Semi-structured Data with Constraints and Incomplete Information

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. In this paper, we base our work on BDFS, which is a formal and elegant model for semi-structured data [Buneman *et al.*, 1997] where schemas are graphs whose edges are labeled with formulae of a theory $\mathcal{T}$. We extend BDFS with the possibility of expressing constraints and dealing with incomplete information. In particular, we consider different types of constraints, and discuss how the expressive power of the constraint language may influence the complexity of checking subsumption between schemas. We then set up a framework for defining BDFS schemas under the assumption that the theory $\mathcal{T}$ is not complete. Finally, we propose a new semi-structured data model, which extends BDFS with both constraints and incomplete theories. We present a technique for checking subsumption in a setting where both the constraints and the theory are expressed in a very powerful language.

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 biological databases, digital libraries, and data integration [Abiteboul, 1997; Buneman *et al.*, 1997; Christophides *et al.*, 1994; Mendelzon *et al.*, 1997; Quass *et al.*, 1995].

Following [Abiteboul, 1997], we define semi-structured data as data that is neither raw, nor strictly typed as in conventional database systems. OEM (Object Exchange Model) [Abiteboul *et al.*, 1997], and BDFS (Basic Data model For Semi-structured data) [Buneman *et al.*, 1997] are recent proposals of models for semi-structured data. They represent data as graphs with labeled edges, where

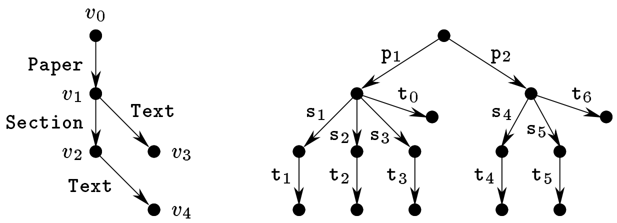


Figure 1: Schema for papers divided in sections and a conforming database

information on both the values and the schema of data are kept.

In particular, BDFS is a formal and elegant data model, where the labels of edges in the schemas are formulae of a certain theory $\mathcal{T}$, and the notion of a database DB conforming 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 DB and those of $\mathcal{S}$ such that, whenever there is an edge labeled a in DB, there is a corresponding edge in $\mathcal{S}$ 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 which models sets of web pages representing papers possibly structured in sections, each with an associated text, and a database that conforms to it. We assume that the theory $\mathcal{T}$ implies that papers, sections, and texts are disjoint sets. ■

In [Buneman *et al.*, 1997], the authors point out that, 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, and they present algorithms and complexity analysis for checking subsumption in BDFS. They also indicate that it would be interesting to extend the model with several types of constraints. Indeed, the analysis in [Buneman *et al.*, 1997] is carried out under the following assumptions:

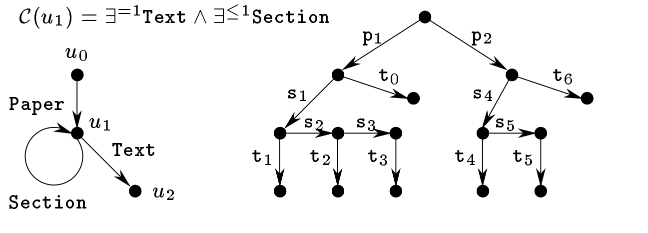


Figure 2: Schema for papers divided in ordered sections and a conforming database

- 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 constraints, 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.
- Since the labels of the edges of the graph are formulae of a complete theory $\mathcal{T}$, the possibility of dealing with incomplete information on databases is ruled out. In other words, it is assumed that, for every database, we have complete information on the objects labeling its edges. This implies, for examples, that in a case where the schema represents, say, home pages of department faculties, and database edges represent faculties, we have complete information on the faculties.

The goal of this paper is to extend the framework of [Buneman *et al.*, 1997] in order to overcome the above limitations.

Example 2 The schema in Figure 1 presents several modeling problems. 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. This calls for adding constraints on nodes v_1 and v_2 to impose restrictions on the number of outgoing edges ($\mathcal{C}(v_1) = \mathcal{C}(v_2) = \exists^{=1}\mathtt{Text}$). Moreover, the order of the sections is not represented in the schema. A possible restructuring of the schema to represent the sequence of sections is shown in Figure 2, where it is essential to impose that each section is followed by at most one other section ($\mathcal{C}(u_1) = \exists^{=1}\mathtt{Text} \wedge \exists^{\leq 1}\mathtt{Section}$). In addition, we would like to impose by means of suitable constraints that such a sequence is well-founded (finite). Finally, we would like to be able to check conformance of papers, even if we do not have complete information on them, e.g. we do not know the language of the text. ∎

Specifically, we present 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 database DB conforming to $\mathcal{S}$, must be satisfied by every node of DB simulating u. We consider different types of constraints, and we discuss how the expressive power of the constraint language may influence the complexity of subsumption checking. In particular, we show that adding edge-existence constraints to BDFS does not increase the complexity of the problem.
- We set up a framework for defining BDFS schemas under the assumption that the theory $\mathcal{T}$ is not complete. We discuss several possibilities of defining subsumption in this new setting, and we show how the incompleteness of $\mathcal{T}$ may influence the complexity of subsumption checking.
- We propose a new semi-structured data model, which extends BDFS with both constraints and incomplete information. Both the constraints and the theory are expressed in a very powerful language, called $\mu\mathcal{ALCQ}$ [De Giacomo and Lenzerini, 1997], which is a decidable fragment of first order logic with fixpoints. Fixpoints are used to impose complex conditions on the schema, such as well-foundedness of subgraphs. We present a technique for checking subsumption in the new data model, showing that the problem is decidable in exponential time.

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, and in Section 4 we study BDFS schemas with incomplete information. In Section 5 we describe our overall framework for specifying schemas with both constraints and incomplete information, and present the results on reasoning about such extended schemas. Finally, Section 6 concludes the paper. Proof sketches appear in the Appendix.

2 Preliminaries

In this section, we describe the basic characteristics of two formalisms that will be used in this paper, namely the BDFS model for semi-structured data, and the description logic $\mu\mathcal{ALCQ}$.

2.1 The BDFS Data Model

The formalism proposed in [Buneman *et al.*, 1997] for specifying semi-structured data schemas, which we call BDFS, is the basis of our investigation. The formalism is appropriate for an edge-labeled graph model of 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 the first order formulae which are true for the elements of $\mathcal{U}$, and it is decidable to check whether a formula p in $\mathcal{L}_\mathcal{T}$ is true in $\mathcal{T}$ (in notation, $\mathcal{T} \models p$).

Definition 3 *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 $u \xrightarrow{a} v$.

Definition 4 *A $\mathcal{T}$-database DB* conforms to a BDFS *$\mathcal{T}$-schema $\mathcal{S}$, in notation $DB \preceq \mathcal{S}$, if there exists a* simulation *from DB to $\mathcal{S}$, i.e. a binary relation $\unlhd$ from the nodes of DB to those of $\mathcal{S}$ satisfying: (1) $root(DB) \unlhd root(\mathcal{S})$, (2) $u \unlhd u'$ implies that for each edge $u \xrightarrow{a} v$ in DB, there exists an edge $u' \xrightarrow{p} v'$ in $\mathcal{S}$ such that $\mathcal{T} \models p(a)$, and $v \unlhd v'$.*

Definition 5 *If $\mathcal{S}$ and $\mathcal{S}'$ are two* BDFS *$\mathcal{T}$-schemas, we say that $\mathcal{S}'$* subsumes *$\mathcal{S}$, in notation $\mathcal{S} \sqsubseteq \mathcal{S}'$, if for every $\mathcal{T}$-database DB, $DB \preceq \mathcal{S}$ implies $DB \preceq \mathcal{S}'$.*

In [Buneman *et al.*, 1997], an algorithm is presented for checking subsumption (and 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 $t_{\mathcal{T}}(x)$ is the time needed to check whether a formula of size x is valid in $\mathcal{T}$, and m is the size of the two schemas. In the setting of [Buneman *et al.*, 1997] it is meaningful not to consider $\mathcal{T}$ to be part of the input of the 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 [De Giacomo and Lenzerini, 1997], where a correspondence was shown with a well-known logic of programs, called *modal mu-calculus* [Kozen, 1983; Streett and 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 (see e.g. [Abiteboul *et al.*, 1995]). We make use of the standard first-order notions of scope, bound and free occurrences of variables, closed formulae, etc., treating μ and ν 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: $C_1 \sqcup C_2$ 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* ρ on an interpretation $\mathcal{I}$ is a mapping from variables to subsets of $\Delta^{\mathcal{I}}$. Given a valuation ρ, we denote by $\rho[X/\mathcal{E}]$ the valuation identical to ρ except for the fact that $\rho[X/\mathcal{E}](X) = \mathcal{E}$.

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

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

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 $\subseteq$. 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 $\mu\mathcal{ALCQ}$ *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 Γ, if $\mathcal{I}$ satisfies all axioms in Γ. A closed concept C is *satisfiable* in a knowledge base Γ if there exists a model $\mathcal{I}$ of Γ such that $C^{\mathcal{I}} \neq \emptyset$.

Theorem 6 ([De Giacomo and Lenzerini, 1997]) *Satisfiability of closed $\mu\mathcal{ALCQ}$ concepts in $\mu\mathcal{ALCQ}$ knowledge bases is an EXPTIME-complete problem.*

3 Schemas with Constraints

In this section, 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 DB conforming to $\mathcal{S}$, must be satisfied by every node of DB 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. In the rest of this section, $\mathcal{T}$ denotes a complete theory, as defined in Section 2.1.

```
function rin(S: T-schema): T-schema;
{ (G',C') ← rnec(S);
  repeat if there is a node u in G' with C'(u) = ∃p_1 ∧ ··· ∧ ∃p_r ∧ ∃^{≤1}f_1 ∧ ··· ∧ ∃^{≤1}f_s,
            that satisfies one of the following conditions:
            (1) u is not connected to root(G') in G'
            (2) r ≥ 1 and u has no outgoing edge in G'
            (3) r ≥ 1, u -q_1-> v_1, ..., u -q_m-> v_m, with m ≥ 1, are all outgoing edges of u in G', and
                T ⊨ ¬∃x_1 ··· ∃x_r(⋀_{1≤i≤r}(p_i(x_i) ∧ ⋁_{1≤j≤n} q_j(x_i)) ∧
                                   ⋀_{1≤k≤s} ⋀_{1≤i<j≤r}((f_k(x_i) ∧ f_k(x_j)) ⊃ x_i = x_j))
        then remove from G' the node u and all edges from and to u;
             remove from C' the pair (u, C'(u));
  until root(G') has been removed from G' or no new node has been removed from G';
  return (G',C')
}
```

Figure 3: Function *rin* that removes inconsistent nodes

Definition 7 *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 8 *A* $\mathcal{T}$*-database* DB conforms to *a* $\mathcal{T}$*-schema with* $\mathcal{L}_{\mathcal{C}}$*-constraints* $\mathcal{S} = (\mathcal{G}, \mathcal{C})$*, in notation* $DB \preceq \mathcal{S}$*, if there exists a binary relation* $\trianglelefteq$ *from the nodes of* DB *to those of* $\mathcal{G}$ *satisfying: (1)* $root(DB) \trianglelefteq root(\mathcal{G})$*, (2)* $u \trianglelefteq u'$ *implies that (2.1)* u *satisfies* $\mathcal{C}(u')$*, and (2.2) for each edge* $u \xrightarrow{a} v$ *in* DB*, there exists an edge* $u' \xrightarrow{p} v'$ *in* $\mathcal{S}$ *such that* $\mathcal{T} \models p(a)$*, 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 9 *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 a 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.

3.1 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 (γ, γ_1 and γ_2 denote constraints, and p denotes a formula of $\mathcal{T}$):

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

We use $\exists^{=1}p$ as an abbreviation for $\exists p \wedge \exists^{\leq 1}p$. 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 DB a $\mathcal{T}$-database. Then a node u of DB *satisfies* a constraint γ, in notation $u \models_c \gamma$, if the following conditions are satisfied:

$u \models_c \top$	always	
$u \models_c \exists p$	iff	$\exists u \xrightarrow{a} v \in Edges(DB).\ \mathcal{T} \models p(a)$
$u \models_c \neg\exists p$	iff	$\forall u \xrightarrow{a} v \in Edges(DB).\ \mathcal{T} \models \neg p(a)$
$u \models_c \exists^{\leq 1} p$	iff	$\#\{u \xrightarrow{a} v \in Edges(DB) \mid \mathcal{T} \models p(a)\} \leq 1$
$u \models_c \gamma_1 \wedge \gamma_2$	iff	$(u \models_c \gamma_1) \ \wedge \ (u \models_c \gamma_2)$

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

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 $\neg\exists p$. In fact, given a $\mathcal{T}$-schema with $\mathcal{L}_l$-constraints $\mathcal{S} = (\mathcal{G}, \mathcal{C})$, we can obtain an equivalent $\mathcal{T}$-schema $\mathcal{S}' = (\mathcal{G}', \mathcal{C})$ not containing constraints of the form $\neg\exists p$ and with the same set of nodes as $\mathcal{S}$ as follows. For every node u in $\mathcal{S}$ with $\mathcal{C}(u) = \exists p_1 \wedge \cdots \wedge \exists p_r \wedge \neg\exists n_1 \wedge \cdots \wedge \neg\exists n_s \wedge \exists^{\leq 1} f_1 \wedge \cdots \wedge \exists^{\leq 1} f_t$ and outgoing edges $u \xrightarrow{q_1} v_1, \ldots, u \xrightarrow{q_k} v_k$, we define $\mathcal{C}'(u) = \exists p_1 \wedge \cdots \wedge \exists p_r \wedge \exists^{\leq 1} f_1 \wedge \cdots \wedge \exists^{\leq 1} f_t$, 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 10 *Let* $\mathcal{S}$ *be a* $\mathcal{T}$*-schema with* $\mathcal{L}_l$*-constraints and* $\mathcal{S}'$ *the* $\mathcal{T}$*-schema* $\mathcal{S}'$ *obtained from* $\mathcal{S}$ *by removing the constraints of the form* $\neg\exists n$ *as described above. Then* $|\mathcal{S}'|$ *is polynomial in* $|\mathcal{S}|$ *and* $\mathcal{S}'$ *is equivalent to* $\mathcal{S}$*.*

We present a method for checking consistency, based on the function *rin* defined in Figure 3, whose role is to first remove the non-existence constraints by calling the function *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

function $\mathit{subs}(S_0\colon \mathcal{T}\text{-schema}, \mathcal{S}_0'\colon \mathcal{T}\text{-schema})$: boolean
$\{\ (\mathcal{G},\mathcal{C}) \leftarrow \mathit{rin}(S_0);$
$(\mathcal{G}',\mathcal{C}') \leftarrow \mathit{rin}(S_0');$
if $\mathcal{G}$ does not contain $\mathit{root}(\mathcal{S}_0)$ **then return** true;
if $\mathcal{G}'$ does not contain $\mathit{root}(\mathcal{S}_0')$ **then return** false;
$R \leftarrow \{(u,u') \mid u \in \mathit{Nodes}(\mathcal{G}), u' \in \mathit{Nodes}(\mathcal{G}')\}$;
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 $\mathcal{G}$,
$\mathcal{C}(u) = \exists p_1 \wedge \cdots \wedge \exists p_r \wedge \exists^{\leq 1} f_1 \wedge \cdots \wedge \exists^{\leq 1} f_s$, $\quad \mathcal{C}'(u') = \exists p_1' \wedge \cdots \wedge \exists p_{r'}' \wedge \exists^{\leq 1} f_1' \wedge \cdots \wedge \exists^{\leq 1} f_{s'}'$,
that satisfies one of the following conditions:
(1) there is $i \in \{1, \ldots, n\}$ such that
$$\begin{aligned}\mathcal{T} \models \exists x_0 \exists x_1 \cdots \exists x_r (q_i(x_0) \ \wedge \ & \textstyle\bigwedge_{1\leq j\leq m} \neg q_j'(x_0) \ \wedge \\ & \textstyle\bigwedge_{1\leq j\leq r} (p_j(x_j) \wedge \bigvee_{1\leq k\leq n} q_k(x_j)) \ \wedge \\ & \textstyle\bigwedge_{1\leq \ell\leq s} \bigwedge_{0\leq j<k\leq r} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))\end{aligned}$$
where $u' \xrightarrow{q_j'} v_j'$, $j \in \{1, \ldots, m\}$ are all edges from u' in $\mathcal{G}'$ 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
$$\begin{aligned}\mathcal{T} \models \exists x_1 \cdots \exists x_r (& \textstyle\bigwedge_{1\leq j\leq r} \neg p_i'(x_j) \ \wedge \\ & \textstyle\bigwedge_{1\leq j\leq r} (p_j(x_j) \wedge \bigvee_{1\leq k\leq n} q_k(x_j)) \ \wedge \\ & \textstyle\bigwedge_{1\leq \ell\leq s} \bigwedge_{1\leq j<k\leq r} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))\end{aligned}$$
(3) there is $i \in \{1, \ldots, s'\}$ such that
$$\begin{aligned}\mathcal{T} \models \exists x_1 \cdots \exists x_r \exists x_{r+1} \exists x_{r+2} (& f_i'(x_{r+1}) \wedge f_i'(x_{r+2}) \wedge x_{r+1} \neq x_{r+2} \ \wedge \\ & \textstyle\bigwedge_{1\leq j\leq r} (p_j(x_j) \wedge \bigvee_{1\leq k\leq n} q_k(x_j)) \ \wedge \\ & \textstyle\bigwedge_{1\leq \ell\leq s} \bigwedge_{1\leq j<k\leq r+2} ((f_\ell(x_j) \wedge f_\ell(x_k)) \supset x_j = x_k))\end{aligned}$$
then remove (u,u') from R;
until no new pair has been removed from R;
return $(\mathit{root}(\mathcal{G}), \mathit{root}(\mathcal{G}')) \in R$
$\}$

Figure 4: Function *subs* that verifies subsumption of schemas with local constraints

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 11 *If $\mathcal{S} = (\mathcal{G},\mathcal{C})$ is a $\mathcal{T}$-schema with $\mathcal{L}_l$-constraints, then $\mathcal{S}$ is consistent if and only if $\mathit{rin}(\mathcal{S})$ contains $\mathit{root}(\mathcal{G})$. Moreover, $\mathit{rin}(\mathcal{S})$ runs in time polynomial in $|\mathcal{S}|$.*

We now turn our attention to the method for checking subsumption of schemas with constraints, which is also a method for checking conformance of databases to schemas. The method is based on the function *subs* defined in Figure 4.

Note that *subs* is an extension of the algorithm in [Buneman *et al.*, 1997]. 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 8 may exist. Intuitively, the algorithm checks locally for the pair (u,u'), whether it is possible to construct a database DB 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 $\mathcal{S}$. Condition (2) checks whether DB 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 12 *If $\mathcal{S}_1$ and $\mathcal{S}_2$ are $\mathcal{T}$-schemas with $\mathcal{L}_l$-constraints, then $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ if and only if $\mathit{subs}(\mathcal{S}_1, \mathcal{S}_2)$ returns true. Moreover, $\mathit{subs}(\mathcal{S}_1, \mathcal{S}_2)$ runs in time polynomial in $|\mathcal{S}_1| + |\mathcal{S}_2|$.*

The above result, together with Lemma 10, shows that adding conjunctions of local constraints to BDFS does not increase the complexity of subsumption.

Example 13 Figure 5 shows two extensions to the schema in Figure 2, in which nesting of sections is considered[1]. 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 2. In fact, the function *subs* in Figure 4 constructs a relation R between the nodes of schema (a) and those of schema (b) such that $(u_0', u_0'') \in R$.

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

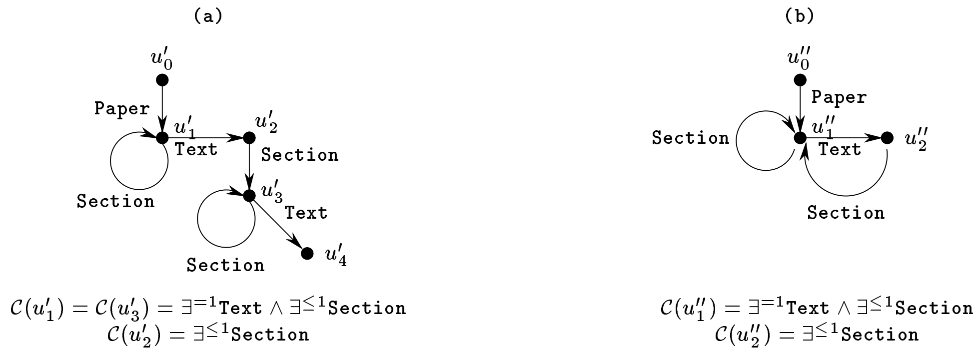


Figure 5: Schemas for papers divided in ordered nested sections

Observe that, if we replace $\exists^{=1}\texttt{Text}$ by $\exists^{\leq 1}\texttt{Text}$ in $\mathcal{C}(u_3')$ (thus 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). ∎

3.2 Non-Local Constraints

In this section we consider a simple constraint language 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 this simple case consistency (and thus subsumption) of $\mathcal{T}$-schemas becomes intractable due to the non-local constraints.

The formulae of the constraint language $\mathcal{L}_{\mathcal{ALE}}$, which is inspired by the DL $\mathcal{ALE}$ [Donini *et al.*, 1992], have the following syntax:

$$\gamma \ ::= \ \top \ | \ \exists p \uparrow \gamma \ | \ \forall p \uparrow \gamma \ | \ \gamma_1 \wedge \gamma_2$$

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

$$u \models_c \exists p \uparrow \gamma \quad \text{iff} \quad \exists u \xrightarrow{a} v \in \mathit{Edges}(DB).\, (\mathcal{T} \models p(a) \wedge v \models_c \gamma)$$
$$u \models_c \forall p \uparrow \gamma \quad \text{iff} \quad \forall u \xrightarrow{a} v \in \mathit{Edges}(DB).\, (\mathcal{T} \models p(a) \supset v \models_c \gamma)$$

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 14 *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**.

Theorem 14 shows that consistency checking stays 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 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 the case of local constraints, where the aspects related to the topology enforced by the constraints can be embedded in an appropriate formula of $\mathcal{T}$.

4 Schemas with Incomplete Theories

In this section, we address the problem of extending the BDFS data model to the case where the theory $\mathcal{T}$ is not necessarily complete. Thus, in the rest of this section, $\mathcal{T}$ denotes a theory which is not necessarily complete, and we assume that $\mathcal{T}$ is presented as a finite set of axioms in a language $\mathcal{L}_{\mathcal{T}}$. In this new setting, we define the notions of conformance and subsumption as follows.

Definition 15 *Let $\mathcal{M}$ be a model of $\mathcal{T}$. A $\mathcal{T}$-database DB* $\mathcal{M}$-conforms *to a* BDFS *$\mathcal{T}$-schema $\mathcal{S}$, in notation $DB \preceq_{\mathcal{M}} \mathcal{S}$, if there exists an $\mathcal{M}$-simulation from DB to $\mathcal{S}$, i.e. a binary relation $\trianglelefteq$ from the nodes of DB to those of $\mathcal{S}$ satisfying: (1) $root(DB) \trianglelefteq root(\mathcal{S})$, (2) $u \trianglelefteq u'$ implies that for each edge $u \xrightarrow{a} v$ of DB, there exists an edge $u' \xrightarrow{p} v'$ in $\mathcal{S}$ such that $\mathcal{M} \models p(a)$ and $v \trianglelefteq v'$.*

Definition 16 *Let $\mathcal{S}$, $\mathcal{S}'$ be two* BDFS *$\mathcal{T}$-schemas. We say that $\mathcal{S}'$* subsumes *$\mathcal{S}$, in notation $\mathcal{S} \sqsubseteq \mathcal{S}'$, if for every $\mathcal{T}$-database DB and every model $\mathcal{M}$ of $\mathcal{T}$, $DB \preceq_{\mathcal{M}} \mathcal{S}$ implies $DB \preceq_{\mathcal{M}} \mathcal{S}'$.*

One can easily verify that, if $\mathcal{T}$ is complete, then the two definitions are equivalent to those presented in Section 2.

It is interesting to compare Definition 16 with the following alternative definitions of subsumption:

1. $\mathcal{S}'$ subsumes $\mathcal{S}$ if for every $\mathcal{T}$-database DB, $DB \preceq_1 \mathcal{S}$ implies $DB \preceq_1 \mathcal{S}'$, where $DB \preceq_1 \mathcal{S}$ means that for every model $\mathcal{M}$ of $\mathcal{T}$, $DB \preceq_{\mathcal{M}} \mathcal{S}$.
2. $\mathcal{S}'$ subsumes $\mathcal{S}$ if for every $\mathcal{T}$-database DB, $DB \preceq_2 \mathcal{S}$ implies $DB \preceq_2 \mathcal{S}'$, where $DB \preceq_2 \mathcal{S}$ means that there exists a binary relation $\trianglelefteq$ from the nodes of DB to those of $\mathcal{S}$ satisfying: (1) $root(DB) \trianglelefteq root(\mathcal{S})$, (2) $u \trianglelefteq u'$ implies that for each edge $u \xrightarrow{a} v$ of DB, there exists an edge $u' \xrightarrow{p} v'$ in $\mathcal{S}$ such that $\mathcal{T} \models p(a)$, and $v \trianglelefteq v'$.

To see why we did not choose any of the above alternative definitions, consider the theory $\mathcal{T}$ with axioms $\{\forall x.(p(x) \to q(x))\}$, and the $\mathcal{T}$-schema $\mathcal{S}_0$ containing only the edge $u \xrightarrow{p} v$. Since for every constant c, $\mathcal{T} \not\models p(c)$, it is easy to see that, in both case (1) and case (2), the only $\mathcal{T}$-database conforming to $\mathcal{S}_0$ is the one with the root and no edges. It follows that, in both cases, $\mathcal{S}_0$ is subsumed by every consistent $\mathcal{T}$-schema, and in particular by the schema $\mathcal{S}'_0$ containing only the edge $u \xrightarrow{\neg p} v$, which is counterintuitive. Note that, on the contrary, with our definition, $\mathcal{S}_0$ is subsumed, for example, by a schema containing only the edge $u \xrightarrow{q} v$, but not by the schema $\mathcal{S}'_0$.

We now discuss how the presence of incomplete theories may influence the computational complexity of subsumption checking. To this end, we show that subsumption checking is at least as hard as validity in propositional logic, even for very simple theories $\mathcal{T}$ (i.e. the axioms of the theory are expressed in a very simple language) and $\mathcal{T}$-schemas.

Let f be a 3-DNF propositional formula of the form $(L_{11} \wedge L_{12} \wedge L_{13}) \vee \cdots \vee (L_{n1} \wedge L_{n2} \wedge L_{n3})$, and let $p_1, \ldots, p_h$ be all letters appearing in f. Let $\mathcal{T}$ be a theory including a constant a, the predicate symbols $\{q, p_1, \ldots, p_h\}$ (all unary), and with axioms $\{\forall x (q(x) \equiv (x = a))\}$. Finally, let $\mathcal{S}_1$ and $\mathcal{S}_2$ be the two $\mathcal{T}$-schemas defined as follows:

- $\mathcal{S}_1$ is $u_0 \xrightarrow{q} u_1 \xrightarrow{q} u_2 \xrightarrow{q} u_3$ and has root u_o.
- $\mathcal{S}_2$ is the set of chains $u'_0 \xrightarrow{L_{i1}} u_{i_1} \xrightarrow{L_{i2}} u_{i_2} \xrightarrow{L_{i3}} u_{i_3}$, for $i \in \{1, \ldots, n\}$, and has root u'_0.

Notice that there is an obvious correspondence between truth assignments of f and models of $\mathcal{T}$ ($p_i(a)$ is true in a model $\mathcal{M}$ of $\mathcal{T}$ iff p_i is true in the corresponding truth assignment). In particular, (1) for every truth assignment $\mathcal{M}$ that satisfies f, the corresponding model $\mathcal{M}'$ of $\mathcal{T}$ is such that, for every $\mathcal{T}$-database DB, $DB \preceq_{\mathcal{M}'} \mathcal{S}_1$ implies $DB \preceq_{\mathcal{M}'} \mathcal{S}_2$; (2) for every truth assignment $\mathcal{M}$ that does not satisfy f, the corresponding model $\mathcal{M}'$ of $\mathcal{T}$ is such that there is a $\mathcal{T}$-database DB such that $DB \preceq_{\mathcal{M}'} \mathcal{S}_1$ and $DB \not\preceq_{\mathcal{M}'} \mathcal{S}_2$. It follows that $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ if and only if f is valid.

One can verify that, if $\mathcal{T}$ is complete, subsumption between schemas of the form of $\mathcal{S}_1$ and $\mathcal{S}_2$ can be checked in polynomial time with respect to the size of the two schemas and the theory. On the contrary, the above considerations show that, in the presence of incomplete theories, checking subsumption between two schemas may require $O(m^{O(1)} \cdot 2^m)$, where m is the number of edges of the two schemas. This means that the assumption of considering $t_{\mathcal{T}}(m)$ to be a constant may not be reasonable in this new setting, because it would hide a cost which is exponential with respect to the size of the schemas.

Example 17 Consider the schema in Figure 6 (ignoring the constraints for the moment), which models papers which are either in Italian or in English, with the condition that all the texts of English papers is in English, while no analogous condition holds for Italian ones.

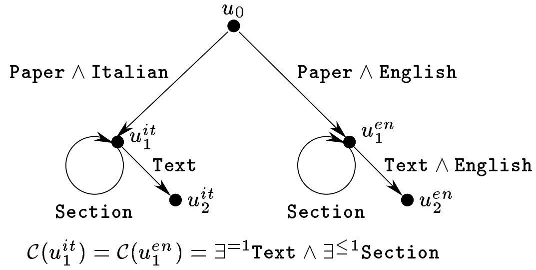


Figure 6: Schema for Italian or English papers

Consider now the database in Figure 2. If the theory $\mathcal{T}$ implies that papers $\mathbf{p}_1$ and $\mathbf{p}_2$ and all their texts are in English, then the database $\mathcal{M}$-conforms to the schema for every model $\mathcal{M}$ of $\mathcal{T}$. However, it may happen that the theory does not have information about the language of papers and texts in the database. In this case there will be models $\mathcal{M}$ of $\mathcal{T}$ such that the database does not $\mathcal{M}$-conform to the schema. ∎

5 Schemas with Constraints and Incomplete Theories

In this section, we consider the combination of the two kinds of extensions (constraints and incomplete theories) to the BDFS model presented above. A *$\mathcal{T}$-schema with $\mathcal{L}$-constraints and incomplete theory $\mathcal{S} = (\mathcal{G}, \mathcal{C})$* is defined as in Definition 7, with the proviso that now $\mathcal{T}$ is not necessarily complete. The previous definitions of conformance and subsumption can then be generalized as follows.

Definition 18 *Let $\mathcal{M}$ be a model of $\mathcal{T}$. A $\mathcal{T}$-database DB $\mathcal{M}$-*conforms *to a $\mathcal{T}$-schema $\mathcal{S} = (\mathcal{G}, \mathcal{C})$, in notation $DB \preceq_{\mathcal{M}} \mathcal{S}$, if there exists a binary relation $\trianglelefteq$ from the nodes of DB to those of $\mathcal{G}$ satisfying: (1) $root(DB) \trianglelefteq root(\mathcal{G})$, (2) $u \trianglelefteq u'$ implies that (2.1) u satisfies $\mathcal{C}(u')$, and (2.2) for each edge $u \xrightarrow{a} v$ in DB, there exists an edge $u' \xrightarrow{p} v'$ in $\mathcal{S}$ such that $\mathcal{M} \models p(a)$, and $v \trianglelefteq v'$.*

Definition 19 *Let $\mathcal{S}$, $\mathcal{S}'$ be two $\mathcal{T}$-schemas. We say that $\mathcal{S}'$* subsumes *$\mathcal{S}$, in notation $\mathcal{S} \sqsubseteq \mathcal{S}'$, if for every $\mathcal{T}$-database DB and every model $\mathcal{M}$ of $\mathcal{T}$, $DB \preceq_{\mathcal{M}} \mathcal{S}$ implies $DB \preceq_{\mathcal{M}} \mathcal{S}'$.*

We specialize this general setting to a specific one, the CDL model, and study schema subsumption in the resulting framework. The following two subsections present CDL and the results on subsumption checking, respectively.

5.1 CDL-Schemas

In CDL the theory is presented as a finite set of axioms in $\mu\mathcal{ALCQ}$, the graph is a BDFS-schema, and the constraint language is a variant of $\mu\mathcal{ALCQ}$. More precisely, if $\mathcal{S} = (\mathcal{G}, \mathcal{C})$ is a CDL $\mathcal{T}$-schema, then the theory $\mathcal{T}$, the graph $\mathcal{G}$, and the constraint language $\mathcal{L}_\mu$ of $\mathcal{C}$ have the following forms.

The Theory $\mathcal{T}$

The theory $\mathcal{T}$ is interpreted over a fixed countably infinite universe U and its language includes one distinct constant $c(d)$ for each element $d \in \mathcal{U}$. $\mathcal{T}$ is presented as a finite set of axioms of the form

$$C_1 \sqsubseteq C_2 \qquad\qquad C(a)$$

where C_1, C_2 and C are closed concepts of $\mu\mathcal{ALCQ}$, and a is a constant. We do not distinguish between $\mathcal{T}$ and the axioms representing $\mathcal{T}$, which will be considered part of the input to subsumption checking.

An interpretation $\mathcal{M} = (\Delta^{\mathcal{M}}, \cdot^{\mathcal{M}})$ of $\mathcal{T}$ is a $\mu\mathcal{ALCQ}$ interpretation, where $\Delta^{\mathcal{M}} = \mathcal{U}$, and where $\cdot^{\mathcal{M}}$ is extended to constants, in such a way that for each constant $c(d)$, $(c(d))^{\mathcal{M}} = d$. $\mathcal{M}$ satisfies an axiom $C_1 \sqsubseteq C_2$ if $C_1^{\mathcal{M}} \subseteq C_2^{\mathcal{M}}$, and an axiom $C(a)$ if $a^{\mathcal{M}} \in C^{\mathcal{M}}$.

The Graph $\mathcal{G}$

$\mathcal{G}$ is a BDFS $\mathcal{T}$-schema, where each edge is labeled by a $\mathcal{T}$-*formula*, i.e. a boolean combination of closed concepts of $\mu\mathcal{ALCQ}$ (in the language of $\mathcal{T}$), and expressions of the form $(\mathit{self} = a)$, where a is a constant of $\mathcal{T}$. Given a model $\mathcal{M}$ of $\mathcal{T}$, we define when a $\mathcal{T}$-formula p *is true for* a *in* $\mathcal{M}$, in notation $\mathcal{M} \models p(a)$, as follows:

$$\begin{array}{lll} \mathcal{M} \models C(a) & \text{iff} & a^{\mathcal{M}} \in C^{\mathcal{M}} \\ \mathcal{M} \models (\mathit{self} = a')(a) & \text{iff} & a^{\mathcal{M}} = a'^{\mathcal{M}} \quad \text{iff} \quad a = a' \\ \mathcal{M} \models (\neg p)(a) & \text{iff} & \mathcal{M} \not\models p(a) \\ \mathcal{M} \models (p_1 \wedge p_2)(a) & \text{iff} & (\mathcal{M} \models p_1(a)) \wedge (\mathcal{M} \models p_2(a)) \end{array}$$

The Constraint Language $\mathcal{L}_\mu$

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

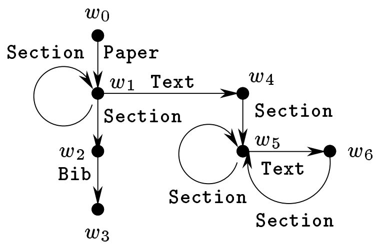
$$\begin{array}{lll} \gamma & ::= & X \mid \exists^{\geq n} F \mid \neg\gamma \mid \gamma_1 \wedge \gamma_2 \mid \mu X.\gamma \\ F & ::= & p \mid \uparrow\gamma \mid \neg F \mid F_1 \wedge F_2 \end{array}$$

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 $\gamma \vee \neg\gamma$, $\perp$ for $\neg\top$, $\forall p\uparrow\gamma$ for $\neg\exists^{\geq 1}(p \wedge \uparrow\neg\gamma)$, and $\nu X.\gamma$ for $\neg\mu X.\neg\gamma[X/\neg X]$.

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

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



$$\begin{array}{lll} \texttt{Paper} & \sqsubseteq & (\texttt{English} \sqcup \texttt{Italian}) \sqcap \exists\texttt{writtenby.Person} \sqcap \\ & & \neg\texttt{Bib} \sqcap \neg\texttt{Section} \sqcap \neg\texttt{Text} \\ \texttt{Section} & \sqsubseteq & \neg\texttt{Bib} \sqcap \neg\texttt{Text} \\ \texttt{Text} & \sqsubseteq & (\texttt{English} \sqcup \texttt{Italian}) \sqcap \neg\texttt{Bib} \\ \texttt{Bib} & \sqsubseteq & \mu X.(\texttt{BibItem} \sqcap (\forall\texttt{next}.\perp \sqcup (= 1\,\texttt{next}.X))) \\ \ldots & & \end{array}$$

$$\begin{array}{l} \mathcal{C}(w_0) = \forall\texttt{Paper}\uparrow\mu X.(\forall\texttt{Section}\uparrow X \wedge \forall\texttt{Text}\uparrow X) \\ \mathcal{C}(w_1) = \mathcal{C}(w_5) = \exists^{=1}\texttt{Text} \wedge \exists^{\leq 1}\texttt{Section} \\ \mathcal{C}(w_2) = \exists^{=1}\texttt{Bib} \\ \mathcal{C}(w_4) = \mathcal{C}(w_6) = \exists^{\leq 1}\texttt{Section} \end{array}$$

Figure 7: A CDL schema for papers with nested sections and a bibliography

where

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

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 20 Consider the schemas in Figure 2 and in Figure 6. It is easy to see that the schema in Figure 2 subsumes the schema in Figure 6, since the latter imposes more constraints. However, suppose we add the constraint

$$\mathcal{C}(u_0) = \forall\texttt{English}\uparrow\nu X.(\forall(\texttt{Text} \wedge \neg\texttt{English})\uparrow\perp \wedge \forall\texttt{Section}\uparrow X)$$

to the schema in Figure 2. Then it is possible to show that also the converse subsumption holds. ■

Example 21 The schema in Figure 7 is a further refinement of our running example, where the last section at the top level may have a bibliography instead of the text. It also includes a constraint on node w_0 that enforces the absence of loops in all chainings of `Text` and `Section`, and hence the finiteness of sequences and nestings of sections.

The theory $\mathcal{T}$ models the content of the different parts of papers. It has several forms of incomplete information (for example, a `Text` may be either `English` or `Italian`, without further information). The bibliography `Bib` is modeled as a list of `BibItem`s (the fixpoint constructor enforces the proper representation of the list).

The schema in Figure 7 subsumes the schemas in the previous figures, provided we add to them suitable constraints that enforce the finiteness of sequences and nestings of sections (e.g. the constraint $\mu X.\forall\texttt{Section}\uparrow X$ to $\mathcal{C}(u_1)$ in the schema in Figure 2). ■

5.2 Checking Subsumption

In CDL, 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, and we concentrate our attention on subsumption only.

The technique we use for checking subsumption in CDL is based on a reduction to unsatisfiability in $\mu\mathcal{ALCQ}$ knowledge bases. 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 $\Phi_{\mathcal{S}_1}$, $\Phi_{\mathcal{S}_2}$, and $\Gamma_{\mathcal{T}}$ 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):

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

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 $\Gamma_{\mathcal{T}}$ contains the following axioms in order to encode the theory $\mathcal{T}$:

$$\begin{array}{lcll}
\top_D & \sqsubseteq & \forall R.\top_D & \text{for each role } R \text{ appearing in } \mathcal{T} \\
\top_D \sqcap C_1 & \sqsubseteq & C_2 & \text{for each axiom } C_1 \sqsubseteq C_2 \text{ in } \mathcal{T} \\
\top_D \sqcap O_a & \sqsubseteq & C & \text{for each axiom } C(a) \text{ in } \mathcal{T}
\end{array}$$

where O_a, one for each constant a and axiom of $\mathcal{T}$, are new atomic concepts, called *object-concepts*. Intuitively these are used to denote constants mentioned in the axioms of $\mathcal{T}$.

Observe that the size of $\Gamma_{\mathcal{T}}$ is polynomial with respect to the size of $\mathcal{T}$.

$\Phi_{\mathcal{S}}$: 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 ψ from constraint expressions to $\mu\mathcal{ALCQ}$ formulae as follows:

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

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

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

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 χ_{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, the size of $\Phi_{\mathcal{S}}$ is exponential with respect to the size of $\mathcal{S}$.

Properties of the encoding

The following three properties of the encoding establish decidability and complexity of subsumption checking.

Theorem 22 *A* CDL *$\mathcal{T}$-schema $\mathcal{S}_1$ is subsumed by a* CDL *$\mathcal{T}$-schema $\mathcal{S}_2$ if and only if there is no model of $\Gamma_{\mathcal{T}}$ that satisfies $\Phi_{\mathcal{S}_1} \sqcap \neg\Phi_{\mathcal{S}_2}$ and interprets every object-concept as a singleton.*

Theorem 23 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two* CDL *$\mathcal{T}$-schemas, and $\Gamma_{\mathcal{T}}$, $\Phi_{\mathcal{S}_1}$, and $\Phi_{\mathcal{S}_2}$ be as defined above. Then there exists a $\mu\mathcal{ALCQ}$ knowledge base Γ' whose size is polynomial in $|\Gamma_{\mathcal{T}}| + |\Phi_{\mathcal{S}_1}| + |\Phi_{\mathcal{S}_2}|$ such that: $\Phi_{\mathcal{S}_1} \sqcap \neg\Phi_{\mathcal{S}_2}$ is satisfied in a model of $\Gamma_{\mathcal{T}}$ that interprets every object-concept as a singleton, if and only if $\Phi_{\mathcal{S}_1} \sqcap \neg\Phi_{\mathcal{S}_2}$ is satisfiable in Γ'.*

Theorem 24 *Given two* CDL *$\mathcal{T}$-schemas $\mathcal{S}_1$ and $\mathcal{S}_2$, checking whether $\mathcal{S}_1 \sqsubseteq \mathcal{S}_2$ is EXPTIME-hard and decidable in time $O(2^{p(|\Gamma_{\mathcal{T}}|+|\Phi_{\mathcal{S}_1}|+|\Phi_{\mathcal{S}_2}|)})$.*

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

6 Conclusions

In this paper we have discussed extensions of BDFS in two main directions: adding constraints to the nodes of the schema, and admitting the possibility of incomplete information in the theory that expresses the knowledge about the edges of databases. Based on these extensions, we have introduced a new model for semi-structured data with constraints and incomplete information, and we have presented techniques and complexity analysis for checking subsumption and conformance. The resulting algorithm works in deterministic double exponential

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

time with respect to the size of the theory and the two schemas.

The analysis presented in Sections 3 and 4 shows that the complexity of subsumption rises even when simple constraints and simple forms of incompleteness are added to BDFS separately. This justifies our approach that aims at adding as much expressive power as possible in specifying both the constraints and the theory, without loosing decidability. We observe also that, in our setting, if $\mathcal{T}$ is a complete theory, conformance can be reduced to model checking, which is polynomial (assuming the alternation of fixpoints in the constraints to be bounded by a constant, see e.g. [Emerson, 1996]).

We are currently working on two aspects of CDL. First, we are developing a new technique which aims at avoiding the worst case exponential blowup in the $\mu\mathcal{ALCQ}$ encoding of the schema. Second, we are considering conjunctive queries with regular expressions over CDL schemas, with the aim of devising techniques for query containment, along the line of [Calvanese *et al.*, 1998].

References

[Abiteboul *et al.*, 1995] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[Abiteboul *et al.*, 1997] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

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

[Buneman *et al.*, 1997] Peter Buneman, Susan Davidson, Mary 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.

[Calvanese *et al.*, 1998] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-98)*, 1998.

[Christophides *et al.*, 1994] V. Christophides, S. Abiteboul, S. Cluet, and M. Scholl. From structured documents to novel query facilities. In R. T. Snodgrass and M. Winslett, editors, *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 313–324, Minneapolis (Minnesota, USA), 1994.

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

[Donini *et al.*, 1992] Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.

[Emerson, 1996] E. Allen Emerson. Automated temporal reasoning about reactive systems. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 41–101. Springer-Verlag, 1996.

[Kozen, 1983] Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.

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

[Quass *et al.*, 1995] 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.

[Steffen and Ingólfsdóttir, 1994] Bernhard Steffen and Anna Ingólfsdóttir. Characteristic formulae for processes with divergence. *Information and Computation*, 110:149–163, 1994.

[Stirling, 1996] Colin Stirling. Modal and temporal logics for processes. In Faron Moller and Graham Birtwistle, editors, *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 149–237. Springer-Verlag, 1996.

[Streett and Emerson, 1989] Robert E. Streett and E. Allen Emerson. An automata theoretic decision procedure for the propositional μ-calculus. *Information and Computation*, 81:249–264, 1989.