Semistructured Data Schemas with Expressive Constraints

Andrea Calì, Diego Calvanese, Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
{cali,calvanese,lenzerini}@dis.uniroma1.it

Abstract

Recently, there have been several proposals of formalisms for modeling semistructured data, which is data that is neither raw, nor strictly typed as in conventional database systems. Semistructured data models are graph-based models, where graphs are used to represent both databases and schemas. We study the basic problem of schema subsumption, which amounts to check whether all databases conforming to a schema also conform to another schema, in the presence of constraints, which are used to enforce additional conditions on databases. In particular, we study the relationship between various constraint languages and the basic property of locality, which allows one to check subsumption between schemas in polynomial time in the number of nodes of the schemas. We show that locality holds when both numeric constraints and disjunction are added to a simple constraint language. On the other hand, locality is lost when we consider constraints both on outgoing and incoming edges of databases.

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 [25, 1, 6, 23, 19, 20]. Semistructured data are data that is neither raw, nor strictly typed as in conventional database systems [1]. Recently, several formalisms for modeling semistructured data have been proposed, such as OEM (Object Exchange Model) [2], and BDFS (Basic Data model For Semistructured data) [6]. In such formalisms, data are represented 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 databases 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 database g conforming to a schema S is given in terms of a special relation, called *simulation*, between the two graphs. 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.

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 database conforming to one schema always conforms to another schema. In [6] an algorithm for checking subsumption in BDFS is presented and its complexity is analyzed.

The problem of extending BDFS with different types of constraints in terms of formulae associated to nodes of the schema, has first been studied in [8, 9]. In particular, so called *locality* of constraints is identified as a crucial property, which allows one to perform the basic inference tasks (namely checking consistency, conformance, and subsumption) by means of several *local checks*, and thus retain efficiency with respect to the total size of the schemas. A simple constraint language with the property of locality, called $\mathcal{L}_{sl}$, is presented, which allows one to express existence and uniqueness of outgoing edges with certain properties.

In this paper we investigate the relationship between the expressive power of constraint languages and the ability to retain locality, thus keeping the complexity of the basic inference tasks over schemas tractable. In particular:

- We add to $\mathcal{L}_{sl}$ more complex forms of numeric constraints (see e.g., [3, 21, 27, 5, 18] and show that this extension preserves locality.
- We add disjunctions of constraints and show that locality is not lost.
- When locality holds, we present algorithms to check consistency and subsumption of schemas which can be run in polynomial time wrt the number of nodes.
- We add to $\mathcal{L}_{sl}$ constraints also on incoming edges (see, e.g., [13, 12, 14, 22]), and show that locality and the finite model property are lost.

2 Framework

The formal model for semistructured data introduced in [6] which we call BDFS (*Basic Data model For Semistructured data*) and its extension with constraints studied in [8, 9] are at the basis of our investigation. BDFS is an edge-labeled graph model for semistructured data where edge-labels are formulae of a first-order language $\mathcal{L}_{\mathcal{T}}$. The language $\mathcal{L}_{\mathcal{T}}$ is built over a set of predicates, including the equality predicate "=", and contains one constant for each element of a fixed (not necessarily finite) universe $\mathcal{U}$. Schemas and databases in BDFS always refer to a *complete* and *decidable* theory $\mathcal{T}$ on $\mathcal{U}$. In BDFS, databases are rooted graphs whose edges are labeled with constants of $\mathcal{T}$, and schemas are rooted graphs whose edges are labeled with unary formulae of $\mathcal{T}$. We consider the extension of BDFS with constraints, as introduced in [8, 9], which allow one to overcome several limitations of the basic BDFS model, e.g., the ability to enforce the existence of edges. *Constraints*, which are expressed in a certain *constraint language* $\mathcal{L}$, label the nodes of the schemas (denoted by $\mathcal{L}$-schemas) and impose additional conditions on the outgoing or incoming edges of the nodes of a database.

To establish if a database is coherent with an $\mathcal{L}$-schema, or if an $\mathcal{L}$-schema is more specific than another one, we define the notions of *conformance* and *subsumption*, which in turn are based on the fundamental notion of simulation. A *simulation* from a database g to an $\mathcal{L}$-schema S is a binary relation $\trianglelefteq$ from the nodes of g to the nodes of S such that, for each pair of nodes v in g and u in S with $v \trianglelefteq u$ the following holds:

1. For each edge $v \xrightarrow{c} v_s$ in g, there exists an edge $u \xrightarrow{\pi} u_s$ in S such that $\mathcal{T} \models \pi(c)$ and $v_s \trianglelefteq u_s$.
2. v satisfies the constraint labeling u (denoted by $\mathcal{C}(u)$).

A database g *conforms* to a schema S, in notation $g \preceq S$, if there exists a simulation $\trianglelefteq$ from g to S such that $\mathrm{root}(g) \trianglelefteq \mathrm{root}(S)$. A schema S_1 *is subsumed by* a schema S_2, in notation $S_1 \sqsubseteq S_2$, if for every database g we have that $g \preceq S_1$ implies $g \preceq S_2$. Two schemas S_1 and S_2 are *equivalent*, in notation $S_1 \equiv S_2$, if both $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$. Since each database can be considered as a schema, conformance is a special case of subsumption and henceforth we concentrate on subsumption only.

Example 1 Figure 1(a) shows a schema modeling the departments of a university, in which professors teach courses, and a student can have either a professor or a student as tutor. We point out that course, prof, stu, tutor, and teaches are *constants*; a constant c labeling an edge of a schema denotes the unary predicate $\lambda x.x = c$. Figure 1(b) shows a conforming database.

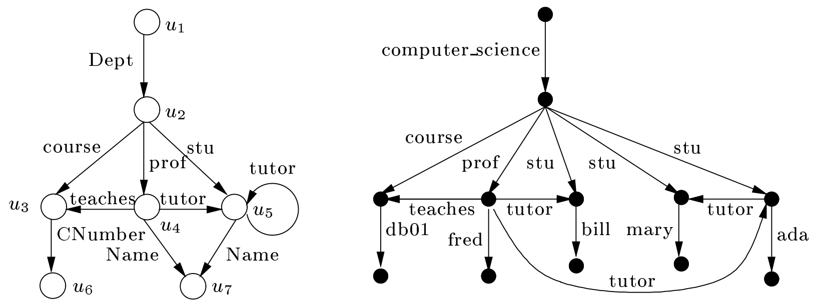


Figure 1: Schema and conforming database

Note that the empty database, which is the database consisting of a single node, conforms to every schema without constraints. However, when we add constraints we also need to consider the notion of consistency. An $\mathcal{L}$-schema which has at least one conforming database is said to be *consistent*. A node u of an $\mathcal{L}$-schema S is said to be *consistent* if there exists a database that conforms to S_u, where S_u is the $\mathcal{L}$-schema identical to S except for the root, which is u. Since all meaningful constraint languages allow one to construct an inconsistent schema, schema inconsistency can be reduced to schema subsumption.

Following [6, 8, 9] we 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), and that the validity of a formula of $\mathcal{T}$ can be checked in constant time.

2.1 Locality of Constraints

In [8, 9] several languages for expressing constraints are presented and the complexity of the basic inference tasks (namely checking consistency, conformance, and subsumption) for such languages is studied. In particular, so called *locality* of constraints is identified as a crucial property that allows one to perform the basic inference tasks by means of several *local checks*, and thus retain efficiency with respect to the total size of the schemas.

A constraint $\mathcal{C}(u)$ labeling a node u of a schema is said to be *local* when it forces conditions that are local to the nodes simulating u. In other words, one can check whether u is consistent by considering only u with $\mathcal{C}(u)$ and the outgoing (and incoming[1]) edges of u (taking as nodes attached to such edges fresh nodes with no constraint on them), while ignoring all other nodes and edges in the database. A constraint language in which all constraints are local is called *local.*

Syntax	**Semantics**
$v \rightarrowtail \top$	Always
$v \rightarrowtail \exists \mathsf{edge}(p)$	$\exists v \xrightarrow{c} v_s \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)$
$v \rightarrowtail \nexists \mathsf{edge}(p)$	$\nexists v \xrightarrow{c} v_s \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)$
$v \rightarrowtail \exists^{\leq 1} \mathsf{edge}(p)$	$\#\{v \xrightarrow{c} v_s \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)\} \leq 1$
$v \rightarrowtail \gamma_1 \wedge \gamma_2$	$(v \rightarrowtail \gamma_1) \wedge (v \rightarrowtail \gamma_2)$

Figure 2: The language $\mathcal{L}_{sl}$ ($\#X$ denotes the cardinality of set X)

Figure 2 shows syntax and semantics of the language $\mathcal{L}_{sl}$ [8] of *simple local constraints*, which is at the basis of our investigation. Constraints of $\mathcal{L}_{sl}$ can force the existence and uniqueness of edges satisfying a certain formula. In Figure 2, p is a unary formula of $\mathcal{T}$ and $v \rightarrowtail \gamma$ means that v satisfies the constraint γ.

We call constraints of the form $\nexists \mathsf{edge}(p)$, which express nonexistence of an outgoing edge satisfying p, *negative constraints*. We observe that negative constraints can be eliminated from an $\mathcal{L}_{sl}$-schema without altering the semantics of the schema. Indeed, for a node u of an $\mathcal{L}_{sl}$-schema S for which $\mathcal{C}(u)$ contains the conjunct $\nexists \mathsf{edge}(p)$, we can remove $\nexists \mathsf{edge}(p)$ from $\mathcal{C}(u)$, provided that we conjoin all formulae labeling the outgoing edges of u with $\neg p$.

Example 2 Referring to the schema of Figure 1, we would like to enforce the following properties:

(1) a course must have exactly one course number;

(2) a professor must teach at least one course;

(3) a professor or student must have exactly one name.

This can be done by using constraints of $\mathcal{L}_{sl}$ as follows:

$$\begin{array}{lcl}
\mathcal{C}(u_1) & = & \mathcal{C}(u_2) \;=\; \mathcal{C}(u_6) \;=\; \mathcal{C}(u_7) \;=\; \top \\
\mathcal{C}(u_3) & = & \exists \mathsf{edge}(\mathrm{CNUmber}) \wedge \exists^{\leq 1} \mathsf{edge}(\mathrm{CNumber}) \\
\mathcal{C}(u_4) & = & \exists \mathsf{edge}(\mathrm{course}) \wedge \exists \mathsf{edge}(\mathrm{Name}) \wedge \exists^{\leq 1} \mathsf{edge}(\mathrm{Name}) \\
\mathcal{C}(u_5) & = & \exists \mathsf{edge}(\mathrm{Name}) \wedge \exists^{\leq 1} \mathsf{edge}(\mathrm{Name})
\end{array}$$

As shown in [8], to check whether a node u of an $\mathcal{L}_{sl}$-schema is consistent, it is sufficient to consider u with its constraint $\mathcal{C}(u)$ and the outgoing edges of u (while neglecting the constraints over the successors of u). Hence $\mathcal{L}_{sl}$ is local, and in [8] it is indeed shown that consistency and subsumption for $\mathcal{L}_{sl}$-schemas can be checked in polynomial time in the size of the schemas. As shown in the next section for two meaningful extensions of $\mathcal{L}_{sl}$, when the constraint language is local, one can always verify consistency and subsumption by performing a polynomial number of local validity checks. The size of the formula to check for validity will typically be small wrt the total size of the two schemas. Hence, the total cost of subsumption can be kept low (e.g., polynomial) wrt the size of the two schemas, even when the cost of performing a single validity check is high wrt the size of the involved constraint and

[1] We will consider in Section 4 also constraints on incoming edges.

edge labels (e.g., the formula to check for validity is exponential in the constraint). For example, under the reasonable assumption that both the size of each constraint attached to a node and the number of outgoing edges for a node are logarithmic in the total size of a schema, checking subsumption can be done in polynomial time in the size of the two schemas.

On the other hand, the lack of locality for a constraint language $\mathcal{L}$ is a strong indication for intractability of the basic inference tasks on $\mathcal{L}$-schemas and databases. For example, in [8] the constraint language $\mathcal{L}_{\mathcal{ALE}}$ is presented, for which it is precisely the lack of locality that makes checking the consistency of an $\mathcal{L}_{\mathcal{ALE}}$-schema coNP-hard in the size of the schemas.

3 Local Constraints on Outgoing Edges

We now study two extensions of the constraint languages $\mathcal{L}_{sl}$ which we will prove to be local.

3.1 Extended Local Constraints

We now introduce a new constraint language $\mathcal{L}_{ext}$ of so called *extended local constraints*, inspired by cardinality constraints typically present in database models and Knowledge representation formalism [3, 21, 27, 5, 18].

Syntax and semantics of $\mathcal{L}_{ext}$ are shown in Figure 3, where p is a unary formula of $\mathcal{T}$, and n, called a *numeric index*, is a number in $\mathbb{N}$ (we assume $0 \in \mathbb{N}$) represented in *unary*. Notice that $\mathcal{L}_{ext}$ extends $\mathcal{L}_{sl}$, since $\exists^{\leq 0}\mathsf{edge}(p)$ is equivalent to $\nexists\mathsf{edge}(p)$, and $\exists^{\geq 1}\mathsf{edge}(p)$ is equivalent to $\exists\mathsf{edge}(p)$.

Syntax	**Semantics**
$v \rightarrowtail \top$	Always
$v \rightarrowtail \exists^{\leq n}\mathsf{edge}(p)$	$\#\{v \xrightarrow{c} v_s \in \text{Edges}(g) \mid \mathcal{T} \models p(c)\} \leq n$
$v \rightarrowtail \exists^{\geq n}\mathsf{edge}(p)$	$\#\{v \xrightarrow{c} v_s \in \text{Edges}(g) \mid \mathcal{T} \models p(c)\} \geq n$
$v \rightarrowtail \gamma_1 \wedge \gamma_2$	$(v \rightarrowtail \gamma_1) \wedge (v \rightarrowtail \gamma_2)$

Figure 3: The language $\mathcal{L}_{ext}$

Example 3 Let us consider again the schema of Figure 1. We would like to enforce, in addition to the properties specified in Example 2, also the following properties:

(4) a professor must teach exactly 2 courses;

(5) a student or professor can have at most 3 students of whom he/she is tutor.

This can be done by adding the following constraints of $\mathcal{L}_{ext}$:

$$\begin{aligned}\mathcal{C}(u_4) &= \exists^{\geq 2}\mathsf{edge}(\text{teaches}) \wedge \exists^{\leq 2}\mathsf{edge}(\text{teaches}) \wedge \exists^{\leq 3}\mathsf{edge}(\text{tutor})\\ \mathcal{C}(u_5) &= \exists^{\leq 3}\mathsf{edge}(\text{tutor})\end{aligned}$$

We now show that $\mathcal{L}_{ext}$ is local and therefore we can check the consistency of an $\mathcal{L}_{ext}$-schema by means of local checks for consistency of the nodes of the schema. Indeed, we present the function `cons`, which checks a node u for consistency by means of local checks. Let u have ℓ outgoing edges, labeled $r_1, \ldots, r_\ell$, let the constraint $\mathcal{C}(u)$ over u be

$$\mathcal{C}(u) = \exists^{\leq n_1}\mathsf{edge}(p_1) \wedge \cdots \wedge \exists^{\leq n_s}\mathsf{edge}(p_s) \wedge \exists^{\geq m_1}\mathsf{edge}(q_1) \wedge \cdots \wedge \exists^{\geq m_t}\mathsf{edge}(q_t)$$

and let $M = \sum_{i=1}^{t} m_i$.

```
function cons(u: node, w: ℒ_ext-constraint): boolean
{ if t = 0 then return true;
  if t > 0 and OutEdges(u, S) = ∅ then return false;
  for K = 1 to M
    if 𝒯 ⊨ ∃x_1 ··· ∃x_K (F ∧ G ∧ H) then return true;
  return false;
}
```

where F, G, and H are the following formulae of $\mathcal{T}$:

$$
\begin{aligned}
F &= \bigwedge_{1\le i\le t} \; \bigvee_{1\le j_1<\cdots<j_{m_i}\le K} \; \bigwedge_{1\le k\le m_i} q_i(x_{j_k}) \\
G &= \bigwedge_{1\le i\le s} \; \bigwedge_{1\le j_1<\cdots<j_{n_i+1}\le K} \neg \bigwedge_{1\le k\le n_i+1} p_i(x_{j_k}) \\
H &= \bigwedge_{1\le i\le K} \; \bigvee_{1\le j\le \ell} r_j(x_i)
\end{aligned}
$$

Theorem 1 *Given a node u of an $\mathcal{L}_{ext}$-schema S, we have that u is consistent iff* `cons`$(u, \mathcal{C}(u))$ *returns true. Moreover,* `cons`$(u, \mathcal{C}(u))$ *runs in time polynomial in the number of conjuncts in $\mathcal{C}(u)$, and in time exponential in the numeric indexes in $\mathcal{C}(u)$.*

Proof (sketch). The function `cons` queries the theory about the existence of K constants (for K going from 1 to M), which can be used to label the outgoing edges of a node simulating u, respecting the conditions imposed by $\mathcal{C}(u)$. In other words, `cons` tries to build a *fragment* of database made up of a single node and up to M outgoing edges, which conforms to the schema S_u obtained from S by considering u as the root (it is possible to show that one can neglect the constraints over the successors of u, and still fix the fragments to a database conforming to S_u). The cost is exponential in the numeric indexes in $\mathcal{C}(u)$ because the size of F and G is exponential in those indexes. □

By exploiting `cons` we can construct the function `rinext`, which iteratively removes the inconsistent nodes from an $\mathcal{L}_{ext}$-schema, as follows:

```
function rinext(S: ℒ_ext-schema): ℒ_ext-schema
{ repeat
    if there is a node u ∈ Nodes(S) that satisfies one of the following conditions:
       • u is not reachable with a direct path starting from root(S);
       • cons(u, 𝒞(u)) = false
    then remove from S: ⋆ the node u,
                        ⋆ all the edges outgoing from u, and
                        ⋆ all the edges incoming in u;
  until root(S) is removed from S or no new node has been removed from S;
  return S
}
```

Theorem 2 *Given an $\mathcal{L}_{ext}$-schema S, we have that S is consistent iff* $\texttt{rinext}(S)$ *does not remove* $\mathrm{root}(S)$. *Moreover,* $\texttt{rinext}(S)$ *runs in time polynomial in the number of nodes of S, and in time exponential in the numeric indexes of the constraints labeling the nodes of S.*

Proof (sketch). The function checks the nodes of S for consistency, calling the function `cons`. It is easy to show that if $\mathrm{root}(S)$ is not removed, the fragments of database conforming to each consistent node in S can *always* be fixed together to obtain a database conforming to S. The claim follows from the fact that the function `cons` is called a number of times which is polynomial in the number of nodes of S. □

Theorem 2 provides a bound on schema consistency which is polynomial in the number of nodes, but exponential in the size of the schema. We can also show that $\mathcal{L}_{ext}$-schema consistency can be checked by a nondeterministic algorithm which runs in polynomial time in the size of the schema.

Theorem 3 *Given an $\mathcal{L}_{ext}$-schema S, verifying the consistency of a node u of S can be done in nondeterministic polynomial time in the size of $\mathcal{C}(u)$.*

Proof (sketch). Let u have ℓ outgoing edges, labeled $r_1, \ldots, r_\ell$, let the constraint $\mathcal{C}(u)$ over u be

$$\mathcal{C}(u) = \exists^{\leq n_1}\mathsf{edge}(p_1) \wedge \cdots \wedge \exists^{\leq n_s}\mathsf{edge}(p_s) \wedge \exists^{\geq m_1}\mathsf{edge}(q_1) \wedge \cdots \wedge \exists^{\geq m_t}\mathsf{edge}(q_t)$$

and let $M = \sum_{i=1}^{t} m_i$. Then we guess a number $K \in \{1, \ldots, M\}$ and try to construct a database fragment that conforms to u consisting of a node with K outgoing edges. To do so, we guess for each of the K outgoing edges, whether the constant labeling the edge should satisfy or not each of the predicates $p_1, \ldots, p_s, q_1, \ldots, q_t$. Once we have done the guess we can check whether the numeric constraints are satisfied by simply counting the constants that satisfy each of the predicates $p_1, \ldots, p_s, q_1, \ldots, q_t$. It remains to check whether the guess is consistent with the theory $\mathcal{T}$. This can be done by querying $\mathcal{T}$ for K times, to see if there are indeed K constants satisfying $p_1, \ldots, p_s, q_1, \ldots, q_t$ or their negation, according to the guess, and satisfying at least one of the predicates $r_1, \ldots, r_\ell$. Each time the formula that is passed to $\mathcal{T}$ is polynomial in the size of $\mathcal{C}(u)$, and since the cost of querying $\mathcal{T}$ is assumed to be constant, the whole procedure runs in polynomial time on a nondeterministic Turing machine. □

3.2 Propositional Local Constraints

We further extend the expressive power of extended local constraints by adding disjunction to $\mathcal{L}_{ext}$; we thus obtain the language $\mathcal{L}_{prop}$ of *propositional local constraints* endowed with all the operators of propositional logics (including negation, since, e.g., $\exists^{\leq n}\mathsf{edge}(p)$ is equivalent to $\neg\exists^{\geq n+1}\mathsf{edge}(p)$).

The syntax and semantics of the additional constraint in $\mathcal{L}_{prop}$ (wrt those in $\mathcal{L}_{ext}$) is shown in Figure 4.

Syntax	Semantics
$v \rightarrowtail \gamma_1 \vee \gamma_2$	$(v \rightarrowtail \gamma_1) \vee (v \rightarrowtail \gamma_2)$

Figure 4: The language $\mathcal{L}_{prop}$

Example 4 Let us consider again the schema of Figure 1. We would like to enforce, in addition to the properties specified in Example 3, also the following property:

(6) each professor who teaches more than one course cannot have more than one student of whom she is a tutor.

This can be done by using the following constraint of $\mathcal{L}_{prop}$:

$$\begin{aligned}\mathcal{C}(u_4) \quad = \quad & \exists^{\leq 3}\mathsf{edge}(\text{tutor}) \wedge \exists^{\leq 2}\mathsf{edge}(\text{teaches}) \wedge \\ & (\exists^{\geq 1}\mathsf{edge}(\text{teaches}) \rightarrow \exists^{\leq 1}\mathsf{edge}(\text{tutor}))\end{aligned}$$

where $a \rightarrow b$ is a shorthand for $\neg a \vee b$.

We will see that there are significant advantages if we put the constraints of $\mathcal{L}_{prop}$ in *disjunctive normal form* (DNF), i.e., given a node u, we have

$$\mathcal{C}(u) = C_1 \vee \cdots \vee C_\nu$$

where each C_k, with $1 \leq k \leq \nu$, is a *clause* of the form

$$C_k \;=\; \exists^{\leq n_1}\mathsf{edge}(p_1) \wedge \cdots \wedge \exists^{\leq n_s}\mathsf{edge}(p_s) \wedge \exists^{\geq m_1}\mathsf{edge}(q_1) \wedge \cdots \wedge \exists^{\geq m_t}\mathsf{edge}(q_t)$$

Exploiting the fact that the constraints are in DNF, and that each clause of $\mathcal{C}(u)$ is a constraint of $\mathcal{L}_{ext}$, we can use the function `cons`, which checks the consistency of nodes labeled with constraints of $\mathcal{L}_{ext}$, to check the consistency of nodes of $\mathcal{L}_{prop}$-schemas. Notice that for $\mathcal{L}_{prop}$ we *cannot* remove the negative constraints as we did for $\mathcal{L}_{sl}$ and $\mathcal{L}_{ext}$. Instead, we can remove the negative constraints *clause by clause*, as each clause of $\mathcal{L}_{prop}$ is a constraint of $\mathcal{L}_{ext}$. So, let `rnec` be the function that removes the negative constraints from a clause.

The function which checks the consistency of an $\mathcal{L}_{prop}$-schema by means of local checks is the following, hence showing that $\mathcal{L}_{prop}$ is a local constraint language.

function `rinprop`(S: schema): schema
{ **repeat**
 if there is a node u in S with $\mathcal{C}(u) = C_1 \vee \cdots \vee C_\nu$,
  such that for each $K \in \{1, \ldots, \nu\}$, at least one of the following conditions is verified:
  - u is not reachable starting from root(S) through a direct path;
  - $t \geq 1$ and $\mathtt{cons}(u, \mathtt{rnec}(C_K)) = \mathit{false}$
   where $\mathtt{rnec}(S, C_K) = \exists^{\leq n_1}\mathsf{edge}(p_1) \wedge \cdots \wedge \exists^{\leq n_s}\mathsf{edge}(p_s) \wedge \exists^{\geq m_1}\mathsf{edge}(q_1) \wedge \cdots \wedge \exists^{\geq m_t}\mathsf{edge}(q_t)$,

 then remove from S: ⋆ the node u,
  ⋆ all the edges outgoing from u, and
  ⋆ all the edges incoming in u;

```
  until root(S) is removed from S or no new node has been removed from S;
  return S;
}
```

As for rinext, it can be shown that $\texttt{rinprop}(S)$ runs in time polynomial in the number of nodes of S, and in time exponential in the numeric indexes of the constraints labeling the nodes of S.

We can prove that also for $\mathcal{L}_{prop}$-schemas, the problem of checking node consistency is in NP. The proof is very similar to that of Theorem 3.

3.3 Subsumption

We now discuss a general technique to check schemas for subsumption which can be applied whenever the constraint language used in the schemas is local. Given two schemas S and S', for each pair of nodes u of S and u of S', we verify by means of a local check whether it is possible to build a fragment of database conforming to S_u and not conforming to $S'_{u'}$ [2] (as usual, we neglect the constraints over the successors of u and u'). If we can build such a fragment, we remove the pair of nodes. In order to make the local check, we call the function cons, giving to it a suitable combination of constraints as second argument. In particular, to check whether there exists a fragment respecting $\mathcal{C}(u)$ that violates an atomic constraint φ, which is part of $\mathcal{C}(u')$, we call $\texttt{cons}(u, \mathcal{C}(u) \wedge \neg\varphi)$. At the end, we answer that S is subsumed by S' if and only if the pair $(\mathrm{root}(S), \mathrm{root}(S'))$ is not removed.

With this technique we can define two functions subsext and subsprop, which check the subsumption between two $\mathcal{L}_{ext}$-schemas and two $\mathcal{L}_{prop}$-schemas respectively. The two functions are very similar; in particular, subsprop exploits the property that the $\mathcal{L}_{prop}$-constraints are in DNF, making *clause by clause* the same checks that subsext makes.

Theorem 4 *Checking subsumption between two $\mathcal{L}_{ext}$-schemas or two $\mathcal{L}_{prop}$-schemas S and S' can be done in time polynomial in the total number of nodes of S and S', and in time exponential in the numeric indexes of the constraints labeling the nodes of the schemas.*

Proof (sketch). It is possible to show that (possibly multiple copies of) the various fragments of databases conforming to S_u and not conforming to $S'_{u'}$, where (u, u') is a pair of nodes that has not been removed, can in any case be fixed together to form a single database. If at the end of the algorithm, $(\mathrm{root}(S), \mathrm{root}(S'))$ has not been removed, then this database conforms to S and does not conform to S', thus being a counterexample for the subsumption between S and S'. On the contrary, if $(\mathrm{root}(S), \mathrm{root}(S'))$ is not removed, we can deduce that $S \sqsubseteq S'$, since the relation from $\mathrm{Nodes}(S)$ to $\mathrm{Nodes}(S')$ that we have constructed can be used to extend every simulation from a database g to S to a simulation from g to S'.

The two functions subsext and subsprop make a number of calls to cons which is polynomial in the total number of nodes of the two schemas. Hence $\texttt{subsext}(S_1, S_2)$

[2] We remind that S_u denotes the schema obtained from S by considering u as the root.

and $\mathtt{subsext}(S_1, S_2)$ run in time polynomial in the total number of nodes of S_1 and S_2.

The only exponential dependence of the execution time of $\mathtt{subsext}(S_1, S_2)$ and $\mathtt{subsext}(S_1, S_2)$ is that from the numeric indexes of the constraints of the nodes of S_1 and S_2. This is due to the fact that the function cons builds formulae to be checked for validity, whose size is exponential in the numeric indexes. $\square$

4 Bidirectional Schemas

We investigate now the extension of the framework obtained by removing the asymmetry between outgoing and incoming edges, and considering both types of edges in the same way. The ability to refer to links in both directions substantially increases the expressive power of a representation formalism, and has been considered important in traditional database models [13, 12], in knowledge representation formalisms [16, 12], and in models and query languages for semistructured data [14, 22, 11]. Notably, XLink [22], the linking language of XML, allows one to express bidirectional and backward links.

We consider now *bidirectional databases* and *bidirectional schemas*, which are rooted graphs in which each node is reachable from the root by a semipath (instead of a path). To properly take into account both outgoing and incoming edges it is necessary to extend the notion of simulation between a bidirectional database g and a bidirectional schema $\mathcal{S}$, by adding to the conditions holding for a simulation (see Section 2) the following:

3. For each edge $v_f \xrightarrow{c} v$ in g, there exists an edge $u_f \xrightarrow{\pi} u$ in S such that $\mathcal{T} \models \pi(c)$ and $v_f \trianglelefteq u_f$.

The notions of conformance, consistency, and subsumption can be extended straightforwardly to bidirectional databases and schemas, by considering bidirectional simulations instead of (unidirectional) simulations. For the constraint languages $\mathcal{L}_{sl}$, $\mathcal{L}_{ext}$, and $\mathcal{L}_{prop}$, all the techniques and results we have found for inference on unidirectional schemas can be extended to bidirectional databases and schemas (and hence bidirectional simulations).

In the context of bidirectional databases and schemas, it is natural to allow one to impose constraints also on the incoming edges of a node. We consider the constraint language $\mathcal{L}_{bid}$, which extends $\mathcal{L}_{sl}$ with the constraints on incoming edges shown in Figure 5.

Syntax	**Semantics**
$u \rightarrowtail \exists\mathsf{edge}^-(p)$	$\exists u_p \xrightarrow{c} u \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)$
$u \rightarrowtail \nexists\mathsf{edge}^-(p)$	$\neg\exists u_p \xrightarrow{c} u \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)$
$u \rightarrowtail \exists^{\leq 1}\mathsf{edge}^-(p)$	$\#\{u_p \xrightarrow{c} u \in \mathrm{Edges}(g) \mid \mathcal{T} \models p(c)\} \leq 1$

Figure 5: The language $\mathcal{L}_{bid}$

Example 5 Let us consider once more the schema of Figure 1. We want to enforce the following properties:

(1) each course must be taught by exactly one professor;

(2) each student can have at most one tutor.

We can do that by using the constraints of $\mathcal{L}_{bid}$ as follows

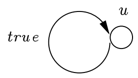
$$\begin{aligned}\mathcal{C}(u_3) &= \exists^{\leq 1}\mathsf{edge}^-(\text{teaches}) \wedge \exists\mathsf{edge}^-(\text{teaches})\\ \mathcal{C}(u_5) &= \exists^{\leq 1}\mathsf{edge}^-(\text{tutor})\end{aligned}$$

$$\begin{aligned}\mathcal{C}(u_1) &= \exists\mathsf{edge}(p)\\ \mathcal{C}(u_2) &= \exists\mathsf{edge}(q)\\ \mathcal{C}(u_3) &= \exists\mathsf{edge}^-(p) \wedge\\ &\quad\ \exists^{\leq 1}\mathsf{edge}^-(p \vee q)\end{aligned}$$

(a) Schema S (b) Fragments of databases

Figure 6: Consistent bidirectional schema

We show that for $\mathcal{L}_{bid}$ we lose the ability to check the consistency of nodes by means of local checks. Indeed, in the presence of $\mathcal{L}_{bid}$-constraints the consistency of a node can be affected by the constraints over other nodes. For example, Figure 6 shows a schema S and the fragments locally conforming to its nodes (we suppose that in $\mathcal{T}$ the formulae $p(c_p)$, $q(c_q)$, and $\neg\exists x(p(x) \wedge q(x))$ are valid); it is evident that the fragments cannot be fixed together to form a database conforming to S. This shows that $\mathcal{L}_{sl}$ is not local. Note that in the example, the schema is consistent, and in particular, a database conforming to it is the one consisting of two nodes conforming to u_1 and u_3 respectively, and connected by an edge satisfying p. However, by using local checks it is not possible to identify the fragments that are to be used to build a conforming database, if it exists. Moreover, there may be many such choices in general.



$\mathcal{C}(u) = \exists\mathsf{edge}(p) \wedge \exists\mathsf{edge}(\neg p) \wedge \exists^{\leq 1}\mathsf{edge}^-(true)$

Figure 7: Schema violating the FMP

A further consequence of the expressiveness of $\mathcal{L}_{bid}$ is that for $\mathcal{L}_{bid}$-schemas the *finite model property* does not hold [15, 7]. Indeed, every database conforming to the $\mathcal{L}_{bid}$-schema shown in Figure 7 must have an infinite number of nodes. This is due to the fact that in such a database, every node simulating u requires the existence of other two[3] *new* nodes, both simulating u.

The lack of the finite model property is typical of representation formalisms that can express functionality on links in both directions, and in general makes it necessary to adopt different techniques for the cases where one wants to reason wrt finite databases only, or one wants to consider arbitrary (possibly infinite) databases [15, 7].

At the moment a lower bound to the complexity of checking consistency (and subsumption) of $\mathcal{L}_{bid}$-schemas is not known, though the lack of locality is a strong indication of intractability. Another open problem is that of finding algorithms to check consistency and subsumption of $\mathcal{L}_{bid}$-schemas.

5 Conclusions and Discussion

We have studied the relationship between various extensions of constraint languages for semistructured data schemas and the basic property of locality, which allows one

[3]Except for the root, which requires the existence of only one new node.

to check subsumption between schemas in polynomial time in the number of nodes of the schemas. We have shown that locality holds when both numeric constraints and disjunction are added to a simple constraint language. On the other hand, locality is lost when we consider constraints both on outgoing and incoming edges of databases.

In all cases where locality holds, the algorithms presented rely only on the ability to do a local check of consistency of a node. Hence we have an algorithmic framework which is completely modular with respect to the function which performs the local consistency check.

We would like to point out that, while the results in this paper have been established in the formal framework of the BDFS data model [6] based on the notion of simulation, all considerations relative to locality hold also for other data models, such as OEM [2].

Formalisms for specifying relationships between data in a flexible way have been investigated extensively in the context of *Description Logics* (DLs) studied in knowledge representation. The results established here for semistructured data models can be recast in terms of DLs (assuming the underlying theory is expressible in a DL). To correctly capture the notion of simulation one must resort to a greatest-fixpoint semantics [24, 4, 8]. Moreover, disjunction is implicit in the BDFS model, and due to the necessity to reify edges, qualified existential quantification and qualified functionality restrictions (resp., qualified number restrictions) are needed to encode the existence and functionality constraints (resp., numeric constraints) of a semistructured data schema. Such a combination of constructs interpreted with a greatest-fixpoint semantics has not been considered before in DLs, with the exception of very expressive DLs with fixpoint constructs [26, 17, 10], which however have a much higher computational complexity. Hence, the results established here represent a new contribution also wrt to reasoning algorithms for DLs.

References

[1] Serge Abiteboul. Querying semi-structured data. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, pages 1–18, 1997.

[2] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[3] J. R. Abrial. Data semantics. In J. W. Klimbie and K. L. Koffeman, editors, *Data Base Management*, pages 1–59. North-Holland Publ. Co., Amsterdam, 1974.

[4] Franz Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18:175–219, 1996.

[5] Alexander Borgida, Ronald J. Brachman, Deborah L. McGuinness, and Lori Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 59–67, 1989.

[6] Peter Buneman, Susan Davidson, Mary F. Fernandez, and Dan Suciu. Adding structure to unstructured data. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, pages 336–350, 1997.

[7] Diego Calvanese. Finite model reasoning in description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 292–303. Morgan Kaufmann, Los Altos, 1996.

[8] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. What can knowledge representation do for semi-structured data? In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 205–210, 1998.

[9] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Modeling and querying semi-structured data. *Network and Information Systems*, 2(2), 1999.

[10] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 84–89, 1999.

[11] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS 2000)*, pages 58–66, 2000.

[12] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. Description logics for conceptual data modeling. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 229–264. Kluwer Academic Publishers, 1998.

[13] Roderick G. G Cattell and Douglas K. Barry, editors. *The Object Database Standard: ODMG 2.0*. Morgan Kaufmann, Los Altos, 1997.

[14] J. Clark and S. Deach. Extensible Stylesheet Language (XSL). Technical report, World Wide Web Consortium, 1999. Available at http://www.w3.org/TR/WD-xsl.

[15] S. S. Cosmadakis, P. C. Kanellakis, and M. Vardi. Polynomial-time implication problems for unary inclusion dependencies. *J. of the ACM*, 37(1):15–46, January 1990.

[16] Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212. AAAI Press/The MIT Press, 1994.

[17] Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th European Conf. on Artificial Intelligence (ECAI'94)*, pages 411–415, 1994.

[18] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[19] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy, and Dan Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 414–425, 1998.

[20] Daniela Florescu, Alon Levy, and Alberto Mendelzon. Database techniques for the World-Wide Web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.

[21] John Grant and Jack Minker. Inferences for numerical dependencies. *Theoretical Computer Science*, 41:271–287, 1985.

[22] Eve Maler and Steve DeRose. XML Linking Language (XLink) – W3C working draft 03-march-1998. Technical report, World Wide Web Consortium, 1998. Available at http://www.w3.org/TR/1998/WD-xlink-19980303.

[23] Alberto Mendelzon, George A. Mihaila, and Tova Milo. Querying the World Wide Web. *Int. J. on Digital Libraries*, 1(1):54–67, 1997.

[24] Bernhard Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.

[25] D. Quass, A. Rajaraman, I. Sagiv, J. Ullman, and J. Widom. Querying semistructured heterogeneous information. In *Proc. of the 4th Int. Conf. on Deductive and Object-Oriented Databases (DOOD'95)*, pages 319–344. Springer-Verlag, 1995.

[26] Klaus Schild. Terminological cycles and the propositional μ-calculus. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of the 4th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 509–520, Bonn (Germany), 1994. Morgan Kaufmann, Los Altos.

[27] Bernhard Thalheim. Fundamentals of cardinality constraints. In G. Pernoul and A. M. Tjoa, editors, *Proc. of the 11th Int. Conf. on the Entity-Relationship Approach (ER'92)*, pages 7–23. Springer-Verlag, 1992.