# What can Knowledge Representation do for 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

**Abstract**

The problem of modeling semi-structured data is important in many application areas such as multimedia data management, biological databases, digital libraries, and data integration. Graph schemas (Buneman *et al.* 1997) have been proposed recently as a simple and elegant formalism for representing semistructured data. In this model, schemas are represented as graphs whose edges are labeled with unary formulae of a theory, and the notions of conformance of a database to a schema and of subsumption between two schemas are defined in terms of a simulation relation. Several authors have stressed the need of extending graph schemas with various types of constraints, such as edge existence and constraints on the number of outgoing edges. In this paper we analyze the appropriateness of various knowledge representation formalisms for representing and reasoning about graph schemas extended with constraints. We argue that neither First Order Logic, nor Logic Programming nor Frame-based languages are satisfactory for this purpose, and present a solution based on very expressive Description Logics. We provide techniques and complexity analysis for the problem of deciding schema subsumption and conformance in various interesting cases, that differ by the expressive power in the specification of constraints.

## 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 biological databases, digital libraries, data integration, and access to web databases (Abiteboul 1997; Buneman *et al.* 1997; Christophides *et al.* 1994; Mendelzon, Mihaila, & Milo 1997; Quass *et al.* 1995). Consider, for example, the set of home pages designed by the faculties for a University web site. Since different home pages may vary considerably one from another, it is extremely hard to describe their structure in a rigid form such as the one imposed, say, by relational databases. Indeed, we need structuring mechanisms that are much more flexible than traditional data models.

Following (Abiteboul 1997), we define semi-structured data as data that is neither raw, nor strictly typed as in conventional database systems. BDFS (Basic Data model For Semi-structured data) (Buneman *et al.* 1997) is a formal and elegant data model, based on graphs with labeled edges, where information on both the values and the schema for the data are kept. The labels of edges in the schemas are formulae of a certain theory $\mathcal{T}$, and the notion of a database $D$ being coherent to a schema $\mathcal{S}$ is given in terms of a special relation, called simulation, between the graph representing the database and the graph representing the schema. Roughly speaking, a simulation is a correspondence between the edges of $D$ and those of $\mathcal{S}$ such that, whenever there is an edge labeled $a$ in $D$, there is a corresponding edge in $\mathcal{S}$ labeled with a formula satisfied by $a$ (but not necessarily vice-versa). 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.

In (Buneman *et al.* 1997), the authors point out that, for several tasks related to data management, it is important to be able to reason about schemas, in particular to check subsumption between two schemas, which is the task of deciding whether every database conforming to one schema always conforms to another schema. They also present algorithms for, and analyze the complexity of checking subsumption in BDFS.

Several papers indicate that in many applications there is the need to extend the BDFS model with different types of constraints. Indeed, in (Buneman *et al.* 1997) all the properties of the schema are expressed in terms of the structure of the graph, and therefore, there is no possibility of specifying additional conditions, such as existence of edges or bounds on the number of edges emanating from a node, or imposing that a certain subgraph is well-founded.

Our intuition suggests that Knowledge Representation (KR) techniques should be very useful for the above purpose. After all, the problem deals with *representing* constraints, and *reasoning* about schemas with constraints. The basic goal of the work reported in this paper was to verify this intuition, and we present here the following results of our investigation:

- We analyze the appropriateness of various KR formalisms for representing and reasoning about graph schemas extended with constraints, and demonstrate that neither First Order Logic, nor Logic Programming nor Frame-based languages are satisfactory for this purpose.
- We show that very expressive Description Logics (DLs), such as the ones studied in (De Giacomo & Lenzerini 1996; Calvanese 1996; De Giacomo & Lenzerini 1997), are the right tools for modeling and reasoning about semi-structured data with constraints. In particular, we propose to express constraints in terms of DLs formulae associated to nodes of the schema. A formula on a node $u$ imposes a condition that, for every database $D$ conforming to $\mathcal{S}$, must be satisfied by every node of $D$ simulating $u$.

Copyright ©1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

- We consider languages for specifying constraints with different expressive power, and present several results on the corresponding reasoning problems. We show that adding various types of local constraints (i.e. constraints that impose conditions only on the edges emanating from a node) does not increase the complexity of reasoning. On the other hand, we present an intractability result for the case of non-local constraints. Finally, we study the case where the constraints are expressed in a very powerful DL, namely, $\mu\mathcal{ALCQ}$ (De Giacomo & Lenzerini 1997), that allows for imposing complex conditions on the schema, such as well-foundedness of subgraphs. We present a technique for checking subsumption in this case, showing that the problem is decidable in double exponential time.

Our presentation starts with a brief description of both BDFS, and the description logic $\mu\mathcal{ALCQ}$.

## Preliminaries

In this section, we describe the basic characteristics of the BDFS model for semi-structured data and the description logic $\mu\mathcal{ALCQ}$.

### The BDFS Data Model

The data model BDFS, which is the basis of our investigation, is an edge-labeled graph model of semi-structured data, where labels are unary formulae of a first order language $\mathcal{L}_\mathcal{T}$. The language $\mathcal{L}_\mathcal{T}$ is constituted by a set of predicates, including the equality predicate "=", and one constant for every element of a universe $\mathcal{U}$.

A schema in BDFS always refers to a complete and decidable theory $\mathcal{T}$ on $\mathcal{U}$. In other words, $\mathcal{T}$ is the set of first order formulae which are true (or valid) for the elements of $\mathcal{U}$, and it is decidable to check whether a formula $p$ in $\mathcal{L}_\mathcal{T}$ is valid in $\mathcal{T}$ (in notation, $\mathcal{T} \models p$).

**Definition 1** *A* BDFS $\mathcal{T}$-schema *is a rooted connected graph whose edges are labeled with unary formulae of $\mathcal{L}_\mathcal{T}$. A $\mathcal{T}$-*database *is a rooted connected graph whose edges are labeled with constants of $\mathcal{T}$.*

For any rooted 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 $a$ with $u \xrightarrow{a} v$.

In order to establish if a database is coherent with a schema, or if a schema is more general than another schema, the notions of conformance and subsumption are defined as follows.

**Definition 2** *A $\mathcal{T}$-database $D$* conforms to *a* BDFS $\mathcal{T}$-*schema $\mathcal{S}$, in notation $D \preceq \mathcal{S}$, if there exists a* simulation *from $D$ to $\mathcal{S}$, i.e. a binary relation $\trianglelefteq$ from the nodes of $D$ to those of $\mathcal{S}$ satisfying: (1) $root(D) \trianglelefteq root(\mathcal{S})$, (2) $u \trianglelefteq u'$ implies that for each edge $u \xrightarrow{a} v$ in $D$, there exists an edge $u' \xrightarrow{p} v'$ in $\mathcal{S}$ such that $\mathcal{T} \models p(a)$, and $v \trianglelefteq v'$.*

**Definition 3** *If $\mathcal{S}$ and $\mathcal{S}'$ are two* BDFS $\mathcal{T}$-*schemas, then $\mathcal{S}'$* subsumes *$\mathcal{S}$, in notation $\mathcal{S} \sqsubseteq \mathcal{S}'$, if for every $\mathcal{T}$-database $D$, $D \preceq \mathcal{S}$ implies $D \preceq \mathcal{S}'$. $\mathcal{S}$* is equivalent to *$\mathcal{S}'$ if $\mathcal{S} \sqsubseteq \mathcal{S}'$ and $\mathcal{S}' \sqsubseteq \mathcal{S}$.*

In (Buneman *et al.* 1997), an algorithm is presented for checking subsumption (and also conformance, being a $\mathcal{T}$-database a special case of $\mathcal{T}$-schema). The algorithm essentially looks for the greatest simulation between the nodes of the two schemas, and works in 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 general it is meaningful not to consider $\mathcal{T}$ to be part of the input of the problem (Buneman *et al.* 1997). Therefore, whenever $t_\mathcal{T}(m)$ may be assumed to be independent of $m$, $t_\mathcal{T}(m)$ can be replaced by a constant (e.g. when $m$ is polynomial in the size $|\mathcal{S}|$ of a $\mathcal{T}$-schema $\mathcal{S}$, which is considerably smaller than $|\mathcal{T}|$).

If not specified otherwise, we also make the assumption that the theory $\mathcal{T}$ is not part of the input to the reasoning problems addressed in the paper (namely, consistency and subsumption).

### The Description Logic $\mu\mathcal{ALCQ}$

Description logics (DLs) 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 DL $\mu\mathcal{ALCQ}$ studied in (De Giacomo & Lenzerini 1997), where a correspondence was shown with a well-known logic of programs, called *modal mu-calculus* (Kozen 1983; Streett & Emerson 1989), that has been recently investigated for expressing temporal properties of reactive and parallel processes (Stirling 1996; Emerson 1996). $\mu\mathcal{ALCQ}$ can be viewed as a well-behaved fragment of first-order logic with fixpoints (Park 1970; Abiteboul, Hull, & Vianu 1995). 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:

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

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

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

Observe that $C^{\mathcal{I}}_{\rho[X/\mathcal{E}]}$ can be seen as an operator from subsets $\mathcal{E}$ of $\Delta^{\mathcal{I}}$ to subsets of $\Delta^{\mathcal{I}}$, and that, by the syntactic restriction enforced on variables, such an operator is guaranteed to be monotonic wrt set inclusion. 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 $\mu\mathcal{ALCQ}$ *knowledge base* is a finite set of *axioms* $C_1 \sqsubseteq C_2$ where $C_1$ and $C_2$ are closed concepts of $\mu\mathcal{ALCQ}$. An interpretation $\mathcal{I}$ *satisfies an axiom* $C_1 \sqsubseteq C_2$, if $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$. $\mathcal{I}$ is a *model* of a knowledge base $\Gamma$, if $\mathcal{I}$ satisfies all axioms in $\Gamma$. A closed concept $C$ is *satisfiable* in a knowledge base $\Gamma$ if there exists a model $\mathcal{I}$ of $\Gamma$ such that $C^{\mathcal{I}} \neq \emptyset$.

**Theorem 4** *(De Giacomo & Lenzerini 1997) Satisfiability of closed $\mu\mathcal{ALCQ}$ concepts in $\mu\mathcal{ALCQ}$ knowledge bases is an EXPTIME-complete problem.*

## Which KR Formalism for Semi-Structured Data Modeling?

In this section we discuss how the technology of KR can contribute to the problem of modeling semi-structured data with constraints.

We start our investigation by verifying whether First Order Logic (FOL) is suited for representing BDFS schemas. We observe that, in principle, FOL would be an interesting tool, because it would allow expressing very complex constraints on the schema. However, we need to check if FOL is able to model the notions of BDFS schemas, databases, and simulation.

A reasonable approach to expressing BDFS schemas in FOL is based on the following observations:

- If $u_1, \ldots, u_N$ are the nodes of the schema, we make use of predicate symbols, $n_1, \ldots, n_N$ and $edge$, where $n_i(x)$ means that $x$ is a node (the $n_i$ corresponding to the root is called the root predicate), and $edge(x, y, z)$ means that there is an edge from $x$ to $y$ labeled with $z$. The fact that the label $z$ satisfies the formula $F$ of $\mathcal{T}$ is represented by $F(z)$.
- We represent the schema $\mathcal{S}$ by means of a set $FOL(\mathcal{S})$ of suitable formulae. For example, the fact that a node $u_i$ has two outgoing edges to $u_j$ and $u_k$ labeled with $P_1$ and $P_2$, is represented by the formula

$$\forall x\,(n_i(x) \Leftrightarrow (\forall y \forall z\; edge(x,y,z) \supset ((P_1(z) \wedge n_j(y)) \vee (P_2(z) \wedge n_k(y)))))$$

- An interpretation of $FOL(\mathcal{S})$ is obtained by choosing a so-called pre-interpretation that assigns a truth value to every ground formula of the form $edge(x, y, z)$, and then extending it to an interpretation by computing the extension of the predicates $n_1, \ldots, n_N$ on the basis of the formulae in $FOL(\mathcal{S})$.
- Given a logical model $M$ of $FOL(\mathcal{S})$, if we traverse the relation $edge$ in $M$ starting from one object satisfying the root predicate, we should obtain a structure that corresponds to a database conforming to $\mathcal{S}$, and, on the contrary, every database conforming to $\mathcal{S}$ should correspond to a logical model of $FOL(\mathcal{S})$ in which the extension of the root predicate is not empty.

Consider, for example, the BDFS schema $\mathcal{S}_1$ with two nodes $u_1$ and $u_2$, and one edge from $u_1$ to $u_2$ labeled with $P$. The corresponding set $FOL(\mathcal{S}_1)$ of formulae in FOL is:

$$\{\ \forall x\,(n_1(x) \Leftrightarrow (\forall y \forall z\; edge(x,y,z) \supset (P(z) \wedge n_2(y)))),\\ \forall x\,(n_2(x) \Leftrightarrow (\neg \exists y \exists z\; edge(x,y,z))\ \}$$

Consider the database $D_1$ with two nodes $d$ and $e$ and an edge from $d$ to $e$ labeled with $t$ such that $P(t)$ is valid in $\mathcal{T}$. It is easy to see that $D_1$ conforms to $\mathcal{S}_1$, and that it corresponds to the pre-interpretation where $edge(d, e, t)$ is true, which is extended to a model of $FOL(\mathcal{S}_1)$ such that $d \in n_1$ and $e \in n_2$. Observe that the empty database with just one node and no edges also conforms to $\mathcal{S}_1$ and corresponds to the pre-interpretation where $edge(x, y, z)$ is always false. This is correctly captured by $FOL(\mathcal{S}_1)$.

On the other hand, consider a schema $\mathcal{S}_2$ with one node $u$ and one edge from $u$ to $u$ labeled with $P$. The set $FOL(S_2)$ is simply

$$\{\ \forall x\,(n(x) \Leftrightarrow (\forall y \forall z\; edge(x,y,z) \supset (P(z) \wedge n(y))))\ \}$$

and it is easy to see that there is at least one model of $FOL(\mathcal{S}_2)$ which does not correspond to any database conforming to $\mathcal{S}_2$. Indeed, consider a pre-interpretation $I$ assigning true to $edge(d, d, t)$, with $P(t)$ valid in $\mathcal{T}$. Clearly, such a pre-interpretation can be extended to a model of $FOL(\mathcal{S}_2)$ simply by letting the extension of $n$ be empty. However, since $d$ is not in the extension of $n$, the resulting model does not reflect the fact that the database corresponding to $I$ conforms to $\mathcal{S}_2$.

The above example shows a general problem that FOL has in representing BDFS schemas. The first order semantics is intrinsically too liberal for capturing the notion of simulation. Indeed, in order to reflect such a notion, we need a type of semantics that forces an object $o$ to be in the extension of a predicate $n_i$ whenever there is no evidence that it cannot satisfy such predicate. We observe that the greatest fixpoint semantics (Baader 1996) satisfies exactly this property. Thus, FOL extended with fixpoints would be suitable, but FOL itself is not.

For the same reason, one can verify that neither Logic Programming languages under the completion semantics (Lloyd 1987), nor Frame-based languages are suited for modeling BDFS schemas. In particular, the difficulties arise when BDFS schemas with cycles are taken into account (see the schema $\mathcal{S}_2$ in the example above). On the contrary, schemas without cycles may be modeled correctly, for example by using KR systems such as CLASSIC (Borgida & Patel-Schneider 1994). Note, however, that the assumption of acyclicity of semi-structured data schemas is too restrictive in practice.

The above observations tell us that, in order to use the KR technology for modeling and reasoning about semi-structured data with constraints, we must resort to KR formalisms that are able:

- to model graphs with no limitations on cycles;
- to interpret such graphs by making use of the greatest fixpoint semantics;
- to express complex constraints on the graphs;
- to provide reasoning procedures for computing the subsumption relation between schemas.

Below we show that $\mu\mathcal{ALCQ}$ has exactly the above characteristics. However, as a first step, we need to formally define graph schemas with constraints and the associated reasoning tasks.

## Schemas with Constraints

We address the problem of extending the BDFS data model in order to express constraints on the graph representing a schema. We conceive a constraint for a BDFS schema $\mathcal{S}$ 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 database $D$ conforming to $\mathcal{S}$, must be satisfied by every node of $D$ 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 5** *A $\mathcal{T}$-*schema with $\mathcal{L}$-constraints *is a pair $\mathcal{S} = (\mathcal{G}, \mathcal{C})$, where $\mathcal{G}$ is a* BDFS *$\mathcal{T}$-schema, and $\mathcal{C}$ is a total function from the nodes of $\mathcal{G}$ to formulae of a constraint language $\mathcal{L}$.*

**Definition 6** *A $\mathcal{T}$-database $D$* conforms to *a $\mathcal{T}$-schema with $\mathcal{L}$-constraints $\mathcal{S} = (\mathcal{G}, \mathcal{C})$, in notation $D \preceq \mathcal{S}$, if there exists a* constraint-consistent simulation*, i.e. a binary relation $\trianglelefteq$ from the nodes of $D$ to those of $\mathcal{G}$ satisfying: (1) $root(D) \trianglelefteq root(\mathcal{G})$, (2) $u \trianglelefteq u'$ implies that (2.1) $u$ satisfies $\mathcal{C}(u')$, and (2.2) for each edge $u \xrightarrow{a} v$ in $D$, there exists an edge $u' \xrightarrow{q} v'$ in $\mathcal{S}$ such that $\mathcal{T} \models q(a)$, and $v \trianglelefteq v'$.*

Since constraints may contradict each other, or may even be incompatible with the structure of the graph, the notion of consistency becomes relevant.

**Definition 7** *For a $\mathcal{T}$-*schema with $\mathcal{L}$-constraints *$\mathcal{S} = (\mathcal{G}, \mathcal{C})$, a node $u \in Nodes(\mathcal{G})$ is* consistent *if there is at least one $\mathcal{T}$-database which conforms to $(\mathcal{G}', \mathcal{C})$, where $\mathcal{G}'$ is equal to $\mathcal{G}$ except that $root(\mathcal{G}') = u$. $\mathcal{S}$ is* consistent*, if $root(\mathcal{G})$ is consistent.*

The notion of subsumption remains unchanged.

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

## Local Constraints

We first 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. $\mathcal{L}_l$ is inspired by DLs with number restrictions and its formulae have the following syntax ($\gamma$, $\gamma_1$ and $\gamma_2$ denote constraints, and $p$ denotes a formula of $\mathcal{T}$):

$$\gamma \ ::= \ \top \ \mid \ \exists p \ \mid \ \neg\exists p \ \mid \ \exists^{\leq 1} p \ \mid \ \gamma_1 \wedge \gamma_2$$

Intuitively, a constraint of the form $\exists p$ 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 $\exists^{\leq 1} p$, called *functionality-constraint*, imposes that $u$ has at most one such outgoing edge. More precisely, let $\mathcal{S} = (\mathcal{G}, \mathcal{C})$ be a $\mathcal{T}$-schema with $\mathcal{L}_l$-constraints. and $D$ a $\mathcal{T}$-database. Then a node $u$ of $D$ *satisfies* a constraint $\gamma$, in notation $u \models_c \gamma$, if the following conditions are satisfied:

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

Note that we can view a $\mathcal{T}$-database $D$ as a $\mathcal{T}$-schema $(D, \mathcal{C})$ with constraints, where $\mathcal{C}(u) = \top$ for every node $u$ of $D$ (such a schema is always consistent).

Checking the consistency of a schema amounts to visiting the graph and removing nodes that violate constraints, which can be detected by a local check. An algorithm for subsumption is obtained essentially by incorporating local checks for constraint violations into the algorithm of (Buneman *et al.* 1997).

**Theorem 8** *Consistency and subsumption of $\mathcal{T}$-schemas with $\mathcal{L}_l$-constraints, can be checked in polynomial time in the size of the schemas.*

## Non-Local Constraints

Next we consider languages 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 even in a simple non-local constraint language, namely $\mathcal{L}_{\mathcal{ALE}}$ inspired by the DL $\mathcal{ALE}$ (Donini *et al.* 1992), consistency and subsumption of $\mathcal{T}$-schemas become intractable.

The formulae of $\mathcal{L}_{\mathcal{ALE}}$ have the following syntax:

$$\gamma \ ::= \ \top \ \mid \ \exists p{\uparrow}\gamma \ \mid \ \forall p{\uparrow}\gamma \ \mid \ \gamma_1 \wedge \gamma_2$$

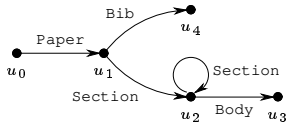
where the additional rules for the satisfaction of constraints of $\mathcal{L}_{\mathcal{ALE}}$ in a node $u$ of a $\mathcal{T}$-database are:

$$\begin{array}{lll}
u \models_c \exists p{\uparrow}\gamma & \text{iff} & \exists u \xrightarrow{a} v \in Edges(D).\, (\mathcal{T} \models p(a) \wedge v \models_c \gamma) \\
u \models_c \forall p{\uparrow}\gamma & \text{iff} & \forall u \xrightarrow{a} v \in Edges(D).\, (\mathcal{T} \models p(a) \supset v \models_c \gamma)
\end{array}$$

Observe that $\mathcal{L}_{\mathcal{ALE}}$ 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 (Donini *et al.* 1992), we can show that consistency checking is coNP-hard.

**Theorem 9** *Checking the consistency of a $\mathcal{T}$-schema $\mathcal{S}$ with $\mathcal{L}_{\mathcal{ALE}}$-constraints is coNP-hard in the size of $\mathcal{S}$, even if $\mathcal{T}$ is empty, i.e. all edges of $\mathcal{S}$ are labeled with* **true***.*

By observing that checking consistency can be reduced to checking subsumption wrt an inconsistent schema, we immediately get that subsumption is NP-hard. Theorem 9 shows also that consistency stays coNP-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 database may exist, whose topology is determined by the constraints. Since $\mathcal{T}$ cannot predict anything about the possible topologies of databases, the validity checker of $\mathcal{T}$ cannot be used to "hide" a potentially exponential calculation. Note that this is different from




$$\mathcal{C}(u_1) = \exists^{\geq 1}\texttt{Section} \wedge \exists^{\leq 1}\texttt{Bib} \wedge \mu X.\forall\texttt{Section}\uparrow X$$
$$\mathcal{C}(u_2) = \exists^{\geq 1}\texttt{Section} \vee \exists^{\geq 1}\texttt{Body}$$
$$\mathcal{C}(u_0) = \mathcal{C}(u_3) = \mathcal{C}(u_4) = \top$$

$$\left\{\begin{array}{rcl} \texttt{Paper} & \sqsubseteq & \neg\texttt{Bib} \sqcap \neg\texttt{Section} \sqcap \neg\texttt{Body} \\ \texttt{Section} & \sqsubseteq & \neg\texttt{Bib} \sqcap \neg\texttt{Body} \\ \texttt{Body} & \sqsubseteq & \neg\texttt{Bib} \\ \texttt{Index} & \sqsubseteq & \texttt{Itemize} \sqcup \texttt{Enumerate} \end{array}\right\} \subseteq \mathcal{T}$$

Figure 1: A $\mathcal{T}$-schema with $\mathcal{L}_\mu$-constraints

the case of local constraints, where the aspects related to the topology enforced by the constraints can be embedded in an appropriate formula of $\mathcal{T}$.

## Fixpoint Constraints

We now extend our framework to a very expressive constraint language, which is a variant of $\mu\mathcal{ALCQ}$, and show decidability of consistency and subsumption.

The constraint language $\mathcal{L}_\mu$ is the set of *closed* formulae constructed according to the following syntax ($p$ denotes a formula of $\mathcal{T}$, $n$ a positive integer, and $X$ a variable):

$$\begin{array}{rcl} \gamma & ::= & X \mid \exists^{\geq n} F \mid \neg\gamma \mid \gamma_1 \wedge \gamma_2 \mid \mu X.\gamma \\ F & ::= & p \mid \uparrow\gamma \mid \neg F \mid F_1 \wedge F_2 \end{array}$$

with the restriction that every free occurrence of $X$ in $\mu X.\gamma$ is in the scope of an even number of negations.

We introduce the abbreviations: $\gamma_1 \vee \gamma_2$ for $\neg(\neg\gamma_1 \wedge \neg\gamma_2)$, $\top$ for $\gamma \vee \neg\gamma$, and $\forall p\uparrow\gamma$ for $\neg\exists^{\geq 1}(p \wedge \uparrow\neg\gamma)$.

Let $D$ be a $\mathcal{T}$-database, and $\mathcal{M}$ be a model of $\mathcal{T}$. A *valuation* $\rho$ on $D$ is a mapping from variables to subsets of $Nodes(D)$. We denote by $\rho[X/\mathcal{E}]$ the valuation identical to $\rho$ except for $\rho[X/\mathcal{E}](X) = \mathcal{E}$. For each node $u \in Nodes(D)$, we define when $u$ *satisfies a constraint* $\gamma$ *under a valuation* $\rho$, in notation $\rho, u \models_c \gamma$, as follows:

$$\begin{array}{lll} \rho, u \models_c X & \text{iff} & u \in \rho(X) \\ \rho, u \models_c \exists^{\geq n} F & \text{iff} & \#\{u \xrightarrow{a} v \in Edges(D) \mid \\ & & \quad \rho, u \xrightarrow{a} v \models_c F\} \geq n \\ \rho, u \models_c \neg\gamma & \text{iff} & \rho, u \not\models_c \gamma \\ \rho, u \models_c \gamma_1 \wedge \gamma_2 & \text{iff} & (\rho, u \models_c \gamma_1) \wedge (\rho, u \models_c \gamma_2) \\ \rho, u \models_c \mu X.\gamma & \text{iff} & \forall \mathcal{E} \subseteq Nodes(D). (\forall v \in Nodes(D). \\ \multicolumn{3}{r}{\rho[X/\mathcal{E}], v \models_c \gamma \supset \rho[X/\mathcal{E}], v \models_c X) \supset \rho[X/\mathcal{E}], u \models_c X} \end{array}$$

where

$$\begin{array}{ll} \rho, u \xrightarrow{a} v \models_c p & \text{iff } \mathcal{T} \models p(a) \\ \rho, u \xrightarrow{a} v \models_c \uparrow\gamma & \text{iff } \rho, v \models_c \gamma \\ \rho, u \xrightarrow{a} v \models_c \neg F & \text{iff } \rho, u \xrightarrow{a} v \not\models_c F \\ \rho, u \xrightarrow{a} v \models_c F_1 \wedge F_2 & \text{iff } (\rho, u \xrightarrow{a} v \models_c F_1) \wedge (\rho, u \xrightarrow{a} v \models_c F_2) \end{array}$$

Since the constraints in $\mathcal{L}_\mu$ are closed formulae, satisfaction is independent of the valuation, and we denote it simply by $u \models_c \gamma$.

**Example** The schema shown in Figure 1 represents a set of web pages such as those generated by "latex2html" when translating a LaTeX article containing nested sections and possibly a bibliography. The connections between the pages are represented by the graph, whereas the content of the pages is modeled by $\mathcal{T}$. Notice the use of constraints to state complex conditions on the structure of the allowed databases. In particular, the constraint $\mu X.\forall\texttt{Section}\uparrow X$ associated with $u_1$ rules out all databases that have loops in the connections of the various sections.

**Checking Subsumption for $\mathcal{L}_\mu$** We develop now a technique for checking subsumption of $\mathcal{T}$-schemas with $\mathcal{L}_\mu$-constraints, which works in the case where the theory $\mathcal{T}$ can be expressed in terms of axioms of $\mu\mathcal{ALCQ}$. In order to illustrate the features of the technique, we further assume that $\mathcal{T}$ is interpreted over a fixed finite universe $U$, includes only unary predicates, one distinct constant $c(d)$ for each element $d \in \mathcal{U}$, and is presented as a finite set containing either $p(a)$ or $\neg p(a)$ for each predicate $p$ and constant $a$[1].

The formulae of $\mathcal{T}$ that label the edges of a $\mathcal{T}$-schema are boolean combinations of atomic formulae in the language of $\mathcal{T}$ and of expressions of the form $(self = a)$, where $a$ is a constant of $\mathcal{T}$. We define when a formula $p(a)$ labeling an edge is valid in $\mathcal{T}$, in notation $\mathcal{T} \models p(a)$, as follows:

$$\begin{array}{lll} \mathcal{T} \models (self = a')(a) & \text{iff} & a = a' \\ \mathcal{T} \models \neg p(a) & \text{iff} & \mathcal{T} \not\models p(a) \\ \mathcal{T} \models (p_1 \wedge p_2)(a) & \text{iff} & \mathcal{T} \models p_1(a) \wedge \mathcal{T} \models p_2(a) \end{array}$$

It is immediate to view a $\mathcal{T}$-database as a $\mathcal{T}$-schema, simply by replacing each edge label $a$ by $(self = a)$. Therefore, as in BDFS, conformance is a special case of subsumption.

The technique we use for checking subsumption is based on a reduction to unsatisfiability in $\mu\mathcal{ALCQ}$ knowledge bases. Differently from the previous cases, in what follows we consider $\mathcal{T}$ to be part of the input to subsumption checking.

Given two $\mathcal{T}$-schemas $\mathcal{S}_1$ and $\mathcal{S}_2$, we reduce the problem of deciding whether $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$, to the problem of deciding the unsatisfiability of the $\mu\mathcal{ALCQ}$ concept $\Phi_{\mathcal{S}_1} \sqcap \neg\Phi_{\mathcal{S}_2}$ in the $\mu\mathcal{ALCQ}$ knowledge base $\Gamma_\mathcal{T}$, where $\Gamma_\mathcal{T}$, $\Phi_{\mathcal{S}_1}$, and $\Phi_{\mathcal{S}_2}$ are defined as follows.

**$\Gamma_\mathcal{T}$: encoding of $\mathcal{T}$ and of the general properties of BDFS graphs** To encode the general properties of BDFS graphs, $\Gamma_\mathcal{T}$ exploits *reification* of edges, as used in (Buneman *et al.* 1997). Specifically, we use a special role $\mathbf{E}$ and split each labeled edge $u \xrightarrow{a} v$ into two edges $u \xrightarrow{\mathbf{E}} e_{uv} \xrightarrow{\mathbf{E}} v$, by introducing an intermediate node $e_{uv}$ labeled by $a$. $\Gamma_\mathcal{T}$ contains the following axioms ($\top_N$, $\top_E$, and $\top_D$ are new atomic concepts, and $\mathbf{L}$ is a new role):

$$\begin{array}{rcl@{\qquad}rcl} \top & \sqsubseteq & \top_N \sqcup \top_E \sqcup \top_D & \top_N & \sqsubseteq & \neg\top_E \\ \top_E & \sqsubseteq & \neg\top_D & \top_D & \sqsubseteq & \neg\top_N \\ \top_N & \sqsubseteq & \forall\mathbf{E}.\top_E & & & \\ \top_E & \sqsubseteq & \forall\mathbf{E}.\top_N \sqcap (=1\,\mathbf{E}.\top) \sqcap \forall\mathbf{L}.\top_D \sqcap (=1\,\mathbf{L}.\top) & & & \end{array}$$

Intuitively, these axioms partition the interpretation domain into objects denoting nodes ($\top_N$), edges ($\top_E$), and constants of $\mathcal{T}$ ($\top_D$), and specify the correct links for those object denoting nodes and edges.

In addition, in order to encode the theory $\mathcal{T}$, we introduce one concept $C_p$ for each predicate of $\mathcal{T}$, and one concept $O_a$ (called an *object-concept*) for each constant $a$ of $\mathcal{T}$, and, for each pair $C_p$, $O_a$ we add to $\Gamma_\mathcal{T}$ the axiom:

[1] We point out that we restrict ourselves to such simple kinds of theories for the sake of simplicity, but our approach works when $\mathcal{T}$ has a more general form.

$$\begin{array}{ll} \top_D \sqcap O_a \sqsubseteq C_p & \text{if} \quad \mathcal{T} \models p(a) \\ \top_D \sqcap O_a \sqsubseteq \neg C_p & \text{if} \quad \mathcal{T} \models \neg p(a) \end{array}$$

Observe that, $|\Gamma_{\mathcal{T}}|$ is linear in $|\mathcal{T}|$.

**$\Phi_{\mathcal{S}}$: encoding of the schema $\mathcal{S}$** In order to define the encoding $\Phi_{\mathcal{S}}$ of a $\mathcal{T}$-schema $\mathcal{S} = (\mathcal{G}, \mathcal{C})$ we define a mapping $\psi$ from constraint expressions to $\mu\mathcal{ALCQ}$ formulae as follows:

$$\begin{array}{rlrl} \psi(X) &= X & \psi(p) &= \forall \mathbf{L}.p \\ \psi(\exists^{\geq n} F) &= (\geq n\, \mathbf{E}.\psi(F)) & \psi(\uparrow\gamma) &= \forall \mathbf{E}.\psi(\gamma) \\ \psi(\neg\gamma) &= \neg\psi(\gamma) & \psi(\neg F) &= \neg\psi(F) \\ \psi(\gamma_1 \wedge \gamma_2) &= \psi(\gamma_1) \sqcap \psi(\gamma_2) & \psi(F_1 \wedge F_2) &= \psi(F_1) \sqcap \psi(F_2) \\ \psi(\mu X.\gamma) &= \mu X.\psi(\gamma) & & \end{array}$$

We construct for each node $u \in \mathit{Nodes}(\mathcal{G}) = \{u_1, \ldots, u_h\}$ a *characteristic $\mu\mathcal{ALCQ}$ concept* $\chi_u$ as follows[2]: Consider the set of mutual recursive equations, one for each node $u_i$ in $\mathit{Nodes}(\mathcal{G})$

$$\begin{array}{l} X_{u_1} \equiv \top_N \sqcap \psi(\mathcal{C}(u_1)) \sqcap \forall \mathbf{E}.(\top_E \sqcap \bigsqcup_{u_1 \xrightarrow{p} v} (\forall \mathbf{L}.p \sqcap \forall \mathbf{E}.X_v)) \\ \quad \ldots \\ X_{u_h} \equiv \top_N \sqcap \psi(\mathcal{C}(u_h)) \sqcap \forall \mathbf{E}.(\top_E \sqcap \bigsqcup_{u_h \xrightarrow{p} v} (\forall \mathbf{L}.p \sqcap \forall \mathbf{E}.X_v)) \end{array}$$

and eliminate, one at the time, each of the above equations, except the one for $X_{u_i}$ as follows: Eliminate the equation $X_{u_j} = C_j$ and substitute each occurrence of $X_{u_j}$ in the remaining equations by $\nu X_{u_j}.C_j$. Let $X_{u_i} = C_i$ be the resulting equation. The concept $\chi_{u_i}$ is $\nu X_{u_i}.C_i$. The encoding $\Phi_{\mathcal{S}}$ of $\mathcal{S}$ is $\Phi_{\mathcal{S}} = \chi_{root(\mathcal{G})}$.

Observe that, in the worst case, $|\Phi_{\mathcal{S}}|$ is exponential with respect to $|\mathcal{S}|$.

**Properties of the encoding** The following three properties of the encoding establish decidability and complexity of checking subsumption between two $\mathcal{T}$-schemas with $\mathcal{L}_\mu$-constraints $\mathcal{S}_1$ and $\mathcal{S}_2$.

**Theorem 10** *$\mathcal{S}_1$ is subsumed by $\mathcal{S}_2$ if and only if there is no model of $\Gamma_{\mathcal{T}}$ that satisfies $\Phi_{S_1} \sqcap \neg\Phi_{S_2}$ and interprets every object-concept as a singleton.*

**Theorem 11** *Let $\Gamma_{\mathcal{T}}$, $\Phi_{\mathcal{S}_1}$, and $\Phi_{\mathcal{S}_2}$ be as defined above. Then there exists a $\mu\mathcal{ALCQ}$ knowledge base $\Gamma'$ whose size is polynomial in $|\Gamma_{\mathcal{T}}| + |\Phi_{\mathcal{S}_1}| + |\Phi_{\mathcal{S}_2}|$ such that: $\Phi_{S_1} \sqcap \neg\Phi_{S_2}$ is satisfied in a model of $\Gamma_{\mathcal{T}}$ that interprets every object-concept as a singleton, if and only if $\Phi_{S_1} \sqcap \neg\Phi_{S_2}$ is satisfiable in $\Gamma'$.*

**Theorem 12** *Checking whether $\mathcal{S}_1$ is subsumed by $\mathcal{S}_2$ is EXPTIME-hard and decidable in time $O(2^{p(|\Gamma_{\mathcal{T}}| + |\Phi_{\mathcal{S}_1}| + |\Phi_{\mathcal{S}_2}|)})$.*

Since $|\Phi_{\mathcal{S}}|$ may be exponential with respect to $|\mathcal{S}|$, it follows that subsumption checking in the presence of $\mathcal{L}_\mu$-constraints can be done in deterministic double exponential time with respect to the size of the two schemas.

[2]This construction is analogous to the one used in Process Algebra for defining a characteristic formula of a process (Steffen & Ingólfsdóttir 1994), i.e. a formula which is satisfied by exactly all processes that are equivalent to the process under bisimulation. In a certain sense, we may say that $\Phi_{\mathcal{S}}$ characterizes, exactly all databases that conform to $\mathcal{S}$.

## Conclusions

The result of our investigation is that very expressive DLs are interesting tools for modeling and reasoning about semi-structured data with constraints. The analysis presented in the paper shows that the complexity of subsumption rises even when simple non-local constraints are added to BDFS. This justifies our approach that aims at adding as much expressive power as possible in specifying the constraints, without loosing decidability.

**Acknowledgments** This work was partly supported by ESPRIT LTR Prj. No. 22469 DWQ and the Italian Space Agency.

## References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts.

Abiteboul, S. 1997. Querying semi-structured data. In *ICDT-97*, 1–18.

Baader, F. 1996. Using automata theory for characterizing the semantics of terminological cycles. *Ann. of Math. and AI* 18:175–219.

Borgida, A., and Patel-Schneider, P. F. 1994. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. of Artificial Intelligence Research* 1:277–308.

Buneman, P.; Davidson, S.; Fernandez, M.; and Suciu, D. 1997. Adding structure to unstructured data. In *ICDT-97*, 336–350.

Calvanese, D. 1996. Finite model reasoning in description logics. In *KR-96*, 292–303.

Christophides, V.; Abiteboul, S.; Cluet, S.; and Scholl, M. 1994. From structured documents to novel query facilities. In *ACM SIGMOD*, 313–324.

De Giacomo, G., and Lenzerini, M. 1996. TBox and ABox reasoning in expressive description logics. In *KR-96*, 316–327.

De Giacomo, G., and Lenzerini, M. 1997. A uniform framework for concept definitions in description logics. *J. of Artificial Intelligence Research* 6:87–110.

Donini, F. M.; Hollunder, B.; Lenzerini, M.; Spaccamela, A. M.; Nardi, D.; and Nutt, W. 1992. The complexity of existential quantification in concept languages. *Artif. Intell.* 2–3:309–327.

Emerson, E. A. 1996. Automated temporal reasoning about reactive systems. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*. Springer-Verlag. 41–101.

Kozen, D. 1983. Results on the propositional $\mu$-calculus. *Theor. Comp. Sci.* 27:333–354.

Lloyd, J. W. 1987. *Foundations of Logic Programming (Second, Extended Edition)*. Springer-Verlag.

Mendelzon, A.; Mihaila, G. A.; and Milo, T. 1997. Querying the World Wide Web. *Int. J. on Digital Libraries* 1(1):54–67.

Park, D. 1970. Fixpoint induction and proofs of program properties. In *Machine Intelligence*, volume 5. Edinburgh University Press. 59–78.

Quass, D.; Rajaraman, A.; Sagiv, I.; Ullman, J.; and Widom, J. 1995. Querying semistructured heterogeneous information. In *DOOD-95*, 319–344. Springer-Verlag.

Steffen, B., and Ingólfsdóttir, A. 1994. Characteristic formulae for processes with divergence. *Information and Computation* 110:149–163.

Stirling, C. 1996. Modal and temporal logics for processes. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *LNCS*. Springer-Verlag. 149–237.

Streett, R. E., and Emerson, E. A. 1989. An automata theoretic decision procedure for the propositional $\mu$-calculus. *Information and Computation* 81:249–264.