Path Constraints from a Modal Logic Point of View

Natasha Alechina [*] Stéphane Demri [†] Maarten de Rijke [‡]

Abstract

We analyze several classes of path constraints for semistructured data in a unified framework and prove some decidability and complexity results for these constraints by embedding them in Propositional Dynamic Logic. While some of our decidability results were known before, we believe that our improved complexity bounds are new. Our proofs, based on techniques from modal logic, shed additional light on the reasons for previously known decidability and complexity results.

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, Lorel, and UnQL are similar in spirit if not in syntax, and 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.

[*]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, 61 Avenue du Président Wilson, 94235 Cachan Cedex, France. E-mail: demri@lsv.ens-cachan.fr

[‡]ILLC, University of Amsterdam, Plantage Muidergracht 24, 1018 TV Amsterdam, The Netherlands. E-mail: mdr@wins.uva.nl

We build on results in [2, 9], and embed several classes of path constraints that have been considered in the literature into a well-known modal logic. This embedding establishes a number of things; it shows how various constraints relate to each other, it sheds light on the known decidability results, and it gives rise to new ones.

The paper is organized as follows. Section 2 provides background information on data models and query languages. Section 3 introduces several kinds of path constraints, and in Section 4 we introduce logical formalisms to capture such constraints. In Section 5 we state our complexity and decidability results. We conclude in Section 6.

2 Background

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]. 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.

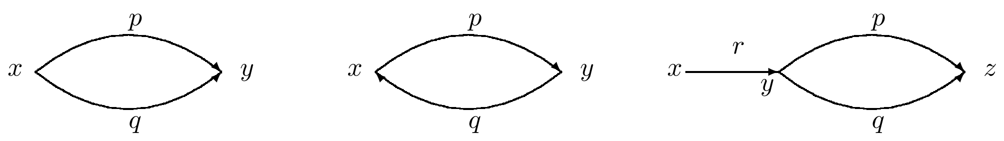
In this paper, we will restrict attention to *rooted connected* graphs: that is, one of the nodes in the graph is designated as the root and every other node is accessible from the root by a directed path of edges. Intuitively, this is because we consider the web from the point of view of browsing, i.e.,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

Definition 1 Let L be a countable set of edge labels. A label $l \in L$, an empty path ϵ and a 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). A *simple path* is a path expression with no occurrence of $\#$, * and $+$.

3 Path Constraints

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



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

Figure 1: Constraints.

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 [2] (see below).

The motivation of the work in [9] is more database-related. One important difference between the constraints considered in [2] and those studied in [9] 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. Another difference is that the former can contain regular expressions, while the latter 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 $p \subseteq_f q$. A path inclusion constraint $p \subseteq_f q$ is *true at a node* x if every node y reachable from x by a path whose labels form a word described by p (i.e., a p-path), is reachable from x by a path whose labels form a word described by q (i.e., q-path). See Figure 1 (a).

The path inclusion constraints defined above are sometimes referred to as *forward* constraints. In [9], *backward* constraints are introduced. We generalize their definition for a language containing regular expressions.

Definition 3 Let p and q be two path expressions. A *backward* path constraint $p \subseteq_b q$ is true at a node x if for every y reachable from x by a p-path, it is possible to come back to x by a q-path. 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 *path constraint* (notation: $p \subseteq q$) is either a path inclusion constraint or a backward constraint. The next class of constraints is a generalization of path constraints as defined in [9] for a language containing regular expressions:

Definition 4 Let p and q be two path expressions. A *lollipop path constraint* is an expression of the form $r \rightsquigarrow p \subseteq q$. A lollipop path constraint $r \rightsquigarrow p \subseteq q$ is *true at a node* x if at every node y reachable from x by an r-path, the path constraint $p \subseteq q$ holds. See Figure 1 (c).

Obviously, a path inclusion constraint $p \subseteq_f q$ is a lollilop path constraint $r \rightsquigarrow p \subseteq_f q$ with $r = \epsilon$.

4 Reasoning about Path Constraints

Now that we have formulated path constraints, we take a closer look at reasoning tasks involving them. These include checking whether a certain constraint holds or whether a certain set of constraints is consistent or implies another constraint. To determine the computational costs of these tasks, we recast them as model checking, satisfiability checking, and implication checking tasks in some logic. Which logic (or logics) should we use? Many formalisms have been proposed for reasoning about graphs. As we will see below, many decidable classes of constraints are definable in suitable modal logics, while constraints that lack a modal flavor (such as the ones studied in [9]) 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' [3].

Thus, we translate constraints into formulas of a flavor of Propositional Dynamic Logic (PDL, [17]) and reformulate reasoning tasks for constraints as reasoning tasks within this flavor of PDL. 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 *id* to denote the diagonal relation. Compound transition terms correspond to path expressions and are built from atomic ones using ;, + and $*$.

In addition to these traditional ingredients of PDL, we add a *wildcard* # and a *converse operator* $(\breve{\cdot})$ (also written as $(\cdot)^{-1}$): # is a transition term, and if t is a transition term, then so is $\breve{t}$ (the labels of t in reverse order). PDL with converse is called *converse PDL* (CPDL). We obtain *CPDL with nominals* by extending CPDL with special propositional symbols, called nominals, that are true of at most one node in a graph; our symbol *root* is an example of a nominal.

Definition 5 (PDL^{path}) We now define the legal formulas of PDL^{path}. Relational terms are typically denoted by t, atomic terms by a, and formulas are typically denoted by ϕ:

$$\begin{array}{rcl} t & ::= & id \mid a \mid \# \mid t+t \mid t\,;t \mid t^* \mid \breve{t} \\ \phi & ::= & \top \mid \bot \mid root \mid \neg\phi \mid \phi\wedge\phi \mid \langle t\rangle\phi \mid [t]\phi. \end{array}$$

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 structures of the form $G = (V, rt, \{R_a : a \in L\})$, where $rt \in V$ is the root and, for every label a, R_a is a binary relation on V. These relations represent the edges of the semistructured data viewed as an edge-labeled graph. Sometimes, we restrict ourselves to so-called *deterministic* structures; these are structures $(V, rt, \{R_a : a \in L\})$ that satisfy the following condition, for all states u, v, $w \in V$ and edge labels $a \in L$: if $uR_a v \wedge uR_a w$ then $v = w$.

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 need first to define *transition relations* $tr(t)$ on $V \times V$ corresponding to the transition terms t:

$tr(a) = R_a$ for $a \in L$

$tr(t^*)$ is the reflexive transitive closure of $tr(t)$

$tr(\#) = \bigcup_{a \in L} R_a$

$tr(\breve{t}) = \{\langle u, v\rangle : \langle v, u\rangle \in tr(t)\}$

$tr(id) = \{\langle u, u\rangle : u \in V\}$

$tr(t_1 \,; t_2) = \{\langle u, v\rangle : \exists z(tr(t_1)(u, z) \wedge tr(t_2)(z, v))\}$

$tr(t_1 + t_2) = tr(t_1) \cup tr(t_2)$

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 a structure* (edge-labeled rooted graph) G if it is true at the root of G. A lollipop path constraint $r \rightsquigarrow p \subseteq q$ is *true on a structure* G (in symbols $G \models r \rightsquigarrow p \subseteq q$) iff $r \rightsquigarrow p \subseteq q$ is true at the root of G.

Definition 7 (Reasoning Tasks) The *model checking problem* is to decide, given a structure G and a formula ϕ, whether ϕ is true on G. The *satisfiability problem* for PDL^{path} is to determine, given a PDL^{path} formula ϕ, whether there is a structure G such that ϕ is true at the root of G. The *implication problem* for path inclusion constraints is to determine, given constraints $\phi_1, \ldots, \phi_n$ and ϕ, whether it the case that for all structure G, $G \models \phi_1, \ldots, G \models \phi_n$ imply $G \models \phi$?

The implication problem for other classes of constraints can be defined accordingly. Below we reduce the implication problem to the satisfiability problem; see Theorem 11.

5 Complexity and Decidability Results

5.1 The Model Checking Problem

Given a formula ϕ, define $|\phi|$, the *length* of ϕ, as the number of symbols in ϕ. Given a structure $G = (V, rt, \{R_a : a \in L\})$ with a finite domain V and a finite set of atomic edge labels L, define $||G||$, the *size* of G, to be the sum of the number of states in V and the number of pairs in $\bigcup\{R_a : a \in L\}$.

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

Theorem 8 There is an algorithm that, given a finite (deterministic or non-deterministic) graph G, a node w of G, and a PDL^{path} formula ϕ, determines, in time $O(||G|| \times |\phi|)$ whether $G, w \models \phi$.

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 $O(||G|| \times |\phi|)$. This follows from the fact that model-checking for the alternation-free modal μ-calculus is in linear-time (see e.g. [11]).

The linear time reduction works as follows. First, given G, we construct a new graph G'. G' has the same vertices and root, and contains all the edges which G does plus, for every edge $u \xrightarrow{a} v$ in G we add two more edges to G': $u \xrightarrow{\#} v$ and $v \xrightarrow{\breve{a}} u$. Construction of G' is obviously linear in the size of G. Second, we rewrite ϕ so that all occurrences of $\breve{}$ are on the atomic labels. This can be done in linear time by using the following equivalences:

$$(\breve{\alpha|\beta}) = (\breve{\alpha}|\breve{\beta})$$

$$(\breve{\alpha;\beta}) = (\breve{\beta};\breve{\alpha})$$

$$(\breve{\alpha^*}) = (\breve{\alpha})^*$$

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

5.2 The Satisfiability Checking Problem

The satisfiability problem for PDL^{path} on non-deterministic graphs can be proved to be decidable by a reduction to the decidability of converse PDL with nominals.

Theorem 9 On non-deterministic graphs, the satisfiability problem for PDL^{path} is decidable.

Proof. We use the fact that CPDL with nominals is decidable [12, Theorem 49] and reduce satisfiability in PDL^{path} to satisfiability in CPDL with nominals.

First consider an easy case: 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 $\{l_1, \ldots, l_n\}$ and possibly #, we construct a CPDL formula ϕ' by replacing # with $l_1 + \cdots + l_{n+1}$ in ϕ, and we show that ϕ is satisfiable iff $\mathit{root} \wedge \phi'$ is.

Observe that if in a model M all edge labels are in the set $\{l_1, \ldots, l_k\}$, then $tr(\#) = tr(l_1 + \cdots + l_k)$. Suppose $M, u \models \phi$. Let M' be obtained from M by replacing all labels not in $\{l_1, \ldots, l_n\}$ by l_{n+1}. It is easy to show that $M', u \models \phi$. By our observation $M', u \models \phi'$. For the converse, assume that $M, u \models \phi'$. Let M' be obtained from M by deleting all edges labeled by modalities not in $\{l_1, \ldots, l_{n+1}\}$. It is easy to see that $M', u \models \phi'$. In M', $\langle \# \rangle$ is definable as a union of $\{l_1, \ldots, l_{n+1}\}$, hence on M', ϕ' and ϕ are equivalent and we obtain $M', u \models \phi$. □

By itself Theorem 9 does not imply an analogous result for deterministic graphs 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 graphs, the satisfiability problem for PDL^{path} is decidable.

In the non-deterministic case, we can actually do better than Theorem 9, and obtain matching lower and upper bounds for the complexity of the satisfiability problem for PDL^{path}.

Theorem 10 On non-deterministic graphs, 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 [12, page 98], 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 reduce the global satisfiability problem for the standard modal logic K (known to be **EXPTIME**-hard, see e.g. [10, 18]) into PDL^{path}-satisfiability restricted to the modal connectives $\langle a_0 \rangle$ and $\langle a_0^{-1} \rangle$. To do so, we take advantage of the spy-point technique [6] by adapting the proof of [4, Theorem 2]. The only difficulty is now to use the spy-point technique and simultaneously to encode the propositional variables.

Let us define a family $(\varphi_i)_{i \in \mathbb{N}}$ of PDL^{path}-formulae encoding propositional variables.

- $\varphi_1 \stackrel{\text{def}}{=} \langle a_0^{-1} \rangle (\langle a_0 \rangle \mathit{root} \wedge \langle a_0^{-1} \rangle \mathit{root})$;
- $\varphi_{i+1} \stackrel{\text{def}}{=} \langle a_0^{-1} \rangle (\neg \mathit{root} \wedge \neg \langle a_0^{-1} \rangle \mathit{root} \wedge \varphi_i)$.

Encoding of propositional variables can be also found in [16, 19] but there are of different nature since we tailor our encoding to the spy-point technique. Con-

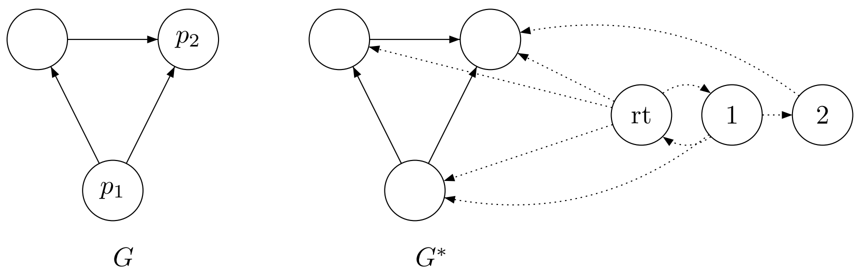


Figure 2: G and G*: an example

sider the mapping f from K-formulae into PDL^{path}-formulae:

$$\begin{array}{rcl} f(p_i) & = & \varphi_i \text{ for } i \geq 1 \\ f & & \text{commutes with the Boolean connectives} \\ f(\Diamond\phi) & = & \langle a_0\rangle(\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \wedge f(\phi)). \end{array}$$

We shall show that ϕ is globally K-satisfiable iff $[a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \Rightarrow f(\phi))$ is PDL^{path}-satisfiable. Without any loss of generality, we can assume that if N distinct propositional variables occur in ϕ, then they are $p_1, \ldots, p_N$.
$(\rightarrow)$ Assume that $G \models \phi$ for some Kripke structure $G = \langle V, R, m\rangle$. Let $G^* = \langle V^*, rt, R_{a_0}\rangle$ be the PDL^{path}-model such that

- $V^* \overset{\text{def}}{=} V \cup \{rt\} \cup \{1, \ldots, N\}$ where rt is not in $V \cup \{1, \ldots, N\}$ and $V \cap \{1, \ldots, N\} = \emptyset$ (N is the number of propositional variables in ϕ);
- $R_{a_0} \overset{\text{def}}{=} R \cup \{\langle rt, w\rangle : w \in V\} \cup \{\langle 1, 2\rangle, \ldots, \langle N-1, N\rangle\} \cup \{\langle 1, rt\rangle, \langle rt, 1\rangle\} \cup \{\langle i, w\rangle : G, w \models p_i\}$;
- the interpretation of $root$ is rt.

The root rt is R_{a_0}-connected to the nodes in $V \cup \{1\}$. The node 1 is distinguished from the nodes from V since it is the only node in $R_{a_0}(rt) \cap R_{a_0}^{-1}(rt)$. In Figure 2, we give a basic example of the construction.

Let us show that for all $w \in V$, for all $\psi \in \text{sub}(\phi)$ (set of subformulae of ϕ), $G, w \models \psi$ iff $G^*, w \models f(\psi)$.

In order to show the base case (for $i \in \{1, \ldots, N\}$, $G, w \models p_i$ iff $G^*, w \models \varphi_i$), one can easily show by induction on i that $\{w \in V^* : G^*, w \models \varphi_i\} = \{w \in V : G, w \models p_i\} \cup (\{i+1\} \setminus \{N+1\})$.

We treat in more details the case $\psi = \Diamond\psi'$. $G, w \models \Diamond\psi'$ iff there is $w' \in R(w)$ such that $G, w' \models \psi'$ iff (i) there is $w' \in V$ such that $\langle w, w'\rangle \in R_{a_0}$, $\langle r, w'\rangle \in R_{a_0}$, $\langle w', r\rangle \not\in R_{a_0}$ and $G, w' \models \psi'$. Furthermore, (i) 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 $w \in V$, $G^*, w \models f(\phi)$ since $G \models \phi$. 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, $G^*, rt \models [a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \Rightarrow f(\phi))$.

($\leftarrow$) 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 PDL^{path}-model $G = \langle V, rt, R_{a_0}\rangle$. Let $G^* = \langle V^*, R, m\rangle$ be the Kripke structure such that $V^* \overset{\text{def}}{=} \{w \in V : \langle rt, w\rangle \in R_{a_0}, \langle w, rt\rangle \notin R_{a_0}\}$, R is the restriction of R_{a_0} to V^* and for $i \in \{1, \ldots, N\}$, $m(p_i) \overset{\text{def}}{=} \{w \in V^* : G, w \models \varphi_i\}$. Let us show that for $w \in V^*$, for $\psi \in \text{sub}(\phi)$, $G, w \models f(\psi)$ iff $G^*, w \models \psi$. The base case is obvious. We treat in more details the case $\psi = \Diamond\psi'$. $G, w \models \langle a_0\rangle(f(\psi') \wedge \langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root)$ 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 $G^*, w' \models \psi'$ iff $G, w \models \psi$. Since $G, rt \models [a_0](\langle a_0^{-1}\rangle root \wedge \neg\langle a_0\rangle root \Rightarrow f(\phi))$, for all $w \in R_{a_0}(rt) \cap V^*$, $G^*, w \models \phi$. Since $V^* \subseteq R_{a_0}(rt)$, for all $w \in V^*$, $G^*, w \models \phi$.

□

As a corollary to the proof of Theorem 10, the minimal tense logic augmented with a single nominal but without proposition letters also has an EXPTIME-hard satisfiability problem.

5.3 The Implication Problem

Our next aim is to obtain sharp complexity results for implication problems for constraints. We start by considering non-deterministic graphs, and the first step is to show the following.

Theorem 11 On non-deterministic graphs, the implication problem for path inclusion constraints is decidable in exponential time, while it is at least PSPACE-hard whenever $|L| \geq 2$.

Proof. The EXPTIME upper bound is from Theorem 10 and the fact that the path inclusion $p \subseteq_f q$ can simply be translated as $[p]\langle \breve{q}\rangle root$. As to the lower bound PSPACE, each relational term t built with $+, ;, ^*$ over the atomic terms $a_0, a_1, \ldots$ can be viewed in the obvious way as regular expressions and we write $L(t)$ to denote the language generated by t. By [20, 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 an easy task to check 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 implication problem restricted to two labels and without inclusion constraints as premisses, is already PSPACE-hard. □

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

Theorem 12 On non-deterministic graphs, the implication problem for backward constraints is decidable in exponential time, while it is at least PSPACE-hard whenever $|L| \geq 2$.

Proof. Since converse is not present in the path expressions, we cannot use the proof of Theorem 11. However, one can show that for any relational term t built over $\{a_0, a_1\}$, $L((a_0 + a_1)^*) \subseteq L(t)$ iff for any structure G, $G \models t \ \subseteq_b \ a_2$ implies $G \models (a_0 + a_1)^* \ \subseteq_b \ a_2$. $\Box$

We now restrict attention to *deterministic* graphs, which makes a substantial difference. The results known so far (see [8]) are: all path constraints without regular expressions (including lollipop path constraints) are decidable on deterministic graphs. Lollipop path constraints that do contain regular expressions are undecidable. Below we consider the case of constraints with regular expressions.

By using the previous correspondences between path constraints and PDL^{path} formulae, one can easily show the following result.

Lemma 13

1. On deterministic graphs, the implication problem for path inclusion constraints is reducible to the satisfiability problem for PDL^{path} on deterministic graphs.

2. On deterministic graphs, the implication problem for backward constraints is reducible to the satisfiability problem for PDL^{path} without converse on deterministic graphs.

As a consequence, since DPDL with nominals is known to be decidable in exponential time [15, 23], we get the following result.

Theorem 14 On deterministic graphs, the implication problem for backward constraints for finite sets of labels L is decidable in exponential time.

The latter result provides a partial positive answer to the last open question from [8]. However, it is open whether on deterministic graphs, the implication problem (for path inclusion constraints) is decidable. Similarly, the decidability of the implication problem for backward constraints (without restrictions on L) is open.

On deterministic graphs, the implication problem with lollipop constraints of the form $r \ \rightsquigarrow \ p \ \subseteq_f \ q$ is undecidable even if L contains only two labels [8]. However, in the lollipop constraints used in the proof the operator * occurs in r, and this is used to encode the word problem.

6 Conclusions

We have given new and transparent decidability proofs for the path inclusion constraints proposed in [2] for optimizing queries on semistructured data, mostly in the context of the web. We have obtained sharp upper and lower bounds that are better than previously known ones (Theorems 11 and 12). Table 1 summarizes the complexity and (un-) decidability results for the logics considered in this paper.

Model checking problem

	non-deterministic graphs	deterministic graphs
PDL	$O(\|\|G\|\| \times \|\phi\|)$ [11]	$O(\|\|G\|\| \times \|\phi\|)$ [11]
PDL^{path}	$O(\|\|G\|\| \times \|\phi\|)$; this this paper, Theorem 8	$O(\|\|G\|\| \times \|\phi\|)$; this paper, Theorem 8

Satisfiability problem

	non-deterministic graphs	deterministic graphs
PDL	EXPTIME-complete [14, 24]	EXPTIME-complete [22, 5]
PDL with nominals	EXPTIME-complete [15]	EXPTIME-complete [15]
CPDL	EXPTIME-complete []	EXPTIME-complete [27]
CPDL with nominals	EXPTIME-complete [12, 4]	open
PDL^{path}	EXPTIME-complete; this paper, Theorem 10	open

Implication problem

	non-deterministic graphs	deterministic graphs
inclusion constraints	PSPACE-hard, in EXPTIME; this paper, Theorem 11	open
backward constraints	PSPACE-hard, in EXPTIME; this paper, Theorem 12	in EXPTIME (with finite L); this paper, Theorem 14
lollipop constraints	undecidable [9, Theorem 3.1]	undecidable [8, Theorem 6.1]

Table 1: A summary

Some of our decidability results were obtained by re-using the results of [12]. In fact, 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 [25], process algebra [21], or knowledge representation [13] which can be usefully applied in database theory. 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, which makes the present application interesting and non-trivial.

Our modal logic perspective on 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 interesting open problems:

1. Complexity of the implication problem (we know PSPACE-hardness and the EXPTIME upper bound).

2. Decidability of the implication problem for forward constraints on deterministic graphs.

3. Decidability of PDL^{path} on deterministic graphs; decidability of PDL with converse and determinism is a long-standing open problem [26].

Acknowledgments. We are grateful to Peter Buneman, Neil Immerman, Achim Jung, Brian Logan, Ulrike Sattler and Alan Sexton for useful comments. This research was supported by the British Council/NWO UK-Dutch Joint Scientific Research Programme under grant JRP548. Maarten de Rijke was also supported by the Spinoza project 'Logic in Action' and 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 and V. Vianu. Regular path queries with constraints. In *Proceedings PODS'97*, pages 122–133, 1997.

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

[4] 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. LNCS 1683, Springer, 1999.

[5] 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.

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

[7] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. Adding structure to unstructured data. In *Proceedings ICDT'97*, pages 336–350, 1997.

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

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

[10] 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. Springer-Verlag, LNCS 813, 1994.

[11] 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. LNCS 575, Springer-Verlag, 1991.

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

[13] 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.

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

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

[16] 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.

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

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

[19] E. Hemaspaandra. The complexity of poor man's logic. In H. Reichel and S. Tison, editors, *STACS'00.* LNCS 1770, Springer Verlag, 2000.

[20] 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.

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

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

[23] S. Passy and T. Tinchev. An essay in combinatory dynamic logic. *Information and Computation*, 93:263–332, 1991.

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

[25] 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.

[26] U. Sattler and M. Vardi. The hybrid mu-calculus. In A. Leitsch, R. Goré, and T. Nipkow, editors, *Proceedings IJCAR 2001*, pages 76–91. LNAI 2083, Springer-Verlag, 2001.

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