RESEARCH

# Modeling and Querying Semi-Structured Data

**Diego Calvanese**
**Giuseppe De Giacomo**
**Maurizio Lenzerini**
*Dipartimento di Informatica e Sistemistica*
*Università di Roma "La Sapienza"*
*Via Salaria 113, 00198 Roma, Italy*
{calvanese,degiacomo,lenzerini}@dis.uniroma1.it
http://www.dis.uniroma1.it/~{calvanese,degiacomo,lenzerini}

ABSTRACT. *We extend the model for semi-structured data proposed in [BUN 97], where both databases and schemas are represented as graphs, with the possibility of expressing different types of constraints on the nodes of the graphs. We discuss how the expressive power of the constraint language may influence the complexity of checking subsumption between schemas, and devise a polynomial algorithm for an interesting class of constraints. We then set up a framework for defining queries which are used to select graphs from a database. The proposed query language allows for expressing sophisticated fixpoint properties of graphs and can be regarded as a basic building block of full-featured languages. We show that reasoning tasks at the basis of query optimization, such as query-schema comparison, query containment, and query satisfiability, are decidable.*

KEY WORDS : *Semi-structured databases, WWW, query languages, description logics*

[0]This is an extended version of a paper submitted for publication to CAiSE'99

Networking and Information Systems Journal. Volume 2 - n° 2/1999, pages 253 à 273

## 1. Introduction

The ability to represent data whose structure is less rigid and strict than in conventional databases is considered a crucial aspect in modern approaches to data modeling, and is important in many application areas, such as web information systems, biological databases, digital libraries, and data integration [QUA 95, ABI 97a, BUN 97, MEN 97, FER 98, FLO 98].

Following [ABI 97a], we define semi-structured data as data that is neither raw, nor strictly typed as in conventional database systems. OEM (Object Exchange Model) [ABI 97c], and BDFS (Basic Data model For Semi-structured data) [BUN 97] are recent proposals of models for semi-structured data. They represent data as graphs with labeled edges, where information on both the values and the schema of data are kept.

In particular, BDFS is an elegant graph-based data model, where graphs are used to represent both portions of a database (called ground graphs) and schemas, the former with edges labeled by data, and the latter with edges labeled by formulae of a suitable logical theory. The notion of a ground graph $g$ conforming to a schema $S$ is given in terms of a special relation, called simulation, between the two graphs. Roughly speaking, a simulation is a correspondence between the edges of $g$ and those of $S$ such that, whenever there is an edge labeled $a$ in $g$, there is a corresponding edge in $S$ labeled with a formula satisfied by $a$. The notion of simulation is less rigid than the usual notion of satisfaction, and suitably reflects the need of dealing with less strict structures of data.

**Example 1** In Figure 1, we show a BDFS schema and a ground graph that conforms to it. The schema models documents representing papers with a title, a sequence of sections, each with an associated text, and a final section of references to other papers. We assume that in the theory specifying the labels of graphs titles, sections, texts, and references are mutually disjoint. ■

For several tasks related to data management, it is important to be able to check subsumption between two schemas, i.e. to check whether every ground graph conforming to one schema always conforms to another schema. In [BUN 97] an algorithm for checking subsumption in BDFS is presented and its complexity is analyzed.

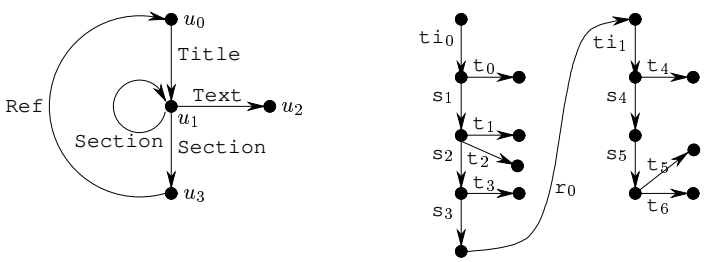



**Figure 1.** *Schema for papers divided in ordered sections and a conforming ground graph*

Additionally, in [BUN 97] the issue of extending the model with different types of constraints is raised. Indeed, in BDFS all the properties of the schema are expressed in terms of the structure of the graph, and the possibility of specifying additional constraints, such as existence of edges, is precluded.

In this paper we extend the framework of [BUN 97] presenting the following contributions:

— We extend BDFS schemas with constraints. The basic idea is to express constraints in terms of formulae associated to nodes of the schema. A formula on a node $u$ imposes a condition that, for every ground graph $g$ conforming to $S$, must be satisfied by every node of $g$ simulating $u$ (see Example 2). We consider different types of constraints, and we discuss how the expressive power of the constraint language influences the complexity of subsumption checking. In particular, we show that by adding edge-existence and functionality constraints the complexity of subsumption remains polynomial.

— We introduce a basic form of queries, called graph selection queries, which are used to select graphs from a database (see Example 3). The query language presented here represents a basic building block of a full-featured query language and has been designed on one hand to express sophisticated fixpoint properties of graphs, and on the other hand to keep several interesting reasoning tasks decidable. These reasoning tasks, such as comparing queries and schemas or checking containment between queries, are at the basis of query optimization techniques applicable to a more expressive query language.

**Example 2** The schema in Figure 1 presents several modeling problems, which are demonstrated by the sample ground graph. Although in principle we would like that each section has exactly one text associated to it, the schema allows for sections with more that one text or no text at all. Similarly, to correctly represent the order of sections it is essential to impose that each section is followed by at most one other section, and that a final section of references, if present, contains at least one reference. This calls for adding constraints on nodes $u_1$ and $u_3$ to impose restrictions on the number of outgoing edges, which we specify as $\mathcal{C}(u_1) = \exists^{=1}\mathsf{edge}\,(\mathtt{Text}) \wedge \exists^{\leq 1}\mathsf{edge}\,(\mathtt{Section})$ and $\mathcal{C}(u_3) = \exists\mathsf{edge}\,(\mathtt{Ref})$. ∎

**Example 3** Given a database containing ground graphs conforming to the schema in Figure 1,

$$\exists\mathsf{path}\,((\mathtt{Title}\circ\mathtt{Section}^*\circ\mathtt{Ref})^*\circ(\mathtt{Title}\wedge(\mathit{self} = \mathtt{GraphQueries})))\,\mathsf{to}\,(\top)$$

is a query that selects all papers that reference either directly or indirectly, via other papers, a paper of title `GraphQueries`. ∎

The paper is organized as follows. In Section 2 we describe the BDFS data model and the description logic $\mu\mathcal{ALCQ}$, which are the basic formalisms in our investigation. In Section 3 we address the problem of adding constraints to BDFS. In Section 4 we define a language for expressing graph selection queries. In Section 5 we describe the evaluation of graph selection queries. Finally, Section 6 concludes the paper.

## 2. Preliminaries

In this section, we describe the basic characteristics of the formalism for modeling semi-structured data proposed in [BUN 97], which we call BDFS, and which is the basis of our investigation. In addition we introduce the description logic $\mu\mathcal{ALCQ}$ [DEG 97] whose reasoning tasks are exploited to reason on semi-structured data schemas.

### 2.1. *The* BDFS *Data Model*

The formalism BDFS is appropriate for an edge-labeled graph model of data, where labels denote properties of edges representing the actual data. To express such properties, a decidable, complete[1] first-order theory $\mathcal{T}$ over a fixed, finite universe $\mathcal{U}$ is considered. The language of $\mathcal{T}$ includes one distinct constant for each element of $\mathcal{U}$ and special unary predicates of the form $(self = a)$, for each constant $a$, where $(self = a)(a')$ is **true** if and only if $a = a'$.

**Definition 4** *A $\mathcal{T}$-ground graph is a rooted connected graph whose edges are labeled with formulae of the form $(self = a)$, where $a$ is a constant of $\mathcal{T}$. A $\mathcal{T}$-graph schema is a rooted connected graph whose edges are labeled with unary formulae of $\mathcal{T}$.*

Note that a $\mathcal{T}$-ground graph is a special case of $\mathcal{T}$-graph schema. In what follows, we omit $\mathcal{T}$, and simply refer to ground graphs, and graph schemas (or simply schemas), respectively. Also, in the labels of ground graphs, we abbreviate $(self = a)$ with $a$, and we use the term graph to denote either a ground graph or a graph schema. A semi-structured database (or simply database) is a finite set of graphs. A database constituted only by ground graphs is called ground database.

For any graph $G$, we denote the root of $G$ by $root(G)$, the set of nodes of $G$ by $Nodes(G)$, and the set of edges of $G$ by $Edges(G)$. We denote an edge from node $u$ to node $v$ labeled by $p$ with $u \xrightarrow{p} v$.

**Definition 5** *Given a ground graph $g$ and a schema $S$, a simulation from $g$ to $S$ is a binary relation $\trianglelefteq$ from the nodes of $g$ to those of $S$ such that $u \trianglelefteq u'$ implies that for each edge $u \xrightarrow{a} v$ in $g$, there exists an edge $u' \xrightarrow{p} v'$ in $S$ such that $\mathcal{T} \models p(a)$, and $v \trianglelefteq v'$.*

**Definition 6** *A ground graph $g$ conforms to a schema $S$, in notation $g \preceq S$, if there exists a simulation from $g$ to $S$ such that $root(g) \trianglelefteq root(S)$.*

**Definition 7** *Given two schemas $S$ and $S'$, $S'$ subsumes $S$, in notation $S \sqsubseteq S'$, if for every ground graph $g$, $g \preceq S$ implies $g \preceq S'$. $S'$ and $S$ are equivalent if both $S \sqsubseteq S'$ and $S' \sqsubseteq S$.*

In [BUN 97], an algorithm is presented for checking subsumption (and conformance, being a ground graph a special case of schema). The algorithm essentially looks for the greatest simulation between the nodes of the two schemas, and works in

[1] The theory is complete in the sense that for every closed formula $f$, either $\mathcal{T}$ entails $f$, or $\mathcal{T}$ entails $\neg f$ [BUN 97].

time $O(m^{O(1)} \cdot t_{\mathcal{T}}(m))$, where $m$ is the size of the two schemas, and $t_{\mathcal{T}}(x)$ is the time needed to check whether a formula of size $x$ is valid in $\mathcal{T}$. In the setting of [BUN 97] it is meaningful not to consider $\mathcal{T}$ to be part of the input of the subsumption problem. Therefore, whenever $t_{\mathcal{T}}(m)$ may be assumed to be independent of $m$, $t_{\mathcal{T}}(m)$ can be replaced by a constant.

### 2.2. *The Description Logic* $\mu\mathcal{ALCQ}$

Description logics allow one to represent a domain of interest in terms of concepts and roles. Concepts model classes of individuals, while roles model relationships between classes. We concentrate on the description logic $\mu\mathcal{ALCQ}$ studied in [DEG 97], where a correspondence was shown with a well-known logic of programs, called modal mu-calculus [KOZ 83, STR 89], that is used for expressing temporal properties of reactive and parallel processes [STI 96, EME 96]. In fact, $\mu\mathcal{ALCQ}$ can be viewed as a variant of modal mu-calculus extended with graded modalities (see e.g. [HOE 95]). $\mu\mathcal{ALCQ}$ can also be viewed as a well-behaved fragment of first-order logic with fixpoints (see e.g. [ABI 95]).

We make use of the standard first-order notions of scope, bound and free occurrences of variables, closed formulae, etc., treating $\mu$ and $\nu$ as quantifiers.

The primitive symbols in $\mu\mathcal{ALCQ}$ are atomic concepts, (concept) variables, and atomic roles (in the following called simply roles). Concepts are formed according to the following syntax:

$$C \ ::= \ A \ \mid \ \neg C \ \mid \ C_1 \sqcap C_2 \ \mid \ \exists R.C \ \mid \ (\geq n\, R.C) \ \mid \ \mu X.C \ \mid \ X$$

where $A$ denotes an atomic concept, $R$ a role, $n$ a natural number, and $X$ a variable, and the restriction is made that every free occurrence of $X$ in $\mu X.C$ is in the scope of an even number of negations.

We introduce the following abbreviations:

$$\begin{aligned}
C_1 \sqcup C_2 \quad &\text{for} \quad \neg(\neg C_1 \sqcap \neg C_2) \\
\top \quad &\text{for} \quad A \sqcup \neg A \\
\bot \quad &\text{for} \quad \neg\top \\
\exists R.C \quad &\text{for} \quad (\geq 1\, R.C) \\
\forall R.C \quad &\text{for} \quad \neg\exists R.\neg C \\
(\leq n\, R.C) \quad &\text{for} \quad \neg(\geq n{+}1\, R.C) \\
(= n\, R.C) \quad &\text{for} \quad (\leq n\, R.C) \sqcap (\geq n\, R.C) \\
\nu X.C \quad &\text{for} \quad \neg\mu X.\neg C[X/\neg X]
\end{aligned}$$

where $C[X/\neg X]$ is the concept obtained by substituting all free occurrences of $X$ with $\neg X$.

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of an interpretation domain $\Delta^{\mathcal{I}}$, and an interpretation function $\cdot^{\mathcal{I}}$, which maps every atomic concept to a subset of $\Delta^{\mathcal{I}}$, and every atomic role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The presence of free variables does not allow us to extend the interpretation function $\cdot^{\mathcal{I}}$ directly to every concept of the logic. For this reason we introduce valuations. A valuation $\rho$ on an interpretation $\mathcal{I}$ is a mapping from variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation $\rho$, we denote by $\rho[X/\mathcal{E}]$ the valuation identical to $\rho$ except for the fact that $\rho[X/\mathcal{E}](X) = \mathcal{E}$.

Let $\mathcal{I}$ be an interpretation and $\rho$ a valuation on $\mathcal{I}$. We assign meaning to concepts of the logic by associating to $\mathcal{I}$ and $\rho$ an extension function $\cdot^{\mathcal{I}}_{\rho}$, mapping concepts to subsets of $\Delta^{\mathcal{I}}$, as follows:

$$\begin{aligned}
X^{\mathcal{I}}_{\rho} &= \rho(X) \subseteq \Delta^{\mathcal{I}} \\
A^{\mathcal{I}}_{\rho} &= A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \\
(\neg C)^{\mathcal{I}}_{\rho} &= \Delta^{\mathcal{I}} - C^{\mathcal{I}}_{\rho} \\
(C_1 \sqcap C_2)^{\mathcal{I}}_{\rho} &= (C_1)^{\mathcal{I}}_{\rho} \cap (C_2)^{\mathcal{I}}_{\rho} \\
(\geq n\, R.C)^{\mathcal{I}}_{\rho} &= \{s \in \Delta^{\mathcal{I}} \mid \#\{s' \mid (s,s') \in R^{\mathcal{I}} \text{ and } s' \in C^{\mathcal{I}}_{\rho}\} \geq n\} \\
(\mu X.C)^{\mathcal{I}}_{\rho} &= \bigcap \{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]} \subseteq \mathcal{E}\ \}
\end{aligned}$$

Observe that the semantics assigned to $\nu X.C$ is

$$(\nu X.C)^{\mathcal{I}}_{\rho} = \bigcup \{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid\ \mathcal{E} \subseteq\ C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}\ \}$$

The expression $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ can be seen as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, and, by the syntactic restriction enforced on variables, such an operator is guaranteed to be monotonic wrt $\subseteq$. The constructs $\mu X.C$ and $\nu X.C$ denote respectively the least fixpoint and the greatest fixpoint of the operator. The extension of closed concepts is independent of the valuation, and therefore for closed concepts we do not consider the valuation explicitly. A closed concept $C$ is satisfiable if there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

A $\mu\mathcal{ALCQ}$ knowledge base is a finite set of axioms $C_1 \sqsubseteq C_2$ where $C_1$ and $C_2$ are closed concepts of $\mu\mathcal{ALCQ}$. We use $C_1 \equiv C_2$ as an abbreviation for the pair of axioms $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. An interpretation $\mathcal{I}$ satisfies an axiom $C_1 \sqsubseteq C_2$, if $C^{\mathcal{I}}_1 \subseteq C^{\mathcal{I}}_2$. $\mathcal{I}$ is a model of a knowledge base $\mathcal{K}$, if $\mathcal{I}$ satisfies all axioms in $\mathcal{K}$. A closed concept $C$ is satisfiable in a knowledge base $\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$.

**Theorem 8 ([DEG 97])** *Satisfiability of (closed) $\mu\mathcal{ALCQ}$ concepts and satisfiability of (closed) $\mu\mathcal{ALCQ}$ concepts in $\mu\mathcal{ALCQ}$ knowledge bases are EXPTIME-complete problems.*

## 3. Schemas with Constraints

We address now the problem of extending the BDFS data model in order to express constraints on a schema. We conceive a constraint for a schema $S$ as a formula associated to a node $u$ of the schema. The formula is expressed in a certain language $\mathcal{L}$, and its role is to impose a condition that, for every ground graph $g$ conforming to $S$, must be satisfied by every node of $g$ simulating $u$. In other words, constraints are used to impose additional conditions on the schema, with respect to those already implied by the structure of the graph.

**Definition 9** *A schema with $\mathcal{L}$-constraints, or simply $\mathcal{L}$-schema, is a schema where each node $u$ is labeled by a formula $\mathcal{C}(u)$ of the constraint language $\mathcal{L}$.*

**Definition 10** *Given a ground graph $g$ and an $\mathcal{L}$-schema $S$, a simulation from $g$ to $S$ is a binary relation $\trianglelefteq$ from the nodes of $g$ to those of $S$ such that $u \trianglelefteq u'$ implies that (1) $u$ satisfies $\mathcal{C}(u')$, and (2) for each edge $u \xrightarrow{a} v$ in $g$, there exists an edge $u' \xrightarrow{p} v'$ in $S$ such that $\mathcal{T} \models p(a)$, and $v \trianglelefteq v'$.*

The notions of conformance, subsumption and equivalence remain unchanged, given the new definition of simulation. We assume that $\mathcal{L}$ contains the formula $\top$, which is satisfied by every node of every ground graph. Therefore, we can view a ground graph $g$ as an $\mathcal{L}$-schema, where $\mathcal{C}(u) = \top$ for every node $u$ of $g$. Thus, conformance is again a special case of subsumption.

Since constraints may contradict each other, or may even be incompatible with the structure of the graph, the notion of consistency becomes relevant (notice that a ground graph is always consistent). Moreover, we can introduce the notion of disjointness between $\mathcal{L}$-schemas.

**Definition 11** *Given an $\mathcal{L}$-schema $S$, a node $u \in Nodes(S)$ is consistent if there is at least one ground graph which conforms to $S'$, where $S'$ is equal to $S$ except that $root(S') = u$. $S$ is consistent, if $root(S)$ is consistent. Two $\mathcal{L}$-schemas $S_1$ and $S_2$ are disjoint, if there is no ground graph that conforms to $S_1$ and $S_2$.*

We consider now different forms of constraints, and study consistency and subsumption checking. Being conformance a special case of subsumption, we do not explicitly deal with conformance.

### 3.1. *Local Constraints*

We consider a language $\mathcal{L}_l$ in which only local constraints can be expressed, i.e. only constraints on the edges directly emanating from a node. Formulae in $\mathcal{L}_l$ have the following syntax ($\gamma$, $\gamma_1$ and $\gamma_2$ denote constraints, and $p$ denotes a formula of $\mathcal{T}$):

$$\gamma \ ::= \ \top \ \mid \ \exists \mathsf{edge}\,(p) \ \mid \ \neg\exists \mathsf{edge}\,(p) \ \mid \ \exists^{\leq 1}\mathsf{edge}\,(p) \ \mid \ \gamma_1 \wedge \gamma_2$$

We use $\exists^{=1}\mathsf{edge}\,(p)$ as an abbreviation for $\exists \mathsf{edge}\,(p) \wedge \exists^{\leq 1}\mathsf{edge}\,(p)$. Intuitively, a constraint of the form $\exists \mathsf{edge}\,(p)$ on a node $u$, called edge-existence constraint,

imposes that $u$ has at least one outgoing edge $u \xrightarrow{a} v$ such that $\mathcal{T} \models p(a)$, while a constraint of the form $\exists^{\leq 1}\mathsf{edge}\,(p)$, called functionality-constraint, imposes that $u$ has at most one such outgoing edge. More precisely, let $S$ be an $\mathcal{L}_l$-schema and $g$ a ground graph. Then a node $u$ of $g$ satisfies a constraint $\gamma$, in notation $u \models \gamma$, if the following conditions are satisfied:

$$
\begin{array}{lll}
u \models \top & & \\
u \models \exists\mathsf{edge}\,(p) & \text{iff} & \exists u \xrightarrow{a} v \in \mathit{Edges}(g).\ \mathcal{T} \models p(a) \\
u \models \neg\exists\mathsf{edge}\,(p) & \text{iff} & \forall u \xrightarrow{a} v \in \mathit{Edges}(g).\ \mathcal{T} \models \neg p(a) \\
u \models \exists^{\leq 1}\mathsf{edge}\,(p) & \text{iff} & \#\{u \xrightarrow{a} v \in \mathit{Edges}(g) \mid \mathcal{T} \models p(a)\} \leq 1 \\
u \models \gamma_1 \wedge \gamma_2 & \text{iff} & (u \models \gamma_1) \ \wedge \ (u \models \gamma_2)
\end{array}
$$

First of all, we show that we do not lose in expressiveness if we omit from $\mathcal{L}_l$ the possibility of using constraints of the form $\neg\exists\mathsf{edge}\,(p)$. In fact, given an $\mathcal{L}_l$-schema $S$, we can obtain an equivalent $\mathcal{L}_l$-schema $\mathit{rnec}(S) = S'$ not containing constraints of the form $\neg\exists\mathsf{edge}\,(p)$ and with the same set of nodes as $S$ as follows. For every node $u$ in $S$ with $\mathcal{C}(u) = \exists\mathsf{edge}\,(p_1)\wedge\cdots\wedge\exists\mathsf{edge}\,(p_r)\wedge\neg\exists\mathsf{edge}\,(n_1)\wedge\cdots\wedge\neg\exists\mathsf{edge}\,(n_s)\wedge \exists^{\leq 1}\mathsf{edge}\,(f_1) \wedge \cdots \wedge \exists^{\leq 1}\mathsf{edge}\,(f_t)$ and outgoing edges $u \xrightarrow{q_1} v_1, \ldots, u \xrightarrow{q_k} v_k$, we set the label of $u$ in $S'$ as $\mathcal{C}(u) = \exists\mathsf{edge}\,(p_1) \wedge \cdots \wedge \exists\mathsf{edge}\,(p_r) \wedge \exists^{\leq 1}\mathsf{edge}\,(f_1) \wedge \cdots \wedge \exists^{\leq 1}\mathsf{edge}\,(f_t)$, and for $i \in \{1, \ldots, k\}$ we replace in $u \xrightarrow{q_i} v_i$ the formula $q_i$ by $q_i' = q_i \wedge \neg n_1 \wedge \cdots \wedge \neg n_s$.

**Lemma 12** *If $S$ is an $\mathcal{L}_l$-schema, then $\mathit{rnec}(S)$ is equivalent to $S$ and its size is polynomial in $|S|$.*

*Proof.* Let $S'$ be $\mathit{rnec}(S)$.

"$S \sqsubseteq S'$" Let $g$ be a ground graph that conforms to $S$ and $\trianglelefteq$ a simulation from $g$ to $S$ respecting the constraints of $S$. (i.e. all conditions in Definition 10). We show that $\trianglelefteq$ is also a simulation from $g$ to $S'$ respecting the constraints of $S'$. Indeed, let $d$ be a node of $g$ and $u$ a node of $S$ (and $S'$) with $d \trianglelefteq u$. For each edge $d \xrightarrow{a} e$ from $u$ in $g$ there is an edge $u \xrightarrow{q} v$ is $S$ such that $\mathcal{T} \models q(a)$ and $e \trianglelefteq v$. Since $d \models_c \mathcal{C}(u)$, we also have $d \models_c \mathcal{C}'(u)$, and moreover $\mathcal{T} \models \neg p(a)$, for all $\neg\exists\mathsf{edge}\,(p)$ appearing in $\mathcal{C}(u)$. Hence $\mathcal{T} \models q'(a)$.

"$S' \sqsubseteq S$" Similar. □

Next we present a method for checking consistency, based on the function $\mathit{rin}$ defined in Figure 2. The role of $\mathit{rin}$ is to first remove the non-existence constraints by calling the function $\mathit{rnec}$, and then remove all inconsistent nodes from a schema. Condition (1) ensures that nodes not connected to the root are removed, while conditions (2) and (3) remove nodes in which a constraint cannot be satisfied. In particular, condition (2) deals with nodes having no outgoing edges but requiring the existence of at least one, while condition (3) verifies the existence in $\mathcal{T}$ of appropriate objects that can simultaneously satisfy the edge-existence and functionality constraints.

**Theorem 13** *An $\mathcal{L}_l$-schema $S$ is consistent if and only if $\mathit{rin}(S)$ contains $\mathit{root}(S)$. Moreover, $\mathit{rin}(S)$ runs in time polynomial in $|S|$.*

**function** $rin(S$: $\mathcal{L}$-schema): $\mathcal{L}$-schema;
{ $S' \leftarrow rnec(S)$;
**repeat**
**if** there is a node $u$ in $S'$ with
$\mathcal{C}(u) = \exists\mathsf{edge}\,(p_1) \wedge \cdots \wedge \exists\mathsf{edge}\,(p_r) \wedge \exists^{\leq 1}\mathsf{edge}\,(f_1) \wedge \cdots \wedge \exists^{\leq 1}\mathsf{edge}\,(f_s)$,
that satisfies one of the following conditions:
(1) $u$ is not connected to $root(S')$ in $S'$
(2) $r \geq 1$ and $u$ has no outgoing edge in $S'$
(3) $r \geq 1$, $u \stackrel{q_1}{\rightarrow} v_1, \ldots, u \stackrel{q_m}{\rightarrow} v_m$, with $m \geq 1$, are all outgoing edges of $u$ in $S'$, and
$\mathcal{T} \models \neg\exists x_1 \cdots \exists x_r(\bigwedge_{1\leq i\leq r}(p_i(x_i) \wedge \bigvee_{1\leq j\leq n} q_j(x_i)) \wedge$
$\bigwedge_{1\leq k\leq s}\bigwedge_{1\leq i<j\leq r}((f_k(x_i) \wedge f_k(x_j)) \supset x_i = x_j))$
**then** remove from $S'$ the node $u$ and all edges from and to $u$;
**until** $root(S')$ has been removed from $S'$ **or** no new node has been removed from $S'$;
**return** $S'$
}

**Figure 2.** *Function rin that removes non-existence constraints and inconsistent nodes*

*Proof.* "$\Leftarrow$" If a node $u$ of $S$ is consistent, then there is a ground graph which conforms to the schema $S_u$ identical to $S$ except for the root which is $u$. Hence neither condition (2) nor (3) of $rin$ can be satisfied for $u$, and if $u = root(S)$ then $u$ is not removed from $S$.

"$\Rightarrow$" Let $S' = rin(S)$ and $u$ a node (connected to $root(S) = root(S')$) in $S'$. If conditions (2) and (3) in $rin$ are not satisfied for $u$, then: either $\mathcal{C}(u)$ contains no edge-existence constraints, and the ground graph consisting of a single node conforms to $S_u$, or $\mathcal{C}(u)$ contains edge-existence constraints $\exists\mathsf{edge}\,(p_1), \ldots, \exists\mathsf{edge}\,(p_r)$, $u$ has outgoing edges $u \stackrel{q_1}{\rightarrow} v_1, \ldots, u \stackrel{q_m}{\rightarrow} v_m$ in $S'$ and the formula in condition (3) is not satisfied. In this case there are (not necessarily distinct) objects $a_1, \ldots, a_r$ in $\mathcal{T}$ which can be used to construct a ground graph with a root $d$, having outgoing edges labeled with $a_1, \ldots, a_r$, and satisfying $\mathcal{C}(u)$.

"Complexity" The number of iterations is bounded by the number of nodes in $S$, and at each iteration, for each node $u$ a validity check is done for a formula of $\mathcal{T}$ whose size is bounded by a polynomial in the sum of the size of $\mathcal{C}(u)$ and the sizes of the formulae labeling the outgoing edges of $u$. Hence $rin$ runs in time $O(|S|^{O(1)} \cdot t_{\mathcal{T}}(|S|^{O(1)}))$, and the thesis follows since $t_{\mathcal{T}}(|S|^{O(1)})$ is assumed to be constant. □

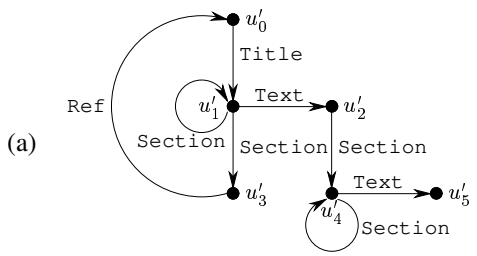
We now turn our attention to the method for checking subsumption of schemas with constraints, which is also a method for checking conformance of ground graphs to schemas. The method is based on the function $subs$ defined in Figure 3. Note that $subs$ is an extension of the algorithm in [BUN 97]. Its basic idea is to look for a simulation between the two schemas by constructing a relation $R$ as the Cartesian product of the two sets of nodes, and then removing from $R$ all the pairs $(u, u')$ for which no relation $\trianglelefteq$ satisfying condition (2) of Definition 10 may exist. Intuitively,

**function** $subs(S_0, S_0'$: $\mathcal{L}$-schema): boolean
{ $S \leftarrow rin(S_0)$;
  $S' \leftarrow rin(S_0')$;
  **if** $S$ does not contain $root(S_0)$ **then return** true;
  **if** $S'$ does not contain $root(S_0')$ **then return** false;
  $R \leftarrow \{(u, u') \mid u \in Nodes(S), u' \in Nodes(S')\}$;
  **repeat**
    **if** there is $(u, u') \in R$, with $u \xrightarrow{q_1} v_1, \ldots, u \xrightarrow{q_n} v_n$ all outgoing edges of $u$ in $S$,
      $\mathcal{C}(u) = \exists \mathsf{edge}(p_1) \wedge \cdots \wedge \exists \mathsf{edge}(p_r) \wedge \exists^{\leq 1} \mathsf{edge}(f_1) \wedge \cdots \wedge \exists^{\leq 1} \mathsf{edge}(f_s)$,
      $\mathcal{C}(u') = \exists \mathsf{edge}(p_1') \wedge \cdots \wedge \exists \mathsf{edge}(p_{r'}') \wedge \exists^{\leq 1} \mathsf{edge}(f_1') \wedge \cdots \wedge \exists^{\leq 1} \mathsf{edge}(f_{s'}')$,
      that satisfies one of the following conditions:
      (1) there is $i \in \{1, \ldots, n\}$ such that
        $\mathcal{T} \models \exists x_0 \exists x_1 \cdots \exists x_r (q_i(x_0) \wedge \bigwedge_{1 \leq j \leq m} \neg q_j'(x_0) \wedge$
        $\bigwedge_{1 \leq j \leq r} (p_j(x_j) \wedge \bigvee_{1 \leq k \leq n} q_k(x_j)) \wedge$
        $\bigwedge_{1 \leq \ell \leq s} \bigwedge_{0 \leq j < k \leq r} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))$
        where $u' \xrightarrow{q_j'} v_j'$, $j \in \{1, \ldots, m\}$ are all edges from $u'$ in $S'$ such that $(v_i, v_j') \in R$
      (2) $r = 0$ and $r' \neq 0$, or $r \neq 0$ and there is $i \in \{1, \ldots, r'\}$ such that
        $\mathcal{T} \models \exists x_1 \cdots \exists x_r (\bigwedge_{1 \leq j \leq r} \neg p_i'(x_j) \wedge$
        $\bigwedge_{1 \leq j \leq r} (p_j(x_j) \wedge \bigvee_{1 \leq k \leq n} q_k(x_j)) \wedge$
        $\bigwedge_{1 \leq \ell \leq s} \bigwedge_{1 \leq j < k \leq r} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))$
      (3) there is $i \in \{1, \ldots, s'\}$ such that
        $\mathcal{T} \models \exists x_1 \cdots \exists x_r \exists x_{r+1} \exists x_{r+2} (f_i'(x_{r+1}) \wedge f_i'(x_{r+2}) \wedge x_{r+1} \neq x_{r+2} \wedge$
        $\bigwedge_{1 \leq j \leq r} p_j(x_j) \wedge$
        $\bigwedge_{1 \leq j \leq r+2} \bigvee_{1 \leq k \leq n} q_k(x_j) \wedge$
        $\bigwedge_{1 \leq \ell \leq s} \bigwedge_{1 \leq j < k \leq r+2} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))$
    **then** remove $(u, u')$ from $R$;
  **until** no new pair has been removed from $R$;
  **return** $(root(S), root(S')) \in R$
}

**Figure 3.** *Function subs that verifies subsumption of schemas with local constraints*

the algorithm checks locally for the pair $(u, u')$, whether it is possible to construct a ground graph $g$ which can be used as a counterexample to the subsumption, and which consists just of a node $d$ and the nodes connected to $d$ by means of its outgoing edges. In particular, condition (1) checks the existence of an object in $\mathcal{T}$ which can label an edge from $d$ which has a corresponding edge from $u$ but none from $u'$. Due to the functionality constraints on $u$, this test must also take into account the constraints on $u$ in $S$. Condition (2) checks whether $g$ could violate the edge-existence constraints on $u'$ while satisfying the constraints on $u$, and condition (3) does a similar check for the functionality constraints on $u'$.

**Theorem 14** *If $S_1$ and $S_2$ are $\mathcal{L}_l$-schemas, then $S_1 \sqsubseteq S_2$ if and only if $subs(S_1, S_2)$*



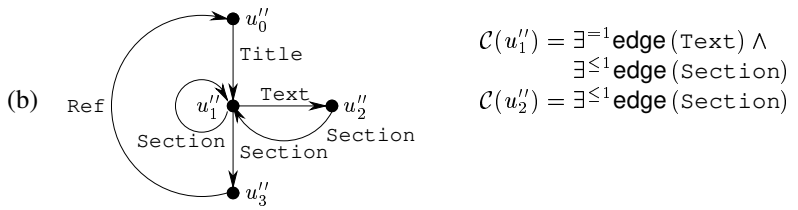




**Figure 4.** *Schemas for papers divided in ordered nested sections*

*returns true. Moreover, $subs(S_1, S_2)$ runs in time polynomial in $|S_1| + |S_2|$.*

*Proof.* For a schema $S$ and a node $u$ of $S$, let $S^u$ denote the schema identical to $S$ except that $root(S^u) = u$. The proof is based on showing that the pair $(u, u')$ is removed from $R$, if and only if there is a ground graph $g$ such that $g \preceq S_1^u$ but $g \not\preceq S_2^{u'}$.

"$\Leftarrow$" Let $g$ be a ground graph such that $g \preceq S_1^u$, and let $\trianglelefteq_1$ be the corresponding simulation respecting the constraints of $S_1^u$. Then the relation $R$ constructed by $subs$ can be used to obtain a simulation $\trianglelefteq_2$ from $g$ to $S_2^{u'}$ respecting the constraints of $S_2^{u'}$.

"$\Rightarrow$" Let $(u, u')$ be a pair removed from $R$ by $subs$ at the $K$-th iteration of the repeat-until loop. The construction of a ground graph $g$ such that $g \preceq S_1^u$ and $g \not\preceq S_2^{u'}$ is by induction on $K$, exploiting the fact that all inconsistent nodes in $S_1$ and $S_2$ have been removed before starting the construction of $R$.

"Complexity" The number of iterations is bound by $|S_1| \cdot |S_2|$, and at each iteration a polynomial number formulae of size polynomial in $m = |S_1| + |S_2|$ are checked for validity in $\mathcal{T}$. Hence $subs$ runs in time $O(m^{O(1)} \cdot m^{O(1)})$. The thesis follows from the fact that $t_{\mathcal{T}}(m^{O(1)})$ is assumed to be constant. □

The above result, together with Lemma 12, shows that adding conjunctions of local constraints to BDFS does not increase the complexity of subsumption.

**Example 15** Figure 4 shows two extensions to the schema in Figure 1, in which nesting of sections is considered[2]. Schema (a) models papers in which sections may contain subsections (i.e. with nesting of depth two). Schema (b), instead, models papers in which sections may be nested at arbitrary depth. It is possible to verify, that schema (b) subsumes schema (a), and that both subsume the schema in Figure 1.

Observe that, if we replace $\exists^{=1}\mathsf{edge}\,(\mathtt{Text})$ by $\exists^{\leq 1}\mathsf{edge}\,(\mathtt{Text})$ in $\mathcal{C}(u'_4)$ (thus

[2]Constraints equal to $\top$ are not shown in the figures.

modeling draft papers with possibly empty sections), the function $subs$ eliminates the pair $(u'_1, u''_1)$ from $R$ because of condition (2), and in turn the pair $(u'_0, u''_0)$ because of condition (1). Hence, in this case, schema (a) is not subsumed by schema (b). ■

In [BUN 97], it is shown that the notion of Least Upper Bound (LUB) of two schemas is useful for several purposes (e.g. for computing the "canonical fragments" of ground graphs). The LUB of two schemas $S_1$ and $S_2$, denoted by $S_1 \sqcap S_2$, is a schema satisfying the following property: the set of ground graphs that conform to $S_1 \sqcap S_2$ is the set of ground graphs that conform to both $S_1$ and $S_2$. We can show that the method mentioned in [BUN 97] for computing the LUB of two schemas can be easily extended in order to compute the LUB of two $\mathcal{L}_l$-schemas $S_1$ and $S_2$ in time $O(|S_1| \cdot |S_2|)$. This implies that we also have a method for checking if two $\mathcal{L}_l$-schemas are disjoint, based on the observation that $S_1$ and $S_2$ are disjoint if and only if $S_1 \sqcap S_2$ is inconsistent.

**Theorem 16** *Checking the disjointness of two $\mathcal{L}_l$-schemas $S_1$ and $S_2$ can be done in time polynomial in $|S_1| \cdot |S_2|$.*

### 3.2. *Non-Local Constraints*

We consider a simple constraint language $\mathcal{L}_n$ in which the constraints are not local, i.e. they can express conditions on edges that are not directly connected to the node labeled with the constraint. We show that consistency (and thus subsumption) of schemas with constraints becomes intractable.

The formulae of the constraint language $\mathcal{L}_n$ have the following syntax:

$$\gamma \ ::= \ \top \ \mid \ \exists \mathsf{edge}\,(p)\,\mathsf{to}\,(\gamma) \ \mid \ \forall \mathsf{edge}\,(p)\,\mathsf{to}\,(\gamma) \ \mid \ \gamma_1 \wedge \gamma_2$$

where the additional rules for the satisfaction of constraints of $\mathcal{L}_n$ in a node $u$ of a ground graph are:

$$u \models \exists \mathsf{edge}\,(p)\,\mathsf{to}\,(\gamma) \quad \text{iff} \quad \exists u \xrightarrow{a} v \in Edges(g).\ (\mathcal{T} \models p(a) \wedge v \models \gamma)$$
$$u \models \forall \mathsf{edge}\,(p)\,\mathsf{to}\,(\gamma) \quad \text{iff} \quad \forall u \xrightarrow{a} v \in Edges(g).\ (\mathcal{T} \models p(a) \supset v \models \gamma)$$

Observe that $\mathcal{L}_n$ is not local since the constraints imposed on one node may imply other constraints on adjacent nodes. By exploiting this property and the hardness results in [DON 92], we can show that consistency checking is coNP-hard.

**Theorem 17** *Checking the consistency of an $\mathcal{L}_n$-schema $S$ is coNP-hard in the size of $S$, even if $\mathcal{T}$ is empty, i.e. all edges of $S$ are labeled with* **true**.

*Proof.* The proof is an adaptation of the proof of coNP-hardness of satisfiability in the Description Logic $\mathcal{ALE}$ [DON 92], and is based on a reduction of the NP-complete ALL-POS ONE-IN-THREE 3SAT problem[3] to inconsistency of an $\mathcal{L}_n$-schema.

Given a set $F = \{f_1, \ldots, f_m\}$ of positive clauses with three literals over $\{x_1, \ldots, x_n\}$, the schema $S_F$ which encodes $F$ is shown in Figure 5, where

[3] ALL-POS ONE-IN-THREE 3SAT is the problem of deciding whether a 3CNF positive formula admits a truth assignment such that each clause has exactly one true literal.

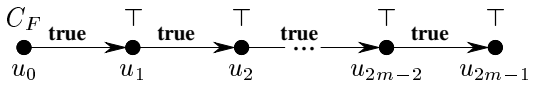



**Figure 5.** *Schema encoding* $F = \{f_1, \ldots, f_m\}$

$root(S_F) = u_0$ and $\mathcal{C}(u_0) = C_1^1 \wedge \cdots \wedge C_1^n$, with $C_i^j$, $i \in \{1, \ldots, m\}$, $j \in \{1, \ldots, n\}$, defined inductively by:

$$C_i^j = \begin{cases} \exists \mathsf{edge}\,(P)\,\mathsf{to}\,(C_{i+1}^j), & \text{if } x_j \in f_i, \\ \forall \mathsf{edge}\,(P)\,\mathsf{to}\,(C_{i+1}^j), & \text{if } x_j \notin f_i. \end{cases}$$

$$C_{m+i}^j = \begin{cases} \exists \mathsf{edge}\,(P)\,\mathsf{to}\,(C_{m+i+1}^j), & \text{if } x_j \in f_i, \\ \forall \mathsf{edge}\,(P)\,\mathsf{to}\,(C_{m+i+1}^j), & \text{if } x_j \notin f_i. \end{cases}$$

It is possible to show that $S_F$ is consistent if and only if there is no truth assignment such that each clause has exactly one true literal. □

Theorem 17 shows that consistency checking remains coNP-hard (and subsumption NP-hard), even if $\mathcal{T}$ can be used as an oracle for validity. The complexity of checking consistency in the presence of non-local constraints lies in the necessity to verify whether a ground graph may exist, whose topology is determined by the constraints. Since $\mathcal{T}$ cannot predict anything about the possible topologies of ground graphs, the validity checker of $\mathcal{T}$ cannot be used to "hide" a potentially exponential computation.

## 4. Graph Selection Queries

In general, query languages on semi-structured data are constituted by two components: one for selecting graphs, and another one for restructuring the selected graph to produce the actual answer [BUN 96, ABI 97c, FER 97, ABI 97b]. Here we introduce a basic form of queries, which we call graph selection queries (gs-queries), which deal only with the selection part. The language of gs-queries allows for expressing sophisticated fixpoint properties of graphs, which are not available in the above mentioned formalisms. Furthermore it has been carefully designed in order to keep several interesting reasoning tasks decidable, such as checking query satisfiability, checking containment or disjointness between queries, and comparing queries and schemas.

Observe that the unit retrieved by a gs-query is a graph, whereas there is no means to extract and further manipulate specific data from a retrieved graph (see for example [GOL 97]). Therefore our language cannot be considered a full-featured query language, such as UnQL [BUN 96], but should rather be regarded as providing basic building blocks for querying semi-structured data, to be exploited in query processing for improving evaluation performance (see Section 5).

In the rest of the paper, we deal only with $\mathcal{L}_l$-schemas, which we simply call schemas. The language for expressing graph selection queries has the following syn-

tax ($p$ denotes a formula of $\mathcal{T}$, $n$ a positive integer, and $X$ a node variable)

$$\begin{array}{lrcl} \text{node formulae:} & N & ::= & X \mid \exists^{\geq n}\mathsf{edge}\,(E) \mid \neg N \mid N_1 \wedge N_2 \mid \mu X.N \\ \text{edge formulae:} & E & ::= & p \mid \mathsf{to}(N) \mid \neg E \mid E_1 \wedge E_2 \end{array}$$

with the restriction that every free occurrence of $X$ in $\mu X.N$ is in the scope of an even number of negations[4]. We introduce the following abbreviations: $\alpha_1 \vee \alpha_2$ for $\neg(\neg\alpha_1 \wedge \neg\alpha_2)$, $\alpha_1 \supset \alpha_2$ for $\neg\alpha_1 \vee \alpha_2$, $\top$ for $\alpha \vee \neg\alpha$, $\bot$ for $\alpha \wedge \neg\alpha$, $\exists^{\leq n}\mathsf{edge}\,(E)$ for $\neg\exists^{\geq n+1}\mathsf{edge}\,(E)$, $\exists\mathsf{edge}\,(E)$ for $\exists^{\geq 1}\mathsf{edge}\,(E)$, and $\forall\mathsf{edge}\,(E)$ for $\neg\exists\mathsf{edge}\,(\neg E)$.

Let $g$ be a ground graph. A valuation $\rho$ on $g$ is a mapping from node variables to subsets of $Nodes(g)$. We denote by $\rho[X/\mathcal{N}]$ the valuation identical to $\rho$ except for $\rho[X/\mathcal{N}](X) = \mathcal{N}$. For each node $u \in Nodes(g)$, we define when $u$ satisfies a node formula $N$ under a valuation $\rho$, in notation $\rho, u \models N$, as follows:

$$\begin{array}{lll} \rho, u \models X & \text{iff} & u \in \rho(X) \\ \rho, u \models \exists^{\geq n}\mathsf{edge}\,(E) & \text{iff} & \#\{u \xrightarrow{a} v \in Edges(g) \mid \rho, u \xrightarrow{a} v \models E\} \geq n \\ \rho, u \models \neg N & \text{iff} & \rho, u \not\models N \\ \rho, u \models N_1 \wedge N_2 & \text{iff} & (\rho, u \models N_1) \wedge (\rho, u \models N_2) \\ \rho, u \models \mu X.N & \text{iff} & \forall \mathcal{N} \subseteq Nodes(g). \\ & & (\forall v \in Nodes(g).\rho[X/\mathcal{N}], v \models N \supset \rho[X/\mathcal{N}], v \models X) \\ & & \supset \rho[X/\mathcal{N}], u \models X \end{array}$$

where

$$\begin{array}{lll} \rho, u \xrightarrow{a} v \models p & \text{iff} & \mathcal{T} \models p(a) \\ \rho, u \xrightarrow{a} v \models \mathsf{to}(N) & \text{iff} & \rho, v \models N \\ \rho, u \xrightarrow{a} v \models \neg E & \text{iff} & \rho, u \xrightarrow{a} v \not\models E \\ \rho, u \xrightarrow{a} v \models E_1 \wedge E_2 & \text{iff} & (\rho, u \xrightarrow{a} v \models E_1) \wedge (\rho, u \xrightarrow{a} v \models E_2) \end{array}$$

Observe that for closed (wrt node variables) node formulae, satisfaction is independent of the valuation, and we denote it simply by $u \models N$.

Note that it is possible to specify node formulae which express the existence of paths that are characterized by regular expressions over edge formulae. In particular, we consider $\exists\mathsf{path}\,(P)\,\mathsf{to}\,(N)$, where $P$ is a regular expression over edge formulae, and $N$ is a node formula, as an abbreviation for the node formula defined inductively over the structure of $P$ as follows:

$$\begin{array}{rcl} \exists\mathsf{path}\,(E)\,\mathsf{to}\,(N) & \doteq & \exists\mathsf{edge}\,(E \wedge \mathsf{to}(N)) \\ \exists\mathsf{path}\,(P_1 \cup P_2)\,\mathsf{to}\,(N) & \doteq & \exists\mathsf{path}\,(P_1)\,\mathsf{to}\,(N) \vee \exists\mathsf{path}\,(P_2)\,\mathsf{to}\,(N) \\ \exists\mathsf{path}\,(P_1 \circ P_2)\,\mathsf{to}\,(N) & \doteq & \exists\mathsf{path}\,(P_1)\,\mathsf{to}\,(\exists\mathsf{path}\,(P_2)\,\mathsf{to}\,(N)) \\ \exists\mathsf{path}\,(P^*)\,\mathsf{to}\,(N) & \doteq & \mu X.(N \vee \exists\mathsf{path}\,(P)\,\mathsf{to}\,(X)) \end{array}$$

We use the abbreviation $\forall\mathsf{path}\,(P)\,\mathsf{to}\,(N)$ for $\neg\exists\mathsf{path}\,(P)\,\mathsf{to}\,(\neg N)$.

[4]This is the usual syntactic monotonicity constraint typical of fixpoint logics, that guarantees the monotonicity of the fixpoint operator.

**Definition 18** *Given a graph $G$ (either a ground graph or a schema) and a closed node formula $N$, we say that $G$ satisfies $N$, in notation $G \sqsubseteq N$, if for every ground graph $g$ conforming to $G$, $root(g) \models N$.*

It is easy to see that if $g$ is a ground graph and $N$ is a node formula, then $g \sqsubseteq N$ if and only if $root(g) \models N$.

**Definition 19** *A graph selection query (gs-query) $Q$ is a closed node formula. The evaluation of $Q$ over a database $DB$ returns the set $Q(DB)$ of all consistent graphs $G \in DB$ such that $G \sqsubseteq Q$.*

**Example 20** The gs-query

$$\forall \mathsf{edge}\,(\mathtt{Title} \supset \mathsf{to}(\mu X.\forall \mathsf{path}\,(\mathtt{Section} \cup (\mathtt{Text} \circ \mathtt{Section}))\,\mathsf{to}\,(X)))$$

selects all graphs representing papers with a finite depth of nesting of sections, and such that at each nesting level, the number of sections is finite. In particular, papers containing a loop between sections, i.e. sections that are followed either directly or indirectly by themselves are not selected by the query. ∎

**Definition 21** *A gs-query $Q$ is satisfiable if there exists a database $DB$ such that $Q(DB)$ is non-empty. Given two gs-queries $Q_1$ and $Q_2$, $Q_1$ is contained in $Q_2$ if for every database $DB$, $Q_1(DB) \subseteq Q_2(DB)$, and $Q_1$ is disjoint from $Q_2$ if for every database $DB$, $Q_(DB) \cap Q_2(DB) = \emptyset$.*

**Theorem 22** *Checking a gs-query for satisfiability and checking containment and disjointness between two gs-queries are EXPTIME-complete problems.*

*Proof.* Since containment between $Q_1$ and $Q_2$ can be verified by simply checking the formula $Q_1 \wedge \neg Q_2$ for unsatisfiability, and disjointness can be verified by checking the formula $Q_1 \wedge Q_2$ for unsatisfiability, we focus on satisfiability only.

It is easy to see that a query is satisfiable if and only if there is a ground graph $g$ such that $g \sqsubseteq Q$. In fact, if there is a database $DB$ such that $Q(DB)$ contains a schema $S$, then there exists also a database $DB' = DB \cup \{g\}$, where $g$ is a ground graph conforming to $S$. Hence $g$ is contained in $Q(DB')$.

To show the EXPTIME upper bound we exploit a polynomial reduction of satisfiability of a gs-query to satisfiability in $\mu\mathcal{ALC}\mathcal{Q}$. We check whether $Q$ is satisfiable, by encoding the problem into satisfiability of a $\mu\mathcal{ALC}\mathcal{Q}$ concept $C_Q$ in a $\mu\mathcal{ALC}\mathcal{Q}$ knowledge base $\mathcal{K}$.

In encoding $Q$ we exploit reification of edges, as used in [BUN 97]. Intuitively, we split each labeled edge $u \xrightarrow{a} v$ of a ground graph into two edges $u \xrightarrow{\mathbf{e}} e_{uv} \xrightarrow{\mathbf{e}} v$, by introducing an intermediate node $e_{uv}$ labeled by $a$ and making use of a special relation $\mathbf{e}$ (which is the only relation used in the encoding).

The knowledge base $\mathcal{K}$ is the union of $\mathcal{K}_0 \sqcap \mathcal{K}_\mathcal{T}$, where $\mathcal{K}_0$ and $\mathcal{K}_\mathcal{T}$ are as follows:

— $\mathcal{K}_0$, which enforces the general structure of graphs, has the form

$$\begin{aligned} \top &\sqsubseteq Node \sqcup Edge \\ Node &\sqsubseteq \neg Edge \end{aligned}$$

$$\begin{aligned} Node &\sqsubseteq \forall \mathbf{e}.Edge \\ Edge &\sqsubseteq (= 1\ \mathbf{e}.\top) \sqcap \forall \mathbf{e}.Node \end{aligned}$$

with $Node$ and $Edge$ new atomic concepts. Intuitively, this part of $\mathcal{K}$ partitions the interpretation domain into objects denoting nodes ($Node$) and objects denoting edges ($Edge$), and specifies the correct links for them.

— $\mathcal{K}_{\mathcal{T}}$, which reflects the properties of the theory $\mathcal{T}$, is formed by the axioms

| | |
|---|---|
| $Edge \equiv O_{a_1} \sqcup \cdots \sqcup O_{a_n}$ | where $a_1, \ldots, a_n$ are all the constants in $\mathcal{T}$, and |
| $O_{a_i} \sqsubseteq \neg O_{a_j}$ | for each pair of constants $a_i$, $a_j$ |

where $O_a$ is a new atomic concept associated with $a$. In addition, for each unary concept $p$ in $Q$ and for each constant $a$, $\Phi_{\mathcal{T}}$ contains an axiom

$$\begin{aligned} O_a \sqsubseteq A_p \quad &\text{if} \quad \mathcal{T} \models p(a) \\ O_a \sqsubseteq \neg A_p \quad &\text{if} \quad \mathcal{T} \models \neg p(a) \end{aligned}$$

where $A_p$ is a new atomic concept associated to $p$.

The concept $C_Q$ has the form $Node \sqcap \psi(Q)$, where $\psi(Q)$ is defined inductively as follows:

$$\begin{aligned} \psi(X) &= X \\ \psi(\exists^{\geq n}\mathsf{edge}\,(E)) &= (\geq n\ \mathbf{e}.\psi(E)) \\ \psi(\neg N) &= \neg\psi(N) \\ \psi(N_1 \wedge N_2) &= \psi(N_1) \sqcap \psi(N_2) \\ \psi(\mu X.N) &= \mu X.\psi(N) \end{aligned} \qquad \begin{aligned} \psi(p) &= A_p \\ \psi(\mathsf{to}(N)) &= \forall \mathbf{e}.\psi(N) \\ \psi(\neg E) &= \neg\psi(E) \\ \psi(E_1 \wedge E_2) &= \psi(E_1) \sqcap \psi(E_2) \end{aligned}$$

It can be shown that each ground graph satisfying $Q$ can be mapped to a model of $\mathcal{K}$ satisfying $C_Q$ and vice-versa, that each model of $\mathcal{K}$ satisfying $C_Q$ can be mapped to a ground graph satisfying $Q$.

To get the EXPTIME upper bound it suffices to observe that $\mathcal{K}$ does not depend on the query and hence has a constant size, while the size of $C_Q$ is linearly bound by the size of the query.

The EXPTIME hardness is a consequence of the EXPTIME hardness of satisfiability of a $\mu\mathcal{ALCQ}$ concept. It is easy to show that given a concept $C$ one can build a concept $C'$ (the "reified" version of $C$) such that every model of $C$ corresponds to a model of the knowledge base $\mathcal{K}$ satisfying $C'$, and vice-versa. $C'$ corresponds directly (by inverting the mapping $\psi$) to a gs-query. □

## 5. Evaluating Graph Selection Queries

We describe now a method for evaluating a gs-query over a graph (either a schema or a ground graph), and over a database.

### 5.1. *Evaluating Queries over Graphs*

Given a ground graph $g$ and a gs-query $Q$, we can verify in polynomial time in the size of $g$ (and in exponential time in the size of $g$ and $Q$) whether $root(g) \models Q$. This follows from the fact that $Q$ can be easily translated into a formula of first-order logic plus fixpoints [ABI 95], and that $g$ can be transformed into a first-order structure. Thus checking whether $g$ is part of the answer set of $Q$ can be reduced to model checking in first-order logic plus fixpoints, which has polynomial data complexity. Therefore, the method verifies in polynomial time in the size of $g$ whether $g \sqsubseteq Q$.

We now turn our attention to checking whether a schema satisfies a gs-query. To this purpose, we exploit the fact that each schema $S$ can be transformed into a gs-query $Q_S$ that is equivalent to $S$, in the sense that the ground graphs conforming to $S$ are exactly those that satisfy $Q_S$. We call $Q_S$ the characteristic query of the schema $S$.

To define $Q_S$, we first consider the set of mutual recursive equations:

$$\begin{aligned} X_{u_1} &= \mathcal{C}(u_1) \wedge \forall \mathsf{edge}\,(\bigvee\nolimits_{u_1 \xrightarrow{p} v}(p \wedge \mathsf{to}(X_v))) \\ &\cdots \\ X_{u_h} &= \mathcal{C}(u_h) \wedge \forall \mathsf{edge}\,(\bigvee\nolimits_{u_h \xrightarrow{p} v}(p \wedge \mathsf{to}(X_v))) \end{aligned}$$

one for each node $u_i$ in $Nodes(S) = \{u_1, \ldots, u_h\}$.

Then we eliminate, one at the time, each of the above equations, except the one for $X_{root(S)}$ as follows: eliminate the equation $X_{u_j} = N_j$ and substitute each occurrence of $X_{u_j}$ in the remaining equations with $\nu X_{u_j}.N_j$. Let $X_{root(S)} = N_S$ be the resulting equation. The characteristic query $Q_S$ of $S$ is $\nu X_{root(S)}.N_S$ [5].

**Theorem 23** *If $S$ is a schema and $Q_S$ is its characteristic query, then, for every ground graph $g$, $g$ conforms to $S$ if and only if $g$ satisfies $Q_S$.*

*Proof.* "$\Leftarrow$" Let $\rho_\nu$ be the valuation assigning the greatest extension to each $X_{u_i}$ while satisfying the equations

$$\begin{aligned} X_{u_1} &= \mathcal{C}(u_1) \wedge \forall \mathsf{edge}\,(\bigvee\nolimits_{u_1 \xrightarrow{p} v}(p \wedge \mathsf{to}(X_v))) \\ &\cdots \\ X_{u_h} &= \mathcal{C}(u_h) \wedge \forall \mathsf{edge}\,(\bigvee\nolimits_{u_h \xrightarrow{p} v}(p \wedge \mathsf{to}(X_v))) \end{aligned}$$

We define

$$\mathcal{R} = \{(u, u') \in Nodes(g) \times Nodes(S) \mid \rho_\nu, u \models X_{u'}\}$$

We show that $\mathcal{R}$ is a simulation from $g$ to $S$, i.e. for each $u$, $u'$, if $(u, u') \in \mathcal{R}$ then (1) $u \models \mathcal{C}(u')$, and (2) for each edge $u \xrightarrow{a} v \in Edges(g)$, there exists an edge $u' \xrightarrow{p} v' \in Edges(S)$ such that $\mathcal{T} \models p(a)$ and $(v, v') \in \mathcal{R}$. Indeed, $\rho_\nu, u \models X_{u'}$ implies $\rho_\nu, u \models \mathcal{C}(u') \wedge \forall \mathsf{edge}\,(\bigvee_{u' \xrightarrow{p} v'}(p \wedge \mathsf{to}(X_{v'})))$. Hence

[5] This construction is analogous to the one used in Process Algebra for defining a characteristic formula of a process [STE 94], i.e. a formula which is satisfied by exactly all processes that are equivalent to the process under bisimulation. Similarly, $Q_S$ characterizes exactly all databases that conform to $S$.

(1) $\rho_\nu, u \models \mathcal{C}(u')$

(2) for each $u \xrightarrow{a} v \in Edges(g)$, there exists an edge $u' \xrightarrow{p} v' \in Edges(S)$ such that $\mathcal{T} \models p(a)$ and $\rho_\nu, v \models X_{v'}$ i.e. $(v, v') \in \mathcal{R}$.

It remains to show that $(root(g), root(S)) \in \mathcal{R}$. The construction applied to build the characteristic formula $Q_S$[6] guarantees that $u \models Q_S$ iff $\rho_\nu, u \models X_{root(S)}$. Hence, since $g \sqsubseteq Q_S$, i.e. $root(g) \models Q_S$, we have that $\rho_\nu, root(g) \models X_{root(S)}$.

"$\Rightarrow$" Let $\preceq \in Nodes(g) \times Nodes(S)$ be the greatest simulation relation such that $u \preceq u'$ implies that (1) $u$ satisfies $\mathcal{C}(u')$, and (2) for each edge $u \xrightarrow{a} v \in Edges(g)$, there exists an edge $u' \xrightarrow{p} v'$ such that $\mathcal{T} \models p(a)$ and $v \preceq v'$. Let $\rho_c$ be a valuation such that $\rho_c(X_{u'}) = \{u \mid u \preceq u'\}$. It is easy to verify that

$$\rho_c, u \models X_{u'} \text{ implies } \rho_c, u \models \mathcal{C}(u') \wedge \forall \mathsf{edge}\,(\bigvee_{u' \xrightarrow{p} v'} p \wedge \mathsf{to}(X_{v'}))$$

Now the valuation $\rho_\nu$ defined above is also the valuation assigning the greatest extension to each $X_{u_i}$ that satisfies these implications[7]. This implies that $\rho_c(X_{u'}) \subseteq \rho_\nu(X_{u'})$ and thus, since $u \models Q_S$ iff $\rho_\nu, u \models X_{root(S)}$, we get that $root(g) \models Q_S$, i.e. $g \sqsubseteq Q_S$. □

**Theorem 24** *If $S$ is a schema and $Q$ is a gs-query, then checking whether $S \sqsubseteq Q$ is EXPTIME-hard and decidable in time $O(2^{p(|Q_S|+|Q|)})$.*

*Proof.* It suffices to verify the unsatisfiability of the gs-query $Q_S \wedge \neg Q$. □

Observe that $|Q_S|$ may be exponential with respect to $|S|$. Therefore checking whether a schema satisfies a gs-query can be done in worst case deterministic double exponential time with respect to the size of the schema (and deterministic exponential time with respect to the size of the gs-query).

### 5.2. *Evaluating Queries over a Database*

We sketch now how to exploit schemas and subsumption and disjointness relations between graphs in order to evaluate gs-queries over databases. We remind the reader that evaluating a gs-query over a database means selecting all graphs in the database that satisfy the query. Without loss of generality we assume that the database does not contain equivalent graphs.

When the database is constituted by a flat set of ground graphs, evaluating a gs-query $Q$ amounts simply to check for each ground graph separately whether it satisfies $Q$. On the contrary, when the database $DB$ is constituted by ground graphs and schemas, and when for each pair of such graphs one knows whether one is subsumed by the other or whether they are disjoint, then the evaluation of $Q$ on $DB$ can take advantage of this information by proceeding as follows.

[6] Note that this construction is exactly the one used e.g. in [BAK 80] to eliminate mutual fixpoints.

[7] This is a direct consequence of Tarski-Knaster's fixpoint theorem [TAR 55].

Let $\mathcal{G}$ be equal to $DB$. While $\mathcal{G}$ is not empty, repeatedly select a graph $G$ from $\mathcal{G}$ such that no graph in $\mathcal{G}$ subsumes $G$, and do the following:

1. If $G$ is equivalent to $Q$, then let $Q(DB)$ be all graphs in $\mathcal{G}$ subsumed by $G$ and stop.
2. If $G$ satisfies $Q$, then move all graphs that are subsumed by $G$ from $\mathcal{G}$ to $Q(DB)$, and continue.
3. If $Q$ is contained in $Q_G$, then remove from $\mathcal{G}$ the graph $G$ and all graphs that are disjoint from $G$ and continue.
4. If $Q_G$ is disjoint from $Q$, then remove from $\mathcal{G}$ all graphs that are subsumed by $G$, and continue.
5. Otherwise, remove $G$ from $\mathcal{G}$ and continue.

Observe that in this way schemas act as semantic indexes on graphs in the database and help in improving performance of query evaluation with respect to the brute approach of evaluating graphs one by one, similarly to DataGuides proposed in [GOL 97]. Therefore, the addition of schemas to a database constituted by ground graphs only allows for a more effective query evaluation process. Obviously, because of the high complexity of comparing schemas and queries, one has to carefully choose the size of schemas to be small (e.g. logarithmic) with respect to the size of the conforming ground graphs in the database.

## 6. Conclusions

In this paper we have extended the BDFS framework by adding constraints to the nodes of the schema, and have devised polynomial time algorithms for checking conformance and subsumption in the case of local constraints. We have also shown that even simple forms of non-local constraints lead to intractability. In addition we have introduced a basic form of queries over schemas with constraints, for which query satisfiability and query containment in our setting are EXPTIME complete. We have discussed how to exploit the semantic information in the schemas for query evaluation.

We are currently working on various aspects. First, we are working to extend the polynomial time algorithm for schema subsumption to other forms of constraints, including cardinality constraints. Second, we are investigating the possibility of avoiding the worst case exponential blowup in the encoding of a schema into a query. Finally, we are considering a more general query language that uses graph selection queries as building blocks, and we are devising techniques for query containment in such a language, along the line of [CAL 98].

***Acknowledgments***

This work was partly supported by MURST, ESPRIT LTR Project No. 22469 DWQ (Foundations of Data Warehouse Quality), and the Italian Space Agency (ASI) under project “Integrazione ed Accesso a Basi di Dati Eterogenee”.

## Bibliographie

[ABI 95] ABITEBOUL S., HULL R. and VIANU V., *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[ABI 97a] ABITEBOUL S., « Querying Semi-Structured Data ». In *Proc. of ICDT-97*, p. 1–18, 1997.

[ABI 97b] ABITEBOUL S., CLUET S., CHRISTOPHIDES V., MILO T. and GUIDO MOERKOTTE J. S., « Querying Documents in Object Databases ». *Int. J. on Digital Libraries*, vol. 1, n. 1, p. 5–19, 1997.

[ABI 97c] ABITEBOUL S., QUASS D., MCHUGH J., WIDOM J. and WIENER J. L., « The Lorel Query Language for Semistructured Data ». *Int. J. on Digital Libraries*, vol. 1, n. 1, p. 68–88, 1997.

[BAK 80] DE BAKKER J., *Mathematical Theory of Program Correctness*. Prentice-Hall, 1980.

[BUN 96] BUNEMAN P., DAVIDSON S., HILLEBRAND G. and SUCIU D., « A Query Language and Optimization Technique for Unstructured Data ». In *Proc. of ACM SIGMOD*, p. 505–516, 1996.

[BUN 97] BUNEMAN P., DAVIDSON S., FERNANDEZ M. and SUCIU D., « Adding Structure to Unstructured Data ». In *Proc. of ICDT-97*, p. 336–350, 1997.

[CAL 98] CALVANESE D., DE GIACOMO G. and LENZERINI M., « On the Decidability of Query Containment under Constraints ». In *Proc. of PODS-98*, p. 149–158, 1998.

[DEG 97] DE GIACOMO G. and LENZERINI M., « A Uniform Framework for Concept Definitions in Description Logics ». *J. of Artificial Intelligence Research*, vol. 6, p. 87–110, 1997.

[DON 92] DONINI F. M., HOLLUNDER B., LENZERINI M., SPACCAMELA A. M., NARDI D. and NUTT W., « The Complexity of Existential Quantification in Concept Languages ». *Artif. Intell.*, vol. 2–3, p. 309–327, 1992.

[EME 96] EMERSON E. A., Automated Temporal Reasoning about Reactive Systems. In MOLLER F. and BIRTWISTLE G., Eds., *Logics for Concurrency: Structure versus Automata*, vol. 1043 of *LNCS*, p. 41–101. Springer-Verlag, 1996.

[FER 97] FERNANDEZ M. F., FLORESCU D., LEVY A. Y. and SUCIU D., « A Query Language for a Web-Site Management System ». *SIGMOD Record*, vol. 26, n. 3, p. 4–11, 1997.

[FER 98] FERNANDEZ M. F., FLORESCU D., KANG J., LEVY A. Y. and SUCIU D., « Catching the Boat with Strudel: Experiences with a Web-Site Management System ». In *Proc. of ACM SIGMOD*, p. 414–425, 1998.

[FLO 98] FLORESCU D., LEVY A. and MENDELZON A., « Database Techniques for the World-Wide Web: A Survey ». *SIGMOD Record*, vol. 27, n. 3, p. 59–74, 1998.

[GOL 97] GOLDMAN R. and WIDOM J., « DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases ». In *Proc. of VLDB-97*, p. 436–445, 1997.

[HOE 95] VAN DER HOEK W. and DE RIJKE M., « Counting Objects ». *J. of Log. and Comp.*, vol. 5, n. 3, p. 325–345, 1995.

[KOZ 83] KOZEN D., « Results on the Propositional $\mu$-Calculus ». *Theor. Comp. Sci.*, vol. 27, p. 333–354, 1983.

[MEN 97] MENDELZON A., MIHAILA G. A. and MILO T., « Querying the World Wide Web ». *Int. J. on Digital Libraries*, vol. 1, n. 1, p. 54–67, 1997.

[QUA 95] QUASS D., RAJARAMAN A., SAGIV I., ULLMAN J. and WIDOM J., « Querying Semistructured Heterogeneous Information ». In *Proc. of DOOD-95*, p. 319–344. Springer-Verlag, 1995.

[STE 94] STEFFEN B. and INGÓLFSDÓTTIR A., « Characteristic Formulae for Processes with Divergence ». *Information and Computation*, vol. 110, p. 149–163, 1994.

[STI 96] STIRLING C., Modal and Temporal Logics for Processes. In MOLLER F. and BIRTWISTLE G., Eds., *Logics for Concurrency: Structure versus Automata*, vol. 1043 of *LNCS*, p. 149–237. Springer-Verlag, 1996.

[STR 89] STREETT R. E. and EMERSON E. A., « An Automata Theoretic Decision Procedure for the Propositional $\mu$-Calculus ». *Information and Computation*, vol. 81, p. 249–264, 1989.

[TAR 55] TARSKI A., « A Lattice-Theoretical Fixpoint Theorem and Its Applications ». *Pacific Journal of Mathematics*, vol. 5, p. 285–309, 1955.