is labeled with l_2 ;
- For each object o in D , we add to D an edge from r to \hat{o} and an edge from \hat{o} to o .

We use the following observations regarding a label $l \in L_a$.

Observation 1 *If T is a uniquely-labeled subtree of D_l that is reduced w.r.t. O , then T is also a subtree of D .*

Observation 2 *Document D_l is acyclically labeled.*

Using Observation 1, it can be shown that $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ if and only if $\mathcal{P}_{\text{all}}^r \models_{D_l} O$ for some $l \in L_a$. Observation 2 and Corollary 4.8 imply that $\mathcal{P}_{\text{all}}^r \models_{D_l} O$ can be tested in polynomial time. We conclude that testing $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ is in polynomial time, as claimed.

The proof of Claim 3 is similar to the proof of Corollary 4.3. □

4.4 Interconnections in Tree Schemas

In this section, we discuss the usage of interconnection semantics in documents that conform to tree schemas. Note that a document that conforms to a tree schema is not necessarily a tree. We will show that for these documents, all the above interconnection semantics are equivalent and the complexity is polynomial, even for unbounded sets of objects. Our results are derived from the following observation.

Observation 4.16 *Let U be a set of nodes in a rooted tree T . Then, T has exactly one rooted subtree T_r and exactly one undirected subtree T_u , such that T_r and T_u are reduced w.r.t. U . Furthermore, T_u is obtained from T_r by ignoring the directions of the edges.*

Using this observation, we prove the following proposition.

Proposition 4.17 *Let D be a document that conforms to a tree schema S , and O be a set of objects in D . Suppose that \mathcal{P} is an interconnection semantics that contains some pattern $p = (L, C)$, where $L = l(O)$ and $C \subseteq S$. Let $p_r = (l(O), C_r)$ be a pattern such that C_r is the rooted subtree of S that is reduced w.r.t. $l(O)$. Then, $\mathcal{P} \models_D O$ if and only if $p_r \models_D O$.*

Proof. Let p_u be the pattern $(l(O), C_u)$, where C_u is the undirected subtree of S that is reduced w.r.t. $l(O)$ (that is, p_u is obtained from p_r by ignoring the directions in C_r). We first prove the following claim.

Claim 1 $p_r \models_D O$ if and only $p_u \models_D O$.

The “only if” direction of Claim 1 is immediate. For the “if” direction, suppose that T_u is an undirected subtree of D that contains the objects of O and is isomorphic to C_u . Let T_r be obtained from T_u by restoring the directions that the edges have in D . Since D conforms to S , the edges of T_r must conform to the edges of C_r and hence, T_r is isomorphic to C_r . This shows that $p_r \models_D O$, as required.

By Observation 4.16, a pattern $p = (L, C) \in \mathcal{P}$, where $L = l(O)$ and $C \subseteq S$, must be either p_r or p_u . Hence, by Claim 1, $\mathcal{P} \models_D O$ if and only if $p_r \models_D O$, as claimed. \square

The main result of this section is presented in the next corollary.

Corollary 4.18 *Let S be a tree schema and D be a document that conforms to S .*

1. $\mathcal{P}_{\text{all}}^r \models_D O$, $\mathcal{P}_{\text{all}}^u \models_D O$, $\mathcal{P}_{\text{min}}^r(S) \models_D O$, $\mathcal{P}_{\text{min}}^u(S) \models_D O$ and $\mathcal{P}_{\text{uca}}^r(S) \models_D O$ are equivalent.
2. $\mathcal{P}_{\text{all}}^r \models_D O$ can be tested in polynomial time in D .
3. Given an interconnection query $q = (L, \mathcal{P}_{\text{all}}^r)$, computing $\mathcal{R}(q, D)$ is in polynomial time in D and the result.

Proof. Consider a subset O of the objects in D . Let p be the pattern $(l(O), C)$, where C is the rooted subtree of S that is reduced w.r.t. $l(O)$. From Proposition 4.17, interconnection of O , in all the above semantics, is equivalent to $p \models_D O$. This proves Part 1 of the corollary. For Part 2, we use Corollary 3.5 that shows the tractability of testing $p \models_D O$. To prove Part 3, consider a subset L of the labels in S . Let p_L be the pattern (L, C_L) , where C_L is the subtree of S that is reduced w.r.t. L . The query $q = (\mathcal{P}_{\text{all}}^r, L)$ is equivalent to the query $q_L = (\{p_L\}, L)$, when these two are computed over D . By Corollary 3.7, $\mathcal{R}(q_L, L)$ can be computed in polynomial time in D and the output, as claimed. \square

Chapter 5

Universal Interconnectivity

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 in every *context* implied by the document there is an evidence of interconnection. As an example, consider the document D_1 in Figure 1.1. Objects 5 and 13 are $\mathcal{P}_{\text{all}}^r$ -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 $\mathcal{P}_{\text{all}}^r$ -interconnected, because that subtree has two objects, 2 and 15, that are labeled by **Department**.

Intuitively, an interconnection semantics \mathcal{P} is applied universally to a set of objects O in a document D as follows. Every context of O that is implied by D should include an evidence of interconnection, according to some pattern $p \in \mathcal{P}$. Clearly, the choice of contexts for O would determine the outcome. We view every subtree T of D that contains O (but is not necessarily uniquely labeled) as a context for O . Thus, T itself should have a subtree T' that is an evidence of interconnection according to some pattern $p \in \mathcal{P}$. This is formally defined as follows.

Definition 5.1 (universal interconnectivity) *Let \mathcal{P} be a rooted (undirected) interconnection semantics, D be a document and O be a subset of the objects of D . We say that O is universally \mathcal{P} -interconnected in D , denoted $\mathcal{P} \models_{\forall, D} O$, if for every rooted (undirected)*

subtree $T \subseteq D$ that contains O , there is a rooted (undirected) pattern $(l(O), C) \in \mathcal{P}$ and a rooted (undirected) subtree $T' \subseteq T$ such that T' contains O and is isomorphic to C .

Note that the above definition cannot be satisfied vacuously, since D has at least one rooted (undirected) subtree that contains O . Thus, $\mathcal{P} \models_{\forall, D} O$ implies $\mathcal{P} \models_D O$, as expected.

Consider an interconnection query $q = (L, \mathcal{P})$ and a document D . We use $\mathcal{R}_{\forall}(q, D)$ to denote the o-relation that results from applying the query q to D in a universal manner; that is, $\mathcal{R}_{\forall}(q, D)$ is the o-relation that consists of all o-tuples t , such that (1) $O[t]$ is a subset of the objects in D , (2) $L[t] = L$, and (3) $\mathcal{P} \models_{\forall, D} O[t]$.

In the rest of this chapter, we discuss specific cases of universal interconnections, and analyze their complexity.

5.1 Universal $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$ interconnections

In this section, we discuss the universal versions of the semantics $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$. These are characterized in the following observation.

Observation 5.2 *A set O of objects in a document D is universally $\mathcal{P}_{\text{all}}^r$ -interconnected ($\mathcal{P}_{\text{all}}^u$ -interconnected) in D if and only if every rooted (undirected) subtree of D that is reduced w.r.t. O is also uniquely labeled.*

Thus, $\mathcal{P}_{\text{all}}^r$, $\mathcal{P}_{\text{all}}^u$ and their universal versions are all equivalent on tree documents (but not necessarily on documents that conform to tree schemas).

As an example, consider document D_1 in Figure 1.1. The set $\{(5, 6), (19, 20), (11, 13)\}$ comprises all pairs of objects for $\{\text{Name}, \text{Email}\}$ that are universally $\mathcal{P}_{\text{all}}^r$ -interconnected. This set also comprises all pairs of objects for $\{\text{Name}, \text{Email}\}$ that are universally $\mathcal{P}_{\text{all}}^u$ -interconnected in D_1 . As another example, objects 15 (a department) and object 10 (a project) are not universally $\mathcal{P}_{\text{all}}^r$ -interconnected, because the project belongs to more than one department. The pairs of objects $(15, 1)$, $(15, 16)$ and $(15, 17)$ (as well as some other pairs containing object 15) are universally $\mathcal{P}_{\text{all}}^r$ -interconnected.

The following proposition shows that universal $\mathcal{P}_{\text{all}}^u$ -interconnection of a set of objects can be reduced to pairwise interconnection and universal $\mathcal{P}_{\text{all}}^r$ -interconnection can be similarly reduced for acyclic document.

Proposition 5.3 *Let D be a document and O be a subset of the objects in D .*

1. $\mathcal{P}_{\text{all}}^u \models_{\forall, D} O$ if and only if $\mathcal{P}_{\text{all}}^u \models_{\forall, D} \{o_1, o_2\}$ for all objects $o_1, o_2 \in O$.
2. If D is acyclic, then $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$ if and only if $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$ for all objects $o_1, o_2 \in O$.

Proof. We first prove Part 1 of the proposition. For the “if” direction, we prove the contrapositive; that is, if $\mathcal{P}_{\text{all}}^u \not\models_{\forall, D} O$ then $\mathcal{P}_{\text{all}}^u \not\models_{\forall, D} \{o_1, o_2\}$ for some objects $o_1, o_2 \in O$. So suppose that D contains an undirected subtree T , such that (1) T is reduced w.r.t. O and (2) T has two distinct objects \hat{o}_1 and \hat{o}_2 that have the same label. Consider the simple undirected path \hat{P} between objects \hat{o}_1 and \hat{o}_2 in the subtree T . Since T is reduced w.r.t. O , the path \hat{P} can be extended to a simple path P in T , such that P is a path between two objects $o_1, o_2 \in O$. Since P contains both objects \hat{o}_1 and \hat{o}_2 , it follows that $\mathcal{P}_{\text{all}}^u \not\models_{\forall, D} \{o_1, o_2\}$.

For the “only if” direction, we also prove the contrapositive. Suppose that o_1 and o_2 are objects in O such that $\mathcal{P}_{\text{all}}^u \not\models_{\forall, D} \{o_1, o_2\}$. By definition, D contains a simple undirected path P between o_1 and o_2 , such that P visits two distinct objects with the same label. From the following observation, it follows that P can be extended to a subtree T that is reduced w.r.t. O .

Observation 1 *Let U be a subset of the nodes of a connected undirected graph G . Every simple path between two objects in U is a subgraph of some subtree $T \subseteq G$, such that T is reduced w.r.t. U .*

Since T has two objects with the same label, it follows that $\mathcal{P}_{\text{all}}^u \not\models_{\forall, D} O$, as required.

The proof of Part 2 is similar to the proof of Part 1. However, in this case, only a weaker version of the above observation holds.

Observation 2 *Let U be a subset of the nodes of a rooted graph G . If G is acyclic, then for every rooted subtree $\hat{T} \subseteq G$, such that \hat{T} is reduced w.r.t. some pair of objects in U , there exists a rooted subtree $T \subseteq G$, such that $\hat{T} \subseteq T$ and T is reduced w.r.t. U .*

□

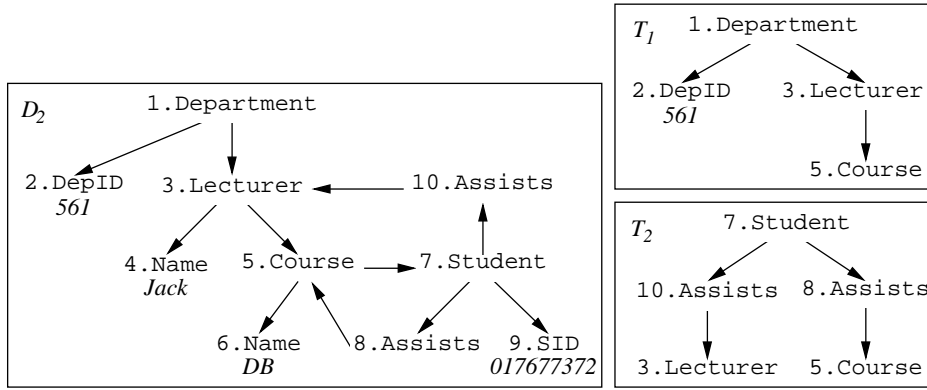


Figure 5.1: A counter example for Proposition 5.3 in a cyclic document.

Claim 2 of Proposition 5.3 does not necessarily hold if D is cyclic. For example, consider the document D_2 depicted in Figure 5.1. The tree T_1 in the figure is the only rooted subtree of D that is reduced w.r.t. the set $\{2, 3, 5\}$. Since T_1 is uniquely labeled, this set is universally $\mathcal{P}_{\text{all}}^r$ -interconnected. However, objects 3 and 5 are also contained in some context that has no evidence for interconnection. This context is represented by the tree T_2 , depicted in the figure. Hence, $\mathcal{P}_{\text{all}}^r \models_{\forall, D_2} \{2, 3, 5\}$ but $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D_2} \{3, 5\}$. The other direction of Claim 2 (i.e., interconnection among pairs of objects implies interconnection of the whole set of objects) always holds, even for cyclic documents..

As a result Proposition 5.3, if a set O of objects is universally $\mathcal{P}_{\text{all}}^u$ -interconnected, then so is every subset of O . Similarly, if a set O of objects in an acyclic document is universally $\mathcal{P}_{\text{all}}^r$ -interconnected, then so is every subset of O .

In the rest of this section, we consider the complexity of testing interconnections and computing interconnection queries using the universal versions of $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$. In the course of proving our results, it is shown that our problems are closely related to the *subgraph homeomorphism problem* (SHP).

Lemma 5.4 *Testing $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$ is coNP-complete, even if $|O| = 2$.*

Proof. To show that $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} O$, one has to provide a subtree T of D such that (1) T is reduced w.r.t. O , and (2) T has two distinct objects with the same label. This shows that this problem is in coNP.

To prove NP-hardness of testing $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} O$, where $|O| = 2$, we use a reduction from the following problem. Given two objects o_s and o_t in a directed o-graph G , determine whether G contains a directed path P , from o_s to o_t , such that P visits two objects with the same label. In Chapter 8, we prove that this problem is NP-hard (Proposition 8.1). The reduction is similar to the one used in the proof of Lemma 4.2. Suppose that G is an o-graph and that o_1 and o_2 are two objects in G . We construct a document D , such that (1) D contains objects o_1 and o_2 , and (2) $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o_2\}$ if and only if some simple directed path in G , from o_1 to o_2 , visits two distinct objects with the same label. Let G' be the subgraph of G induced by all objects that are reachable from o_1 through a directed path (we assume that o_2 is among these objects). We define D to be the document that is obtained from G' by removing all incoming edges of object o_1 . The subtrees of D that are reduced w.r.t. $\{o_s, o_t\}$ are exactly the simple directed paths from o_s to o_t in G . Hence, D is the desired document. \square

The following theorem summarizes the complexity of testing universal $\mathcal{P}_{\text{all}}^r$ -interconnectivity and $\mathcal{P}_{\text{all}}^u$ -interconnectivity.

Theorem 5.5 *Let D be a document and O be a subset of the objects in D .*

1. *If D is acyclic, then testing $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$ is polynomial in D ; otherwise, it is coNP-complete.*
2. *Testing $\mathcal{P}_{\text{all}}^u \models_{\forall, D} O$ is polynomial in D .*

Proof. The coNP-complete result of Part 1 follows from Corollary 5.4. To prove the tractability of this problem when D is acyclic, consider two objects o_1 and o_2 in D . By Proposition 5.3, it is enough to show that $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$ can be tested in polynomial time. We assume that $l(o_1) \neq l(o_2)$, otherwise $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o_2\}$ trivially holds. Our strategy is to efficiently look for a subtree $T \subseteq D$, such that T is reduced w.r.t. $\{o_1, o_2\}$ and T has two distinct objects with the same label. If this search fails, we decide that $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$. Consider a rooted tree T from the ones depicted in Figure 5.2. We use T as a *template* for trees with objects from D , as follows. The tree T has objects o_1 and o_2 as two of its nodes. The rest of the nodes in T are variables from the set $\{x_1, x_2, y, y', z\}$.

An *assignment* for T is a one-to-one mapping ϕ from the variables of T to objects in D that satisfies the following constraints.

- The object ϕx_1 and ϕx_2 are labeled with $l(o_1)$ and $l(o_2)$, respectively.
- The objects ϕy and $\phi y'$ have the same label.
- The image of ϕ includes neither o_1 nor o_2 .

By ϕT we denote the result of the assignment ϕ . We say that the tree ϕT is *homeomorphic* to a subgraph of D if there exists a mapping θ , from the edges of T to directed paths in D , such that

- if $e = (o, o')$, then $\theta(e)$ is a path from o to o' , and
- if $e \neq e'$, then $\theta(e)$ and $\theta(e')$ have no objects in common (except possibly for their endpoints).

Our proof uses the following claims.

Claim 1 $\mathcal{P}_{\text{all}}^r \not\equiv_{\forall, D} \{o_1, o_2\}$ if and only if ϕT is homeomorphic to a subgraph of D for some rooted tree T in Figure 5.2 and some assignment ϕ for T .

Claim 2 Let T be one of the rooted trees in Figure 5.2 and ϕ be an assignment for T . Testing whether T is homeomorphic to a subgraph of D is in polynomial time in D .

Claim 1 is proved as follows. For the “if” direction, suppose that ϕ is an assignment for some rooted tree T in Figure 5.2, and that ϕT is homeomorphic to a subgraph of D . Consider the rooted subtree T' of D that is obtained by replacing the edges in ϕT with disjoint paths between the corresponding objects. By observing each of the rooted trees in the above figure, the following hold.

- Two distinct objects in ϕT (and hence, in T') have same label.
- T' is reduced w.r.t. $\{o_1, o_2\}$.

Thus, T' is an evidence for $\mathcal{P}_{\text{all}}^r \not\equiv_{\forall, D} \{o_1, o_2\}$.

For the “if” direction, suppose that T' is a rooted subtree of D , such that T' is reduced w.r.t. $\{o_1, o_2\}$ and T' has two objects with the same label. It is easy but tedious to show

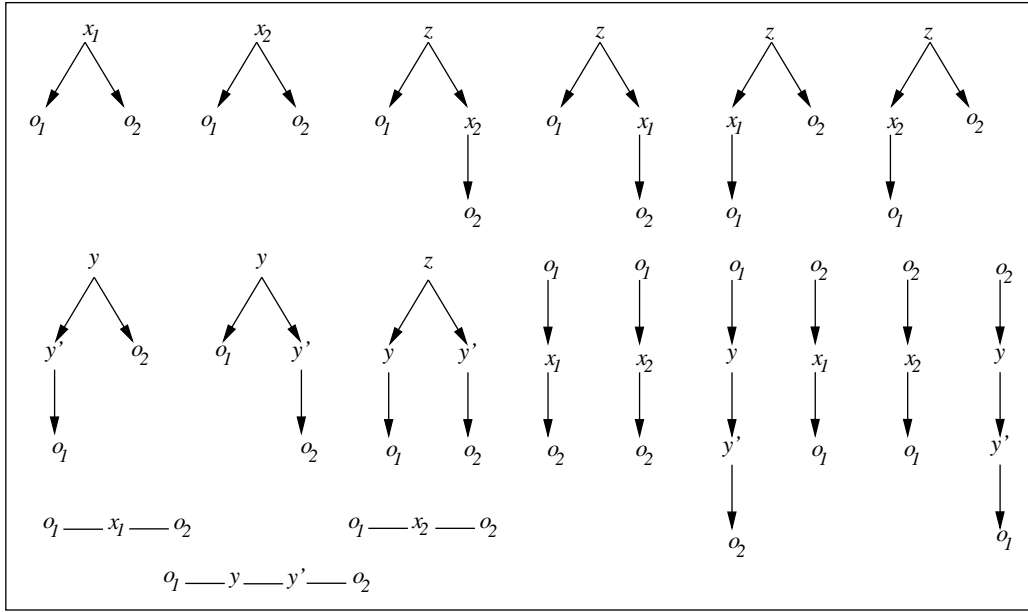


Figure 5.2: Templates for trees.

that the following holds. For at least one of the rooted trees T in Figure 5.2, there exists an assignment ϕ for T , such that the image of ϕ consists of objects in T' , and ϕT is homeomorphic to a subgraph of T' . Since T' is a subgraph of D , it follows that ϕT is also homeomorphic to a subgraph of D , as required.

Claim 2 is a special case of the result of [6], that testing whether a directed graph of constant size is homeomorphic to a subgraph of a directed acyclic graph is in polynomial time.

By Claim 1, in order to test whether $\mathcal{P}_{\text{all}}^r \not\equiv_{\forall, D} \{o_1, o_2\}$, one can traverse all trees T in Figure 5.2 and all possible assignments ϕ for T , and test whether ϕT is homeomorphic to a subgraph of D . By Claim 2, this can be done in polynomial time in D . This proves Part 1 of the theorem.

The proof of Part 2 is similar to the proof of the previous part, except for the following modifications. Instead of using the rooted subtrees in Figure 5.2, we use the undirected ones. In this part, we use the result of [9] that testing subgraph homeomorphism is tractable for undirected graphs, where the source graphs are fixed. \square

Interestingly, if S_D is acyclic and D has n objects and m edges, then $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$ can be tested in $\mathcal{O}(|O|^2)$ time after applying an $\mathcal{O}(nm)$ -time preprocessing step. The preprocessing step computes $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$ for all pairs o_1 and o_2 of objects in D . From Proposition 5.3 and the fact that D is acyclic (since D is a document that conforms to an acyclic schema), it follows that one can test $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$ by testing $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$ for all $o_1, o_2 \in O$. We first prove the following proposition (observe its similarity to Lemma 4.6).

Proposition 5.6 *Let D be a document, such that S_D is acyclic. Let l_1, \dots, l_m be the labels of S_D in a topological order. If $l(o_1) = l_i$ and $l(o_2) = l_j$ for some $i < j$, then $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o_2\}$ if and only if $\mathcal{P}_{\text{all}}^r \models_{\forall, D} \{o_1, o\}$ for all parents o of o_2 in D .*

Proof. For the “if” direction, suppose that $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o_2\}$. Let T be a subtree of D that is reduced w.r.t. $\{o_1, o_2\}$ and contains two objects with the same label. Since T is reduced, for all objects o in T , the schema S_D contains a directed path from $l(o)$ to either $l(o_1)$ or $l(o_2)$. This implies that $l(o_2)$ is the last, among all labels in T , in the topological order of the labels. We conclude that (1) o_2 is not the root of T and (2) o_2 is the only object in T that is labeled with $l(o_2)$. Let o be the parent of o_2 in T , and T' be the tree that is obtained from T by removing object o_2 . Then, T' is reduced w.r.t. $\{o_1, o\}$ and T' has two objects with the same label. Thus, $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o\}$.

For the “only if” direction, suppose that object o_2 has a parent o such that $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o\}$. Let T' be a rooted subtree of D , such that T' is reduced w.r.t. $\{o_1, o\}$ and T' has label repetition. Note that T' cannot have any object that is labeled with $l(o_2)$, since all labels in T' precede either $l(o_1)$ or $l(o)$ in the topological order of the labels. In particular, T' does not contain object o_2 . Consider the tree T that is obtained from T' by adding the edge from o to o_2 . Then T is reduced w.r.t. $\{o_1, o_2\}$ and T has a repeated label. Hence, $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} \{o_1, o_2\}$, as required. \square

Consider the algorithm $\mathcal{P}_{\text{all}}^r\text{-ARRAY}(D, k)$, presented in Figure 4.2. In this algorithm, we used dynamic programming in order to find all sets O with k or fewer objects in D , such that O is $\mathcal{P}_{\text{all}}^r$ -interconnected (assuming that S_D is acyclic). From Proposition 5.6, we can use a similar algorithm in order to find all universally $\mathcal{P}_{\text{all}}^r$ -interconnected pairs of objects. We modify this algorithm as follows.

- We set the variable k to the fixed value 2 (hence, k need not be an argument of our algorithm).
- In Line 6, we replace the condition “ o_i has a parent o s.t. $I[O \cup \{o\}] = \mathbf{true}$ ” with “ o_i has no parent o s.t. $I[O \cup \{o\}] = \mathbf{false}$ ”.

Assuming that D has n objects and m edges, the modified algorithm runs in time $O(nm)$, as implied by Proposition 4.7.

It is not known whether a preprocessing step with a similar complexity exists for the other polynomial-time case of Theorem 5.5.

In the rest of this section, we discuss the complexity of computing interconnection queries with the universal versions of $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$. In many cases, computing such queries is intractable, as shown in the following theorem. A tractable case of this problem will be discussed in the next section.

Theorem 5.7 *Let L be a set of labels, q_u be the interconnection query $(L, \mathcal{P}_{\text{all}}^r)$, q_r be the interconnection query $(L, \mathcal{P}_{\text{all}}^u)$ and D be a document.*

1. *Testing $\mathcal{R}_{\forall}(q_r, D) \neq \emptyset$ is an NP-hard and coNP-hard member of Σ_2^P .*
2. *If S_D is acyclic, testing $\mathcal{R}_{\forall}(q_r, D) \neq \emptyset$ is NP-complete.*
3. *If D is acyclic, testing $\mathcal{R}_{\forall}(q_r, D) \neq \emptyset$ is NP-complete.*
4. *If L is of constant size, testing $\mathcal{R}_{\forall}(q_r, D) \neq \emptyset$ is coNP-complete.*
5. *Testing $\mathcal{R}_{\forall}(q_u, D) \neq \emptyset$ is NP-complete.*

Proof. For Part 1 of the theorem, observe that hardness is derived from Parts 3–4, and membership in Σ_2^P is obvious.

Part 2 is proved as follows. In [2], it was shown that $\mathcal{R}_{\forall}(q_r, D) \neq \emptyset$ is NP-complete when D is a tree. Their proof showed that the problem is NP-complete even when S_D is acyclic. This proves NP-hardness. Membership in NP is derived from Theorem 5.5. Parts 3 and 5 are similarly proved (note that q_r and q_u are equivalent when applied universally to tree documents).

For Part 4, membership in coNP is proved as follows. For a specific set O of objects in D , an evidence for $\mathcal{P}_{\text{all}}^r \not\models_{\forall, D} O$ is a subtree $T_O \subseteq D$, such that T is reduced w.r.t. O and T has two distinct objects with the same label. If L is of constant size, an evidence for $\mathcal{R}_{\forall}(q_r, D) = \emptyset$ may consist of an evidence T_O , as described above, for all subsets O of objects in D , such that (1) the objects in O have distinct labels, and (2) $l(O) = L$. To prove coNP-hardness, a reduction from testing $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$, where $|O| = 2$, can be used. Recall that the latter proved was proved to be coNP-complete in Corollary 5.4. The reduction is similar to the reduction used in the proof of Corollary 4.3. \square

5.2 Universal Interconnectivity in Tree Schemas

In this section, we discuss universal interconnectivity in documents that conform to tree schemas, i.e., documents D such that S_D is a tree. Proposition 4.17 showed that for such documents, there exists exactly one pattern that can interconnect a set of objects. The next two propositions state a result in this spirit for universal interconnectivity.

Proposition 5.8 *Let D be a document, such that S_D is a tree, and let O be a subset of the objects in D . Let \mathcal{P} be a rooted interconnection semantics that contains some pattern $(l(O), C)$, such that $C \subseteq S_D$. Then, $\mathcal{P} \models_{\forall, D} O$ if and only if $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$.*

Proof. Let C be a rooted subtree of S_D that is reduced w.r.t. $l(O)$. Since S_D is a tree, the subtree C is unique. Hence, $p = (l(O), C)$ is the only pattern in \mathcal{P} that has the label set $l(O)$ as its first component. Let \mathcal{P}' be the singleton interconnection semantics $\{p\}$. Then $\mathcal{P} \models_{\forall, D} O$ if and only if $\mathcal{P}' \models_{\forall, D} O$. We conclude by observing that \mathcal{P} can be replaced with $\mathcal{P}_{\text{all}}^r$. \square

Following is the undirected version of the above proposition. It can be proved similarly.

Proposition 5.9 *Let D be a document, such that S_D is a tree, and let O be a subset of the objects in D . Let \mathcal{P} and be an undirected interconnection semantics that contains some pattern $(l(O), C)$, such that $C \subseteq S_D$. Then, $\mathcal{P} \models_{\forall, D} O$ if and only if $\mathcal{P}_{\text{all}}^u \models_{\forall, D} O$.*

In the reminder of this section, we show that for an interconnection query q , the \mathcal{O} -relation $\mathcal{R}_{\forall}(q, D)$ is either trivially empty or can be efficiently generated.

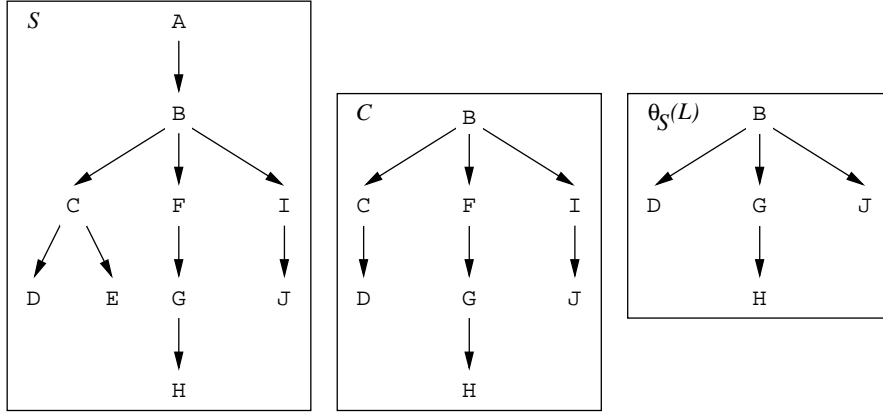


Figure 5.3: A schema S , the subtree $C \subseteq S$ that is reduced w.r.t. the set $L = \{D, G, H, J\}$, and the l-tree $\Theta_S(L)$.

Consider an interconnection query $q = (\mathcal{P}, L)$ and a document D , such that S_D is a tree. Our method for efficiently computing $\mathcal{R}_\forall(q, D)$ is described intuitively as follows. First, we transform D into a tree document T_L , in a way that all universal \mathcal{P} -interconnections among object sets O , where $l(O) = L$, are preserved. Since T_L is a tree, universal interconnectivity in T_L is equivalent to ordinary interconnectivity. Hence, we use a reduction to a tractable case of computing interconnection queries by extracting the proper tuples from T_L .

We begin with some definitions. Consider a tree schema S and a subset L of the labels in S . Let C be the rooted subtree of S that is reduced w.r.t. L . A label l in S is *topographic* w.r.t. L if either $l \in L$ or l has more than one child in the subtree C (hence, l must be in C). Equivalently, a label l in S is topographic w.r.t. L if either $l \in L$ or l is the least common ancestor of two distinct labels in L . A directed path P in S is *reducible* w.r.t. L if the endpoints are the only labels in P that are topographic w.r.t. L . The *topography* of L in S , denoted $\Theta_S(L)$, is the directed l-graph with the following properties:

- the labels of $\Theta_S(L)$ are the labels in S that are topographic w.r.t. L and
- the edges of $\Theta_S(L)$ are all pairs (l_1, l_2) , such that l_2 is reachable from l_1 through a directed path in S that is reducible w.r.t. L .

It can easily be shown that $\Theta_S(L)$ is always a rooted l-tree that is reduced w.r.t. L .

As an example, consider the tree schema S in Figure 5.3. Let L be the label set $\{D, G, H, J\}$. The l-tree C in the figure is the subtree of S that is reduced w.r.t. L . The

figure also depicts the topography of L in S . Note that the label B is topographic w.r.t. L , since it has three children in C . The path that comprises the labels B , C and D is an example to a path that is reducible w.r.t. L .

Consider a document D and an interconnection semantics \mathcal{P} . Assume that L is a subset of the labels in S_D . Let G be the directed o-graph with the following properties.

- The objects of G are all objects o in D , such that $l(o)$ is a label in $\Theta_{S_D}(L)$ (that is, $l(o)$ is topographic w.r.t. L in the schema S_D), and
- The edges of G are all pairs (o_1, o_2) such that:
 1. $(l(o_1), l(o_2))$ is an edge in $\Theta_S(L)$, and
 2. $\mathcal{P} \models_{\forall, D} \{o_1, o_2\}$.

Clearly, G conforms to the l-tree $\Theta_S(L)$ and hence, G is acyclic. Let U be the set of all objects in G that have no incoming edges. We define $\mathcal{T}_{\forall}(D, L, \mathcal{P})$ to be the document that is obtained by adding to G a root r with a unique label, and edges from r to all objects in U . It can be proved that the document $\mathcal{T}_{\forall}(D, L, \mathcal{P})$ is a tree. However, this is not necessary for our results and hence, the proof is omitted. The following lemma shows how the universal interconnections of interest can be extracted from the document $\mathcal{T}_{\forall}(D, L, \mathcal{P})$.

Lemma 5.10 *Let $q = (L, \mathcal{P})$ be an interconnection query, such that \mathcal{P} is either $\mathcal{P}_{\text{all}}^r$ or $\mathcal{P}_{\text{all}}^u$. Let D be a document, such that S_D is a tree, and assume that S_D contains all labels from L . Let p be the pattern $(L, \Theta_{S_D}(L))$ and \hat{q} be the interconnection query $(L, \{p\})$. Then,*

$$\mathcal{R}_{\forall}(q, D) = \mathcal{R}(\hat{q}, \mathcal{T}_{\forall}(D, L, \mathcal{P})).$$

Proof. We prove the lemma for $\mathcal{P} = \mathcal{P}_{\text{all}}^r$. The proof for $\mathcal{P}_{\text{all}}^u$ is similar (just replace the word “rooted” with “undirected”).

Let $\hat{D} = \mathcal{T}_{\forall}(D, L, \mathcal{P})$ and $\hat{C} = \Theta_{S_D}(L)$. We first prove that $\mathcal{R}(\hat{q}, \hat{D}) \subseteq \mathcal{R}_{\forall}(q, D)$. We will use the following claim.

Claim 1 *Let l_a, l_b and l_c be three distinct labels in S_D , such that l_b is a label in the subtree of S_D that is reduced w.r.t. $\{l_a, l_b\}$. Let o_a, o_b and o_c be objects in D that are labeled with l_a, l_b and l_c , respectively. If $\mathcal{P} \models_{\forall, D} \{o_a, o_b\}$ and $\mathcal{P} \models_{\forall, D} \{o_b, o_c\}$ then $\mathcal{P} \models_{\forall, D} \{o_a, o_c\}$.*

Let C_{ac} be the rooted subtree of S_D that is reduced w.r.t. $\{l_a, l_c\}$, and let T_{ac} be some rooted subtree of D that is reduced w.r.t. $\{o_a, o_c\}$. We will show that T_{ac} must be isomorphic to C_{ac} . Let C_{ab} and C_{bc} be the rooted subtrees of S_D that are reduced w.r.t. $\{l_a, l_b\}$ and $\{l_b, l_c\}$, respectively. The label l_b is the only label that appears in both C_{ab} and C_{bc} , because l_b appears in the subtree C_{ac} . Let T_{abc} be some extension of T_{ac} to a rooted subtree of D that is reduced w.r.t. $\{o_a, o_b, o_c\}$ (that is, $T_{ac} \subseteq T_{abc} \subseteq D$, T_{abc} is rooted and T_{abc} is reduced w.r.t. $\{o_a, o_b, o_c\}$). Note that T_{abc} must exist, since D is acyclic. Let T_{ab} and T_{bc} be the rooted subtrees of T_{abc} that are reduced w.r.t. $\{o_a, o_b\}$ and $\{o_b, o_c\}$, respectively. Since $\mathcal{P} \models_{\forall, D} \{o_a, o_b\}$ and $\mathcal{P} \models_{\forall, D} \{o_b, o_c\}$, the subtrees T_{ab} and T_{bc} have to be isomorphic to C_{ab} and C_{bc} , respectively. Thus, we can construct a rooted tree T that is isomorphic to C_{ac} by concatenating the subtrees T_{ab} and T_{bc} . Hence, T is a rooted subtree of T_{abc} that is reduced w.r.t. $\{o_a, o_c\}$. Since T_{abc} has exactly one rooted subtree that is reduced w.r.t. $\{o_a, o_c\}$, the subtrees T and T_{ac} must be the same. Hence, T_{ac} is isomorphic to C_{ac} , as required. This completes the proof of Claim 1.

Now, suppose that $t \in \mathcal{R}(\hat{q}, \hat{D})$. We will show that $\mathcal{P} \models_{\forall, D} O[t]$ (i.e., $t \in R_{\forall}(q, D)$). From the construction of \hat{D} , it follows that $O[t]$ is a subset of the objects in D . Let \hat{T} be a subtree of \hat{D} , such that \hat{T} is reduced w.r.t. $O[t]$ and \hat{T} is isomorphic to \hat{C} . Consider a label l_b in \hat{C} , and two distinct labels l_a and l_c that are adjacent to l_b in \hat{C} . Let o_a, o_b and o_c be the objects in \hat{T} that are labeled with l_a, l_b and l_c , respectively. From the construction of \hat{C} , it follows that (1) label l_b is in the subtree of S_D that is reduced w.r.t. $\{l_a, l_c\}$, and (2) $\mathcal{P} \models_{\forall, D} \{o_a, o_b\}$ and $\mathcal{P} \models_{\forall, D} \{o_b, o_c\}$. From Claim 1, it follows that $\mathcal{P} \models_{\forall, D} \{o_a, o_c\}$. By repeating this argument, it can be shown that every two objects in \hat{T} are universally \mathcal{P} -interconnected. By Proposition 5.3, it holds that $\mathcal{P} \models_{\forall, D} O[t]$, as required.

Next, we will show that $\mathcal{R}_{\forall}(q, D) \subseteq \mathcal{R}(\hat{q}, \hat{D})$. Consider an o -tuple $t \in \mathcal{R}_{\forall}(q, D)$. In the rest of this proof, we denote by C the subtree of S_D that is reduced w.r.t. L . Our proof uses the following claims.

Claim 2 *Let o_1 and o_2 be two objects in $O[t]$, and let l be the least common ancestor of $l(o_1)$ and $l(o_2)$ in S_D . There exists exactly one object o in D , such that $l(o) = l$ and o is a common ancestor of o_1 and o_2 in D .*

Claim 3 *For all labels l in \hat{C} , there exists an object o , such that o is labeled with l and*

$\mathcal{P} \models_{\forall, D} O[t] \cup \{o\}$.

The proof of Claim 2 is as follows. Consider a subtree $T \subseteq D$, such that T is reduced w.r.t. $O[t]$. Then, T is isomorphic to C and hence, T contains an object o , such that $l(o) = l$ and o is a common ancestor of o_1 and o_2 . Assume, by way of contradiction, that another object $o' \neq o$ has these properties. Let P_1 be a directed path in D from o to o_1 and P_2 be a directed path in D from o' to o_2 . Since l is a least common ancestor of $l(o_1)$ and $l(o_2)$, the paths P_1 and P_2 must be node disjoint. Let T' be some rooted subtree of D that is reduced w.r.t. $\{o, o'\}$. Then, the concatenation of the subtrees P_1 , P_2 and T' is a rooted subtree of D that is reduced w.r.t. $\{o_1, o_2\}$ and has label repetition. However, this contradicts the fact that $\mathcal{P} \models_{\forall, D} \{o_1, o_2\}$. We conclude that no such o' exists, as required.

To prove Claim 3, consider a label l in \hat{C} . Observe that if $l \in L$, the claim is trivial. Hence, we assume that $l \notin L$. From the definition of $\Theta_{S_D}(L)$, the label l has at least two children in \hat{C} . Let l_1 and l_2 be two leaves in different subtrees of l in \hat{C} . Then, l is a least common ancestor of l_1 and l_2 . Let o_1 and o_2 be the objects in $O[t]$ that are labeled with l_1 and l_2 , respectively. By Claim 2, there exists a unique common ancestor of o_1 and o_2 that is labeled with l . Let o be such an object. We will show that $\mathcal{P} \models_{\forall, D} O[t] \cup \{o\}$. Suppose that $T \subseteq D$ is a rooted subtree of D that is reduced w.r.t. $O[t] \cup \{o\}$. Let T' be the subtree of T that is reduced w.r.t. $O[t]$. Then, T' is isomorphic to C and hence, T' must include the object o . Thus, T' is also reduced w.r.t. $O[t] \cup \{o\}$. We conclude that $T' = T$ and hence, T is uniquely labeled, as required.

Finally, we use the following observation.

Observation 1 *If l is a label in $\Theta_{S_D}(L)$, then $\Theta_{S_D}(L) = \Theta_{S_D}(L \cup \{l\})$.*

We complete the proof as follows. By repeatedly using Claim 3 and Observation 1, it can be shown that there exists a set \hat{O} with the following properties:

- $O[t] \subseteq \hat{O}$;
- $\mathcal{P} \models_{\forall, D} \hat{O}$;
- $l(\hat{O})$ is exactly the set of nodes of \hat{C} (i.e., the labels in S_D that are topographic w.r.t. L).

Hence, the document \hat{D} contains an edge (o_1, o_2) for every two objects $o_1, o_2 \in \hat{O}$ such that $(l(o_1), l(o_2))$ is an edge in \hat{C} . Therefore, the subgraph of \hat{D} that is induced by \hat{O} is isomorphic to \hat{C} and contains the objects of $O[t]$. It follows that $t \in \mathcal{R}(\hat{q}, \hat{D})$, as required.

□

We conclude this chapter by proving the tractability of computing universal interconnection queries over documents that conform to tree schemas.

Theorem 5.11 *Let $q = (L, \mathcal{P})$ be an interconnection query, such that \mathcal{P} is either rooted or undirected, and let D be a document such that S_D is a tree. Assume that \mathcal{P} contains a pattern (L, C) such that $C \subseteq S_D$. Computing $\mathcal{R}_\forall(q, D)$ is in polynomial time in D and the output.*

Proof. From Propositions 5.8 and 5.9, it is sufficient to show that the theorem holds when \mathcal{P} is either $\mathcal{P}_{\text{all}}^r$ or $\mathcal{P}_{\text{all}}^u$. The query computation is as follows. First, we generate the document $\hat{D} = \mathcal{T}_\forall(D, L, \mathcal{P})$. From Theorem 5.5, this graph can be generated in polynomial time. Let p be the rooted pattern $(L, \Theta_{S_D}(L))$, and let \hat{q} be the interconnection query $(L, \{p\})$. We complete the computation by generating the o-relation $\mathcal{R}(\hat{q}, \hat{D})$. This is done by executing `PATTERNINTERCONNECTION`(\hat{D}, L, p) (Figure 3.2). By Lemma 3.6, this algorithm runs in polynomial time in \hat{D} and the output. From Lemma 5.10, it follows that the result of this algorithm is exactly the o-relation $\mathcal{R}(q, D)$. □

Chapter 6

Practical Considerations of Interconnectivity

6.1 A Discussion on the Complexity Results

We have considered two related problems. The first is determining whether a set O of objects is interconnected and the second is evaluating interconnection queries. The complexity of these problems has been analyzed according to several parameters, depending on whether $|O|$ (i.e., the size of the set of objects) is fixed, whether $|L|$ (i.e., the size of the query) is fixed, whether the schema S is acyclic, or whether the document D is acyclic. Table 6.1 summarizes the complexity of testing whether a given set of objects is interconnected. The results in Table 6.1 were proved in Chapter 4 and Chapter 5.

Query evaluation is usually analyzed in terms of *data complexity* [11], i.e., assuming that the query is fixed. If testing interconnection is polynomial when $|O|$ is fixed, then the data complexity of evaluating interconnection queries is also polynomial. Thus, Table 6.1 implies that in the general case (i.e., no restrictions on either the schema or the document), the semantics \mathcal{P}_{\min}^r , \mathcal{P}_{\min}^u , $\mathcal{P}_{\text{uca}}^r$ (when defined) and universal- $\mathcal{P}_{\text{all}}^u$ are tractable (i.e., query evaluation has a polynomial data complexity). If the document is acyclic (even if the schema is cyclic), then universal- $\mathcal{P}_{\text{all}}^r$ is also tractable. If the schema is acyclic, then all the semantics considered in this thesis, except for $\mathcal{P}_{\text{all}}^u$, are tractable. Usually schemas of XML documents are acyclic and, consequently, our results are not merely of theoretical

Problem	No restrictions	$ O $ is fixed	D is acyclic	S is acyclic	S is acyclic, $ O $ is fixed
$\mathcal{P}_{\text{all}}^r \models_D O$	NPC	NPC	NPC	NPC	Poly
$\mathcal{P}_{\text{all}}^u \models_D O$	NPC	NPC	NPC	NPC	NPC
$\mathcal{P}_{\text{min}}^r(S) \models_D O$	$\text{NPH} \cap \text{coNPH}$	Poly	$\text{NPH} \cap \text{coNPH}$	$\text{NPH} \cap \text{coNPH}$	Poly
$\mathcal{P}_{\text{min}}^u(S) \models_D O$	$\text{NPH} \cap \text{coNPH}$	Poly	$\text{NPH} \cap \text{coNPH}$	$\text{NPH} \cap \text{coNPH}$	Poly
$\mathcal{P}_{\text{uca}}^r(S) \models_D O$	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	NPC	Poly
$\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$	coNPC	coNPC	Poly	Poly	Poly
$\mathcal{P}_{\text{all}}^u \models_{\forall, D} O$	Poly	Poly	Poly	Poly	Poly

Table 6.1: The complexity of testing interconnection of a set O of objects in a document D that conforms to a schema S . If S is a tree, the complexity is always polynomial.

importance but also have a practical significance.

One could also measure the complexity of query evaluation assuming that the query is part of the input. However, this does not lead to interesting conclusions simply because the result of evaluating a query could be exponential in the size of the query (and hence query evaluation is always exponential when the query is not fixed). A different approach is measuring the *input-output complexity* [12] of query evaluation. Under this approach, the complexity is measured as a function of the combined size of the input (consisting of the query and the data) and the output. Note that if the query (i.e., $|L|$) is fixed, then input-output complexity is the same as data complexity. However, if the query is not fixed, then input-output complexity is capable of differentiating between the following two cases. First, a case where query evaluation is a hard problem simply because it is hard to determine whether the result of the query is not empty. Second, a case where query evaluation is not hard, but the result is exponential in the size of the query. (Note that data complexity cannot distinguish between these two cases.) For example, in the general case, the input-output complexity of evaluating a sequence of joins is not polynomial, because determining whether the result is not empty is NP-complete. For acyclic joins, however, query evaluation has a polynomial input-output complexity. Thus, polynomial input-output complexity is a stronger evidence of tractability than polynomial data complexity.

Table 6.2 summarizes the input-output complexity of evaluating interconnection queries. As mentioned earlier, when $|L|$ is fixed, input-output complexity is the same as data com-

Interconnection Query	No restrictions	$ L $ is fixed	D is acyclic	D is acyclic $ L $ is fixed	S is acyclic	S is acyclic $ L $ is fixed
$\mathcal{R}(q, D), q = (L, \mathcal{P}_{\text{all}}^r)$	**	**	**	**	**	*
$\mathcal{R}(q, D), q = (L, \mathcal{P}_{\text{all}}^u)$	**	**	**	**	**	**
$\mathcal{R}(q, D), q = (L, \mathcal{P}_{\text{min}}^r(S))$	****	*	****	*	****	*
$\mathcal{R}(q, D), q = (L, \mathcal{P}_{\text{min}}^u(S))$	****	*	****	*	****	*
$\mathcal{R}(q, D), q = (L, \mathcal{P}_{\text{uca}}^r(S))$	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	<i>undefined</i>	**	*
$\mathcal{R}_{\forall}(q, D), q = (L, \mathcal{P}_{\text{all}}^r)$	****	***	**	*	**	*
$\mathcal{R}_{\forall}(q, D), q = (L, \mathcal{P}_{\text{all}}^u)$	**	*	**	*	**	*

- * polynomial
- ** verifying non-emptiness is NP-complete
- *** verifying non-emptiness is coNP-complete
- **** verifying non-emptiness is both NP-hard and coNP hard

Table 6.2: Input-output complexity of computing interconnection queries. D is always assumed to conform to S . If S is a tree, the complexity is always polynomial in D and the output.

plexity. The results in Table 6.2 either follow from this observation or were proved in Chapter 4 and Chapter 5. The input-output complexity (and hence, the data complexity) is always polynomial when the schema is a tree, and the data complexity is always polynomial when the document is a tree.

6.2 Heuristics For Intractable Cases

The intractable cases of Tables 6.1 and 6.2 can still be handled efficiently, in practice, by using an approach based on Theorem 3.8. We illustrate this approach on the semantics $\mathcal{P}_{\text{all}}^r$. Consider a document D with the derived schema S_D . Given a query $q = (L, \mathcal{P}_{\text{all}}^r)$, we first generate all the patterns of $\mathcal{P}_{\text{all}}^r$ that are relevant to D , i.e., all patterns $p = (L, C)$, where C is a rooted subtree of S_D that is reduced w.r.t. L . Intuitively, there is only a small number of such patterns, because in the real world and in its reflection in XML data, there are only a few different semantic relationships among any given set of labels L . Thus, generating all the required patterns would be done efficiently if we can devise an algorithm that is polynomial in the size of the input and the output. Once these patterns are available, we can apply the efficient algorithm of Theorem 3.8. This approach can also be used for the

semantics \mathcal{P}_{\min}^r and $\mathcal{P}_{\text{uca}}^r$ by choosing only those patterns that are minimal and structurally minimal, respectively.

In Appendix A, we describe the required algorithm that enumerates, for a given graph, all the rooted subtrees that are reduced w.r.t. a given set of nodes V . This algorithm has a polynomial input-output complexity. If the size of the output of this algorithm is polynomial in the size of the input, the whole evaluation process is polynomial. This holds, for example, in either one of the following cases:

- The schema S is of fixed size;
- The schema S has $n + k$ edges, where n is the number of labels in S and k is a bounded integer;
- S is acyclic, and there exists a fixed integer k , such that in each directed path from the root of S to some leaf of S , at most k nodes have more than one parent.

This algorithm can be generalized to the undirected case and, hence, the semantics $\mathcal{P}_{\text{all}}^u$ and \mathcal{P}_{\min}^u can be handled similarly.

As mentioned above, in practice there are only a few distinct semantic relationships among any set of labels. Hence, enumeration of subtrees may be a good approach when dealing with cases that are intractable according to Tables 6.1 and 6.2, even if the enumeration is not guaranteed to be in polynomial time.

A similar approach can be used for the intractable cases of the semantics universal- $\mathcal{P}_{\text{all}}^r$. However, in this case, we generate subtrees of the document instead of subtrees of the schema, as follows. Consider a document D and a subset O of the objects in D . In order to test $\mathcal{P}_{\text{all}}^r \models_{\forall, D} O$, we enumerate the subtrees of D that are reduced w.r.t. O , and then check whether each one of these subtrees is uniquely labeled. Hence, using the assumption that documents have a small number of trees that are reduced w.r.t. a given set of objects, the complexity of universal- $\mathcal{P}_{\text{all}}^r$ becomes tractable under data complexity.

Chapter 7

Constructing Meaningful Documents

In the previous chapters, we described how meaningfully-related sets of objects can be extracted from a document using an interconnection semantics. In this chapter, we consider the complementary problem: constructing a meaningful document from a given set of objects and a given set of meaningful relationships among these objects.

7.1 Layouts

Suppose that \mathcal{U} is a set of objects, S is a schema and \mathcal{I} is a set of uniquely-labeled subsets of objects from \mathcal{U} . The set \mathcal{I} is used to indicate subsets of objects from \mathcal{U} that are known to be meaningfully related. We would like to construct a document D that (1) contains the objects of \mathcal{U} (and possibly additional objects), (2) conforms to S and (3) reflects the meaningful relationships in \mathcal{I} , i.e., D implies a meaningful relationship between the objects of each $O \in \mathcal{I}$.

The third condition is formalized using interconnection semantics. Thus, our input consists of an interconnection semantics \mathcal{P} for the target document, in addition to \mathcal{U} , S and \mathcal{I} . Formally, the third condition is the requirement that $\mathcal{P} \models_D O$ for all $O \in \mathcal{I}$. We use the term $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -*layout* to describe a document D that satisfies the three conditions above.

In principle, a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout D may introduce new meaningful relationships that do not appear in \mathcal{I} . In other words, it may hold that $\mathcal{P} \models_D O$, even though $O \notin \mathcal{I}$. For some cases, every $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout D interconnects sets of objects not appearing in \mathcal{I} . This is the case, for example, if $\mathcal{P} = \mathcal{P}_{\text{all}}^r$ and there is a set $O \in \mathcal{I}$ for which not all subsets of O are in \mathcal{I} . (Note that if $\mathcal{P}_{\text{all}}^r \models_D O$, then by definition $\mathcal{P}_{\text{all}}^r \models_D O'$ for all $O' \subseteq O$.) Obviously, it is desirable that D introduces as few new interconnections as possible.

Observe that a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout can exist only if for every set $O \in \mathcal{I}$, there is a pattern (L, C) in \mathcal{P} , such that $L = l(O)$ and $C \subseteq S$. Hence, in the remainder of this section, we will assume that this is always the case, i.e., for all $O \in \mathcal{I}$, there is a pattern $p \in \mathcal{P}$ that can interconnect O in some document conforming to S . We make two additional assumptions to simplify the presentation: (1) \mathcal{P} is rooted and (2) none of the patterns in \mathcal{P} contains the root of S . The results of this section can be extended to cases where these assumptions are relaxed.

We present an algorithm that constructs a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout. We start by defining some notation used in our algorithm. If G_1 and G_2 are o-graphs (possibly with common objects), we use $G_1 \cup G_2$ to denote the o-graph that concatenates G_1 and G_2 , i.e., $G_1 \cup G_2$ consists of all the objects and edges in either G_1 or G_2 .

Let O be a set of uniquely-labeled objects. Let C be a rooted l-tree that is reduced w.r.t. $l(O)$. Let L' be the set of labels of C that are not in $l(O)$. We use $copy(O, C)$ to denote the o-tree that

- is isomorphic to C and
- contains the objects of O , along with a new object o_l , labeled with l , for each label $l \in L'$.

Intuitively, $copy(O, C)$ is an o-tree version of C that contains the objects from O instead of the labels from $l(O)$, and new objects instead of the labels from L' .

The algorithm LAYOUT, presented in Figure 7.1, constructs a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout. The execution of this algorithm can be visualized as follows. In the first phase, an o-tree is created for each $O \in \mathcal{I}$ by copying an l-tree appearing in some pattern. These o-trees are concatenated together. The result of the first phase is an o-graph. This o-graph may not contain every object in \mathcal{U} , since some objects in \mathcal{U} may not participate in any subset of \mathcal{U}

<p><u>LAYOUT($\mathcal{U}, S, \mathcal{I}, \mathcal{P}$)</u></p> <ol style="list-style-type: none"> 1: let D be an empty o-graph 2: for all subset $O \in \mathcal{I}$ do 3: let $(L, C) \in \mathcal{P}$ be a pattern, such that $L = l(O)$ and $C \subseteq S$ 4: $D := D \cup \text{copy}(O, C)$ 5: let l_r be the root of S 6: let r be a new object labeled with l_r 7: $D := D \cup \{r\} \cup \mathcal{U}$ 8: while D contains an object o that is not reachable from r do 9: let C be a path from the root l_r to $l(o)$ in S 10: $D := D \cup \text{copy}(\{r, o\}, C)$ 11: return D

Figure 7.1: The algorithm LAYOUT.

that belongs to \mathcal{I} . In addition, the o-graph is not necessarily a document, since it may not be rooted. In the second phase, the algorithm adds the missing objects of \mathcal{U} and a root object to the o-graph. It then connects the root to the rest of the o-graph, in order to form a document. An example of applying the algorithm is given in the next section.

Consider Line 3 of the algorithm. Finding a pattern $(L, C) \in \mathcal{P}$, such that C is subtree of the schema S , may not be a tractable task when \mathcal{P} is given implicitly. However, for the interconnection semantics of Chapter 4, finding such a pattern is easier than checking \mathcal{P} -interconnectivity.

Note that the choices of o and C in Lines 8 and 9 are nondeterministic. Different choices of o and C can result in documents of different sizes. Heuristics can be used to reduce the size of the result. This is not discussed further.

The following theorem states that LAYOUT returns a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout, as required. In addition, we show that under certain conditions, LAYOUT only introduces new interconnections that are already implied by the input.

Theorem 7.1 *Let \mathcal{U} be a set of objects, \mathcal{I} be a set of uniquely-labeled subsets of objects from \mathcal{U} , S be a schema and \mathcal{P} be an interconnection semantics. Let D be the result of calling LAYOUT($\mathcal{U}, S, \mathcal{I}, \mathcal{P}$). Then:*

1. D is a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout.

2. Suppose that for each subset $O \in \mathcal{I}$, there is exactly one pattern $(L, C) \in \mathcal{P}$ such that $L = l(O)$ and $C \subseteq S$. If D' is also a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout, then all the \mathcal{P} -interconnections among objects of \mathcal{U} that exist in D also exist in D' , i.e., for all $O \subseteq \mathcal{U}$, if $\mathcal{P} \models_D O$ then $\mathcal{P} \models_{D'} O$.

Proof. Claim 1 is immediate from the construction of the layout. For Claim 2, assume that the conditions of this claim hold. Let D' be a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout. Suppose that O is a subset of \mathcal{U} such that $\mathcal{P} \models_D O$. We will prove that $\mathcal{P} \models_{D'} O$.

Let T be a subtree of D and $p = (l(O), C) \in \mathcal{P}$ be a pattern, such that T contains O and T is isomorphic to C . From our assumptions, C does not contain the root of T . We will prove the following claim.

Claim 1 Consider a subset $\hat{O} \subseteq O$. Let $\hat{C} \subseteq C$ be the subtree of C that is reduced w.r.t. $l(\hat{O})$. Then D' has a subtree that is isomorphic to \hat{C} and contains \hat{O} .

Observe that for $\hat{O} = O$, this claim shows that D' contains a subtree that contains O and is isomorphic to C . This proves that $\mathcal{P} \models_{D'} O$. Hence, we complete the proof of this theorem proving Claim 1. Our proof is by induction on the number of labels in \hat{C} . For the base case, we assume that \hat{C} has only one label. Then $|\hat{O}| = 1$ and hence, the claim obviously holds. For the inductive step, consider the subtree $\hat{T} \subseteq T$ that is reduced w.r.t. \hat{O} . We consider two cases.

Case 1: All objects of \hat{O} have exactly one incident edge in \hat{T} .

Consider an object o in \hat{T} , such that $o \notin \hat{O}$. Then o was created in some call to *copy* (Line 4). Observe that all the incident edges of o in \hat{T} must have been created in the same call to *copy*. We conclude that all the edges of \hat{T} were created in this call. Suppose that this call to *copy* had the arguments O' and C' . Then $\hat{O} \subseteq O'$, $O' \in \mathcal{I}$, and $p = (l(O'), C')$ is the unique pattern in \mathcal{P} that has the label set $l(O')$ as its first component, and a subtree of S as its second. Since D' is a $(\mathcal{U}, S, \mathcal{I}, \mathcal{P})$ -layout, it must have a subtree T' that is isomorphic to C' and includes O' . In particular, the subtree T' itself has a subtree that is isomorphic to \hat{C} and contains \hat{O} , as required.

Case 2: Some object $o \in \hat{O}$ has two or more incident edges in \hat{T} (therefore, $|\hat{O}| \geq 3$).

Let T' be the subtree of \hat{T} that is rooted at one of the children of o . Let T_1 be obtained from T' by adding object o together with the edge from o to the root. Let T_2 be obtained

from \hat{T} by pruning the subtree T' . Observe that object o is the only object that T_1 and T_2 have in common. Let O_1 and O_2 be the sets of all objects in \hat{O} that are also in T_1 and T_2 , respectively. Note that T_1 and T_2 are reduced w.r.t. O_1 and O_2 , respectively. Let C_1 and C_2 be the subtrees of \hat{C} that are isomorphic to T_1 and T_2 , respectively. By the inductive hypothesis, document D' contains two subtrees T'_1 and T'_2 , such that for $i \in \{1, 2\}$,

- T'_i contains O_i and
- T'_i is isomorphic to C_i .

We conclude by observing that $T'_1 \cup T'_2$ is a subtree of D' that is isomorphic to \hat{C} and contains \hat{O} . □

The condition in Part 2 of Theorem 7.1 states that \mathcal{P} is unambiguous in defining how the required interconnections should be reflected in the result. If this holds, then the algorithm only generates unavoidable interconnections among the objects of \mathcal{U} , in addition to those specified by \mathcal{I} . The unambiguity condition always holds if S is a tree, and can hold (depending on S , \mathcal{I} and \mathcal{P}) even if S is not a tree.

Unambiguity is critical when deciding how to create the target document. For example, suppose that \mathcal{I} contains a set O with $l(O) = \{\text{Name}, \text{Email}\}$. Suppose that the labels **Manager** and **Project** could potentially have a name and an email according to S and \mathcal{P} . In such a case, it is not possible to determine whether, in the target document, the given name and email should be properties of a manager or properties of a project. Unambiguity requires that the set O also contain an object with either **Manager** or **Project** as its label.

7.2 Combining Data Extraction and Document Construction

Up to now, we have described how meaningfully-related data can be extracted from a given document and presented an algorithm that constructs a document from a given set of objects, such that the document preserves the given relationships among these objects. In this section, we demonstrate how data extraction and document construction can be combined in order to translate a given document so that it conforms to a new schema, while preserving semantic relationships.

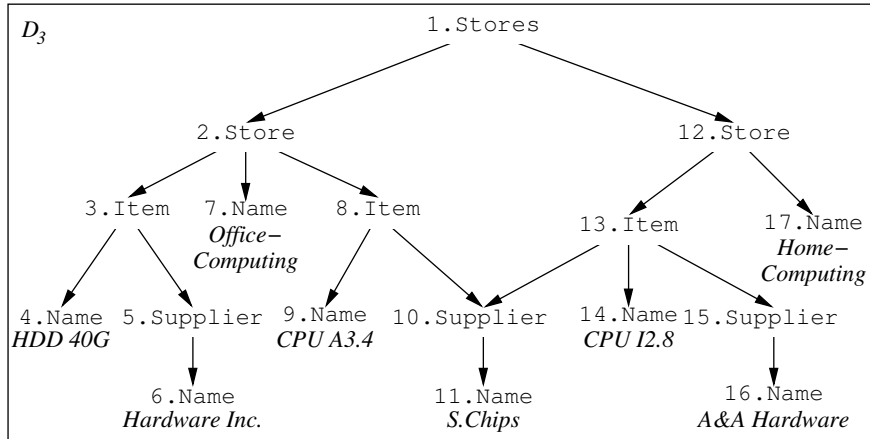


Figure 7.2: A document D_3 describing stores, items and suppliers.

We consider the following problem. Given a document D and a schema S , we would like to present the content of D as a document that conforms to S . Our approach for solving this problem is to construct a document that contains objects from the source document and preserves the semantic relationships among these objects.¹

The input to our problem consists of a document D , a set of sets of labels \mathcal{L} and a schema S . We would like to create a target document D' that (1) conforms to S , (2) contains all the objects from D that are labeled with labels from \mathcal{L} and (3) preserves all semantic relationships among sets of objects O from D such that $l(O) \in \mathcal{L}$. Intuitively, \mathcal{L} is used to specify the objects from D that should appear in the result and the semantic relationships that should be preserved. In order to formalize the third condition, we need two interconnection semantics (for the input document and for the target document), as explained below.

We demonstrate a solution to this problem by showing how it is possible to translate the document D_3 (depicted in Figure 7.2) into a new document D_4 (depicted in Figure 7.4) that conforms to the schema S_2 (depicted in Figure 7.3).

The document D_3 describes stores that sell computer hardware. For each store, D_3 specifies the items that are sold in the store. For each item, the suppliers that supply that item are displayed. Note that D_3 has a *store-centric* view of the data, since

¹We assume that D and S use the same labels to represent entities of the same type. In practice, this assumption can be relaxed by using methods for ontology mapping.

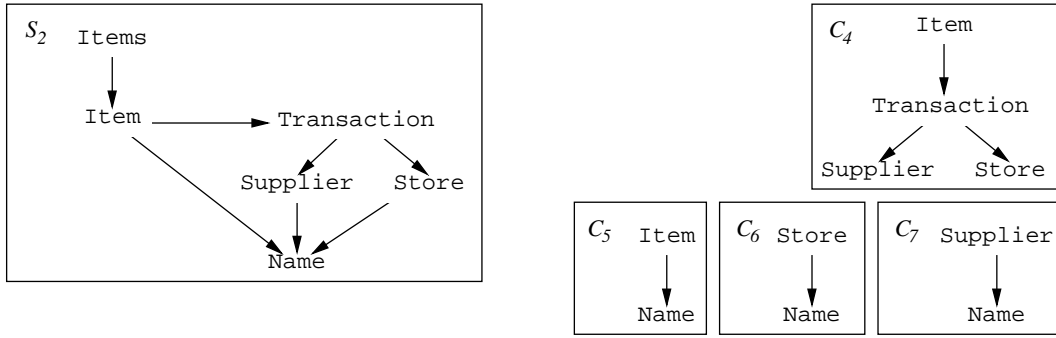


Figure 7.3: A schema S_2 and its rooted subtrees C_4 , C_5 , C_6 and C_7 .

stores are at the top of the hierarchy. Our goal is to present the data of D_3 in a new document that conforms to the schema S_2 . The schema S_2 has an *item-centric* view of the data that puts items at the top of the hierarchy. In a target document that conforms to S_2 , **Transaction** elements represent triples of related items, suppliers and stores. The set \mathcal{L} is used to specify both the objects in D that should be preserved and the sets of objects whose relationships should be preserved. In our example, we define \mathcal{L} as: $\{\{\text{Store,Name}\}, \{\text{Item,Name}\}, \{\text{Supplier,Name}\}, \{\text{Store,Supplier,Item}\}\}$.

We now show how our methods can be used to perform the translation of D_3 into D_4 (which conforms to S_2). The translation is a two-step process.

Step 1: Extract data from the source document. We extract sets O of objects from D_3 , such that $l(O) \in \mathcal{L}$ and the objects in O are meaningfully related. Hence, before extracting data from D_3 , we must choose an interconnection semantics that defines the semantic relationships among objects in D_3 . In our example, we use the semantics $\mathcal{P}_{\min}^r(S_{D_3})$.

The result of Step 1 is presented in Table 7.1. We use \mathcal{I} to denote the set of sets of objects that appear in Table 7.1. Note that \mathcal{I} has the same type as the set that is used in the input to the LAYOUT algorithm to describe the semantic relationships that should be preserved.

Step 2: Construct the target document. We will construct the target document using the algorithm LAYOUT. In order to use this algorithm, we must define the semantics

$\{\text{Store, Name}\}$	$\{\text{Item, Name}\}$	$\{\text{Supplier, Name}\}$	$\{\text{Store, Supplier, Item}\}$
$\{2, 7\}$	$\{3, 4\}$	$\{5, 6\}$	$\{2, 5, 3\}$
$\{12, 17\}$	$\{8, 9\}$	$\{10, 11\}$	$\{2, 10, 8\}$
	$\{13, 14\}$	$\{15, 16\}$	$\{12, 10, 13\}$
			$\{12, 15, 13\}$

Table 7.1: The set \mathcal{I} of $\mathcal{P}_{\min}^r(S_{D_3})$ -interconnections in D_3 for the sets of \mathcal{L} .

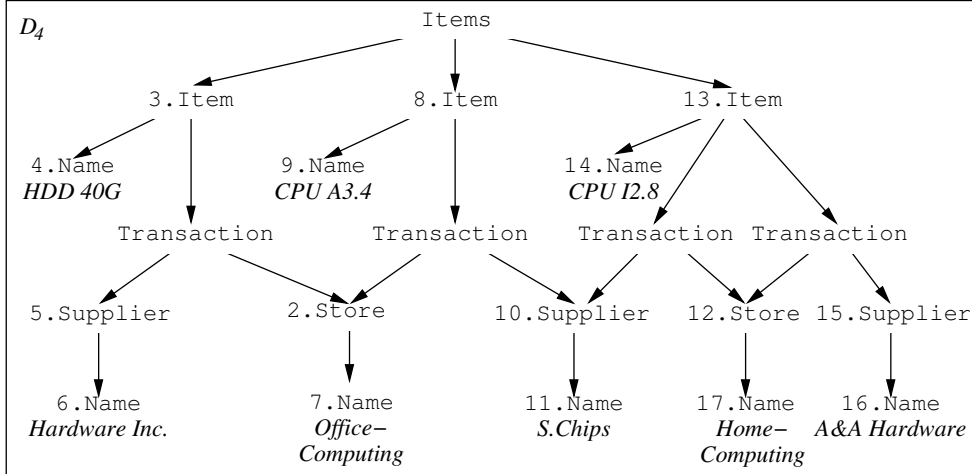


Figure 7.4: The result document D_4 .

that will be used for the target document. In our example, we choose $\mathcal{P}_{\min}^r(S_2)$. We use \mathcal{U} to denote the set of all the objects in \mathcal{I} , i.e., all objects that appear in Table 7.1. It is now possible to produce the required target document by executing $\text{LAYOUT}(\mathcal{U}, S_2, \mathcal{I}, \mathcal{P}_{\min}^r(S_2))$. The result of the algorithm is the document D_4 , shown in Figure 7.4.

Line 3 of the algorithm LAYOUT looks for patterns in the target semantics $\mathcal{P}_{\min}^r(S_2)$, which is not given explicitly. The necessary patterns can be generated by applying to the schema S_2 an algorithm that finds Steiner trees. This algorithm must be executed for each set of labels in \mathcal{L} . If these sets are small, the Steiner trees can be found efficiently and hence, the patterns can be efficiently generated. Note that the number of patterns that must be generated is at most the number of sets in \mathcal{L} .

For our example, the following four patterns are created during the execution of the algorithm: $(\{\text{Store, Supplier, Item}\}, C_4)$, $(\{\text{Store, Name}\}, C_6)$, $(\{\text{Item, Name}\}, C_5)$ and

$(\{\text{Supplier}, \text{Name}\}, C_7)$, where C_4 , C_5 , C_6 and C_7 are the l-trees that are shown in Figure 7.3. These patterns are minimal w.r.t. S_2 .

Chapter 8

NP-Complete Properties of Labeled Graphs

In this chapter, we discuss some NP-complete problems in labeled graphs. For compliance with the rest of this work, we use o-graphs to model labeled graphs.

8.1 Uniquely Labeled Paths

Consider a directed o-graph G and two objects o_1 and o_2 in G . We will show that the following problems are NP-complete.

1. Determine whether G has a path from o_1 to o_2 that is uniquely labeled.
2. Determine whether G has a simple path from o_1 to o_2 that is not uniquely labeled.

We will show that the undirected version of Problem 1 is also NP-complete. However, the undirected version of Problem 2 is in polynomial time, as shown in Theorem 5.5.

The following lemma is used to derive some of our NP-completeness results.

Proposition 8.1 *Let o_1 and o_2 be two objects in a directed o-graph G . It is NP-complete to test whether o_2 is reachable from o_1 through a directed (undirected) path that is uniquely labeled, even if G is acyclic.*

Proof. Membership in NP is immediate. To prove NP-hardness, we use a reduction from *exact cover by 3-sets* (X3C). This problem is defined as follows. Given a set \mathcal{X} of size $3q$

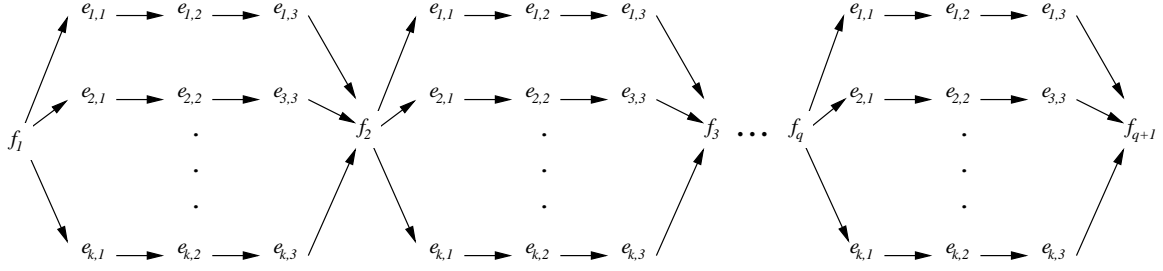


Figure 8.1: The o-graph G constructed from an instance of the X3C problem.

and a collection \mathcal{T} of 3-element subsets of \mathcal{X} , determine whether \mathcal{T} contains a cover of \mathcal{X} by q (pairwise-disjoint) subsets. X3C is known to be NP-complete. Suppose that $(\mathcal{X}, \mathcal{T})$ is an instance of X3C. Let $\mathcal{X} = \{e_1, \dots, e_{3q}\}$ and $\mathcal{T} = \{\{e_{i,1}, e_{i,2}, e_{i,3}\} \mid 1 \leq i \leq k\}$. We construct the following o-graph G from \mathcal{X} and \mathcal{T} .

- G contains objects o_1, \dots, o_{q+1} labeled with f_1, \dots, f_{q+1} , respectively.
- For all $1 \leq j \leq q$, object o_j is the source of k pairwise-disjoint paths ending at o_{j+1} , where the i th path ($1 \leq i \leq k$) consists of three objects that are labeled with $e_{i,1}$, $e_{i,2}$ and $e_{i,3}$.

Note that we use the elements of \mathcal{X} as labels. The constructed o-graph is illustrated in Figure 8.1 (only the labels of the objects are shown). Observe that this graph is acyclic. The following claim completes our proof.

Claim 1 *The o-graph G contains a directed (undirected) uniquely-labeled path from o_1 to o_{q+1} if and only if \mathcal{X} can be covered by exactly q subsets in \mathcal{T} .*

To prove the claim, consider a simple path P (either directed or undirected) between objects o_1 and o_{q+1} . The path P contains all the objects o_1, \dots, o_{q+1} , and $3q$ objects with labels from q (not necessarily distinct) subsets in \mathcal{T} . There is such a path that is uniquely labeled if and only if \mathcal{T} contains q subsets that are pairwise disjoint (i.e., exactly q subsets that cover \mathcal{X}). This proves the claim. \square

Proposition 8.2 *Let o_1 and o_2 be two objects in a directed o-graph G . It is NP-complete to test whether G contains a directed path from o_1 to o_2 that has two distinct objects with the same label.*

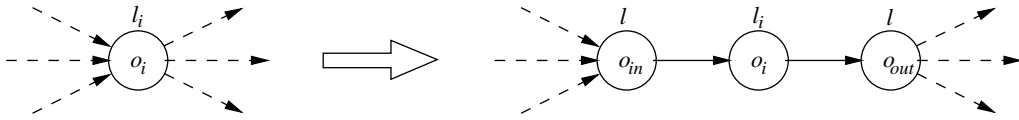


Figure 8.2: Transformation of G into G' .

Proof. Membership in NP is immediate. To prove NP-hardness of this problem, we use a reduction from a hard case of the *subgraph homeomorphism problem* (SHP). In [6], it was shown that the following problem is NP-complete. Given three nodes v_s , v_t and v_i in a graph, determine whether v_t is reachable from v_s through a simple directed path that visits v_i . For simplicity, we assume that an instance of this problem consists of a uniquely-labeled o-graph G and three objects o_s , o_t and o_i in G . Our goal is to determine whether G contains a simple directed path that starts with o_s , visits o_i and ends with o_t . Given such an instance, we construct the o-graph G' that is obtained by adding to G two objects o_{in} and o_{out} , both labeled with a new label l , as described in Figure 8.2. The incoming edges of o_{in} are the edges that entered o_i in G , and the outgoing edges of o_{out} are the edges that emanated o_i in G . Object o_i has only two edges in G' , one arrives from o_{in} and the other enters o_{out} . All the other objects and edges of G' are the same as in G . Note that o_{in} and o_{out} are the only objects in G' that share a common label. Thus, the following are equivalent:

1. G contains a simple directed path from o_s to o_t that visits o_i .
2. G' contains a simple directed path from o_s to o_t that visits some label twice.

This proves the NP-hardness of our problem. □

8.2 Uniquely-Labeled Subgraph Isomorphisms

The *labeled-subgraph isomorphism* problem is known to be NP-complete. The *subgraph isomorphism* problem is a special case of this problem. The latter remains hard even if the source graph is assumed to be a tree. For example, a graph G with n nodes has a *Hamiltonian path* if and only if it has a subgraph that is isomorphic to a path of length n .

Thus far, isomorphism between o-graphs and l-graphs has only been defined for trees. Isomorphism between arbitrary o-graphs and l-graphs is similarly defined as follows. A directed (undirected) o-graph G is isomorphic to a directed (undirected) l-graph S if (1) G is uniquely labeled, (2) the labels of G are exactly the nodes of S , and (3) S has an edge (l_1, l_2) if and only if G has an edge (o_1, o_2) , where o_1 and o_2 are the objects in G that are labeled with l_1 and l_2 , respectively.

The *uniquely-labeled-subgraph isomorphism* is defined as follows. Given a directed (undirected) o-graph G and a directed (undirected) l-graph S , determine whether G contains a subgraph that is isomorphic to S . We show that both the directed and undirected versions of this problem are NP-complete. We begin with the undirected case.

Proposition 8.3 *Let G be an undirected o-graph and S be an undirected l-graph. Testing whether S is isomorphic to a subgraph of G is NP-complete.*

Proof. Membership in NP is easy. We prove NP-hardness by a reduction from 3SAT. Let $U = \{u_1, \dots, u_k\}$ be a set of variables and $F = C_1 \wedge \dots \wedge C_m$ be a conjunction of clauses over U . We denote $C_i = a_{i,1} \vee a_{i,2} \vee a_{i,3}$, where each $a_{i,j}$ is a literal. We construct the following o-graph G_F from F .

- G_F has an object $o_{i,j}$ for each literal $a_{i,j}$. The label of object $o_{i,j}$ is the clause C_i .
- G_F has an edge between every two objects $o_{i,j}$ and $o_{i',j'}$, such that $i \neq i'$ and $a_{i,j}$ is not the negation of $a_{i',j'}$.

Let S_F be the complete l-graph over the labels C_1, \dots, C_m . We complete the proof by showing that the following holds.

Claim 1 *S_F is isomorphic to a subgraph of G_F if and only if F has a satisfying assignment.*

To prove this claim, observe that a subgraph $H \subseteq G_F$ that is isomorphic to S_F can only be obtained by choosing an object $o_{i,j}$ for every clause C_i , such that the following holds. If $o_{i,j}$ and $o_{i',j'}$ are chosen, then $a_{i,j}$ and $a_{i',j'}$ do not contradict each other (i.e., they are not a literal and its negation). Hence, a subgraph of G_F that is isomorphic to S_F exists if and only if F is satisfiable. □

Following is the directed version of the above proposition.

Proposition 8.4 *Let G be a directed o-graph and S be a directed l-graph. Testing whether S is isomorphic to a subgraph of G is NP-complete, even if S is acyclic and G is acyclically labeled.*

Proof. Membership in NP is obvious. We prove NP-hardness by a reduction from the previous problem. Let G be an undirected o-graph and S be an undirected l-graph. Our goal is to determine whether a subgraph of G is isomorphic to S . We assume that G contains all the labels in S . We construct the directed o-graph G_r from G as follows. We arbitrarily choose some complete order \prec over the labels of G . The o-graph G_r consists of all objects in G , and all the edges (o_1, o_2) such that (1) (o_1, o_2) is an edge of G , and (2) $l(o_1) \prec l(o_2)$. We similarly construct the directed l-graph S_r from S ; that is, (1) S and S_r have the same set of labels, and (2) S_r has an edge (l_1, l_2) if and only if (l_1, l_2) is an edge of S and $l_1 \prec l_2$.

Obviously, S_r is acyclic and G_r is acyclically labeled. Furthermore, from the construction of G_r and S_r , it follows that G_r has a subgraph that is isomorphic to S_r if and only if G has a subgraph that is isomorphic to S , as required. \square

Chapter 9

Conclusion

We presented interconnection semantics as a general framework for defining meaningful relationships among nodes of XML documents. Interconnection semantics can be defined explicitly for specific documents or queries. We also described several interconnection semantics that can be derived automatically and have different precision-versus-recall trade-offs. These semantics turn out to be tractable for many important cases and hence, can be efficiently applied to real-world documents. We also investigated the problem of constructing a new document from a given set of objects. The new document should conform to a given schema and preserve the original interconnections among the given objects. We gave a construction algorithm and showed that under a natural condition of unambiguity, the new document has a minimal set of new interconnections among the given objects. Our approach can be used to automatically translate data from one XML document to another.

In some cases, meaningful relationships exist among objects that are not uniquely labeled, e.g., when a book has several authors. A formalism for dealing with such cases was developed in [2] for tree documents. $\mathcal{P}_{\text{all}}^r$ and $\mathcal{P}_{\text{all}}^u$ (and their universal versions) can be extended to deal with such cases, along the lines of [2], without changing the complexity.

Interconnection semantics can be useful in many applications. For example, in common approaches to querying heterogeneous information sources, each source is described by some relations. Interconnection semantics can be used to compute these relations from XML documents.

Few other papers have dealt directly with the problem of determining semantic re-

relationships among nodes of an XML document. This problem was considered in [2] for tree documents only and without taking the structure of the schema into consideration. No general framework for interconnection semantics was introduced in [2], but different alternatives for defining interconnections in tree documents were proposed; one of those is equivalent to $\mathcal{P}_{\text{all}}^r$. Compact skeletons were introduced in [8] for a different problem than ours—automatically inferring the structure of documents for the purpose of automatic wrapper generation. The approach of [8] can also be used for discovering semantic relationships among the nodes of XML documents; however, it only applies to “regular” documents. Interestingly, the approach of [8] can be modeled by interconnection semantics, under the reasonable assumption that a subtree of a document indicates a meaningful relationship only if it is uniquely labeled.

Data translation was considered in [7]. Their approach is similar to ours in the sense that it has the following two steps (however, these steps are done differently from our approach). First, they discover semantic relationships in the source database and represent them as relations. Second, they transform those relations into the target database in a manner that preserves the original semantic relationships (but may also introduce new ones). Discovering semantic relationships is based on “primary” paths (i.e., a limited form of hierarchies) and referential constraints, using a chase process. They cannot express all the semantic relationships that can be expressed in our framework. But since their chase is bounded (to guarantee termination), it is quite possible that their approach for discovering semantic relationships can be emulated by interconnection semantics. They did not address the issue of the new semantic relationships that may be introduced in the target database. Their approach is semi-automatic and requires user assistance.

The idea of creating XML documents from relational data by means of writing queries in an appropriate language was discussed in several papers (e.g., [3, 5, 10]). Our approach is simpler and can even be fully automatic, at the expense of possibly having recall and precision that are lower than those that can be obtained by meticulous human intervention.

Bibliography

- [1] M. Barg and R. Wong. Structural proximity searching for large collections of semi-structured data. In *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pages 175–182, Atlanta (Georgia, USA), November 2001. ACM Press.
- [2] S. Cohen, Y. Kanza, and Y. Sagiv. Generating relations from XML documents. In *Proc 9th International Conference on Database Theory*, Siena (Italy), January 2003. Springer-Verlag.
- [3] Sara Cohen, Yaron Kanza, and Yehoshua Sagiv. Sql4x: A flexible query language for xml and relational databases. In *Revised Papers from the 8th International Workshop on Database Programming Languages*, pages 263–280. Springer-Verlag, 2002.
- [4] S.E. Dreyfus and R.A. Wagner. The Steiner problem in graphs. *Networks*, 1:195–207, 1972.
- [5] Mary Fernández, Wang-Chiew Tan, and Dan Suci. Silkroute: trading between relations and xml. *Comput. Networks*, 33(1-6):723–745, 2000.
- [6] S.J. Fortune, J. Hopcroft, and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10:11–121, 1980.
- [7] L. Popa, Y. Velegrakis, R.J. Miller, M.A. Hernandez, and R. Fagin. Translating web data. In *Proc. 28th International Conference on Very Large Data Bases*, pages 598–609, Hong Kong (China), August 2002. Morgan Kaufmann Publishers.

- [8] A. Rajaraman and J.D. Ullman. Querying websites using compact skeletons. In *Proc. 20th Symposium on Principles of Database Systems*, pages 16–27, Santa Barbara (California, USA), May 2001. ACM Press.
- [9] N. Robertson and P.D. Seymour. Graph minors. XIII. the disjoint paths problem. *Journal of Combinatorial Theory Series B*, 63(1):65–110, 1995.
- [10] Jayavel Shanmugasundaram, Eugene Shekita, Rimon Barr, Michael Carey, Bruce Lindsay, Hamid Pirahesh, and Berthold Reinwald. Efficiently publishing relational data as xml documents. *The VLDB Journal*, 10(2-3):133–154, 2001.
- [11] Moshe Y. Vardi. The complexity of relational query languages (extended abstract). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, pages 137–146. ACM Press, 1982.
- [12] Mihalis Yannakakis. Algorithms for acyclic database schemes. In *Very Large Data Bases, 7th International Conference, September 9-11, 1981, Cannes, France, Proceedings*, pages 82–94. IEEE Computer Society, 1981.

Appendix A

Enumeration of Reduced Subtrees

In this chapter we present efficient algorithms for enumerating the rooted (undirected) subtrees, of a given graph, that are reduced w.r.t. a given subset of nodes.

The following notations are used. Let G be a directed graph and V be a subset of the nodes of G . By $\rho^r(G, V)$ we denote the set of rooted subtrees of G that are reduced w.r.t. V . Similarly, by $\rho^u(G, V)$ we denote the set of undirected subtrees of G that are reduced w.r.t. V .

A.1 Rooted Subtrees

In the following section, assume that G is a directed graph and that V is a set of nodes in G . We will describe an algorithm for generating $\rho^r(G, V)$ (i.e., the set of rooted subtrees of G that are reduced w.r.t. V). Our algorithm is recursive, and has a polynomial input-output complexity.

A.1.1 Extensions of Reduced Subtrees

Consider a node $v \in V$. Assume that T is a subtree of G that is reduced w.r.t. $V \setminus \{v\}$ and that T does not contain the node v . The *extensions* of T w.r.t. v are the subtrees \hat{T} of G such that (1) T is a subtree of \hat{T} , and (2) \hat{T} is reduced w.r.t. V . We distinguish between two types of extensions, as illustrated in Figure A.1.

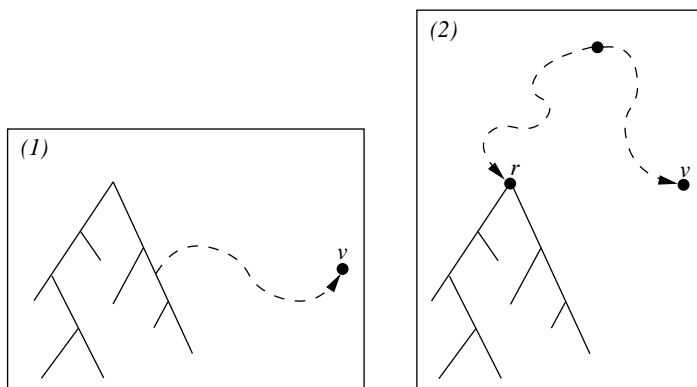


Figure A.1: Two types of subtree extensions: (1) an extension by a directed path, and (2) an extension by a rooted subtree.

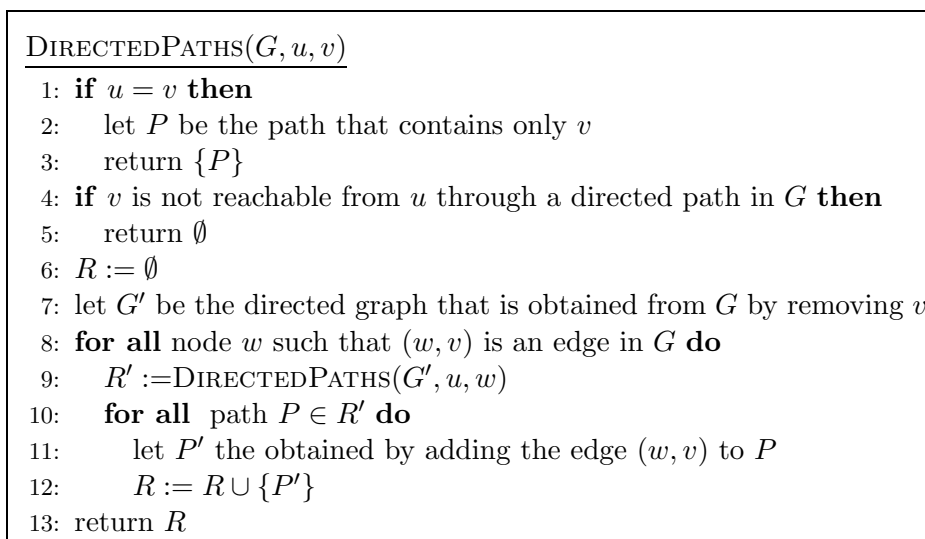


Figure A.2: An algorithm that generates all simple paths from u to v in G .

- **Extensions by directed paths**—rooted subtrees \hat{T} that are obtained by concatenating T and some directed path P from a non-root node u in T to v . The node u is the only node in P that is also in T .
- **Extensions by rooted subtrees**—rooted subtrees \hat{T} that are obtained by concatenating T and some rooted subtree T' such that T' is reduced w.r.t. $\{v, r\}$. The node r is the only node in T' that is also in T . Note that the root of \hat{T} is the root of T' , and hence, may be different from r .

<p><u>REDUCEDSUBTREES(G, u, v)</u></p> <ol style="list-style-type: none"> 1: if $u = v$ then 2: let P be the subtree that contains only v 3: return $\{P\}$ 4: if G does not contain an object w such that both u and v are reachable from w through directed paths in G then 5: return \emptyset 6: $R := \text{DIRECTEDPATHS}(G, v, u)$ 7: let G' be the directed graph that is obtained from G by removing v 8: for all node w such that (w, v) is an edge in G do 9: $R' := \text{REDUCEDSUBTREES}(G', u, w)$ 10: for all subtree $T \in R'$ do 11: let T' be the obtained by adding the edge (w, v) to T 12: $R := R \cup \{T'\}$ 13: return R
--

Figure A.3: An algorithm that generates all rooted subtrees of G that are reduced w.r.t. $\{u, v\}$.

In order to find all extensions of T by directed paths, we do the following. For each non-root node u in T , we create the graph G_u that is obtained from G by removing all nodes in T , except u . We use the algorithm $\text{DIRECTEDPATHS}(G_u, u, v)$ in order to generate the set of all simple directed paths from u to v in G_u . Finally, we output the concatenation of T with each of the paths that were generated.

In order to find all extensions of T by rooted subtrees, we do the following. We create the graph G_r that is obtained from G by removing all nodes in T , except for the root r of T . We then use the algorithm $\text{REDUCEDSUBTREES}(G_r, r, v)$ in order to generate the set of all the subtrees of G_r that are reduced w.r.t. $\{r, v\}$. Finally, we output the concatenation of T with each of the reduced subtrees that were generated.

A.1.2 Generating $\rho^r(G, V)$

In order to generate $\rho^r(G, V)$ in polynomial input-output complexity, we have to make sure that not too many intermediate results are generated. One problem that has to be dealt with is the following. For some subset $V' \subseteq V$, the set $\rho^r(G, V')$ may be much larger than the set $\rho^r(G, V)$. For example, consider the directed graph G_1 depicted in Figure A.4. Let V be the node set $\{a, b, c\}$. While G_1 contains exactly one rooted subtree that is reduced

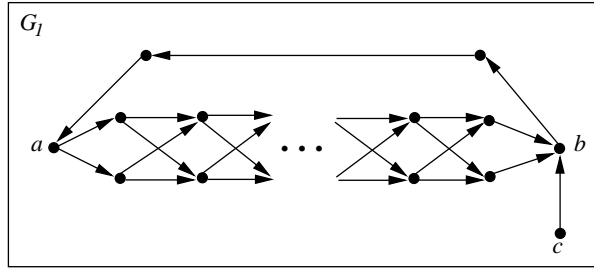


Figure A.4: A directed graph G_1 . This graph has exactly one subtree that is reduced w.r.t. $V = \{a, b, c\}$, and has an exponential number of subtrees that are reduced w.r.t. $V \setminus \{c\}$.

w.r.t. V , the number of its subtrees that are reduced w.r.t. $\{a, b\}$ is exponential in the size of G_1 . Hence, a naive recursion will not provide the desired complexity.

Our approach is to recursively generate subtrees that are either in $\rho^r(G, V)$ or can be extended to subtrees in $\rho^r(G, V)$. Our algorithm is described as follows.

1. We first deal with the trivial cases: $|V| = 1$ or G has no rooted subtrees that are reduced w.r.t. V . In order to test whether G has some rooted subtree that is reduced w.r.t. V , one has to test whether, for some node u in G , all the nodes in V are reachable from u through a directed path.
2. If some node $v \in V$ is reachable from another node in V , then we do the following:
 - We recursively generate $R = \rho^r(G, V \setminus \{v\})$;
 - For each $T \in R$, if T contains v then we output T , otherwise we output all the extensions of T w.r.t. v .

The fact that v is reachable from some other node in V guarantees that every subtree in $\rho^r(G, V \setminus \{v\})$ that does not contain v , has at least one extension w.r.t. v .

3. If no node in V is reachable from another one, we conclude that the following holds. In every tree of $\rho^r(G, V)$, the nodes of V are all leaves. Hence, we arbitrarily choose some node $v \in V$, and do the following. First, we generate the graph G_v that is obtained from G by removing the node v . Next, for each node u in G_v with an outgoing edge to v , we generate the set $\rho^r(G_v, V \cup \{u\} \setminus \{v\})$ and output the concatenation of each tree in this set with the edge (u, v) .

A.2 Undirected Subtrees

To generate $\rho^u(G, V)$, we use a simple variation of the algorithm that was used in the previous section. The main difference between the directed and the undirected cases is the following. An undirected subtree T that is reduced w.r.t. $V \setminus \{v\}$ can always be extended to a subtree \hat{T} that is reduced w.r.t. V by concatenating T with a simple path P , such that (1) P is a path between v and some node u in T , and (2) u is the only node in T that is visited by P . Hence, we generate $\rho^u(G, V)$ as follows.

1. We verify that the nodes in V are connected through undirected paths in G ;
2. We choose an arbitrary node $v \in V$;
3. We recursively generate $R = \rho^u(G, V \setminus \{v\})$;
4. For each $T \in R$, if T contains v then we output T ; otherwise, we output all the extensions of T w.r.t. v . An extension of an undirected subtree is obtained by concatenating undirected paths between nodes in T and v , as described above.