Path Constraints on Deterministic Graphs

Peter Buneman[*] peter@central.cis.upenn.edu
Wenfei Fan[†] wfan@saul.cis.upenn.edu
Scott Weinstein[‡] weinstein@linc.cis.upenn.edu

Department of Computer and Information Science
University of Pennsylvania

September 1998

(Revised April 1999)

Abstract

Path constraints have been studied in [4, 10, 11] for semistructured data modeled as a rooted edge-labeled directed graph. They have proven useful in the optimization of path queries. However, in this graph model, the implication problems associated with many natural path constraints are undecidable [10]. A variant of the graph model, called *the deterministic data model*, was recently proposed in [9]. In this model, data is represented as a graph with deterministic edge relations, i.e., the edges emanating from any node in the graph have distinct labels. The deterministic graph model is more appropriate for representing, for example, ACeDB [25] databases and Web pages.

This paper investigates path constraints for the deterministic data model. It demonstrates the application of path constraints to, among other things, query optimization. Four classes of path constraints are considered: the class of word constraints P_w proposed in [4], the constraint language P_c introduced in [10], an extension of P_c, denoted by P_c^-, by including wildcards in path expressions, and a generalization of P_c^-, denoted by P_c^*, by representing paths as regular expressions. The implication problems for these constraint languages are studied in the context of the deterministic data model. It shows that the implication and finite implication problems for P_w are decidable in cubic-time and are finitely axiomatizable. Moreover, in contrast to the undecidability result of [10], these results also hold for P_c. In addition, the implication problems are decidable for P_c^-. However, the implication problems for P_c^* are undecidable.

[*]Supported partly by the Army Research Office (DAAH04-95-1-0169) and NSF Grant CCR92-16122.

[†]Supported by a graduate fellowship from the Institute for Research in Cognitive Science, University of Pennsylvania.

[‡]Supported by NSF Grant CCR-9403447.

1 Introduction

Semistructured data is usually modeled as an edge-labeled rooted directed graph [1, 7]. Let us refer to this graph model as the *semistructured data model* (SM). For data found in many applications, the graph is *deterministic*, i.e., the edges emanating from each node in the graph have distinct labels. For example, when modeling Web pages as a graph, a node stands for an HTML document and an edge represents a link with an HTML label from one document (source) to another (target). It is reasonable to assume that the HTML label uniquely identifies the target document. Even if this is not literally the case, one can achieve this by including the URL (Universal Resource Locator) of the target document in the edge label. This yields a deterministic graph. As another example, consider ACeDB [25], which is a database management system popular with biologists. A graph representing an ACeDB database is also deterministic. In general, any database with "exportable" data identities can be modeled as a deterministic graph by including the identities in the edge labels. Here by exportable identities we mean directly observable identities such as keys. Some relational and object-oriented database management systems support exportable identities. In particular, in the OEM model (see, e.g., [3]), there are exportable object identities. To capture this, we consider a data model for semistructured data which is a variant of SM, referred to as *the deterministic data model* (DM). In DM, data is represented as a deterministic, rooted, edge-labeled, directed graph. An important feature of DM is that in this model, each component of a database is uniquely identified by a path.

A number of query languages (e.g., [3, 8, 13, 22]) have been developed for semistructured data. The study of semistructured data has also generated the design of query languages (e.g., [12]) for XML (eXtensible Markup Language [6]) documents. In these languages, queries are described in terms of navigation paths. To optimize path queries, it often appears necessary to use structural information about the data described by path constraints. Path constraints are capable of expressing natural integrity constraints that are a fundamental part of the semantics of the data, such as inclusion dependencies and inverse relationships. In traditional structured databases such as object-oriented databases, this semantic information is described in schemas. Unlike structured databases, semistructured data does not have a schema, and path constraints are used to convey the semantics of the data. The approach to querying semistructured data with path constraints was proposed in [4] and later studied in [10, 11]. Several proposals (e.g., [5, 15, 19, 20]) for adding structure or type systems to XML data also advocate the need for integrity constraints that can be expressed as path constraints.

To use path constraints in query optimization, it is important to be able to reason about them. That is, we need to settle the question of constraint implication: given that certain constraints are known to hold, does it follow that some other constraint is necessarily satisfied? In the context of databases, only finite instances (graphs) are considered, and constraint implication is referred to as *finite implication*. In the traditional logic framework, both infinite and finite instances (graphs) are permitted, and constraint implication is called *unrestricted implication* or simply *implication*. For the graph model SM, it has been shown that the implication problems associated with many natural integrity constraints are undecidable. For example, the implication problem for the simple constraint language P_c studied in [10, 11] is

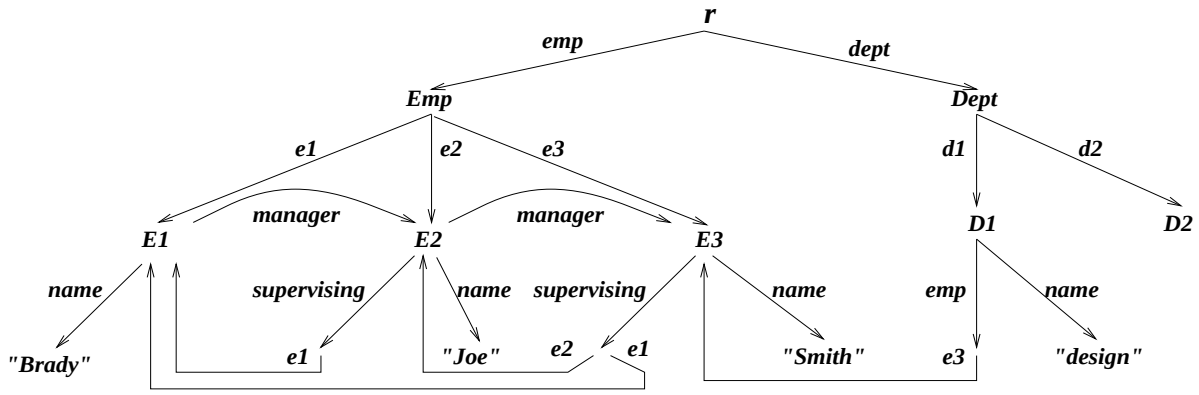

Figure 1: An example semistructured database in DM

r.e. complete, and the finite implication problem for P_c is co-r.e. complete [10].

In this paper, we investigate path constraints for the deterministic data model DM. We demonstrate applications of path constraints to semantic specification and query optimization, and study the implication problems associated with path constraints. Four constraint languages are considered. We first investigate the class of word constraints P_w proposed in [4] and the path constraints language P_c introduced in [10]. In the context of SM, it has been shown that word constraint implication is finitely axiomatizable and is decidable in PTIME [4], and the implication problems for P_c are undecidable [10]. However, we show that in the context of DM, the set of inference rules given in [4] is no longer complete for word constraint implication. For DM, we present a finite set of inference rules that is sound and complete for word constraint implication, and develop an algorithm for testing word constraint implication in time $O(n^3)$, where n is the length of constraints. In addition, we show that in contrast to the undecidability result of [10], the implication and finite implication problems for P_c are also decidable in cubic-time and are finitely axiomatizable in the context of DM. This demonstrates that the determinism condition of DM simplifies the analysis of path constraint implication. We also introduce and investigate two generalizations of P_c. One generalization, denoted by P_c^-, is defined by including wildcards in path expressions. The other, denoted by P_c^*, represents paths by regular expressions. We show that in the context of DM, the implication and finite implication problems for P_c^- are also decidable. However, the implication and finite implication problems for P_c^* are undecidable in the context of DM. This undecidability result shows that the determinism condition of DM does not reduce the analysis of path constraint implication to a trivial problem.

An example. To demonstrate applications of path constraints, let us consider Figure 1, which collects information on employees and departments. It is an example of semistructured data represented in the deterministic data model. In Figure 1, there are two edges emanating from the root node r, which are labeled `emp` and `dept` and connected to nodes `Emp` and `Dept`, respectively. Edges emanating from `Emp` are labeled with employee ID's and connected to vertices representing employees. An employee node may have three edges emanating from it: an edge labeled `manager` and connected to his/her manager, an edge labeled `supervising` that

connects to a node from which there are outgoing edges connected to employees under his/her supervision, and an edge labeled `name`. Similarly, there are vertices representing departments that may have edges connected to employees. Observe that Figure 1 is deterministic.

Path constraints. Typical path constraints on Figure 1 include:

$$\forall\, x\,(emp \cdot \_ \cdot manager(r,\, x) \rightarrow emp \cdot \_\,(r,\, x)) \qquad (\phi_1)$$
$$\forall\, x\,(emp \cdot \_ \cdot supervising \cdot \_\,(r,\, x) \rightarrow emp \cdot \_\,(r,\, x)) \qquad (\phi_2)$$
$$\forall\, x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall\, y\,(manager(x,\, y) \rightarrow supervising \cdot \_\,(y,\, x))) \qquad (\phi_3)$$

Here r is a constant denoting the root of the graph, variables x and y range over vertices, and "`_`" is a "wildcard" symbol, which matches any edge label. A path in the graph is a sequence of edge labels, which can be expressed as a logic formula $\alpha(x,\, y)$ that holds in the graph if α is a sequence of edge labels from vertex x to y. For example, `emp`$\cdot$`e1`$\cdot$`manager` is a path and can be expressed as a logic formula, which holds in Figure 1. Path formulas can be naturally generalized to include wildcards. The path constraints above describe inclusion relations. More specifically, ϕ_1 states that if a node is reached from the root r by following `emp`$\cdot$`_`$\cdot$`manager`, then it is also reachable from r by following `emp`$\cdot$`_`. It asserts that the manager of any employee is also an employee that occurs in the database. Similarly, ϕ_2 states that if a node is reached from r by following `emp`$\cdot$`_`$\cdot$`supervising`$\cdot$`_`, then it is also reachable from r by following `emp`$\cdot$`_`. Constraint ϕ_3 states that for any employee x and for any y, if x is connected to y by a `manager` edge, then x is reachable from y by following `supervising`$\cdot$`_`. These are constraints of P_c^-, one of the path constraint languages introduced and studied in this paper.

We generalize P_c^- by representing paths as regular expressions. This generalization is denoted by P_c^*. For example, the following are constraints of P_c^*:

$$\forall\, x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall\, y\,(manager \cdot manager^*(x,\, y) \rightarrow supervising \cdot \_\,(y,\, x))) \qquad (\psi_1)$$
$$\forall\, x\,(emp \cdot \_\,(r,\, x) \rightarrow \forall\, y\,(supervising \cdot \_\,(x,\, y) \rightarrow manager \cdot manager^*(y,\, x))) \qquad (\psi_2)$$

Here $*$ is the Kleene closure. These constraints describe an inverse relationship between `manager`$\cdot$`manager`* and `supervising`$\cdot$`_`. More specifically, ψ_1 asserts that for any employee x and for any y, if y is reachable from x by following one or more `manager` edges, then x is reachable from y by following path `supervising`$\cdot$`_`. Similarly, ψ_2 asserts that if y is reachable from x by following `supervising`$\cdot$`_`, then x is reachable from y by following one or more `manager` edges.

A subclass of P_c^*, P_c, has been investigated in [10, 11] for the graph model SM for semistructured data. As opposed to P_c^* constraints, path constraints of P_c contain neither wildcards nor the Kleene star. In the deterministic data model, P_c constraints express path equalities. For example, the following can be described by P_c constraints:

$$emp \cdot e1 \cdot manager \;=\; emp \cdot e2 \qquad (\varphi_1)$$
$$dept \cdot d1 \cdot emp \cdot e1 \;=\; emp \cdot e1 \qquad (\varphi_2)$$

These can also be expressed as word constraints introduced in [4]. However, the following is a P_c constraint but is not an example of word constraint:

$$\forall\, x\, (emp \cdot e1\ (r,\ x) \rightarrow \forall\, y\, (manage(x,\ y) \rightarrow supervising \cdot e1(y,\ x))) \qquad (\varphi_3)$$

Observe that the paths in the P_c constraints above do not contain wildcards and the Kleene closure.

Semantic specification with path constraints. The path constraints above describe certain typing information about the data. For example, abusing object-oriented database terms, ϕ_1 asserts that a manager of an employee has an "`employee` type", and in addition, is in the "extent" of "class" `employee`. By using ϕ_1, it can be shown that for any employee x and any y, if y is reachable from x by following zero or more `manager` edges, then y also has an "`employee` type" and is in the "extent" of `employee`. A preliminary type system was proposed in [9] for the deterministic data model, in which the types of paths are defined by means of path constraints. This is a step in unifying the (programming language) notion of a type with the (database) notion of a schema.

Query optimization with path constraints. To illustrate how path constraints can be used in query optimization, consider again the database represented in Figure 1. Suppose, for example, we want to find the name of the employee with ID $e1$ in department $d1$. One may write the query as Q_1 (in Lorel syntax [3]):

Q_1:
```
select X.name
from   r.dept.d1.emp.e1 X
```

Given path constraint φ_2, the query Q_1 can be rewritten as Q_1':

Q_1':
```
select X.name
from   r.emp.e1 X
```

One can easily verify that Q_1 and Q_1' are equivalent.

As another example, suppose we want to find the names of the employees connected to Smith by one or more `manager` edges. Without path constraints, one would write the query as Q_2 (in Lorel syntax):

Q_2:
```
select X.name
from   r.emp.% X, X(.manager)+ Y
where  Y.name = "Smith"
```

In Lorel, `%` denotes wildcard and `(.manager)+` means one or more occurrences of `.manager`. Given constraints ψ_1, ψ_2, ϕ_1 and ϕ_2, we can rewrite Q_2 as Q_2', which finds the names of the employees under the supervision of Smith:

Q_2':
```
select X.name
from   r.emp.% Y, Y.supervising.% X
where  Y.name = "Smith"
```

It can be verified that given those path constraints, Q_2 and Q_2' are equivalent. In addition, Q_2' is more efficient than Q_2 because it does not require the traversal of sequences of `manager` edges. It should be mentioned that to show that Q_2 and Q_2' are equivalent, we need to verify that certain constraints necessarily hold given that ψ_1, ψ_2, ϕ_1 and ϕ_2 hold. That is, they are implied by ψ_1, ψ_2, ϕ_1 and ϕ_2. In particular, we need to show that ψ_3 below is implied by ψ_1, ψ_2, ϕ_1 and ϕ_2:

$$\forall\, x\, (emp \cdot \_ \cdot manager^*(r,\, x) \rightarrow emp \cdot \_\, (r,\, x)) \qquad (\psi_3)$$

Related work. A more general deterministic data model, DDM, was proposed in [9]. In DDM, edge labels also have structure, and a number of database operations may be obtained by manipulation of this structure. In particular, annotations can be described in this structure for the purpose of data provenance, i.e., to keep track by what process some piece of data got into the database. To simplify the discussion we do not consider this general model here.

Path constraints have been studied in [4, 10, 11]. The constraints of [4] have either the form $p \subseteq q$ or $p = q$, where p and q are path expressions represented by regular expressions. When p and q are simply paths, i.e., sequences of edge labels, the constraint is called a *word constraint*. These constraints were investigated for the graph model SM for semistructured data. The decidability of the implication problems for this form of constraints was established in [4] in the context of SM. A mild generalization of word constraints, P_c, was introduced and studied in [10] for SM. It was shown there that despite the simple syntax of P_c, its associated implication and finite implication problems are undecidable in the context of SM. The interaction between P_c constraints and type systems was investigated in [11]. However, none of these papers has considered the deterministic data model. In addition, path constraint languages P_c^- and P_c^* were not studied in these papers.

Recently, the application of integrity constraints to query optimization was also studied in [23]. Among other things, [23] developed an equational theory for query rewriting by using a certain form of constraints.

The results established on path constraint implication in this paper may find applications to other fields. Indeed, if we view vertices in a graph as states and labeled edges as actions, then the deterministic graphs considered here are in fact Kripke models studied in deterministic propositional dynamic logic (DPDL. See, e.g., [17, 26]), which is a powerful language for describing programs. These deterministic graphs may also be viewed as feature structures studied in feature logics [24]. They are also studied in deterministic transitive closure logics (DTCLs. See, e.g., [16, 18]). It should be mentioned that DPDL and feature logics are modal logics, in which our path constraints are not expressible. The path constraint languages P_w, P_c and P_c^- considered here can be viewed as decidable fragments of DTCLs, which possess undecidable implication problems.

Organization. The rest of the paper is organized as follows. Section 2 reviews the definitions of P_w and P_c constraints proposed in [4, 10], and introduces two extensions of P_c, namely, P_c^- and P_c^*. Sections 3, 4, 5 and 6 study the implication and finite implication problems for P_w, P_c, P_c^- and P_c^* for the deterministic data model, respectively. Finally, Section 7 identifies open problems and directions for further work.

2 Deterministic graphs and path constraints

In this section, we first give an abstraction of semistructured databases in DM in terms of first-order logic, and then present four path constraint languages: P_w, P_c, P_c^- and P_c^*.

2.1 The deterministic data model

In the graph model SM, a database is represented as an edge-labeled rooted directed graph [1, 7]. An abstraction of databases in SM has been given in [10] as (finite) first-order logic structures of a relational signature

$$\sigma = (r,\ E),$$

where r is a constant denoting the root and E is a finite set of binary relation symbols denoting the edge labels.

In the deterministic data model DM, a database is represented as an edge-labeled rooted directed graph with deterministic edge relations. That is, for any edge label K and node a in the graph, there exists at most one edge labeled K going out of a. Along the same lines of the abstraction of databases in SM, we represent a database in DM as a (finite) σ-structure satisfying the *determinism condition*:

$$\bigwedge_{K \in E} \forall\, x\, y\, z\, (K(x,\ y) \wedge K(x,\ z) \rightarrow y = z).$$

Such structures are called *deterministic structures*. A deterministic structure G is specified by $(|G|,\ r^G,\ E^G)$, where $|G|$ is the set of nodes in G, r^G is the root node, and E^G is the set of binary relations on $|G|$, each of which is named by a relation symbol of E.

In a deterministic structure, a path is a sequence of edge labels. Formally, *paths* are defined by the syntax:

$$\rho\ ::=\ \epsilon \mid K \mid K \cdot \rho$$

Here ϵ is the empty path, $K \in E$, and $\cdot$ denotes path concatenation. Paths defined above are the simplest form of path expressions. We shall present more general forms of path expressions shortly in this section.

A path ρ is said to be a *prefix* of ϱ if there exists γ, such that $\varrho = \rho \cdot \gamma$.

We have seen many examples of paths in Section 1. Among them are:

$$emp \cdot e1 \cdot manager$$
$$dept \cdot d1 \cdot emp \cdot e1$$

A path can be expressed as a first-order logic formula $\rho(x, y)$ with two free variables x and y, which denote the tail and head nodes of the path, respectively. For example, the paths above can be described by the following formulas:

$$\exists\, z\,(emp(x,\, z) \wedge \exists\, w\,(e1(z,\, w) \wedge manager(w,\, y)))$$
$$\exists\, z\,(dept(x,\, z) \wedge \exists\, w\,(d1(z,\, w) \wedge \exists\, u\,(emp(w,\, u) \wedge e1(u,\, y))))$$

We write $\rho(x, y)$ as ρ when the parameters x and y are clear from the context.

By treating paths as logic formulas, we are able to borrow the standard notion of models from first-order logic [14]. Let G be a deterministic structure, $\rho(x,\, y)$ be a path formula and $a,\, b$ be nodes in $|G|$. We use $G \models \rho(a,\, b)$ to denote that $\rho(a,\, b)$ holds in G, i.e., there is a path ρ from a to b in G.

The *length* of path ρ, $|\rho|$, is defined by:

$$|\rho| \;=\; \begin{cases} 0 & \text{if } \rho = \epsilon \\ 1 & \text{if } \rho = K \\ 1 + |\varrho| & \text{if } \rho = K \cdot \varrho \end{cases}$$

For example, $|emp \cdot e1| = 2$ and $|dept \cdot d1 \cdot emp \cdot e1| = 4$.

By a straightforward induction on the lengths of paths, it can be verified that deterministic graphs have the following property.

Lemma 2.1: Let G be a deterministic structure. Then for any path ρ and node $a \in |G|$, there is at most one node b such that $G \models \rho(a,\, b)$. ∎

This lemma shows that in DM, any component of a database can be uniquely identified by a path.

2.2 Path constraint languages

We next present our path constraint languages. We begin with the definition of P_c^*, which is the most powerful language among the four constraint languages considered in this paper, and continue with the definitions of P_c^-, P_c and P_w presented as restrictions of P_c^*.

2.2.1 Path constraint language P_c^*

To define P_c^*, we first generalize the syntax of path expressions.

We represent path expressions as regular expressions, defined by the syntax:

$$e \;::=\; \epsilon \mid K \mid e \cdot e \mid e + e \mid e^*$$

Here ϵ is a singleton set consisting of the empty path (also denoted by ϵ), $K \in E$ (E is the set of binary relation symbols in σ), $\cdot$, $+$ and $*$ represent concatenation, union and the Kleene closure, respectively.

Let p be a regular expression and ρ be a path. We use $\rho \in p$ to denote that ρ is in the regular language generated by p.

We also treat a regular expression p as a logic formula $p(x, y)$, where x and y are free variables. We say that a deterministic structure G *satisfies* $p(x, y)$, denoted by $G \models p(x, y)$, if there exist path $\rho \in p$ and nodes $a, b \in |G|$ such that $G \models \rho(a, b)$.

Recall that the wildcard symbol "_" matches any edge label. We can express "_" as a regular expression. More specifically, let E, the finite set of binary relation symbols in signature σ, be enumerated as K_1, K_2, ..., K_n. Then "_" can be defined as a regular expression:

$$K_1 + K_2 + ... + K_n.$$

In Section 1, we have seen the following path expressions that can be represented as regular expressions:

$$manager \cdot manager^*$$
$$emp \cdot \_ \cdot manager^*$$

Using regular expressions, we define P_c^* as follows.

Definition 2.1: A constraint ψ of P_c^* is an expression of either the *forward form*:

$$\forall x\,(p(r, x) \rightarrow \forall y\,(q(x, y) \rightarrow s(x, y))),$$

or the *backward form:*

$$\forall x\,(p(r, x) \rightarrow \forall y\,(q(x, y) \rightarrow s(y, x))),$$

where p, q and s are regular expressions, denoted by $pf(\psi)$, $lt(\psi)$ and $rt(\psi)$, respectively. ■

For example, all the path constraint given in Section 1 are (or can be expressed as) P_c^* constraints.

A deterministic structure G *satisfies* a constraint ϕ of P_c^*, denoted by $G \models \phi$, if the following condition is satisfied:

- when ϕ is a forward constraint: for all $a, b \in |G|$, if there exist paths $\alpha \in p$ and $\beta \in q$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, then there exists a path $\gamma \in s$ such that $G \models \gamma(a, b)$;
- when ϕ is a backward constraint: for all $a, b \in |G|$, if there exist paths $\alpha \in p$ and $\beta \in q$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, then there exists $\gamma \in s$ such that $G \models \gamma(b, a)$.

2.2.2 Path constraint language P_c^-

We next present P_c^-. Path constrains of P_c^- are defined in terms of path expressions represented by a restricted form of regular expressions:

$$w \ ::= \ \epsilon \mid K \mid w \cdot w \mid w + w$$

That is, we define path expressions to be regular expressions which do not contain the Kleene closure. Let us refer to such expressions as $*$*-free regular expressions*.

The following should be noted about $*$-free regular expressions.

- The regular language generated by a $*$-free regular expression is finite.
- The wildcard symbol "_" can be expressed as, in fact, a $*$-free regular expression.

For example, we have seen in Section 1 the following path expressions that can be represented as $*$-free regular expressions:

$$emp \cdot \_ \cdot manager$$
$$emp \cdot \_ \cdot supervising \cdot \_$$

Using $*$-free regular expressions, we define P_c^- as follows.

Definition 2.2: The path constraint language P_c^- is defined to be

$$P_c^- = \{\phi \mid \phi \in P_c^*,\ pf(\phi),\ lt(\phi) \text{ and } rt(\phi) \text{ are } *\text{-free regular expressions}\},$$

where $pf(\phi)$, $lt(\phi)$ and $rt(\phi)$ are described in Definition 2.1. ■

For example, path constraints ϕ_1, ϕ_2 and ϕ_3 given in Section 1 are P_c^- constraints, but ψ_1, ψ_2 and ψ_3 are not in P_c^-.

2.2.3 Path constraint language P_c

A subclass of P_c^-, P_c, has been introduced and studied for SM in [10]. The path constraint language P_c is defined in terms of paths, i.e., sequences of edge labels. In other words, it is defined in terms of regular expressions containing neither the wildcard symbol nor the Kleene closure.

Definition 2.3 [10]: The path constraint language P_c is defined to be

$$P_c = \{\varphi \mid \varphi \in P_c^-,\ pf(\varphi),\ lt(\varphi) \text{ and } rt(\varphi) \text{ are paths}\},$$

where $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$ are described in Definition 2.1. ■

For example, φ_3 given in Section 1 is a P_c constraint, and φ_1 and φ_2 can be described by P_c constraints:

$$\forall\, x\, (emp \cdot e1 \cdot manager(r,\ x) \rightarrow emp \cdot e2(r,\ x))$$
$$\forall\, x\, (emp \cdot e2(r,\ x) \rightarrow emp \cdot e1 \cdot manager(r,\ x))$$
$$\forall\, x\, (dept \cdot d1 \cdot emp \cdot e1(r,\ x) \rightarrow emp \cdot e1(r,\ x))$$
$$\forall\, x\, (emp \cdot e1(r,\ x) \rightarrow dept \cdot d1 \cdot emp \cdot e1(r,\ x))$$

Note that ϕ_1, ϕ_2 and ϕ_3 are not P_c constraints.

2.2.4 Path constraint language P_w

A proper subclass of P_c was introduced and studied in [4] for SM:

Definition 2.4 [4]: The class of word constraints, P_w, is defined to be

$$P_w = \{\varphi \mid \varphi \in P_c,\ \varphi \text{ is a forward constraint, } pf(\varphi) = \epsilon\}.$$

where $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$ are described in Definition 2.1. ■

In other words, a word constraint is a forward constraint of P_c with its prefix being the empty path ϵ. It has been shown in [10] that many P_c constraints cannot be expressed as word constraints or even by the more general constraints given in [4].

For example, φ_1 and φ_2 given in Section 1 can be described by P_w constraints, but φ_3 is not a word constraint.

2.3 Path constraint implication

Comparing the expressive powers of these constraint languages, we have

$$P_w \quad \subset \quad P_c \quad \subset \quad P_c^- \quad \subset \quad P_c^*$$

It should be noted that while $*$-free regular expressions, P_w, P_c and P_c^- are definable in first-order logic, regular expressions and P_c^* are not.

Next, we describe implication and finite implication of path constraints in the context of the deterministic data model.

Let $C \in \{P_w, P_c, P_c^-, P_c^*\}$, G be a deterministic structure and φ be a constraint in C. We use $G \models \varphi$ to denote that G *satisfies* φ (i.e., G *is a model of* φ). Let Σ be a finite subset of C. We use $G \models \Sigma$ to denote that G *satisfies* Σ (i.e., G *is a model of* Σ). That is, for every $\phi \in \Sigma$, $G \models \phi$.

Let $\Sigma \cup \{\varphi\}$ be a finite subset of C. We use $\Sigma \models \varphi$ to denote that Σ *implies* φ in the context of DM. That is, for every deterministic structure G, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_f \varphi$ to denote that Σ *finitely implies* φ. That is, for every finite deterministic structure G, if $G \models \Sigma$, then $G \models \varphi$.

In the context of DM, the *implication problem for* C is the problem to determine, given any finite subset $\Sigma \cup \{\varphi\}$ of C, whether $\Sigma \models \varphi$. Similarly, the *finite implication problem for* P_c is the problem of determining whether $\Sigma \models_f \varphi$.

For example, let $\Sigma = \{\psi_1, \psi_2, \phi_1, \phi_2\}$, where ψ_1, ψ_2, ϕ_1 and ϕ_2 are given in Section 1. Then the question whether $\Sigma \models \psi_3$ ($\Sigma \models_f \psi_3$) is an instance of the (finite) implication problem for P_c^*. In Section 1, this implication is used in the proof of the equivalence of the queries Q_2 and Q_2'.

In the context of the graph model SM, the structures considered in the implication problems for C are σ-structures, which are not necessarily deterministic.

The following was established in [4].

Theorem 2.2 [4]: In the context of SM, the implication and finite implication problems for P_w are finitely axiomatizable and are decidable in PTIME. ■

However, it was shown in [10] that in SM, the implication and finite implication problems for P_c are undecidable.

Theorem 2.3 [10]: In the context of SM, the implication problem for P_c is r.e. complete, and the finite implication problem for P_c is co-r.e. complete. ■

As immediate corollaries of Theorem 2.3, we have the following:

Corollary 2.4: In the context of SM, the implication and finite implication problems for P_c^- are undecidable. ■

Corollary 2.5: In the context of SM, the implication and finite implication problems for P_c^* are undecidable. ■

We shall show that in the context of DM, the set of inference rules for word constraint implication given in [4] is no longer complete. In the next section, we present a finite axiomatization for word constraint implication in the context of DM. We shall also show that the undecidability results of Theorem 2.3 and Corollary 2.4 break down in DM. However, the implication and finite implication problems for P_c^* remain undecidable in DM.

3 The implication problems for P_w

In this section, we show that in the context of the deterministic data model DM, P_w has the following properties:

- The implication and finite implication problems for P_w are decidable in linear-space.
- P_w is finitely axiomatizable.
- There is an algorithm for testing implication and finite implication of constraints of P_w in cubic-time.

3.1 The decidability

We begin with a small model argument for the decidability of the implication and finite implication problems for P_w.

Proposition 3.1: In the context of DM, the implication and finite implication problems for P_w coincide and are decidable in linear-space. ∎

Proof: It suffices to show:

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_w, and $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there is a deterministic structure H such that $H \models \phi$ and the size of H is at most the length of ϕ.

For if the claim holds, then the satisfiability problem corresponding to the implication problem for P_w has the small model property. Therefore, the implication and finite implication problems for P_w coincide and are decidable in linear-space.

To show the claim, assume that there is a deterministic structure G satisfying ϕ. Recall that a constraint ψ of P_w can be described as

$$\forall\, x\, (lt(\psi)(r,\, x) \rightarrow rt(\psi)(r,\, x)),$$

where $lt(\psi)$ and $rt(\psi)$ are paths. Let

$$\begin{aligned} Pts(\phi) &= \{lt(\psi),\, rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\}\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\, \rho \preceq \varrho\}. \end{aligned}$$

Here $\rho \preceq \varrho$ stands for that ρ is a prefix of ϱ. Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H to be $(|H|,\, r^H,\, E^H)$ such that

- $|H| = \{a \mid a \in |G|,\, \rho \in CloPts(\phi),\, G \models \rho(r^G,\, a)\}$,
- $r^H = r^G$,
- for all $a, b \in |H|$ and $K \in E$, $H \models K(a,\, b)$ iff $K \in E_\phi$ and $G \models K(a,\, b)$.

It is easy to verify that $H \models \phi$ and H is deterministic, since G has these properties. In addition, by Lemma 2.1, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is at most the length of ϕ. ∎

3.2 A finite axiomatization

It is desirable to develop a finite set of inference rules for a class of constraints. Inference rules can be used not only for generating symbolic proofs of implication, but also for studying the essential properties of the constraints. In general, the existence of a finite set of inference rules is a stronger property than the existence of an algorithm for testing implication. There are classes of constraints for which there is no finite set of inference rules but there is an algorithm for testing their logical implication.

In the context of SM, it has been shown in [4] that the following inference rules are sound and complete for implication and finite implication of constraints of P_w:

- Reflexivity:

$$\frac{}{\forall x\ (\alpha(r,x) \rightarrow \alpha(r,x))}$$

- Transitivity:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \beta(r,x)) \quad \forall x\ (\beta(r,x) \rightarrow \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \gamma(r,x))}$$

- Right-congruence:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \beta(r,x))}{\forall x\ (\alpha \cdot \gamma(r,x) \rightarrow \beta \cdot \gamma(r,x))}$$

These rules, however, are not complete for P_w in the context of DM. To illustrate this, let α be a path and consider the following constraints of P_w:

$$\begin{aligned} \varphi &= \forall\, x\, (\epsilon(r,\, x) \rightarrow \alpha(r,\, x)) \\ \phi &= \forall\, x\, (\alpha(r,\, x) \rightarrow \epsilon(r,\, x)) \end{aligned}$$

By Lemma 2.1, it is easy to verify that $\varphi \models \phi$. However, this implication cannot be derived by using the rules given above.

Next, we show that P_w is still finitely axiomatizable in the context of DM. To do this, we give the following lemma.

Lemma 3.2: For every finite subset $\Sigma \cup \{\varphi\}$ of P_w,

$$\begin{aligned} \Sigma \models \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi, \\ \Sigma \models_f \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models_f \varphi. \end{aligned}$$

■

Proof: Obviously, if $\Sigma \models \varphi$ then $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$.

Conversely, if $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$, then

$$\Sigma \models \exists\, x\, (lt(\varphi)(r,\, x)) \rightarrow \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

That is,

$$\Sigma \models \forall\, x\, \neg lt(\varphi)(r,\, x) \vee \forall\, x\, (\neg lt(\varphi)(r,\, x) \vee rt(\varphi)(r,\, x)).$$

Since $\forall\, x\, \neg lt(\varphi)(r,\, x) \models \forall\, x\, (\neg lt(\varphi)(r,\, x) \vee rt(\varphi)(r,\, x))$, we have

$$\Sigma \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

That is, $\Sigma \models \varphi$.

The same proof can be used to show the finite case. ■

Based on this observation, we extend P_w by including constraints of the *existential form:*

$$\exists\, x\, \rho(r,\, x)$$

Here ρ is a path. Constraints of the existential form enable us to assert the existence of paths. As pointed out by [21], this ability is important for specifying Web link characteristics.

Let

$$P_w^e = P_w \cup \{\exists\, x\, \rho(r,\, x) \mid \rho \text{ is a path}\}.$$

For P_w^e, we consider a set of inference rules, $\mathcal{I}_w$, which consists of the following and Reflexivity, Transitivity, and Right-congruence given above.

- Empty-path:

$$\frac{}{\exists x\ \epsilon(r, x)}$$

- Prefix:

$$\frac{\exists x\ (\alpha \cdot \beta(r, x))}{\exists x\ \alpha(r, x)}$$

- Entail:

$$\frac{\exists x\ \alpha(r, x) \qquad \forall x\ (\alpha(r, x) \rightarrow \beta(r, x))}{\exists x\ \beta(r, x)}$$

- Symmetry:

$$\frac{\exists x\ \alpha(r, x) \qquad \forall x\ (\alpha(r, x) \rightarrow \beta(r, x))}{\forall x\ (\beta(r, x) \rightarrow \alpha(r, x))}$$

Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_w^e. We use $\Sigma \vdash_{\mathcal{I}_w} \varphi$ to denote that φ is provable from Σ using $\mathcal{I}_w$. That is, there is an $\mathcal{I}_w$-proof of φ from Σ. For example, it is easy to verify the following:

$$\{\exists\, x\ \alpha(r,\, x),\ \forall\, x\,(\alpha(r,\, x) \rightarrow \alpha \cdot \beta(r,\, x)),\ \forall\, x\,(\alpha(r,\, x) \rightarrow \gamma(r,\, x))\}$$
$$\vdash_{\mathcal{I}_w} \forall\, x\,(\alpha \cdot \beta(r,\, x) \rightarrow \gamma(r,\, x))$$
$$\{\exists\, x\ (\alpha \cdot \rho(r,\, x)),\ \forall\, x\,(\alpha(r,\, x) \rightarrow \beta(r,\, x)),\ \forall\, x\,(\beta(r,\, x) \rightarrow \alpha \cdot \rho(r,\, x))\}$$
$$\vdash_{\mathcal{I}_w} \forall\, x\,(\alpha \cdot \rho(r,\, x) \rightarrow \beta(r,\, x))$$

The theorem below shows that $\mathcal{I}_w$ is a finite axiomatization of P_w in the context of DM.

Theorem 3.3: In the context of DM, for every finite subset $\Sigma \cup \{\varphi\}$ of P_w,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi,$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi.$$

■

Proof: By Lemma 3.2, we only need to show that

$$\Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\, x))\} \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\,(lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi.$$

Soundness of $\mathcal{I}_w$ can be verified by induction on the lengths of $\mathcal{I}_w$-proofs. For the proof of completeness, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_w and k be any natural number such that

$$k \geq max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

- $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\}$, and
- for every path ρ such that $|\rho| \leq k$, if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$, then

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x)).$$

For if the claim holds and suppose that $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models \varphi$, then we have $G \models \varphi$ because $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\}$. In addition, because G is finite, if it is the case where $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \models_f \varphi$, then we also have $G \models \varphi$. Thus again by the claim,

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow rt(\varphi)(r,\, x)).$$

Next, we show the claim. To define the structure G described in the claim, let

$$N = \{\rho \mid \rho \text{ is a path},\ |\rho| \leq k,\ \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \exists\, x\, \rho(r, x)\}.$$

Then the following should be noted:

- By Empty-path in $\mathcal{I}_w$, $\epsilon \in N$.
- By Prefix in $\mathcal{I}_w$, if $\rho \in N$, then all the prefixes of ρ are also in N. That is, N is prefix-closed.

We also define an equivalence relation $\sim$ on N as follows:

$$\rho \sim \varrho \ \text{ iff } \ \Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x)).$$

It should be noted that for all $\rho, \varrho \in N$, $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x))$ iff $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (\varrho(r,\, x) \rightarrow \rho(r,\, x))$, by Symmetry in $\mathcal{I}_w$.

Let $[\rho]$ be the equivalence class of ρ with respect to $\sim$. For each $\rho \in N$, let $o([\rho])$ be a distinct node. We then define G to be $(|G|,\, r^G,\, E^G)$, where

- $|G| = \{o([\rho]) \mid \rho \in N\}$,
- $r^G = o([\epsilon])$,
- for every $K \in E$ and $o([\rho]), o([\varrho]) \in |G|$, $G \models K(o([\rho]),\, o([\varrho]))$ iff $[\varrho] = [\rho \cdot K]$. This is well-defined by Transitivity, Right-congruence and Symmetry in $\mathcal{I}_w$.

We next show that G is indeed the structure described in the Claim.

(1) G is a finite deterministic structure.

It should be noted that N is finite. Therefore, $|G|$ is finite. In addition, G is deterministic because of Symmetry, Transitivity and Right-congruence in $\mathcal{I}_w$.

(2) $G \models \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\}$.

It suffices to show

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G,\, o([\rho]))$ iff $\varrho \in [\rho]$.

For if Claim 1 holds, then by $lt(\varphi) \in N$, we have that $G \models lt(\varphi)(r^G,\, o([lt(\varphi)])$. That is,

$$G \models \exists\, x\, (lt(\varphi)(r,\, x)).$$

In addition, for every $\phi \in \Sigma$, if there exists $a \in |G|$ such that $G \models lt(\phi)(r^G,\, a)$, then by Claim 1 and the fact that G is deterministic, $lt(\phi) \in N$ and $a = o([lt(\phi)])$. By Entail in $\mathcal{I}_w$, we have $rt(\phi) \in N$, and moreover, $rt(\phi) \sim lt(\phi)$. Therefore, again by Claim 1, we have $G \models rt(\phi)(r^G,\, o([lt(\phi)]))$. Thus $G \models \phi$. Hence $G \models \Sigma$.

We show Claim 1 by induction on $|\varrho|$.

Base case: $\varrho = \epsilon$.

Clearly, $G \models \epsilon(o([\epsilon]),\, o([\rho]))$ iff $[\rho] = [\epsilon]$ iff $\epsilon \in [\rho]$.

Inductive step: Assume Claim 1 for ϱ. We next show that Claim 1 also holds for $\varrho \cdot K$.

If $G \models \varrho \cdot K(r^G,\, o([\rho]))$, then by the induction hypothesis and Lemma 2.1, we have

$$G \models \varrho(r^G,\, o([\varrho])) \wedge K(o([\varrho]),\, o([\rho])).$$

By the definition of E^G, $G \models K(o([\varrho]),\, o([\rho]))$ iff $[\rho] = [\varrho \cdot K]$. Therefore, $\varrho \cdot K \in [\rho]$.

Conversely, if $\varrho \cdot K \in [\rho]$, then by Prefix in $\mathcal{I}_w$, we have $\varrho \in N$. By the induction hypothesis,

$$G \models \varrho(r^G,\, o([\varrho])).$$

By the definition of E^G, $G \models K(o([\varrho]),\, o([\varrho \cdot K]))$. Moreover, by $\varrho \cdot K \in [\rho]$, we have $[\varrho \cdot K] = [\rho]$ and therefore, $G \models \varrho \cdot K(r^G,\, o([\rho]))$.

(3) For any ρ such that $|\rho| \leq k$, $\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$ if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$.

Note that $lt(\varphi) \in N$. By Claim 1,

$$G \models lt(\varphi)(r,\, o([lt(\varphi)])).$$

Thus if $G \models \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x))$, then $G \models \rho(r^G,\, o([lt(\varphi)]))$. Again by Claim 1, we have $\rho \in [lt(\varphi)]$. That is, $\rho \sim lt(\varphi)$. Hence by the definition of $\sim$ and Symmetry in $\mathcal{I}_w$,

$$\Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \forall\, x\, (lt(\varphi)(r,\, x) \rightarrow \rho(r,\, x)).$$

This completes the proof of Theorem 3.3. ∎

In fact, $\mathcal{I}_w$ is a finite axiomatization of P_w^e.

Theorem 3.4: In the context of DM, for any finite subset $\Sigma \cup \{\varphi\}$ of P_w^e, if $\varphi \in P_w$, then

$$\Sigma \models \varphi \text{ iff } \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi \quad \text{and} \quad \Sigma \models_f \varphi \text{ iff } \Sigma \cup \{\exists\, x\, (lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_w} \varphi.$$

Otherwise, i.e., when φ is a constraint of the existential form,

$$\Sigma \models \varphi \text{ iff } \Sigma \vdash_{\mathcal{I}_w} \varphi \quad \text{and} \quad \Sigma \models_f \varphi \text{ iff } \Sigma \vdash_{\mathcal{I}_w} \varphi.$$

∎

Proof: Soundness of $\mathcal{I}_w$ can be verified by induction on the lengths of $\mathcal{I}_w$-proofs. For the proof of completeness, we consider two cases.

(1) $\varphi \in P_w$.

The proof for this case is similar to the argument for Theorem 3.3.

(2) $\varphi \notin P_w$. That is, $\varphi = \exists\, x\, \alpha(r, x)$.

In this case, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_w^e and k be any natural number such that

$$k \geq max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cap P_w\} \text{ and } k \geq max\{|\rho| \mid \exists\, x\, \rho(r,\, x) \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

- $G \models \Sigma$, and
- for every path ρ such that $|\rho| \leq k$, if $G \models \exists\, x\, \rho(r,\, x)$, then $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\, \rho(r,\, x)$.

For if the claim holds and $\Sigma \models \exists\, x\, \alpha(r,\, x)$, then we have $G \models \exists\, x\, \alpha(r,\, x)$, since $G \models \Sigma$. In addition, since G is finite, if $\Sigma \models_f \exists\, x\, \alpha(r,\, x)$, then we also have $G \models \exists\, x\, \alpha(r,\, x)$. Thus again by Claim, $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\, \alpha(r,\, x)$. That is, $\Sigma \vdash_{\mathcal{I}_w} \varphi$.

Next, we show the claim. We first define the following:

$$\begin{aligned} N &= \{\rho \mid \rho \text{ is a path, } |\rho| \leq k,\ \Sigma \vdash_{\mathcal{I}_w} \exists\, x\, \rho(r, x)\} \\ \rho \sim \varrho &\text{ iff } \Sigma \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x)) \end{aligned}$$

In addition, we define $[\rho]$, $o([\rho])$ and G as in the proof of Theorem 3.3. Then using the same argument given in the proof of Theorem 3.3, we can verify

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G,\, o([\rho]))$ iff $\varrho \in [\rho]$.

Using Claim 1, we can show the following.

Statement 1: $G \models \Sigma$.

Let ϕ be any constraint in Σ. If ϕ is $\exists\, x\ \rho(r,\, x)$, then we have $\rho \in N$ and moreover, by Claim 1, $G \models \rho(r^G,\, o([\rho]))$. That is, $G \models \phi$.

If $\phi \in P_w$, then by using the same argument given in the proof of Theorem 3.3, it can also be shown that $G \models \phi$.

Statement 2: For every path ρ such that $|\rho| \leq k$, if $G \models \exists\, x\ \rho(r,\, x)$, then $\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r,\, x)$.

If $G \models \exists\, x\ \rho(r,\, x)$, then there exists $\varrho \in N$ such that $G \models \rho(r^G,\, o([\varrho]))$. By Claim 1, $\rho \in [\varrho]$. By the definition of N and $\sim$, we have $\rho \in N$ and in addition,

$$\Sigma \vdash_{\mathcal{I}_w} \exists\, x\ \rho(r,\, x).$$

This completes the proof of Theorem 3.4. ∎

3.3 An algorithm

Next, we present an algorithm for testing word constraint implication. This algorithm takes as input a finite subset Σ of P_w and a path α. It returns as output a deterministic structure G having the following properties: there is $o \in |G|$ such that $G \models \alpha(r^G,\, o)$, and moreover, for any path β,

$$G \models \beta(r^G, o) \ \text{ iff } \ \Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \rightarrow \beta(r,\, x)).$$

By Theorem 3.3, this algorithm can be used for testing implication and finite implication of word constraints.

The algorithm (Algorithm 3.1) is given in Table 1. The procedure $merge(a,\, b)$ used in the algorithm is shown in Table 2.

The structure G computed by Algorithm 3.1 can be naturally extended to a σ-structure by letting $K^G = \emptyset$ for any $K \in E \setminus E_\phi$.

It should be noted that the rationale behind step 4 (1) of Algorithm 3.1 is Lemma 2.1. In a deterministic graph G, for any path ρ, if there is $o \in |G|$ such that $G \models \rho(r^G,\, o)$, then o is unique. As a result, every constraint in Σ is used at most once by the algorithm. In general, this does not hold in graphs of SM. It is because of this property that Algorithm 3.1 has low complexity.

For the complexity of the algorithm, the following should be noted. Let n_E be the cardinality of E_ϕ, n_G the size of $|G|$, n the length of Σ and α, and n_Σ the cardinality of Σ.

- $n_E \leq n$, $n_G \leq n$ and $n_\Sigma \leq n$.
- Step 4(1), (2) and (3) are executed at most n_Σ times.
- Testing whether $G \models \rho(r^G,\, o_\rho)$ in step 4 can be done in at most $O(n_G\, |\rho|)$ time. Therefore, it can be done in $O(n^2)$ time. By using appropriate data structure, e.g., (variable length) array indexed by edge labels in E_ϕ, this can be done in $O(|\rho|)$ time, i.e., $O(n)$ time.

Algorithm 3.1:

Input: a finite subset Σ of P_w and a path α
Output: the structure G described above

1. E_ϕ := the set of edge labels appearing in either α or some path in constraints of Σ;
2. $Rules := \Sigma$;
3. $G := (|G|, r^G, E_\phi^G)$, where
   - $|G| = \{o(\rho) \mid \rho \preceq \alpha,\ o(\rho)$ is a distinct node$\}$,
   - $r^G = o(\epsilon)$,
   - E_ϕ^G is populated such that $G \models K(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot K$;
4. repeat until no further change:
   if $\forall x\,(\rho(r, x) \rightarrow \varrho(r, x)) \in \Sigma$ and there is $o_\rho \in |G|$ such that $G \models \rho(r^G, o_\rho)$ then
   (1) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \rightarrow \varrho(r, x))\}$;
   (2) for each $\gamma \cdot K \preceq \varrho$ do
   if there is no $o \in |G|$ such that $G \models \gamma \cdot K(r^G, o)$ then
   (i) add to $|G|$ a distinct node $o_{\gamma \cdot K}$;
   (ii) add to E_ϕ^G an edge labeled K from o_γ to $o_{\gamma \cdot K}$,
   where $o_\gamma \in |G|$ such that $G \models \gamma(r^G, o_\gamma)$;
   (3) $merge(o_\rho, o_\varrho)$;
5. output G.

Table 1: An algorithm for testing word constraint implication in DM

- The procedure *merge* is executed at most n_G times. Each step takes $O(n_E\ n_G)$ time. Hence the total cost of executing *merge* is $O(n_G^2\, n_E)$, i.e., $O(n^3)$. Again, by using appropriate data structure, this can be done in $O(n^2)$ time.

Therefore, Algorithm 3.1 is in $O(n^3)$ time. In addition, when implemented using appropriate data structures, this algorithm is in $O(n^2)$ time.

Next, we show that Algorithm 3.1 is correct.

Proposition 3.5: Given a finite subset Σ of P_w and a path α, Algorithm 3.1 computes a finite deterministic structure G having the following property: there exists $o \in |G|$, such that $G \models \alpha(r^G, o)$, and in addition, for any path β,

$$G \models \beta(r^G, o) \ \text{ iff } \ \Sigma \cup \{\exists x\,\alpha(r, x)\} \vdash_{\mathcal{I}_w} \forall x\,(\alpha(r, x) \rightarrow \beta(r, x)).$$

■

Proof: The step 4 of Algorithm 3.1 ensures that $G \models \Sigma$, taking advantage of the fact that G is deterministic and because of Lemma 2.1. In addition, step 3 ensures that there is $o \in |G|$, such that $G \models \alpha(r^G, o)$. Therefore, if $\Sigma \cup \{\exists x\,\alpha(r, x)\} \vdash_{\mathcal{I}_w} \forall x\,(\alpha(r, x) \rightarrow \beta(r, x))$, then by Theorem 3.3, $G \models \beta(r^G, o)$.

procedure *merge*$(a,\, b)$

1. for each $K \in E_\phi$ do
   if there is $o \in |G|$ such that $G \models K(o,\, b)$ then
   (1) delete from E_ϕ^G the edge labeled K from o to b;
   (2) add to E_ϕ^G an edge labeled K from o to a;
2. for each $K \in E_\phi$ do
   if there is $o_b \in |G|$ such that $G \models K(b,\, o_b)$ then
   (1) delete from E_ϕ^G the edge labeled K from b to o_b;
   (2) add to E_ϕ^G an edge labeled K from a to o_b;
   (3) if there is $o_a \in |G|$ such that $G \models K(a, o_a)$ and $o_a \neq o_b$ then
   merge$(o_a,\, o_b)$;
3. $|G| := |G| \setminus \{b\}$;

Table 2: Procedure *merge*

Conversely, by a straightforward induction on the number of steps in the construction of G by the algorithm, we can show that for all paths ρ and ϱ, if there is $a \in |G|$ such that $G \models \rho(r^G,\, a) \wedge \varrho(r^G,\, a)$, then $\Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\rho(r,\, x) \rightarrow \varrho(r,\, x))$. Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_w$. For example, step 4(2) corresponds to an application of Prefix, and *merge* corresponds to applications of Transitivity, Right-Congruence and Symmetry in $\mathcal{I}_w$. As a result, if $G \models \beta(r^G, o)$, then we have $\Sigma \cup \{\exists\, x\, \alpha(r,\, x)\} \vdash_{\mathcal{I}_w} \forall\, x\, (\alpha(r,\, x) \rightarrow \beta(r,\, x))$. ∎

From Proposition 3.5, Algorithm 3.1, and Theorem 3.3, the corollary below follows immediately.

Corollary 3.6: In the context of DM, the implication and finite implication problems for P_w are decidable in cubic-time. ∎

4 The implication problems for P_c

This section generalizes the results established in the last section to P_c. In contrast to the undecidability of the implication and finite implication problems for P_c in SM, we show that in the context of DM, P_c has the following properties:

- The implication and finite implication problems for P_c coincide and are decidable in linear-space.
- There is a finite axiomatization for (finite) implication of constraints of P_c.
- There is a cubic-time algorithm for testing (finite) implication of constraints of P_c.

These results show that the determinism condition of DM simplifies reasoning about path constraints.

4.1 The decidability

With slight modification, the small model argument for Proposition 3.1 is also applicable to the proposition below.

Proposition 4.1: In the context of DM, the implication and finite implication problems for P_c coincide and are decidable in linear-space. ∎

Proof: As in the proof of Proposition 3.1, it suffices to show:

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c, and $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there is a deterministic structure H such that $H \models \phi$ and the size of H is at most the length of ϕ.

To show the claim, assume that ϕ has a deterministic model G. Let

$$\begin{aligned} Pts(\phi) &= \{pf(\psi)\cdot lt(\psi),\ pf(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the forward form}\} \\ &\quad \cup \{pf(\psi)\cdot lt(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the backward form}\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\ \rho \preceq \varrho\}. \end{aligned}$$

Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H in the same way as in the proof of Proposition 3.1. It is easy to verify that H is a deterministic structure, $H \models \phi$, and in addition, by Lemma 2.1, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is at most the length of ϕ. ∎

4.2 A finite axiomatization

Before we present a finite axiomatization for P_c, we first study basic properties of constraints of P_c in DM.

Lemma 4.2: Let φ be a forward constraint of P_c:

$$\varphi = \forall\, x\, (\alpha(r,\ x) \rightarrow \forall\, y\, (\beta(x,\ y) \rightarrow \gamma(x,\ y))),$$

and ψ be a constraint of P_w:

$$\psi = \forall\, x\, (\alpha\cdot\beta(r,\ x) \rightarrow \alpha\cdot\gamma(r,\ x)).$$

Then for any deterministic structure G, $G \models \varphi$ iff $G \models \psi$. ∎

Proof: If $G \models \neg\psi$, then there is $b \in |G|$ such that

$$G \models \alpha\cdot\beta(r^G,\ b) \wedge \neg\alpha\cdot\gamma(r^G,\ b).$$

Thus there exists $a \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. In addition, $G \models \neg\gamma(a, b)$ since otherwise $G \models \alpha \cdot \gamma(r^G, b)$. Hence there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(a, b).$$

Thus $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(a, b).$$

By Lemma 2.1, a is the unique node such that $G \models \alpha(r^G, a)$. Thus

$$G \models \alpha \cdot \beta(r^G, b) \wedge \neg\alpha \cdot \gamma(r^G, b).$$

That is, $G \models \neg\psi$. ∎

Lemma 4.3: Let φ be a backward constraint of P_c:

$$\varphi = \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))),$$

and ψ be a constraint of P_w:

$$\psi = \forall x\,(\alpha(r, x) \rightarrow \alpha \cdot \beta \cdot \gamma(r, x)).$$

Then for any deterministic structure G, if $G \models \exists x\,(\alpha \cdot \beta(r, x))$, then $G \models \varphi$ iff $G \models \psi$. ∎

Proof: Assume that $G \models \exists x\,(\alpha \cdot \beta(r, x))$. Then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b).$$

By Lemma 2.1, a is the unique node such that $G \models \alpha(r^G, a)$, and b is the unique node such that $G \models \beta(a, b)$.

If $G \models \neg\psi$, then

$$G \models \alpha(r^G, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G, a).$$

Clearly, $G \models \neg\gamma(b, a)$ since otherwise $G \models \alpha \cdot \beta \cdot \gamma(r^G, a)$. Therefore,

$$G \models \alpha(r^G, a) \wedge \beta(a, b) \wedge \neg\gamma(b, a).$$

That is, $G \models \neg\varphi$.

Conversely, if $G \models \neg\varphi$, then there are $a', b' \in |G|$ such that

$$G \models \alpha(r^G, a') \wedge \beta(a', b') \wedge \neg\gamma(b', a').$$

By Lemma 2.1, $a' = a$ and $b' = b$. In addition, $G \models \neg\alpha \cdot \beta \cdot \gamma(r^G, a)$ since otherwise $G \models \gamma(b, a)$. Hence

$$G \models \alpha(r^G, a) \wedge \neg\alpha \cdot \beta \cdot \gamma(r^G, a).$$

Thus $G \models \neg\psi$. ∎

Lemma 4.4: For every finite subset $\Sigma \cup \{\varphi\}$ of P_c,

$$\begin{array}{rcl} \Sigma \models \varphi & \text{iff} & \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi, \\ \Sigma \models_f \varphi & \text{iff} & \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models_f \varphi. \end{array}$$

∎

Proof: Obviously, if $\Sigma \models \varphi$ then $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$.

Conversely, if $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$, then

$$\Sigma \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)) \rightarrow \varphi.$$

That is,

$$\Sigma \models \forall\, x\, \neg pf(\varphi) \cdot lt(\varphi)(r,\, x) \vee \varphi.$$

Note that φ is of either the forward form:

$$\forall\, x\, (\neg pf(\varphi)(r,\, x) \vee \forall\, y\, (\neg lt(\varphi)(x,\, y) \vee rt(\varphi)(x,\, y))),$$

or the backward form:

$$\forall\, x\, (\neg pf(\varphi)(r,\, x) \vee \forall\, y\, (\neg lt(\varphi)(x,\, y) \vee rt(\varphi)(y,\, x))).$$

Since $\forall\, x\, \neg(pf(\varphi) \cdot lt(\varphi)(r,\, x)) \models \forall\, x \forall\, y\, (\neg pf(\varphi)(r,\, x) \vee \neg lt(\varphi)(x,\, y))$, we have

$$\forall\, x\, \neg(pf(\varphi) \cdot lt(\varphi)(r,\, x)) \models \varphi.$$

Hence $\Sigma \models \varphi$.

The same proof is also applicable to the case of finite implication. ∎

Based on Lemma 4.4, we extend P_c by including constraints of the existential form as follows:

$$P_c^e = P_c \cup \{\exists\, x\, \rho(r,\, x) \mid \rho \text{ is a path}\}.$$

As mentioned in the last section, constraints of the existential form assert existence of paths.

For P_c^e, we consider a set of inference rules, $\mathcal{I}_c$, which consists of the following and those in $\mathcal{I}_w$ given in the last section. Note that the inference rules below are sound in DM because of Lemmas 4.2 and 4.3.

- Forward-to-word:
$$\frac{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}$$
- Word-to-forward:
$$\frac{\forall x\ (\alpha \cdot \beta(r,x) \rightarrow \alpha \cdot \gamma(r,x))}{\forall x\ (\alpha(r,x) \rightarrow \forall y\ (\beta(x,y) \rightarrow \gamma(x,y)))}$$

- Backward-to-word:

$$\frac{\exists x\ (\alpha \cdot \beta(r, x)) \quad \forall x\ (\alpha(r, x) \rightarrow \forall y\ (\beta(x, y) \rightarrow \gamma(y, x)))}{\forall x\ (\alpha(r, x) \rightarrow \alpha \cdot \beta \cdot \gamma(r, x))}$$

- Word-to-backward:

$$\frac{\exists x\ (\alpha \cdot \beta(r, x)) \quad \forall x\ (\alpha(r, x) \rightarrow \alpha \cdot \beta \cdot \gamma(r, x))}{\forall x\ (\alpha(r, x) \rightarrow \forall y\ (\beta(x, y) \rightarrow \gamma(y, x)))}$$

Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c^e. We use $\Sigma \vdash_{\mathcal{I}_c} \varphi$ to denote that φ is provable from Σ using $\mathcal{I}_c$. That is, there is an $\mathcal{I}_c$-proof of φ from Σ.

Similar to Theorem 3.3, the following theorem shows that $\mathcal{I}_c$ is indeed a finite axiomatization of P_c.

Theorem 4.5: In the context of DM, for every any subset $\Sigma \cup \{\varphi\}$ of P_c,

$$\begin{aligned} \Sigma \models \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi, \\ \Sigma \models_f \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi. \end{aligned}$$

■

Proof: By Lemma 4.4, we only need to show

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi \quad \text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi.$$

The proof is similar to the proof of Theorem 3.3. Soundness of $\mathcal{I}_c$ can be verified by induction on the lengths of $\mathcal{I}_c$-proofs. For the proof of completeness, it suffices to show

Claim: Let $\Sigma \cup \{\varphi\}$ be any finite subset of P_c and k be any natural number such that

$$k \geq max\{|pf(\psi)| + |lt(\psi)| + |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\}.$$

Then there is a finite deterministic structure G such that

1. $G \models \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G,\, x))\}$,
2. for every path ρ such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$,
   - if $G \models \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)))$, then

     $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(x,\, y)))$;

   - if $G \models \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x)))$, then

     $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x)))$.

To see why this claim suffices, suppose that $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models \varphi$. Then by $G \models \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\}$, we have $G \models \varphi$. In addition, since G is finite, if it is the case where $\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \models_f \varphi$, then we also have $G \models \varphi$. Thus again by the claim, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi.$$

Next, we show the claim. Let

$$N = \{\rho \mid \rho \text{ is a path, } |\rho| \leq k,\ \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G,\, x))\} \vdash_{\mathcal{I}_c} \exists\, x\ \rho(r, x)\}.$$

Recall the equivalence relation $\sim$ on N defined in the proof of Theorem 3.3. We also use $[\rho]$ to denote the equivalence class of ρ with respect to $\sim$. For each $\rho \in N$, we create a distinct node $o([\rho])$. Let G be the structure defined in the proof of Theorem 3.3. Using the same proof, we can show that G is a finite deterministic structure. In addition, we can also show the following claim:

Claim 1: For every $\rho \in N$ and path ϱ such that $|\varrho| \leq k$, $G \models \varrho(r^G,\, o([\rho]))$ iff $\varrho \in [\rho]$.

Using Claim 1, we show the following.

(1) $G \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r^G,\, x))$.

Clearly, $pf(\varphi) \cdot lt(\varphi) \in N$. Thus by Claim 1,

$$G \models pf(\varphi) \cdot lt(\varphi)(r^G,\, o([pf(\varphi) \cdot lt(\varphi)])).$$

(2) $G \models \Sigma$.

Suppose, for *reductio*, that there is $\psi \in \Sigma$ such that $G \models \neg\psi$. Then we show that the assumption leads to a contradiction.

If ψ is a forward constraint $\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(a,\, b).$$

Thus by Lemma 2.1 and Claim 1, we have $\alpha \cdot \beta \in N$, $a = o([\alpha])$ and $b = o([\alpha \cdot \beta])$. By Forward-to-word and Entail in $\mathcal{I}_c$, we have $\alpha \cdot \gamma \in N$ and moreover,

$$\alpha \cdot \beta \sim \alpha \cdot \gamma.$$

Therefore, again by Claim 1, we have $G \models \alpha \cdot \gamma(r^G,\, o([\alpha \cdot \beta]))$. By Lemma 2.1, we have

$$G \models \gamma(o([\alpha]),\, o([\alpha \cdot \beta])).$$

This contradicts the assumption.

If ψ is a backward constraint $\forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x)))$, then there are $a, b \in |G|$ such that

$$G \models \alpha(r^G,\, a) \wedge \beta(a,\, b) \wedge \neg\gamma(b,\, a).$$

Again by Lemma 2.1 and Claim 1, we have $\alpha \cdot \beta \in N$, $a = o([\alpha])$ and $b = o([\alpha \cdot \beta])$. That is,

$$G \models \exists\, x\, (\alpha \cdot \beta(r,\, x)).$$

By Backward-to-word and Entail, we have $\alpha \cdot \beta \cdot \gamma \in N$ and moreover,

$$\alpha \sim \alpha \cdot \beta \cdot \gamma.$$

Therefore, again by Claim 1, we have $G \models \alpha \cdot \beta \cdot \gamma(r^G,\, o([\alpha]))$. By Lemma 2.1, we have

$$G \models \gamma(o([\alpha \cdot \beta]),\, o([\alpha])).$$

This again contradicts the assumption.

Thus $G \models \psi$. Hence $G \models \Sigma$.

(3) G has the property described by (2) of Claim.

Let ρ be a path such that $|\rho| \leq k - |pf(\varphi) \cdot lt(\varphi)|$.

If $G \models \forall\, x\, (pf(\varphi)(r,\, x) \to \forall\, y\, (lt(\varphi)(x,\, y) \to \rho(x,\, y)))$, then by Lemma 4.2,

$$G \models \forall\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r,\, x)).$$

By $pf(\varphi) \cdot lt(\varphi) \in N$ and Claim 1, we have

$$G \models pf(\varphi) \cdot \rho(r^G,\, o([pf(\varphi) \cdot lt(\varphi)])),$$

and moreover,

$$pf(\varphi) \cdot lt(\varphi) \sim pf(\varphi) \cdot \rho.$$

Thus by the definition of $\sim$ and Symmetry in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x) \to pf(\varphi) \cdot \rho(r^G,\, x)).$$

By Word-to-forward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi)(r,\, x) \to \forall\, y\, (lt(\varphi)(x,\, y) \to \rho(x,\, y))).$$

If $G \models \forall\, x\, (pf(\varphi)(r,\, x) \to \forall\, y\, (lt(\varphi)(x,\, y) \to \rho(y,\, x)))$, then by $G \models \exists(pf(\varphi) \cdot lt(\varphi)(r,\, x))$ and Lemma 4.3, we have

$$G \models \forall\, x\, (pf(\varphi)(r,\, x) \to pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

Since $pf(\varphi) \cdot lt(\varphi) \in N$, by Prefix in $\mathcal{I}_c$, we have $pf(\varphi) \in N$. By Claim 1, we have

$$G \models pf(\varphi) \cdot lt(\varphi) \cdot \rho(r^G,\, o([pf(\varphi)])),$$

and moreover,

$$pf(\varphi) \sim pf(\varphi) \cdot lt(\varphi) \cdot \rho.$$

Thus by the definition of $\sim$ and Symmetry in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi) \rightarrow pf(\varphi) \cdot lt(\varphi) \cdot \rho(r,\, x)).$$

By Word-to-backward in $\mathcal{I}_c$ and $G \models \exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))$, we have

$$\Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x)))\} \vdash_{\mathcal{I}_c} \forall\, x\, (pf(\varphi)(r,\, x) \rightarrow \forall\, y\, (lt(\varphi)(x,\, y) \rightarrow \rho(y,\, x))).$$

This completes the proof of Claim, and therefore, the proof of Theorem 4.5. ∎

In addition, it can be shown that $\mathcal{I}_c$ is also a finite axiomatization of P_c^e.

Theorem 4.6: In DM, for every finite subset $\Sigma \cup \{\varphi\}$ of P_c^e, if $\varphi \in P_c$, then

$$\begin{aligned} \Sigma \models \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi, \\ \Sigma \models_f \varphi \quad &\text{iff} \quad \Sigma \cup \{\exists\, x\, (pf(\varphi) \cdot lt(\varphi)(r,\, x))\} \vdash_{\mathcal{I}_c} \varphi. \end{aligned}$$

Otherwise, i.e., when φ is an existential constraints,

$$\begin{aligned} \Sigma \models \varphi \quad &\text{iff} \quad \Sigma \vdash_{\mathcal{I}_c} \varphi, \\ \Sigma \models_f \varphi \quad &\text{iff} \quad \Sigma \vdash_{\mathcal{I}_c} \varphi. \end{aligned}$$

∎

The proof of this theorem is similar to the proof of Theorem 4.5.

4.3 An algorithm

Next, we present an algorithm for testing implication of constraints of P_c. This algorithm takes as input a finite subset Σ of P_c and two paths α and β. It computes a deterministic structure G having the following properties: there are $a, b \in |G|$ such that $G \models \alpha(r^G,\, a) \wedge \beta(a,\, b)$, and in addition, for any path γ,

$$\begin{aligned} G \;\models\; \gamma(a,\, b) \;\text{ iff }\; \Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(x,\, y))), \\ G \;\models\; \gamma(b,\, a) \;\text{ iff }\; \Sigma \cup \{\exists\, x\, (\alpha \cdot \beta(r,\, x))\} \vdash_{\mathcal{I}_c} \forall\, x\, (\alpha(r,\, x) \rightarrow \forall\, y\, (\beta(x,\, y) \rightarrow \gamma(y,\, x))). \end{aligned}$$

By Theorem 4.5, this algorithm can be used for testing implication and finite implication of P_c constraints.

The algorithm (Algorithm 4.1) is shown in Table 3, which uses procedure *merge* given in Table 2. The structure G computed by the algorithm can be extended to a σ-structure by letting $K^G = \emptyset$ for any $K \in E \setminus E_\phi$.

Similar to the analysis of Algorithm 3.1 given in the last section, it can also be shown that Algorithm 4.1 runs in $O(n^3)$ time, where n is the length of Σ and $\alpha \cdot \beta$. In addition, when implemented using appropriate data structures, the algorithm runs in $O(n^2)$ time.

The proposition below shows that Algorithm 4.1 is correct.

Algorithm 4.1:

Input: a finite subset Σ of P_c and paths α, β
Output: the structure G described above

1. E_ϕ := the set of edge labels appearing in either $\alpha \cdot \beta$ or some path in constraints of Σ;

2. $Rules := \Sigma$;

3. $G := (|G|, r^G, E_\phi^G)$, where
   - $|G| = \{o(\rho) \mid \rho \preceq \alpha \cdot \beta,\ o(\rho)$ is a distinct node$\}$,
   - $r^G = o(\epsilon)$,
   - E_ϕ^G is populated such that $G \models K(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot K$;

4. repeat until no further change:
   (1) if $\forall\, x\,(\rho(r, x) \rightarrow \forall\, y\,(\varrho(x, y) \rightarrow \zeta(x, y))) \in \Sigma$ and there are $o_\rho, o_{\rho\cdot\varrho} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \varrho(o_\rho, o_{\rho\cdot\varrho})$ then
      (i) $Rules := Rules \setminus \{\forall\, x\,(\rho(r, x) \rightarrow \forall\, y\,(\varrho(x, y) \rightarrow \zeta(x, y)))\}$;
      (ii) for each $\xi \cdot K \preceq \zeta$ do
         if there is no $o \in |G|$ such that $G \models \xi \cdot K(o_\rho, o)$ then
            (a) add to $|G|$ a distinct node $o_{\rho\cdot\xi\cdot K}$;
            (b) add to E_ϕ^G an edge labeled K from $o_{\rho\cdot\xi}$ to $o_{\rho\cdot\xi\cdot K}$,
               where $o_{\rho\cdot\xi} \in |G|$ such that $G \models \xi(o_\rho, o_{\rho\cdot\xi})$;
      (iii) $merge(o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta})$;
   (2) if $\forall\, x\,(\rho(r, x) \rightarrow \forall\, y\,(\varrho(x, y) \rightarrow \zeta(y, x))) \in \Sigma$ and there are $o_\rho, o_{\rho\cdot\varrho} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \varrho(o_\rho, o_{\rho\cdot\varrho})$ then
      (i) $Rules := Rules \setminus \{\forall\, x\,(\rho(r, x) \rightarrow \forall\, y\,(\varrho(x, y) \rightarrow \zeta(y, x)))\}$;
      (ii) for each $\xi \cdot K \preceq \zeta$ do
         if there is no $o \in |G|$ such that $G \models \xi \cdot K(o_{\rho\cdot\varrho}, o)$ then
            (a) add to $|G|$ a distinct node $o_{\rho\cdot\varrho\cdot\xi\cdot K}$;
            (b) add to E_ϕ^G an edge labeled K from $o_{\rho\cdot\varrho\cdot\xi}$ to $o_{\rho\cdot\varrho\cdot\xi\cdot K}$,
               where $o_{\rho\cdot\varrho\cdot\xi} \in |G|$ such that $G \models \xi(o_{\rho\cdot\varrho}, o_{\rho\cdot\varrho\cdot\xi})$;
      (iii) $merge(o_\rho, o_{\rho\cdot\varrho\cdot\zeta})$;

5. output G.

Table 3: An algorithm for testing path constraint implication in DM

Proposition 4.7: Given a finite subset Σ of P_c and paths α, β, Algorithm 4.1 computes a finite deterministic structure G having the following property: there are $a, b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$, and in addition, for any path γ,

$$G \models \gamma(a, b) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))),$$
$$G \models \gamma(b, a) \text{ iff } \Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))).$$

■

Proof: The step 4 of Algorithm 4.1 ensures that $G \models \Sigma$, taking advantage of the fact that G is deterministic and by using Lemma 2.1. In addition, step 3 ensures that there are $a, b \in |G|$, such that

$$G \models \alpha(r^G, a) \wedge \beta(a, b).$$

Thus if $\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y)))$, then by Theorem 4.5, $G \models \gamma(a, b)$. Similarly, if $\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x)))$, then $G \models \gamma(b, a)$.

Conversely, by a straightforward induction on the number of steps in the construction of G by the algorithm, we can show that for all paths ρ and ϱ, if there exists node $o \in |G|$ such that $G \models \rho(r^G, o) \wedge \varrho(r^G, o)$, then $\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\rho(r, x) \rightarrow \varrho(r, x))$. Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_c$. For example, step 4 (1) corresponds to an application of Forward-to-word, step 4 (2) corresponds to an application of Backward-to-word, step 4 (1) (ii) and 4 (2) (ii) correspond to applications of Prefix, and *merge* corresponds to applications of Transitivity, Right-Congruence and Symmetry in $\mathcal{I}_c$. As a result, if $G \models \gamma(a, b)$, then by Word-to-forward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(x, y))).$$

Similarly, if $G \models \gamma(b, a)$, then by Word-to-backward in $\mathcal{I}_c$, we have

$$\Sigma \cup \{\exists x\,(\alpha \cdot \beta(r, x))\} \vdash_{\mathcal{I}_c} \forall x\,(\alpha(r, x) \rightarrow \forall y\,(\beta(x, y) \rightarrow \gamma(y, x))).$$

■

From Proposition 4.7, Algorithm 4.1 and Theorem 4.5, the corollary below follows immediately.

Corollary 4.8: In the context of DM, the implication and finite implication problems for P_c are decidable in cubic-time. ■

5 The implication problems for P_c^-

In this section, we show that in contrast to Corollary 2.4, the implication and finite implication problems for P_c^- are decidable in the context of the deterministic data model.

Proposition 5.1: In the context of DM, the implication and finite implication problems for P_c^- are decidable. ∎

Proof: To establish the decidability of the implication and finite implication problems for P_c^-, it suffices to give a finite model argument. That is, it suffices to show the following claim.

Claim: Let $\Sigma \cup \{\varphi\}$ be a finite subset of P_c^-, and let $\phi = \bigwedge \Sigma \wedge \neg\varphi$. If there is a deterministic structure G such that $G \models \phi$, then there is a finite deterministic structure H such that $H \models \phi$.

For if the claim holds, then the implication and finite implication problems for P_c^- coincide and are decidable.

To show the claim, assume that there is a deterministic structure G satisfying ϕ. Recall that a constraint ψ of P_c^- is of either the form

- $\forall x\,(pf(\psi)(r,\, x) \rightarrow \forall y\,(lt(\psi)(x,\, y) \rightarrow rt(\psi)(x,\, y)))$ (i.e., the forward form), or the form
- $\forall x\,(pf(\psi)(r,\, x) \rightarrow \forall y\,(lt(\psi)(x,\, y) \rightarrow rt(\psi)(y,\, x)))$ (i.e., the backward form),

where $pf(\psi)$, $lt(\psi)$ and $rt(\psi)$ are $*$-free regular expressions, as described in Definition 2.2. Let

$$\begin{aligned} PEs(\phi) &= \{pf(\psi)\cdot lt(\psi),\ pf(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the forward form}\} \ \cup \\ &\qquad \{pf(\psi)\cdot lt(\psi)\cdot rt(\psi) \mid \psi \in \Sigma \cup \{\varphi\},\ \psi \text{ is of the backward form}\}, \\ Pts(\phi) &= \{\varrho \mid \varrho \text{ is a path},\ p \in PEs(\phi),\ \varrho \in p\}, \\ CloPts(\phi) &= \{\rho \mid \varrho \in Pts(\phi),\ \rho \preceq \varrho\}. \end{aligned}$$

Here $\varrho \in p$ means that path ϱ is in the regular language generated by $*$-free regular expression p, and $\rho \preceq \varrho$ stands for that path ρ is a prefix of path ϱ. Let E_ϕ be the set of edge labels appearing in some path in $Pts(\phi)$. Then we define H to be $(|H|,\, r^H,\, E^H)$ such that

- $|H| = \{a \mid a \in |G|,\ \rho \in CloPts(\phi),\ G \models \rho(r^G,\, a)\}$,
- $r^H = r^G$,
- for all $a, b \in |H|$ and $K \in E$, $H \models K(a,\, b)$ iff $K \in E_\phi$ and $G \models K(a,\, b)$.

It is easy to verify that $H \models \phi$ and H is deterministic, since G has these properties. By Lemma 2.1, the size of $|H|$ is at most the cardinality of $CloPts(\phi)$, which is finite because the regular language generated by a $*$-free regular expression is finite. This proves the claim. It should be noted that E_ϕ and $CloPts(\phi)$ are determined by ϕ only. ∎

6 The implication problems for P_c^*

This section establishes the undecidability results of the paper:

Theorem 6.1: In the context of DM, the implication and finite implication problems for P_c^* are undecidable. ■

Theorem 6.1 shows that the determinism condition of DM does not trivialize the problem of path constraint implication.

We prove Theorem 6.1 by reduction from the word problem for (finite) monoids. Before we present the details of the proof, we first review the word problem for (finite) monoids.

6.1 The word problem for (finite) monoids

Recall the following notions from [2].

A *monoid* is a triple $(M, \circ, 1)$, where

- M is a nonempty set,
- $\circ$ is an associative binary relation on M, and
- 1 is an element of M that is the identity for $\circ$. That is, for any $a \in M$, $1 \circ a = a = a \circ 1$.

A monoid $(M, \circ, 1)$ is said to be finite if M is finite.

Let $?$ be a finite alphabet. The *free monoid generated by* $?$ is $(?^*, \cdot, \epsilon)$, where

- $?^*$ is the set of all finite strings with letters in $?$,
- $\cdot$ is the concatenation operator on strings, and
- ϵ is the empty string.

Let $?$ be a finite alphabet. An *equation* (over $?$) is a pair (α, β) of strings in $?^*$.

Let a finite set of equations

$$\Theta = \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in ?^*,\ i \in [1, n]\},$$

and a *test equation* θ be (α, β), where $\alpha, \beta \in ?^*$. Then $\Theta \models \theta$ ($\Theta \models_f \theta$) if for every (finite) monoid $(M, \circ, 1)$ and every homomorphism $h : ?^* \to M$, if $h(\alpha_i) = h(\beta_i)$ for each $i \in [1, n]$, then $h(\alpha) = h(\beta)$.

The *word problem for (finite) monoids* is the problem of determining, given Θ and θ, whether $\Theta \models \theta$ ($\Theta \models_f \theta$).

The following result is well-known (see, e.g., [2]).

Theorem 6.2: Both the word problem for monoids and the word problem for finite monoids are undecidable. ■

6.2 Reduction from the word problem for (finite) monoids

Next, we present an encoding of the word problem for (finite) monoids in terms of the (finite) implication problem for P_c^* in the context of DM.

Let $?_0$ be a finite alphabet and Θ_0 be a finite set of equations over $?_0$. Without loss of generality, assume $?_0 \subseteq E$, where E is the set of binary relation symbols in σ. Assume

$$\begin{aligned} ?_0 &= \{K_j \mid j \in [1,m],\ K_i \neq K_j \text{ if } i \neq j\}, \\ \Theta_0 &= \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in ?_0^*,\ i \in [1,n]\}. \end{aligned}$$

Note here that each symbol in $?_0$ is a binary relation symbol in E. Therefore, every α in $?_0^*$ can be represented as a path formula, also denoted by α. In addition, we use $\cdot$ to denote the concatenation operator for both paths and strings.

Let e_0 be the regular expression defined by:

$$e_0 = (K_1 + K_2 + \ldots + K_m)^*$$

We encode Θ_0 in terms of a subset Σ of P_c^*, as follows:

$$\begin{aligned} \Sigma = &\{\forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\alpha_i(x,\,y) \rightarrow \beta_i(x,\,y))) \mid i \in [1,n]\} \ \cup \\ &\{\forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\beta_i(x,\,y) \rightarrow \alpha_i(x,\,y))) \mid i \in [1,n]\}. \end{aligned}$$

Let $(\alpha,\,\beta)$ be a test equation, where α and β are arbitrary strings in $?_0^*$. We encode this test equation as

$$\varphi = \forall x\,(e_0(r,\,x) \rightarrow \forall y\,(\alpha(x,\,y) \rightarrow \beta(x,\,y))).$$

It should be noted that in the encoding above, only forward constraints of P_c^* are used. In addition, for each $\psi \in \Sigma \cup \{\varphi\}$, $lt(\psi)$ and $rt(\psi)$ are simply paths rather than complex regular expressions, where $lt(\psi)$ and $rt(\psi)$ are described in Definition 2.1.

The lemma below shows that the encoding above is indeed a reduction from the word problem for (finite) monoids.

Lemma 6.3: Let Θ_0, $(\alpha,\,\beta)$, Σ and φ be described as above. Then in the context of DM,

$$\Theta_0 \models (\alpha,\,\beta) \quad \text{iff} \quad \Sigma \models \varphi, \tag{a}$$

$$\Theta_0 \models_f (\alpha,\,\beta) \quad \text{iff} \quad \Sigma \models_f \varphi. \tag{b}$$

■

Proof: We prove (b) only. The proof of (a) is similar and simpler.

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha,\,\beta)$. Then we show that $\Sigma \not\models_f \varphi$. That is, we show that there exists a finite deterministic structure G, such that $G \models \Sigma$ and $G \not\models \varphi$.

To do this, we first define some notations. By $\Theta_0 \not\models_f (\alpha, \beta)$, there exist a finite monoid $(M, \circ, 1)$ and a homomorphism $h : ?_0^* \to M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Based on M and h, we define an equivalence relation $\approx$ on $?_0^*$ as follows:

$$\rho \approx \varrho \quad \text{iff} \quad h(\rho) = h(\varrho).$$

For every $\rho \in ?_0^*$, let $\widehat{\rho}$ be the equivalence class of ρ with respect to $\approx$. Let

$$C_{\Theta_0} = \{\widehat{\rho} \mid \rho \in ?_0^*\}.$$

Using these notations, we construct a deterministic structure $G = (|G|, r^G, E^G)$ as follows.

(1) $|G|$.

For each $\widehat{\rho} \in C_{\Theta_0}$, let $o(\widehat{\rho})$ be a distinct node. Then we define

$$|G| = \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o(\widehat{\epsilon})$.

(3) The binary relations are populated as follows: For any $K \in E$ and $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$, $G \models K(o(\widehat{\rho}), o(\widehat{\varrho}))$ iff $\rho \cdot K \in \widehat{\varrho}$.

Next, we show that G is indeed the structure desired. More specifically, we verify the following claims.

Claim 1: G is a finite deterministic structure.

To show that G is finite, it is sufficient to show that C_{Θ_0} is finite. Consider a function $f : C_{\Theta_0} \to M$ defined by

$$f : \ \widehat{\rho} \mapsto h(\rho).$$

Clearly, f is well-defined, total and injective. Therefore, because M is finite, C_{Θ_0} is also finite.

We next show that G is deterministic. By the construction of G, it is easy to see that for every $\rho \in ?_0^*$ and $j \in [1, m]$, $o(\widehat{\rho \cdot K_j})$ is the unique node such that $G \models K_j(o(\widehat{\rho}), o(\widehat{\rho \cdot K_j}))$. This is because h is a homomorphism, and as a result, if $\rho_1 \approx \rho_2$, then

$$\begin{aligned} h(\rho_1 \cdot K_j) &= h(\rho_1) \circ h(K_j) \\ &= h(\rho_2) \circ h(K_j) \\ &= h(\rho_2 \cdot K_j). \end{aligned}$$

Claim 2: $G \models \Sigma$.

Suppose, for *reductio*, that there is $i \in [1, n]$ such that

$$G \not\models \forall\, x\, (e_0(r,\, x) \to \forall\, y\, (\alpha_i(x,\, y) \to \beta_i(x,\, y))).$$

Then there is $\gamma \in e_o$ and $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$ such that

$$G \models \gamma(r^G,\, o(\widehat{\rho})) \wedge \alpha_i(o(\widehat{\rho}),\, o(\widehat{\varrho})) \wedge \neg\beta_i(o(\widehat{\rho}),\, o(\widehat{\varrho})).$$

To see that this leads to a contradiction, it suffices to show:

Fact 1: For all $o(\widehat{\rho}), o(\widehat{\varrho}) \in |G|$ and $\xi \in ?^*_0$,

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\varrho})) \;\text{ iff }\; \widehat{\rho \cdot \xi} = \widehat{\varrho}.$$

For if Fact 1 holds, then by the assumption, $\widehat{\rho \cdot \alpha_i} = \widehat{\varrho}$, but $\widehat{\rho \cdot \beta_i} \neq \widehat{\varrho}$. However, since h is a homomorphism and $\alpha_i \approx \beta_i$, we have

$$\begin{aligned} h(\rho \cdot \alpha_i) &= h(\rho) \circ h(\alpha_i) \\ &= h(\rho) \circ h(\beta_i) \\ &= h(\rho \cdot \beta_i). \end{aligned}$$

Thus $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. Hence $\widehat{\rho \cdot \alpha_i} = \widehat{\rho \cdot \beta_i}$. This contradicts the assumption.

Similarly, we can also show that for every $i \in [1, n]$,

$$G \models \forall\, x\, (e_0(r,\, x) \to \forall\, y\, (\beta_i(x,\, y) \to \alpha_i(x,\, y))).$$

We show Fact 1 by induction on $|\xi|$.

Base case: $\xi = \epsilon$.

Clearly, $G \models \epsilon(o(\widehat{\rho}),\, o(\widehat{\varrho}))$ iff $o(\widehat{\rho}) = o(\widehat{\varrho})$ iff $\widehat{\rho} = \widehat{\varrho}$.

Inductive step: Assume Fact 1 for ξ. We next show that Fact 1 also holds for $\xi \cdot K$.

If $G \models \xi \cdot K(o(\widehat{\rho}),\, o(\widehat{\varrho}))$, then by induction hypothesis and Lemma 2.1, we have

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \xi})) \wedge K(o(\widehat{\rho \cdot \xi}),\, o(\widehat{\varrho})).$$

By the definition of G, $G \models K(o(\widehat{\rho \cdot \xi}),\, o(\widehat{\varrho}))$ iff $\rho \cdot \xi \cdot K \in \widehat{\varrho}$. Therefore,

$$\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}.$$

Conversely, suppose that $\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}$. By induction hypothesis,

$$G \models \xi(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \xi})).$$

Again by the definition of G, $G \models K(o(\widehat{\rho \cdot \xi}),\, o(\rho \cdot \widehat{\xi \cdot K}))$. Hence by $\rho \cdot \widehat{\xi \cdot K} = \widehat{\varrho}$, we have

$$G \models \xi \cdot K(o(\widehat{\rho}),\, o(\widehat{\varrho})).$$

Claim 3: $G \not\models \varphi$.

By Fact 1, $G \models \alpha(r^G,\, o(\widehat{\alpha}))$. By $h(\alpha) \neq h(\beta)$, we have

$$o(\widehat{\alpha}) \neq o(\widehat{\beta}).$$

Therefore, again by Fact 1, we have $G \not\models \beta(r^G,\, o(\widehat{\alpha}))$. Hence

$$G \models \alpha(r^G,\, o(\widehat{\alpha})) \wedge \neg\beta(r^G,\, o(\widehat{\alpha})).$$

Note that $\epsilon \in e_0$. That is, the empty path ϵ is in the language generated by the regular expression e_0. Thus

$$G \models \exists\, x\, y\, (e_0(r^G,\, x) \wedge \alpha(x,\, y) \wedge \neg\beta(x,\, y)).$$

That is, $G \not\models \varphi$.

(*only if*) Suppose that there exists a finite deterministic structure G such that $G \models \Sigma$ and $G \not\models \varphi$. Then we show that $\Theta_0 \not\models_f (\alpha,\, \beta)$. More specifically, we define a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : ?_0^* \to M$ such that for every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation $\sim$ on $?_0^*$, as follows:

$$\rho \sim \varrho \;\text{ iff }\; G \models \forall\, x(e_0(r, x) \to \forall\, y\, (\rho(x, y) \to \varrho(x, y))) \wedge \forall\, x\, (e_0(r, x) \to \forall\, y\, (\varrho(x, y) \to \rho(x, y))).$$

Then by $G \models \Sigma$, for every $i \in [1, n]$, we have $\alpha_i \sim \beta_i$. In addition, by $G \not\models \varphi$, we have $\alpha \not\sim \beta$.

For every $\rho \in ?_0^*$, let $[\rho]$ denote the equivalence class of ρ with respect to $\sim$. Then clearly, for every $i \in [1, n]$, $[\alpha_i] = [\beta_i]$. However, we have $[\alpha] \neq [\beta]$.

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in ?_0^*\}.$$

An important property of M is described as follows.

Claim 4: M is finite.

To show this, for every $\rho \in ?_0^*$, let

$$S_\rho = \{(a, b) \mid a, b \in |G|,\; G \models e_0(r^G,\, a) \wedge \rho(a,\, b)\}.$$

In addition, let

$$S_G = \{S_\rho \mid \rho \in ?_0^*\}.$$

Since $S_\rho \subseteq |G| \times |G|$ and $|G|$ is finite, S_G is finite. Moreover, it is easy to verify the following:

Fact 2: For all $\rho, \varrho \in ?_0^*$, $\rho \sim \varrho$ iff $S_\rho = S_\varrho$.

To see that Fact 2 holds, first assume that $\rho \sim \varrho$. Then for each $(a,\, b) \in S_\rho$, by the definition of S_ρ, we have

$$G \models e_0(r^G,\, a) \wedge \rho(a,\, b).$$

By the definition of $\sim$ and the assumption that $\rho \sim \varrho$, we have

$$G \models e_0(r^G, a) \wedge \varrho(a, b).$$

Hence $(a, b) \in S_\varrho$. Therefore, $S_\rho \subseteq S_\varrho$. Similarly, it can be shown that $S_\varrho \subseteq S_\rho$. Hence

$$S_\rho = S_\varrho.$$

Conversely, assume that $S_\rho = S_\varrho$. Suppose, for *reductio*, that $\rho \not\sim \varrho$. Without loss of generality, assume that

$$G \not\models \forall x(e_0(r, x) \rightarrow \forall y\,(\rho(x, y) \rightarrow \varrho(x, y))).$$

Then there exist $a, b \in |G|$, such that

$$G \models e_0(r^G, a) \wedge \rho(a, b) \wedge \neg\varrho(a, b).$$

That is, $(a, b) \in S_\rho$ but $(a, b) \notin S_\varrho$. Hence $S_\rho \neq S_\varrho$. This contradicts the assumption. Therefore, Fact 2 holds.

Next, consider a function $g : M \rightarrow S_G$ defined by

$$g : [\rho] \mapsto S_\rho.$$

Using Fact 2 above, it is easy to see that g is well-defined, total and injective. Therefore, because S_G is finite, M is also finite.

Next, we define a binary operation $\circ$ on M by

$$[\rho] \circ [\varrho] = [\rho \cdot \varrho].$$

It is easy to verify the following claims.

Claim 5: $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in ?^*_0$ such that $\rho_1 \sim \rho_2$ and $\varrho_1 \sim \varrho_2$, we show that

$$\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2.$$

To do this, consider all $o, o_1 \in |G|$ such that

$$G \models e_0(r^G, o) \wedge \rho_1 \cdot \varrho_1(o, o_1).$$

Clearly, there exists $o' \in |G|$ such that

$$G \models \rho_1(o, o') \wedge \varrho_1(o', o_1).$$

By $\rho_1 \sim \rho_2$, we have

$$G \models \rho_2(o, o').$$

Note that $e_0 \cdot \rho_2 \subseteq e_0$. That is, the language generated by the regular expression $e_0 \cdot \rho$ is contained in the language generated by e_0. By $\varrho_1 \sim \varrho_2$, we also have

$$G \models \varrho_2(o', o_1).$$

Hence

$$G \models \rho_2 \cdot \varrho_2(o, o_1).$$

Therefore,

$$G \models \forall x\,(e_0(r, x) \to \forall y\,(\rho_1 \cdot \varrho_1(x, y) \to \rho_2 \cdot \varrho_2(x, y))).$$

Similarly, we can show that

$$G \models \forall x\,(e_0(r, x) \to \forall y\,(\rho_2 \cdot \varrho_2(x, y) \to \rho_1 \cdot \varrho_1(x, y))).$$

Therefore, $\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2$. Hence $\circ$ is well-defined.

Claim 6: $\circ$ is associative.

This is because for all $[\rho], [\varrho], [\xi] \in M$,

$$\begin{aligned} ([\rho] \circ [\varrho]) \circ [\xi] &= [\rho \cdot \varrho] \circ [\xi] \\ &= [\rho \cdot \varrho \cdot \xi] \\ &= [\rho] \circ ([\varrho \cdot \xi]) \\ &= [\rho] \circ ([\varrho] \circ [\xi]). \end{aligned}$$

Claim 7: $[\epsilon]$ is the identity for $\circ$. This is because for any $[\rho] \in M$,

$$[\epsilon] \circ [\rho] = [\rho] = [\rho] \circ [\epsilon].$$

These claims show that $(M, \circ, [\epsilon])$ is a finite monoid.

Finally, we define $h : ?_0^* \to M$ by

$$h : \ \rho \mapsto [\rho].$$

Clearly, h is a homomorphism since

$$h(\rho \cdot \varrho) = [\rho \cdot \varrho] = [\rho] \circ [\varrho] = h(\rho) \circ h(\varrho).$$

In addition, for every $i \in [1, n]$, by $[\alpha_i] = [\beta_i]$, $h(\alpha_i) = h(\beta_i)$. Moreover, by $[\alpha] \neq [\beta]$, $h(\alpha) \neq h(\beta)$. Therefore,

$$\Theta_0 \not\models_f (\alpha, \beta).$$

This completes the proof of Lemma 6.3. ∎

From Lemma 6.3 and Theorem 6.2, Theorem 6.1 follows immediately.

7 Conclusions

We have investigated path constraints for the deterministic data model DM. Four path constraint languages have been considered: P_w, P_c, P_c^- and P_c^*. While P_w and P_c were studied for the graph model SM for semistructured data [4, 10], P_c^- and P_c^* have not appeared in any literature. We have demonstrated how constraints of these languages might be used for, among other things, query optimization. We have also studied implication problems associated with these constraint languages in the context of DM. More specifically, we have presented a finite axiomatization for P_w in the context of DM. We have also shown that in contrast to the undecidability result of [10] established for SM, the implication and finite implication problems for P_c and P_c^- are decidable in the context of DM. In particular, the implication problems associated with P_c are decidable in cubic-time and are finitely axiomatizable. These results demonstrate that the determinism condition of DM simplifies the analysis of path constraint implication. However, we have also established the undecidability of the implication and finite implication problems for P_c^* in the context of DM. This undecidability result shows that the determinism condition of DM does not trivialize the problem of path constraint implication.

A number of important questions are open.

First, a more general deterministic data model for semistructured data, DDM, was proposed in [9], in which edge labels may also have structure. A type system for DDM is currently under development, in which certain path constraints are embedded. A natural question here is: do the decidability and undecidability results established here hold in DDM? This question becomes more intriguing when types are considered. As shown in [11], adding a type to the data in some cases simplifies reasoning about path constraints, and in other cases makes it harder.

Second, to define a richer data model for semistructured data, one may want to replace the set of edge labels with a set of logic formulas, which possesses a decidable satisfiability problem. A question here is: in this new setting, do the decidability results of this paper still hold?

Third, can path constraints help in reasoning about the equivalence of data representations?

Fourth, how should path constraints be used in reasoning about the containment and equivalence of path queries? What kind of automatic tools should be developed to achieve this?

Finally, another path constraint language, denoted by P_w^*, was also investigated in [4]. The language P_w^* is a subclass of P_c^*, defined by:

$$P_w^* = \{\phi \mid \phi \in P_c^*,\ \phi \text{ is of the forward form, } pf(\phi) = \epsilon\}.$$

It has been shown in [4] that in SM, the implication and finite implication problems for P_w^* are decidable in EXPSPACE. In the context of DM, however, it remains open whether these problems are decidable.

Acknowledgements. We thank Victor Vianu for comments and discussions.

References

[1] S. Abiteboul. "Querying semi-structured data". In *Proceedings of the 6th International Conference on Database Theory*, 1997.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesly Publishing Company, 1995.

[3] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner. "The lorel query language for semistructured data". *Journal of Digital Libraries*, 1(1), 1997.

[4] S. Abiteboul and V. Vianu. "Regular path queries with constraints". In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[5] T. Bray, C. Frankston, and A. Malhotra. "Document Content Description for XML". W3C Note NOTE-dcd-19980731. Available as `http://www.w3.org/TR/NOTE-dcd`.

[6] T. Bray, J. Paoli, and C. M. Sperberg-McQueen. "Extensible Markup Language (XML) 1.0". W3C Recommendation REC-xml-19980210. See `http://www.w3.org/TR/REC-xml`.

[7] P. Buneman. "Semistructured data". Tutorial in *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, 1997.

[8] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. "A query language and optimization techniques for unstructured data". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 505-516, 1996.

[9] P. Buneman, A. Deutsch, and W. Tan. "A deterministic model for semi-structured data". In *Proceedings of Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats*, 1999.

[10] P. Buneman, W. Fan, and S. Weinstein. "Path constraints on semistructured and structured data". In *Proceedings of the 17th ACM Symposium on Principles of Database Systems*, 1998. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/pods98.ps.gz`.

[11] P. Buneman, W. Fan, and S. Weinstein. "Interaction between path and type constraints". In *Proceedings of the 18th ACM Symposium on Principles of Database Systems*, 1999. Available as `ftp://ftp.cis.upenn.edu/pub/papers/db-research/pods99.ps.gz`.

[12] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "XML-QL: a query language for XML". W3C Note NOTE-xml-ql-19980819. See `http://www.w3.org/TR/NOTE-xml-ql`.

[13] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu. "Catching the boat with Strudel: experience with a Web-site management system". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 414-425, 1998.

[14] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[15] M. Fuchs, M. Maloney, and A. Milowski. "Schema for object-oriented XML". W3C Note NOTE-SOX-19980930. See `http://www.w3.org/TR/NOTE-SOX`.

[16] E. Grädel, M. Otto, and E. Rosen. "Undecidability results on two-variable logics". In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, *LNCS 1200*, 1997.

[17] D. Harel. "Dynamic logic". In D. M. Gabbay and F. Guenthner, eds., *Handbook of Philosophical Logic. II: Extensions of Classical Logic*, pp. 497-604, 1984.

[18] N. Immerman. "Languages that capture complexity classes". *SIAM Journal of Computing*, 16: 760-778, 1987.

[19] O. Lassila and R. R. Swick. "Resource Description Framework (RDF) model and syntax specification". W3C Working Draft WD-rdf-syntax-19981008. See `http://www.w3.org/TR/WD-rdf-syntax`.

[20] A. Layman, E. Jung, E. Maler, H. S. Thompson, J. Paoli, J. Tigue, N. H. Mikula, and S. De Rose. "XML-Data". W3C Note NOTE-XML-data-980105. Available as `http://www.w3.org/TR/1998/NOTE-XML-data`.

[21] E. Maler and S. DeRose. "XML Linking language (XLink)". W3C Working Draft WD-xlink-19980303. Available as `http://www.w3.org/TR/WD-xlink`.

[22] A. O. Mendelzon, G. A. Mihaila, and T. Milo. "Querying the World Wide Web". In *Proceedings of the 4th International Conference on Parallel and Distributed Information Systems*, pp. 80-91, 1996.

[23] L. Popa and V. Tannen. "An equational chase for path-conjunctive queries, constraints, and views". In *Proceedings of the 7th International Conference on Database Theory*, 1999.

[24] W. C. Rounds. "Feature logics". In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*. Elsevier, 1997.

[25] J. Thierry-Mieg and R. Durbin. "Syntactic definitions for the ACEDB data base manager". Technical Report MRC-LMB xx.92, MRC Laboratory for Molecular Biology, Cambridge, CB2 2QH, UK, 1992.

[26] M. Y. Vardi and P. Wolper. "Automata-theoretic techniques for modal logic of programs". *Journal of Computer and System Sciences*, 32(2): 182-221, April 1986.