A Modal Perspective on Path Constraints

Natasha Alechina* Stéphane Demri† Maarten de Rijke‡

Abstract

We analyze several classes of path constraints for semistructured data and prove a umber of decidability and complexity results for such constraints. While some of our decidability results were known before, we believe that our improved complexity bounds are new. Our proofs are based on techniques from modal logic and automata theory. We believe that our modal logic perspective sheds additional light on the reasons for previously known decidability and complexity results.

Keywords: semistructured data, path constraints, modal logic, automata theory.

1 Introduction

In recent years, a lot of interesting work has been done to extend database techniques to semistructured collections of data, in particular the World Wide Web or fragments of it; an overview of this work can be found in [1]. It is generally agreed that the appropriate data model for semistructured data is an edge-labeled graph. More specifically, the web can be viewed as a set of objects linked by labeled edges; an object represents a page, and the labeled edges represent hypertext links.

Query languages proposed for semistructured data and querying the web, such as WebSQL [35], Lorel [2], and UnQL [11] are similar in spirit if not in syntax, and all include a form of recursion (regular expressions). Making effective use of whatever information is available about the format of data is obviously a very important issue. In the context of the web, it is often useful to know that everything accessible by a given sequence of links is cached, or available locally; or that the site reachable by a given sequence of links is mirrored elsewhere, etc. To express such information, one can use so-called *path constraints*, that is: statements about paths in the graph. It is reasonable to expect that the language of constraints forms a well-behaved (preferably decidable) sublanguage of the query language.

In this paper we build on results in [3, 13], and embed several classes of path constraints that have been considered in the literature into well-known modal logics.

*School of Computer Science & IT, University of Nottingham, Nottingham, NG8 1BB, England. E-mail: n.alechina@cs.nott.ac.uk.

†Laboratoire Spécification et Vérification, ENS de Cachan, CNRS UMR 8643, 61 Avenue du Président Wilson, 94235 Cachan Cedex, France. E-mail: demri@lsv.ens-cachan.fr.

‡Language and Inference Technology group, ILLC, U. of Amsterdam, Nieuwe Achtergracht 166, 1018 WV Amsterdam, The Netherlands. E-mail: mdr@science.uva.nl.

Earlier work has considered the modal μ-calculus, but we consider variants of PDL (see e.g., [28]). Our embeddings into a flavor of PDL establish a number of things; they shed light on known decidability results and give rise to new ones and to new complexity bounds. In some cases, the complexity bounds obtained by translation are not the tighest possible; in such cases we provide tighter bounds by using other methods (see e.g., Theorems 14 and 15).

But perhaps more importantly, adopting a logical pespective on data modeling and description languages often yields conceptual clarity, as demonstrated, for instance by Hayes' paper on a model-theoretic semantics for RDF and RDFs [30]. In our case, we think that the main benefit of our modal perspective on path constraints lies in the insights it yields on the way various constraints relate to each other. Of course, rephrasing reasoning tasks on constraints in terms reasoning inside a suitable logic will not always give the sharpest possible complexity bounds: in some cases the corresponding logic simply has not been explored yet.

This paper is an extended and updated version of [4]. It is organized as follows. Section 2 provides background information on data models and query languages and it introduces several kinds of path constraints. In Section 3 we introduce logical formalisms to capture such constraints. In Section 4 we state our complexity and decidability results for logical problems. In Section 5, we establish corresponding results for reasoning problems on path constraints. We conclude in Section 6.

2 Path Constraints

Semistructured data is often represented as an edge-labeled graph. In particular, the World Wide Web can be modeled as a graph where the vertices are uniquely identified by URLs and the labels are hypertext links between them [1]; richer structures able to deal with the frame structure of the pages can be found in [5]. An important special class of graphs are deterministic graphs. A graph is called *deterministic* if for every node u and label a there is at most one node v such that $u \xrightarrow{a} v$ holds. In the case of the web (unlike the case of most object-oriented databases) it is reasonable to expect a graph to be deterministic.

In this paper, we will restrict attention to *rooted connected* graphs: that is, one of the nodes in the graph is designated as the root and every other node is accessible from the root by a directed path of edges. Intuitively, this is because we consider the web from the point of view of browsing, i.e., only the sites accessible from the current site (the root) really matter.

Languages for querying semistructured data use so-called path queries. These have
emerged as an important class of browsing-style queries, and in their simplest version they are of the form 'find all objects reachable by paths whose edge labels form a regular expression over some given alphabet of labels.'

Let us make things more formal. Let L be a countable set of edge labels. An L-structure G is a tuple of the form $G = \langle V, rt, (R_a)_{a\in L}\rangle$ such that V is a set of nodes, rt is a distinguished element of V (the root), and $(R_a)_{a\in L}$ is a family of binary relations on V acting as links between nodes. We say that G is finite whenever V is finite and R_a is non-empty for only finitely many labels. In that case, the size of

G, written $|G|$, is $|V| + \Sigma_{a \in L}|R_a|$. In other words, L-structures are rooted labeled transitions systems over the (possibly infinite) alphabet L.

Definition 1 Let L be a countable set of edge labels. A label $a \in L$, the empty path ϵ and wildcard $\#$ are *path expressions*. If p_1 and p_2 are path expressions, then so are $p_1 ; p_2$ (sequential composition), $p_1 + p_2$ (union), and p^* (finite iteration).

Given an L-structure G, we define *transition relations* $tr(p)$ on $V \times V$ corresponding to the path expressions p:

$$\begin{array}{rcl} tr(a) & = & R_a \text{ for } a \in L \\ tr(p^*) & \text{is} & \text{the reflexive transitive closure of } tr(p) \\ tr(\#) & = & \bigcup_{a \in L} R_a \\ tr(\epsilon) & = & \{\langle u, u\rangle : u \in V\} \\ tr(p_1 ; p_2) & = & \{\langle u, v\rangle : \exists z\, (tr(p_1)(u, z) \wedge tr(p_2)(z, v))\} \\ tr(p_1 + p_2) & = & tr(p_1) \cup tr(p_2) \end{array}$$

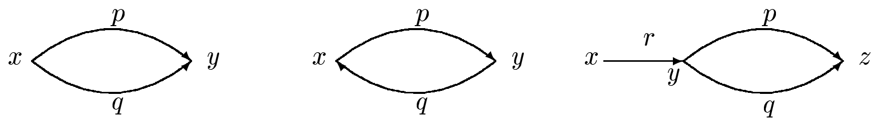
As usual, given a binary relation R on V, we write $R(u)$ to denote the set of R-successors of u: $\{v \in V : \langle u, v\rangle \in R\}$.

In the absence of information about the format of data, evaluating queries with regular expressions can be very inefficient. A natural way to express useful information about the data represented as a graph is to impose constraints on possible paths in the graph, such as 'all objects reachable by a path p are also reachable by a path q,' where p and q are sequences of labels, possibly involving regular expressions. Examples of constraints which may be useful for query optimization in the context of the web are constraints saying that everything accessible by such-and-such sequence of labels is also accessible locally; that the answer to such-and-such query is cached; that such-and-such site is mirrored elsewhere, and so on. All these examples can be expressed by means of path inclusion constraints as defined in [3] (see below).

The motivation of the work in [13] is more database-related than the work in [3]. Indeed, one important difference between the constraints considered in [3] and those studied in [13] is that the former correspond to unary properties and are evaluated relative to a node. The latter are closed sentences and can be evaluated anywhere and don't have to mention the root; the motivation for the latter is much more database-related than the work in [3]. Another difference is that the constraints from [3] can contain regular expressions, while those in [13] are strictly first-order definable.

Definition 2 Let p and q be two path expressions. A *path inclusion constraint* is a statement of the form $p \subseteq_f q$. Let G be an L-structure. A path inclusion constraint $p \subseteq_f q$ is *true at* G, written $G \models p \subseteq_f q$, if $tr(p)(rt) \subseteq tr(q)(rt)$: every node y reachable from rt by a path whose labels form a word described by p (i.e., a p-path), is reachable from rt by a path whose labels form a word described by q (i.e., a q-path). See Figure 1 (a).

The path inclusion constraints defined above are sometimes referred to as *forward* constraints. In [13], *backward* constraints are introduced. We generalize their definition to a language containing regular expressions.



(a) Inclusion constraint (b) Backward constraint (c) Lollipop constraint

Figure 1: Constraints.

Definition 3 Let p and q be two path expressions. A *backward* path constraint is a statement of the form $p \subseteq_b q$. Let G be an L-structure. A backward path constraint $p \subseteq_b q$ is *true at* G, written $G \models p \subseteq_b q$, if $tr(p)(rt) \subseteq (tr(q))^{-1}(rt)$ where $^{-1}$ denotes the converse operator on binary relations. See Figure 1 (b).

Notice that a backward constraint can be rewritten as an inclusion constraint, and vice versa, by rewriting the regular expressions involved in the presence of the converse operator.

A *standard path constraint* (notation: $p \subseteq q$) is either a forward constraint or a backward constraint. The next class of constraints is a generalization of path constraints as defined in [13] to a language containing regular expressions:

Definition 4 Let p, q, and r be path expressions. A *lollipop path constraint* is an expression of the form $r \rightsquigarrow p \subseteq q$. Let $G = \langle V, rt, (R_a)_{a \in L} \rangle$ be an L-structure. A lollipop path constraint $r \rightsquigarrow p \subseteq q$ is *true at* G, written $G \models r \rightsquigarrow p \subseteq q$, if for every $x \in tr(r)(rt)$, $\langle V, x, (R_a)_{a \in L} \rangle \models p \subseteq q$. See Figure 1 (c).

Obviously, a forward constraint $p \subseteq_f q$ is a lollilop path constraint $r \rightsquigarrow p \subseteq_f q$ with $r = \epsilon$. In the sequel, a lollipop path constraint will simply be called a path constraint.

Our syntax for talking about path constraints is obviously much more abstract than, for example, the XPath syntax [45, 29], which was designed to describe paths in XML trees. XPath involves predicates specific to attributes and names, and it allows navigation along the sibling axis. However, many constraints which are formulated using XPath expressions are very similar to the path constraints we are interested in. Consider the following example (taken from [15]):

```
<consistencyrule id="r1">
    <description>
      The product name of an advertised product must be
      in the catalogue
    </description>
    <forall var="a" in="$adverts">
        <exists var="p" in="$products">
            <equal op1="$a/ProductName/text()"
            op2="$p/Name/text()"/>
        </exists>
    </forall>
</consistencyrule>
```

This is the same as the following path inclusion constraint:

$$\text{Adverts}; \text{ProductName} \subseteq_f \text{Products}; \text{Name}.$$

A further exploration of the formal logical aspects of XPath is beyond the scope of this paper; we refer the reader to [25, 37].

Now that we have formulated path constraints, we take a closer look at important reasoning tasks involving them.

- The *query evaluation* problem for a class C of path constraints is defined as follows:

  instance: a finite L-structure G and a constraint c in C;

  question: $G \models c$?

- A more difficult problem is the *containment* problem for the class C of path constraints. It is defined as follows:

  instance: constraints $c_1, \ldots, c_{n+1}$, $n \geq 0$, in C;

  question: is it the case that for every L-structure G, $G \models c_1$ and $\ldots$ and $G \models c_n$ imply $G \models c_{n+1}$? (if so, we write $c_1, \ldots, c_n \rightarrow c_{n+1}$.)

Variants of the above problems can be defined by considering only (finite, deterministic, ...) L-structures or/and by distinguishing the class of constraints for c_1, ..., c_n and c separately (if applicable).

In the sequel we investigate the decidability and complexity issues of the problems we have just introduced, mostly from a modal logic point of view.

3 Capturing Path Constraints by Modal Means

To determine the computational costs of reasoning tasks on path constraints, we recast them as model checking, validity, and satisfiability tasks in some logic. The logic used for this purpose should be such that it can easily encode problems on path constraints, especially the containment problem for the class of standard path constraints. Moreover, we wish to reduce the complexity gap between problems on path constraints and logical problems. In this way, a model checker or a theorem prover for such a logic will allow us to solve problems on path constraints in an efficient manner.

3.1 Choosing a Logic

A modal flavor. Which logic (or logics) should we use for capturing path constraints? Many formalisms have been proposed for reasoning about graphs. As we will see below, many decidable classes of constraints can be defined in terms of suitable modal logics, while constraints that lack a modal flavor (such as the ones studied in [13]) are generally undecidable. Rather than the presence or absence of regular expressions or even the need for two vs. three variables to express a constraint, the 'modal flavor' of constraints seems to be important. By this we mean the fact that modal formulas can only express local properties and the fact that the quantification implicit in modal formulas is 'guarded' [6]. We opt for Propositional Dynamic Logics (PDL, [28]) here since these are modal logics that incorporate regular expressions. Thus, we translate constraints into formulas of a flavor of PDL and reformulate reasoning tasks for constraints as reasoning tasks within this flavor of PDL.

PDL-like features. The language of PDL has two kinds of primitive symbols: propositional symbols and atomic transitions. Propositional symbols stand for properties that are true or false of a node in a graph; we only need three propositional symbols: $\top$ (tautology), $\bot$ (falsum), and *root* (to denote the root of the graph). Atomic transitions are used to label edges; we include a distinguished label ϵ to denote the diagonal relation. Compound transition terms correspond to path expressions and are built from atomic ones using ;, + and $*$. We indifferently use "path expression", "transition term" and "relational term" as synonyms.

In addition to these traditional ingredients of PDL, we add a *wildcard* # and a *converse operator* $(\cdot)^{-1}$: # is a transition term, and if t is a transition term, then so is $(t)^{-1}$. For instance, the satisfaction of the forward constraint $c = p \subseteq_f q$ at rt in the L-structure G will be expressed as $G, rt \models [p]\langle (q)^{-1}\rangle root$. The node rt is the only node w in G satisfying $G, w \models root$. Hence, our symbol *root* is an example of a so-called *nominal* (a proposition letter that is true of at most a single node of a model). PDL with converse is called *converse PDL* (CPDL). We obtain *CPDL with nominals* (see e.g., [20]) by extending CPDL with nominals.

3.2 Defining the Logic PDL^{path}

The logic PDL^{path} we study is a fragment of CPDL with nominals augmented with the wildcard #. Here's a more precise definition.

Definition 5 (PDL^{path}) The path expressions of PDL^{path} considered are those introduced in Definition 1 with the inclusion of the converse operator $(\cdot)^{-1}$; they are denoted by p, q and also by t. Formulas of PDL^{path} are typically denoted by ϕ:

$$\phi \quad ::= \quad \top \mid \bot \mid root \mid \neg\phi \mid \phi \wedge \phi \mid \langle t\rangle\phi \mid [t]\phi.$$

Given a formula ϕ, define $|\phi|$, the *length* of ϕ, as the number of symbols in ϕ. A formula $\langle t\rangle\phi$ is read as 'after some transition t, ϕ holds,' or, more precisely, as 'there is a sequence of labels which forms a word in a regular language defined by t and it leads to a node where ϕ holds.' Dually, $[t]\phi$ is definable as $\neg\langle t\rangle\neg\phi$ and means 'after every transition t, ϕ holds,' that is: 'if labels of a path form a word in t, then at the end of the path ϕ holds.'

To give an example, $\langle a^*\rangle\neg\langle b\rangle\top$ means that after 0 or finitely many a links one can reach a node which has no outgoing links labeled b.

Definition 6 (Semantics) PDL^{path} is interpreted on L-structures. We now define truth of a formula ϕ at a node w in a structure G (notation: $G, w \models \phi$). For atomic propositional symbols, $\top$ is true at all nodes, $\bot$ is false at all nodes, and *root* is true only at the root of the graph. Further, $\neg\phi$ is true if ϕ is false, and $\phi \wedge \psi$ is true if both ϕ and ψ are true. For modalities, we shall use the transition relations $tr(p)$. We just need to extend $tr(\cdot)$ such that $tr((p)^{-1}) = \{\langle u, v\rangle : \langle v, u\rangle \in tr(p)\}$.

We say that v is *accessible* from u by a transition t if $\langle u, v\rangle \in tr(t)$. Then, for modal formulas, $\langle t\rangle\phi$ is *true at a node* u if there exists a node v accessible from u by t such that ϕ is true at v. Dually, $[t]\phi$ is true at u if for every v accessible from u by t, ϕ is true. A PDL^{path} formula ϕ is *true on an L-structure G* if it is true at the root of G. A PDL^{path} formula ϕ is *satisfiable* if it is true on some L-structure G. A formula is valid if it is true on all L-structures.

Roughly speaking, L-structures equipped with $tr(t)$ for every relational term t in PDL^{path} can (almost) be viewed as PDL-models with a unique proposition letter interpreted by a singleton. This is not quite true because of the presence of #, but we will show below that it is correct when non-deterministic L-structures are considered (see the proof of Theorem 11). Hence, modulo the presence of #, PDL^{path} can be viewed as a fragment of CPDL with nominals [20] or as a fragment of the hybrid μ-calculus [43]. Furthermore, the constructive μ-calculus introduced in [5] also contains nominals (i.e., proposition letters interpreted by singletons) as well as a form of recursion. We don't need the full expressive power of the μ-calculus, however: we are interested in path queries, and regular expressions from PDL are sufficient to express standard path constraints.

Just like the modal logic HML [33], PDL^{path} has no proposition letters except the truth constant $\top$ and the unique nominal *root*. Indeed, PDL^{path} is designed to reason about relations.

A final comment: one of our main aims is to reduce as much as possible the complexity gaps between problems on path constraints and logical problems. We will show that PDL^{path} is well-designed in this respect.

3.3 Standard Logical Reasoning Tasks

For the purposes of this paper, the following logical reasoning tasks (involving PDL^{path}) are important:

- The *model checking* problem for PDL^{path} is:

  instance: a finite L-structure G and a formula ϕ;

  question: Is ϕ true at G?

- The *validity* problem for PDL^{path} is defined as follows:

  instance: a formula ϕ;

  question: Is it the case that for every L-structure G, ϕ is true at G?

- The *satisfiability problem* is defined accordingly in the usual way.

Before we explore the computational costs of the above reasoning tasks for PDL^{path} we will show that PDL^{path} suffices for expressing the reasoning tasks on the standard path constraints that we defined earlier.

3.4 From Path Constraints to PDL^{path} and Back

Given a forward constraint $c = p \subseteq_f q$ (or a backward constraint $c = p \subseteq_b q$), we write ϕ_c to denote the formula $[p]\langle(q)^{-1}\rangle root$ (or $[p]\langle q\rangle root$, respectively). Lemma 7 relates reasoning with path constraints to reasoning tasks with PDL^{path}.

Lemma 7

1. Let G be an L-structure and $c = p \subseteq q$ a standard path constraint. Then $G \models c$ iff ϕ_c is true at G.
2. Let $c_1, \ldots, c_{n+1}$ be standard path constraints. Then $c_1, \ldots, c_n \rightarrow c_{n+1}$ iff $(\phi_{c_1} \wedge \cdots \wedge \phi_{c_n}) \Rightarrow \phi_{c_{n+1}}$ is PDL^{path} valid.

The proof of Lemma 7 is by an easy verification.

By contrast, there is no way to express (lollipop) path constraints $r \rightsquigarrow p \subseteq q$ by PDL^{path} formulas in a similar fashion since the containment problem for the class of (lollipop) path constraints is undecidable [13] and PDL^{path} validity is decidable, as we will see below. As an aside, (lollipop) path constraints can be expressed in modal logics with reference pointers (see e.g., [24]), but the validity problem of such logics is usually undecidable.

By using Lemma 7, one can easily establish the following results:

Lemma 8 Let C be either the full class of L-structures or the class of deterministic L-structures.

1. The query evaluation problem for standard path constraints is LOGSPACE reducible to the model checking problem for PDL^{path}.
2. The containment problem for forward constraints restricted to L-structures in C is LOGSPACE reducible to the validity problem for PDL^{path} restricted to L-structures in C.
3. The containment problem for backward constraints restricted to L-structures in C is LOGSPACE reducible to the validity problem for PDL^{path} without converse and restricted to L-structures in C.

To conclude this section, we will briefly contrast our modal formalization of path constraints with others in the literature. In [37], a fragment of the Computation Tree Logic CTL is shown to be equivalent to $P^{[],*,//}$, a tree pattern language; more precisely, the authors prove equivalence between the implication problem in this fragment of CTL and the containment problem for $P^{[],*,//}$. In addition, in [25], Core XPath is translated into full CTL. It is important to notice that, unlike the authors of the papers just quoted, we consider graph structures — and not just tree structures.

In [16], a spatial logic is introduced for reasoning about labelled directed graphs; the logic is closely related to monadic second-order logic (MSO) for graphs (see e.g., [19]), and its main use is in querying structures from the semi-structured data model. Both logics have a model checking problem in PSPACE, whereas satisfiability for MSO is undecidable. By contrast, our logic PDL^{path} is better attuned to the reasoning tasks about path constraints and we do not actually need the full power of MSO-like formalisms, especially if we wish to deduce complexity upper bounds of problems on path constraints from translations into logical problems.

4 The Complexity of Reasoning with PDL^{path}

We now consider complexity and decidability results for PDL^{path} problems. In the next section we will use these results, together with Lemma 8, to derive complexity and decidability results for path constraints.

4.1 The Model Checking Problem

The model checking problem for PDL^{path} is no more expensive than for PDL:

Theorem 9 Let $G = \langle V, rt, (R_a)_{a \in L} \rangle$ be an L-structure, $w \in V$, and ϕ a PDL^{path} formula. Checking whether $G, w \models \phi$ can be done in time $\mathcal{O}(|G| \times |\phi|)$.

Proof There is a simple linear reduction of the model checking problem for PDL^{path} to the model checking problem for PDL. The latter problem is $\mathcal{O}(|G| \times |\phi|)$. This follows from the fact that model checking for the alternation-free modal μ-calculus is in linear time (see e.g., [18]).

The linear time reduction works as follows. First, given G, we construct a new L-structure G'. G' has the same vertices and root, and contains all the edges which G has plus, for every edge $u \xrightarrow{a} v$ in G we add three more edges to G':

$$u \xrightarrow{\#} v,\ v \xrightarrow{(\#)^{-1}} u, \text{ and } v \xrightarrow{(a)^{-1}} u.$$

The construction of G' is obviously linear in the size of G. Second, we rewrite ϕ so that all occurrences of $(\cdot)^{-1}$ are on the atomic labels. This can be done in linear time by using the following standard equivalences:

$$\begin{array}{rcl}
(p+q)^{-1} & = & (p)^{-1} + (q)^{-1} \\
(p\,;q)^{-1} & = & (q)^{-1}\,;(p)^{-1} \\
(p^*)^{-1} & = & ((p)^{-1})^* \\
((p)^{-1})^{-1} & = & p
\end{array}$$

Note that the resulting formula ϕ' is linear in the size of ϕ. Finally, it is easy to show that $G, w \models_{\mathrm{PDL}^{path}} \phi$ iff $G', w \models_{\mathrm{PDL}} \phi'$.

Corollary 10 The model checking problem for PDL^{path} is P-complete.

The polynomial upper bound is a consequence of Theorem 9 and the polynomial lower bound can easily be obtained by a reduction from the P-complete problem SAM2CVP (synchronous alternating monotone fanout 2 circuit value problem; see e.g., [26, page 123]), as has been done to show the P-hardness of CTL model-checking restricted to $\forall \mathsf{X}$ and $\exists \mathsf{X}$, a folklore result in model checking. The proof for CTL can easily be adapted to the model-checking problem for PDL^{path} restricted to deterministic L-structures, as soon as $|L| \geq 2$.

4.2 The Satisfiability Problem

The satisfiability problem for PDL^{path} can be proved to be decidable by a reduction to the (decidable) satisfiability problem of CPDL with nominals.

Theorem 11 The satisfiability and validity problems for PDL^{path} are decidable in EXPTIME.

Proof We use the fact that CPDL with nominals is decidable [20, Theorem 49] and reduce satisfiability in PDL^{path} to satisfiability in CPDL with nominals. If the set of labels L is finite, $\#$ can be replaced by the finite union of labels from L and the reduction to CPDL with nominals is immediate.

Suppose that L is infinite, in which case $\#$ is a non-trivial addition to the language. We proceed as follows. Given a PDL^{path} formula ϕ which uses labels $\{a_1, \ldots, a_n\}$ and possibly $\#$, we construct a CPDL formula ψ by replacing $\#$ with $(a_1 + \cdots + a_{n+1})$ in ϕ, and we show that ϕ is satisfiable (which means ϕ is true at the root of some L-structure) iff $root \wedge \psi$ is.

Suppose $G = \langle V, rt, (R_a)_{a \in L}\rangle, rt \models \phi$. We will define an L-structure G' and view it as a partial description of a model for CPDL with nominals. Let $G' = \langle V, rt, (R'_a)_{a \in L}\rangle$ be the L-structure defined as follows:

1. for every $1 \leq i \leq n$, $R'_{a_i} = R_{a_i}$;
2. $R'_{a_{n+1}} = \bigcup_{a \in L} R_a$.

It is easy to show that $G, rt \models \phi$ iff $G', rt \models root \wedge \psi$ since $tr(\#)$ in G is equal to $tr(a_1 + \cdots + a_{n+1})$ in G'.

Now, suppose $G = \langle V, rt, (R'_a)_{a \in L}\rangle, rt \models root \wedge \psi$ for some partial description of a model for CPDL with nominals. Let $G = \langle V, rt, (R_a)_{a \in L}\rangle$ be the L-structure defined as follows:

1. for every $1 \leq i \leq n$, $R_{a_i} = R'_{a_i}$;
2. for every $a \in (L \setminus \{a_1, \ldots, a_n\})$, $R_a = R'_{a_{n+1}}$.

It is now easy to show that $G', rt \models root \wedge \psi$ iff $G, rt \models \phi$.

By itself, Theorem 11 does not imply an analogous result for deterministic L-structures, which remains an open problem to date. However, if the set of edge labels L is finite, deterministic CPDL with nominals is decidable only if on deterministic L-structures, the satisfiability problem for PDL^{path} is decidable.

In the non-deterministic case, we can actually do better than Theorem 11, and obtain matching lower and upper bounds for the complexity of the satisfiability problem for PDL^{path}.

Theorem 12 The satisfiability problem for PDL^{path} is EXPTIME-complete whenever $|L| \geq 1$.

Proof To see that the satisfiability problem for PDL^{path} is decidable in exponential time, recall that, by [20, page 98], the satisfiability problem for CPDL with nominals is EXPTIME-complete. The reduction of the PDL^{path} satisfiability problem to the satisfiability problem for CPDL with nominals is polynomial in the size of the input formula.

As to the lower bound, we use a reduction from the global satisfiability problem for the standard modal logic K. A formula ϕ is *globally* satisfiable if there exists a model such that ϕ is satisfied in every state of the model. The global satisfiablity problem for K is known to be EXPTIME-hard, see e.g., [17, 31]. Our reduction only uses a restricted form of the PDL^{path}-satisfiability problem, viz. restricted to the modal operators $\langle a_0 \rangle$ and $\langle a_0^{-1} \rangle$. We use the spy-point technique as described in [10] by adapting the proof of [7, Theorem 2]. As part of this technique, we introduce a node (the "spy point") in the model that can see any other node, and, therefore, universal quantification can be simulated by exploring the successors of the spy point. The only difficulty is to use the spy-point technique and simultaneously encode the proposition letters which occur in the K-formula.

We set out to define a map f from K-formulas into PDL^{path}-formulas such that ϕ is globally satisfiable iff $f(\phi)$ is PDL^{path} satisfiable. In order to show the 'only-if' direction, given a Kripke structure $\mathcal{M} = \langle V, R, m \rangle$ such that $\mathcal{M} \models \phi$, we construct an L-structure $G = \langle V^*, rt, R_{a_0} \rangle$ such that $f(\phi)$ is true at G and the fact that $\mathcal{M}, w \models p_i$ is encoded by the existence of an edge $i \xrightarrow{a_0} w$ in G where i is a node in G associated with the proposition letter p_i. To simplify notation we now write $\longrightarrow$ instead of $\xrightarrow{a_0}$. The structure $\langle V^*, R_{a_0} \rangle$ consists of:

- a copy of $\langle V, R \rangle$, that is to say, the restriction of $\langle V^*, R_{a_0} \rangle$ to V is $\langle V, R \rangle$;
- an extra node rt (the spy point) that can see any node in V (i.e., for every $w \in V$, $rt \longrightarrow w$ in G);
- extra nodes $1, \ldots, N$ that are used to encode the satisfaction of proposition letters $p_1, \ldots, p_N$ in such a way that $\mathcal{M}, w \models p_i$ iff $i \longrightarrow w$ is in G;
- a chain $rt \longleftrightarrow 1 \longrightarrow 2 \longrightarrow \ldots \longrightarrow N$ that allows us to identify the elements of $V^* \setminus V$:
  1. rt is the only element w in V^* such that $G, w \models \mathit{root}$ (not related to the chain);
  2. 1 is the only element w of V^* such that $rt \longrightarrow w$ and $w \longrightarrow rt$;
  3. $i+1$ $(1 \leq i \leq N-1)$ is the only element w in V^* such that $i \longrightarrow w$, $w \neq rt$ (just here for the case $i = 1$), and not $rt \longrightarrow w$.

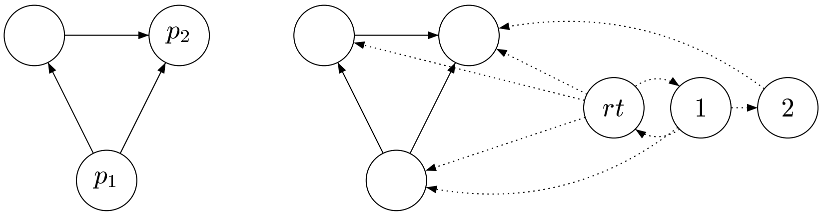
Consequently, the elements of V in G are precisely the elements w in V^* such that $rt \longrightarrow w$ and not $w \longrightarrow rt$.

Let us define a family $(\pi_i)_{i \in \mathbb{N}}$ of PDL^{path} formulas encoding proposition letters.

$$\begin{aligned} \pi_1 &:= \langle a_0^{-1} \rangle \overbrace{(\langle a_0 \rangle \mathit{root} \wedge \langle a_0^{-1} \rangle \mathit{root})}^{\text{"only true at 1"}} \\ \pi_{i+1} &:= \langle a_0^{-1} \rangle \overbrace{(\neg \mathit{root} \wedge \neg \langle a_0^{-1} \rangle \mathit{root} \wedge \pi_i)}^{\text{"only true at } i+1\text{"}}. \end{aligned}$$

Encodings of proposition letters can be also found in [27, 32], but our encoding is of a different nature since we tailor it to the spy-point technique. The mapping f is now defined as follows:

$$f(p_i) \quad = \quad \pi_i \text{ for } i \geq 1$$



$\mathcal{M}$ $\qquad$ G

Figure 2: $\mathcal{M}$ and G: an example.

$$
\begin{array}{rcl}
f & & \text{commutes with the Boolean connectives} \\
f(\Diamond\phi) & = & \langle a_0\rangle(\overbrace{\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root}^{\text{“only true at elements of } V\text{”}} \wedge f(\phi)).
\end{array}
$$

Now, let ϕ be a K-formula. We will show that ϕ is globally K-satisfiable iff the following formula is PDL^{path}-satisfiable:

$$
[a_0](\overbrace{\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root}^{\text{“only true at elements of } V\text{”}} \rightarrow f(\phi)).
$$

Without loss of generality, we can assume that if N distinct proposition letters occur in ϕ, then they are $p_1, \ldots, p_N$.
(‘Only-if’) Assume that $\mathcal{M} \models \phi$ for some Kripke structure $\mathcal{M} = \langle V, R, m\rangle$. Let $G^* = \langle V^*, rt, R_{a_0}\rangle$ be the L-structure such that

- $V^* := V \cup \{rt\} \cup \{1, \ldots, N\}$ where $rt \notin V \cup \{1, \ldots, N\}$ and $V \cap \{1, \ldots, N\} = \emptyset$;
- $R_{a_0} := R \cup \{\langle rt, w\rangle : w \in V\} \cup \{\langle 1, 2\rangle, \ldots, \langle N-1, N\rangle\} \cup \{\langle 1, rt\rangle, \langle rt, 1\rangle\} \cup \{\langle i, w\rangle : \mathcal{M}, w \models p_i\}$;
- the interpretation of $root$ is rt.

In Figure 2, we give a simple example of the construction. Let us show that for all $w \in V$, for all $\psi \in \mathrm{sub}(\phi)$ (the set of subformulas of ϕ), $\mathcal{M}, w \models \psi$ iff $G, w \models f(\psi)$. For the base case (for $i \in \{1, \ldots, N\}$, $\mathcal{M}, w \models p_i$ iff $G, w \models \pi_i$), one can easily show by induction on i that $\{w \in V^* : G, w \models \pi_i\} = \{w \in V : \mathcal{M}, w \models p_i\} \cup (\{i+1\} \setminus \{N+1\})$.

We now turn to the case $\psi = \Diamond\psi'$. Observe that $\mathcal{M}, w \models \Diamond\psi'$ iff there is $w' \in R(w)$ such that $\mathcal{M}, w' \models \psi'$ iff (i) there is $w' \in V$ such that $\langle w, w'\rangle \in R_{a_0}$, $\langle r, w'\rangle \in R_{a_0}$, $\langle w', r\rangle \notin R_{a_0}$, and $\mathcal{M}, w' \models \psi'$. Furthermore, (i) holds iff there is $w' \in R_{a_0}(w)$ such that $G, w' \models f(\psi') \wedge \langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root$ iff $G, w \models f(\psi)$. So for all $w \in V$, we have $G, w \models f(\psi)$ since $\mathcal{M} \models \psi$. Moreover, $R_{a_0}(rt) \cap \{w \in V^* : G, w \models \langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root\} = V$. So, $G, rt \models [a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \rightarrow f(\psi))$.
(‘If’) Assume that $G, rt \models [a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \rightarrow f(\phi))$ for some L-structure $G = \langle V, rt, R_{a_0}\rangle$. Define a model $\mathcal{M} = \langle V^*, R, m\rangle$ such that

- $V^* := \{w \in V : \langle rt, w\rangle \in R_{a_0}, \langle w, rt\rangle \notin R_{a_0}\}$;

- R is the restriction of R_{a_0} to V^*;
- for every $i \in \{1, \ldots, N\}$, $m(p_i) := \{w \in V^* : G, w \models \pi_i\}$.

Let us show that for all $w \in V^*$ and for $\psi \in \text{sub}(\phi)$, $G, w \models f(\psi)$ iff $\mathcal{M}, w \models \psi$. The base case is obvious. We treat the case $\psi = \Diamond\psi'$ in a bit more detail. We have $G, w \models \langle a_0\rangle(f(\psi') \wedge \langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root)$ iff there is $w' \in R_{a_0}(w) \cap V^*$ such that $G, w' \models f(\psi')$ iff there is $w' \in R(w)$ such that $\mathcal{M}, w' \models \psi'$ iff $\mathcal{M}, w \models \psi$. Since $G, rt \models [a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \rightarrow f(\psi))$, we obtain $\mathcal{M}, w \models \psi$ for all $w \in R_{a_0}(rt) \cap V^*$s. Since $V^* \subseteq R_{a_0}(rt)$, it follows that $\mathcal{M}, w \models \phi$ for all $w \in V^*$.

The minimal tense logic (the logic K_t in [41]) is a bimodal logic with modal operators $[a_0]$, $[a_0^{-1}]$, and a countably infinite set of proposition letters. As a corollary to the proof of Theorem 12, the minimal tense logic augmented with a single nominal but without proposition letters has an EXPTIME-hard satisfiability problem. This improves the result in [7], which states that the minimal tense logic with one nominal *and* proposition letters is EXPTIME-hard. From the proof of Theorem 12, it is clear that the main ingredient for EXPTIME-hardness is not the presence of regular expressions in PDL^{path} but rather the presence of a unique nominal with future and past-time operators.

To conclude this section we summarize the results we have obtained so far, and situate them amongst related results in the literature; see Table 1.

Model checking problem

	non-deterministic graphs	deterministic graphs
PDL	P-complete; see e.g., [18]	P-complete; see e.g., [18]
PDL^{path}	P-complete; this paper, Corollary 10	P-complete; this paper, Corollary 10

Satisfiability problem

	non-deterministic graphs	deterministic graphs
PDL	EXPTIME-complete [22, 40]	EXPTIME-complete [39, 9]
PDL with nominals	EXPTIME-complete [23]	EXPTIME-complete [23]
CPDL	EXPTIME-complete [22, 40]	EXPTIME-complete [44]
CPDL with nominals	EXPTIME-complete [20, 7]	open
PDL^{path}	EXPTIME-complete; this paper, Theorem 12	open

Table 1: A summary of results on logical reasoning tasks.

5 The Complexity of Reasoning with Path Constraints

In order to characterize the complexity of reasoning problems on path constraints, we will either use translations into PDL^{path} (mainly via Lemma 8) or use direct methods.

5.1 The Query Evaluation Problem

An immediate corollary to Lemma 8, item 1, and Theorem 9 is that checking whether $G \models p \subseteq q$ can be done in time $\mathcal{O}(|G| \times (|p| + |q|))$. The rest of this section is devoted to showing that this tractability result can be improved significantly. We need the following lemma.

Lemma 13 The problem below is in NLOGSPACE in $|G|$ and $|t|$:

instance: a finite L-structure G, a path expression t, and $u, v \in V$;

question: is $\langle u, v \rangle \in tr(t)$?

Proof Without loss of generality, we can assume that t does not contain $\#$, since G is finite. We write $\mathcal{A}_{G,u,v}$ to denote the finite state automaton obtained from G in the obvious way with initial state u and final state v. The following facts are known:

1. constructing a finite state automaton $\mathcal{A}(t)$ recognizing $L(t)$ (the language generated by t) can be done in LOGSPACE in $|t|$;
2. constructing a product automaton $\mathcal{B}$ recognizing the intersection of the languages from $\mathcal{A}(t)$ and $\mathcal{A}_{G,u,v}$ can be done in LOGSPACE in $|\mathcal{A}(t)| + |G|$;
3. the class of LOGSPACE transformations is closed under composition (see e.g., [8, Theorem 3.37]).

Now, note that the question whether $\langle u, v \rangle \in tr(t)$ is equivalent to checking whether $L(\mathcal{B})$ is non-empty. By [8, Theorem 3.36], we get that the latter can be done in NLOGSPACE in $|\mathcal{B}|$.

Lemma 13 is an improvement of [36], which only states that the problem formulated in Lemma 13 is in P.

In the proof of Lemma 13, if t contains $\#$, then we consider t' instead of t by replacing every occurrence of $\#$ by $(a_1 + \cdots + a_k)$, where either a_i occurs in t or R_{a_i} is non-empty in G. Hence, $|t'|$ is in $\mathcal{O}(|G| \times |t|)$ which guarantees that we also have an NLOGSPACE upper bound in this case.

Theorem 14 The query evaluation problem for the class of path constraints is NLOGSPACE-complete in $|G|$ and $|c|$.

Proof The Graph Accessibility Problem (GAP) can easily be reduced to the query evaluation problem, which provides NLOGSPACE-hardness. Indeed, let $G = \langle V, R \rangle$ be a graph and $u, v \in V$. We have $\langle u, v \rangle \in R^*$ iff $G' = \langle V, u, R_{a_1}, R_{a_2} \rangle \models a_2 \subseteq_f a_1^*$ with $R_{a_1} = R$ and $R_{a_2} = \{\langle u, v \rangle\}$.

Now, let us establish the NLOGSPACE upper bound. Let $c = r \rightsquigarrow p \subseteq_f q$ (respectively, $c = r \rightsquigarrow p \subseteq_b q$) be a path constraint and G a finite L-structure. We provide an NLOGSPACE algorithm to check whether $G \not\models c$. Since NLOGSPACE = co-NLOGSPACE, we are done. The steps are the following:

1. non-deterministically choose u and v in V;
2. check in NLOGSPACE whether $\langle rt, u \rangle \in tr(r)$ (see Lemma 13);

3. check in NLOGSPACE whether $\langle u, v\rangle \in tr(p)$ (see Lemma 13);
4. check in NLOGSPACE whether $\langle u, v\rangle \notin tr(q)$ (respectively $\langle v, u\rangle \notin tr(q)$).

To see that the final step can also be done in NLOGSPACE, use Lemma 13 together with the fact that NLOGSPACE = co-NLOGSPACE.

Notice that the proof for NLOGSPACE-hardness is actually a lowerbound for forward constraints. Moreover (and by taking $R_{a_2} = \{\langle v, u\rangle\}$) it can be adapted for backward constraints. In a similar manner, the proof can be adapted for deterministic structures. Summarizing, then, we have the following:

Theorem 15 The query evaluation problems for the classes of forward constraints, backward constraints, and path constraints c are all NLOGSPACE-complete in $|G|$ and $|c|$, for both deterministic and non-deterministic graphs G.

As an aside, we designed the logic PDL^{path} in such a way that we can express reasoning problems for standard path constraints as easily as possible. We can 'measure' how well PDL^{path} fits this purpose by comparing the complexity results of reasoning in PDL^{path} to those of reasoning in other logics, and hopefully getting as close as possible to the complexity of the corresponding problem on standard path constraints. For instance, the hybrid μ-calculus has a model-checking problem in NP $\cap$ co-NP; given Theorems 14 and 15 this result disqualifies the hybrid μ-calculus as a logic that nicely fits the descriptive requirements of standard path constraints.

5.2 The Containment Problem

Our next aim is to obtain sharp complexity results for containment problems for classes of path constraints. We start by considering non-deterministic L-structures, and the first step is to show the following.

Theorem 16 The containment problem for forward constraints is decidable in exponential time, while it is at least PSPACE-hard whenever $|L| \geq 2$.

Proof The EXPTIME upper bound is a corollary of Lemma 8(2) and Theorem 12. As to the lower bound PSPACE, each relational term t built with $+, ;, ^*$ over the atomic terms $a_0, a_1, \ldots$ can obviously be viewed as regular expressions and we write $L(t)$ to denote the language generated by t. By [34, Theorem 2.12(c)], the problem of checking whether $L((a_0+a_1)^*) \not\subseteq L(t)$, where t is a relational term built over $\{a_0, a_1\}$, is PSPACE-complete. The complement problem belongs to the same complexity class. One can show that for any regular expression t, t', $L(t) \subseteq L(t')$ iff for any structure $\langle V, r, R_{a_0}, R_{a_1}\rangle$, $tr(t) \subseteq tr(t')$. Consequently, it is easily verified that for any relational term t built over $\{a_0, a_1\}$, $L((a_0 + a_1)^*) \subseteq L(t)$ iff $\models (a_0 + a_1)^* \subseteq_f t$. Hence, the containment problem restricted to two labels and without forward constraints as premisses, is already PSPACE-hard.

For backward constraints one can obtain results similar to those for forward constraints.

Theorem 17 The containment problem for backward constraints is decidable in exponential time, while it is at least PSPACE-hard whenever $|L| \geq 3$.

Proof Since the converse construction $(\cdot)^{-1}$ is not present in the path expressions, we cannot simply use the proof of Theorem 16. However, one can easily show that for any regular expression p built over $\{a_0, a_1\}$, (i) $L((a_0 + a_1)^*) \subseteq L(p)$ iff (ii) for any L-structure G, $G \models p \subseteq_b a_2$ implies $G \models (a_0 + a_1)^* \subseteq_b a_2$, and from this we easily get our theorem. So let us prove that the equivalence (i) iff (ii) holds.
(Only-if) If $L((a_0+a_1)^*) \subseteq L(p)$, then obviously for every L-structure G, $G \models p \subseteq_b a_2$ implies $G \models (a_0 + a_1)^* \subseteq_b a_2$.
(If) Assume that (ii) holds and suppose that there is a finite word $w \in L((a_0 + a_1)^*) \setminus L(p)$ such that $w = b_1 \cdot \ldots \cdot b_n$ for some $n \geq 0$ ($n = 0$ if $w = \epsilon$). Let $G_0 = \langle \{1, \ldots, n+1\}, 1, R_{a_0}, R_{a_1}, R_{a_2} \rangle$ be the structure such that

- for every $i \in \{0, 1\}$, for all $k, k' \in \{1, \ldots, n+1\}$, $kR_{a_i}k'$ iff $k' = k + 1$ and $b_k = a_i$;
- $R_{a_2} := \{\langle k+1, 1 \rangle : k \in \{0, \ldots, n\},\ b_1 \cdot \ldots \cdot b_k \in L(t)\}$.

So $G_0 \models p \subseteq_b a_2$ but not $G \models (a_0 + a_1)^* \subseteq_b a_2$ since $\langle 1, n+1 \rangle \in (R_{a_0} \cup R_{a_1})^*$ and $\langle n+1, 1 \rangle \notin R_{a_2}$, a contradiction.

We now restrict attention to *deterministic* L-structures, which makes a substantial difference. The containment problem with (lollipop) path constraints of the form $r \rightsquigarrow p \subseteq_f q$ restricted to deterministic L-structures is undecidable even if L contains only two labels [12]. However, in the lollipop constraints used in the proof, the operator * occurs in r, and this is used to encode the word problem. Moreover by imposing restrictions of r, p, and q (for instance by forbidding *), decidable restrictions of the containment problem on deterministic L-structures have been found (again, see [12]). Using the results in this paper, we are able to identify a new decidable case. By combining Lemma 8, item 3, and Theorem 11, we get

Lemma 18 The containment problem for backward constraints restricted to deterministic L-structures for finite sets of labels L is decidable in exponential time.

Lemma 18 above provides a positive answer to an open question from [12]. However, in spite of Lemma 8 it is open whether the containment problem for forward constraints restricted to deterministic L-structures is decidable. Similarly, the decidability of the containment problem for backward constraints restricted to deterministic L-structures (without restrictions on L) is also open.

6 Conclusions

By moving back and forth between reasoning tasks for PDL^{path} and reasoning tasks for semistructured data, we have given new and transparent decidability proofs for the forward constraints proposed in [3] for optimizing queries on semistructured data, mostly in the context of the web. In some cases we have obtained sharp upper and lower bounds that are better than previously known ones (see e.g., Theorem 14 and 15), and in other cases we have improved known bounds (Theorems 16 and 17). Tables 1 and 2 summarize the complexity and (un-) decidability results for the reasoning tasks considered in this paper.

Query evaluation problem

	non-deterministic graphs	deterministic graphs
forward constraints	NLOGSPACE-complete; this paper, Theorem 15	NLOGSPACE-complete; this paper, Theorem 15
backward constraints	NLOGSPACE-complete; this paper, Theorem 15	NLOGSPACE-complete; this paper, Theorem 15
constraints	NLOGSPACE-complete; this paper, Theorem 14	NLOGSPACE-complete; this paper, Theorem 15

Containment problem

	non-deterministic graphs	deterministic graphs
forward constraints	PSPACE-hard, in EXPTIME; this paper, Theorem 16	open
backward constraints	PSPACE-hard, in EXPTIME; this paper, Theorem 17	in EXPTIME (L finite); this paper, Lemma 18
constraints	undecidable [13, Theorem 3.1]	undecidable [12, Theorem 6.1]

Table 2: A summary of results on reasoning tasks with path constraints.

It is worth observing that some of our decidability results were obtained by reusing the results of [20]. More generally, there are many areas in computer science in which describing and reasoning about finite graphs is a key issue. There exists a large body of work in e.g., feature structures [42], process algebra [38], or knowledge representation [21] which can be usefully applied in the theory of semistructured datas. But there are differences in the kind of questions asked and in the emphasis in descriptions of linguistic structures, processes, or knowledge on the one hand, and in descriptions of database schemas on the other hand; these differences make the present application interesting and non-trivial.

Our modal logic perspective on standard path constraints moves many decidability and complexity issues for semistructured data into the realm of PDL-like logics. Here are just some of the many remaining open problems:

1. Complexity of the containment problem for the class of forward constraints (respectively backward constraints) (we know PSPACE-hardness and the EXPTIME upper bound). The containment problem for the class of forward constraints cannot be expressed naturally by query containement between a conjunctive two-way regular path query with constants and a tree two-way regular path query with constants, although this problem is in PSPACE [14]. Query containment between conjunctive two-way regular path queries with constants is roughly about the validity of questions of the form $p_1 \cap \cdots \cap p_n \subseteq q_1 \cap \cdots \cap q_m$.

2. Decidability of the containment problem for forward constraints restricted to deterministic L-structures.

3. Decidability of PDL^{path} restricted to deterministic L-structures; decidability of PDL with converse and determinism is a long-standing open problem [43].

4. Extending our results to a richer path expressions language containing additional predicates such as, for example, XPath [45].

Acknowledgments. We are grateful to Ulrike Sattler and the anonymous referees for useful comments. We are especially grateful to Philippe Schnoebelen, who gave us the proof for Lemma 13. This research was supported by the British Council/NWO UK-Dutch Joint Scientific Research Programme under grant JRP548. Natasha Alechina was supported by the EPSRC grant M98050. Maarten de Rijke was supported by grants from the Netherlands Organization for Scientific Research (NWO), under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, and 220-80-001.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann, 2000.

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

[3] S. Abiteboul and V. Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58(3):428–452, 1999.

[4] N. Alechina, S. Demri, and M. de Rijke. Path constraints from a modal logic point of view (extended abstract). In *8th International Workshop on Knowledge Representation meets Databases (KRDB-2001), Roma*, September 2001. Available via `http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS` on WWW.

[5] L. de Alfaro. Model checking the world wide web. In G. Berry, H. Comon, and A. Finkel, editors, *Computer Aided Verification*, pages 337–349. volume 2102 of Lecture Notes in Computer Science, Springer-Verlag, 2001.

[6] H. Andréka, J. van Benthem, and I. Németi. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic*, 27(3):217–274, 1998.

[7] C. Areces, P. Blackburn, and M. Marx. A road-map on complexity for hybrid logics. In J. Flum and M. Rodríguez-Artalejo, editors, *Computer Science Logic*, pages 307–321. volume 1683 of Lecture Notes in Computer Science, Springer Verlag, 1999.

[8] J. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer, Berlin, 2nd edition, 1988.

[9] M. Ben-Ari, J. Halpern, and A. Pnueli. Deterministic propositional dynamic logic: finite models, complexity and completeness. *Journal of Computer and System Sciences*, 25:402–417, 1982.

[10] P. Blackburn and J. Seligman. Hybrid languages. *Journal of Logic, Language and Information*, 4:251–272, 1995.

[11] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *ACM SIGMOD International Conference on the Management of Data*, pages 505–516, 1996.

[12] P. Buneman, W. Fan, and S. Weinstein. Path constraints on deterministic graphs. Technical Report MS-CIS-98-33, LINCS, CIS, UPenn, 1998.

[13] P. Buneman, W. Fan, and S. Weinstein. Path constraints on semistructured and structured data. In *Proceedings PODS'98, Seattle, Washington*, 1998.

[14] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. View-based query answering and query containment over semistructured data. Submitted for journal publication.

[15] L. Capra, E. Ellmer, W. Emmerich, A. Finkelstein, Ch. Nentwich, D. Smolko, and A. Zisman. xlinkit: the consistency checking and link generation engine developed at the University College London. Available from `http://www.xlinkit.com`.

[16] L. Cardelli, Ph. Gardner, and G. Ghelli. A spatial logic for querying graphs. In *ICALP'02, Malaga, Spain*, volume 2380 of *LNCS*, pages 597–610. Springer, Berlin, 2002.

[17] C. Chen and I. Lin. The complexity of propositional modal theories and the complexity of consistency of propositional modal theories. In A. Nerode and Yu. V. Matiyasevich, editors, *LFCS-3, St. Petersburg*, pages 69–80. volume 813 of Lecture Notes in Computer Science, Springer Verlag, 1994.

[18] R. Cleaveland and B. Steffen. A linear-time model-checking algorithm for the alternation-free modal mu-calculus. In K. Larsen and A. Skou, editors, *Computer-Aided Verification (CAV '91), Aalborg, Denmark*, pages 48–58. volume 575 of Lecture Notes in Computer Science, Springer Verlag, 1991.

[19] B. Courcelle. The expression of graph properties and graph transformations in monadic second-order logic. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations - Volume 1: Foundations*, pages 313–400. World Scientific, New-Jersey, London, 1997.

[20] G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms.* PhD thesis, Universitá degli Studi di Roma "La Sapienza", 1995.

[21] F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 191–236. CSLI Publications, 1996.

[22] N.J. Fisher and R.E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Sciences*, 18:194–211, 1979.

[23] G. Gargov and S. Passy. Determinism and looping in combinatory PDL. *Theoretical Computer Science*, 61:259–277, 1988.

[24] V. Goranko. Hierarchies of modal and temporal logics with references pointers. *Journal of Logic, Language and Information*, 5:1–24, 1996.

[25] G. Gottlob and C. Koch. Monadic queries over tree-structured data. In *LICS'02*, pages 189–202. IEEE Computer Society, 2002.

[26] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P-completeness theory.* Oxford University Press, New York, Oxford, 1995.

[27] J. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372, 1995.

[28] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic.* The MIT Press, 2000.

[29] E.R. Harold and W.S. Means. *XML in a Nutshell.* O'Reilly, 2001.

[30] P. Hayes. RDF model theory. W3C Working Draft 25 September 2001. Available from `http://www.w3.org/TR/rdf-mt/`.

[31] E. Hemaspaandra. The price of universality. *Notre Dame Journal of Formal Logic*, 37(2):173–203, 1996.

[32] E. Hemaspaandra. The complexity of poor man's logic. *Journal of Logic and Computation*, 11(4):609–622, 2001.

[33] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the Association for Computing Machinery*, 32(1):137–161, 1985.

[34] H. Hunt, D. Rosenkrantz, and Th. Szymanski. On the equivalence, containment, and covering problems for the regular and context-free languages. *Journal of Computer and System Sciences*, 12:222–268, 1976.

[35] A. Mendelzon, G. Mihaila, and T. Milo. Querying the world wide web. *International Journal on Digital Libraries*, 1(1):54–67, 1997.

[36] A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM Journal on Computing*, 24(6), 1995.

[37] G. Miklau and D. Suciu. Containment and equivalence for an XPath fragment. In *21st ACM Symposium on Principles of Databases Systems (PODS), Madison, Wisconsin*, pages 65–76, 2002.

[38] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[39] R. Parikh. Propositional logics of programs: systems, models and complexity. In *7th Annual ACM Symp. on Principles of Programming Languages*, pages 186–192, 1980.

[40] V.P. Pratt. Models of program logics. In *Proc. 20th IEEE Symp. Foundations of Computer Science*, pages 115–222, 1979.

[41] N. Rescher and A. Urquhart. *Temporal Logic.* Springer-Verlag, 1971.

[42] W.C. Rounds. Feature logics. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 475–534. Elsevier Science, 1997.

[43] U. Sattler and M. Vardi. The hybrid mu-calculus. In A. Leitsch, R. Goré, and T. Nipkow, editors, *Proc. IJCAR 2001*, pages 76–91. volume 2083 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2001.

[44] M. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 32:183–221, 1986.

[45] W3C. XML Path Language (XPath) 1.0. Available from `http://www.w3.org/TR/xpath`.