# Interaction between Path and Type Constraints

**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

July 1998

(Revised November 1998)

**Abstract**

XML [7], which is emerging as an important standard for data exchange on the World Wide Web, highlights the importance of semistructured data. Although the XML standard itself does not require any schema or type system, a number of proposals [6, 15, 18] have been developed that roughly correspond to data definition languages. These allow one to constrain the structure of XML data by imposing a schema on it. These and other proposals also advocate the need for integrity constraints, another form of constraints that should, for example, be capable of expressing inclusion constraints and inverse relationships. The latter have recently been studied as path constraints in the context of semistructured data [4, 11]. It is likely that future XML proposals will involve both forms of constraint, and it is therefore appropriate to understand the interaction between them.

This paper investigates that interaction. In particular it studies constraint implication problems, which are important both in understanding the semantics of type/constraint systems and in query optimization. A number of results on path constraint implication are established in the presence and absence of type systems. These results demonstrate that adding a type system may in some cases simplify reasoning about path constraints and in other cases make it harder. For example, it is shown that there is a path constraint implication problem that is decidable in PTIME in the untyped context, but that becomes undecidable when a type system is added. On the other hand, there is an implication problem that is undecidable in the untyped context, but becomes not only decidable in cubic time but also finitely axiomatizable when a type system is imposed.

[*]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

Among the numerous proposals for adding structure or semantics to XML documents [7], several [6, 15, 17, 18] advocate the need for integrity constraints. However, concrete proposals for constraint systems have yet to be developed. Whether such constraints will be specified as extensions to existing type systems such as XML-Data [18], SOX [15], DCD [6], or whether they will be added as independent constructs, is not yet clear, and, in all probability, they will be added in both ways. XLink [19], for example, is independent of any type system and can express simple co-reference constraints. It is therefore appropriate to study constraints and type systems separately and to understand their interaction.

Integrity constraints for semistructured data were originally studied as path constraints in [4]. While these constraints could specify inclusions between paths, they were not expressive enough to capture, say, inverse constraints. Extensions were studied in [9, 10, 11] to overcome this limitation. The central technical problem investigated in these papers has been the question of constraint implication: given that certain constraints are known to hold, does it follow that some other constraint is necessarily satisfied? A number of decidability and undecidability results were established in these papers for semistructured data, i.e., data unconstrained by any type system or schema. In this paper, we extend the work reported in [9, 10, 11] by investigating the interaction between type systems and constraint systems. An interesting result presented here is that adding a type system may in some cases simplify the analysis of path constraint implication and in other cases make it harder. On the one hand, we exhibit an implication problem associated with path constraints that is undecidable in the context of semistructured data, but becomes decidable in cubic-time when a (restricted) type system is added. On the other hand, we give an example of a constraint implication problem that is decidable in PTIME in the untyped context, but becomes undecidable when a (generic) type system is imposed. The practical interest of this phenomena is addressed in Section 2, where we discuss two instances of these implication problems.

**An example.** To cast the problem concretely, the structure represented in Figure 1 describes the following XML document:

```
<?XML version = "1.0">
<bib>
     <book ISBN = "12"   author = "345">
           <title> ...  </title>
           <year> ...  </year>
     </book>
     <book ISBN = "23"   author = "123"   ref = "12">
           <title> ...  </title>
     </book>
     <book ISBN = "34"   author = "123"   ref = "23">
           <title> ...  </title>
     </book>
```

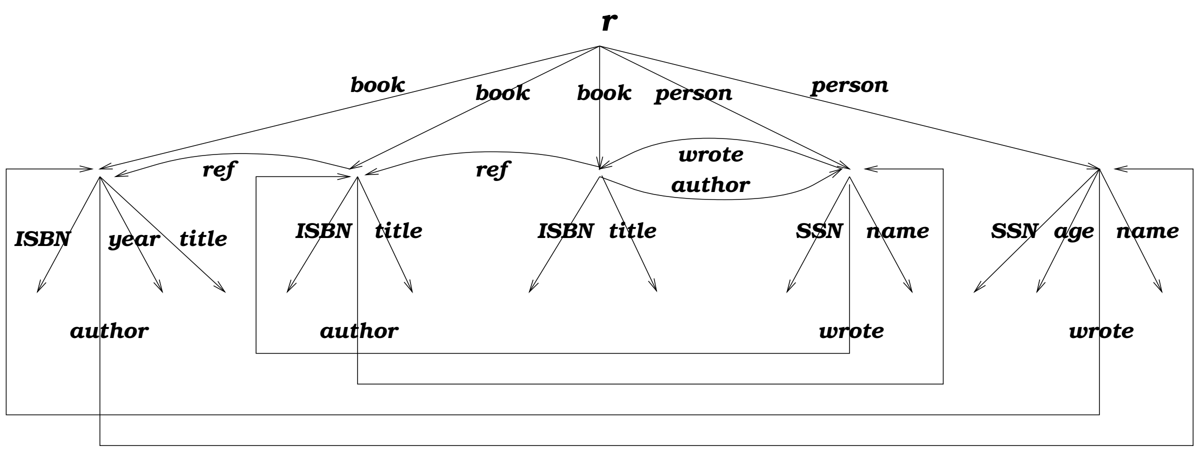

Figure 1: Representation of an XML document

```
      <person SSN = "123"   wrote = "34 23">
              <name> ...  </name>
      </person>
      <person SSN = "345"   wrote = "12">
              <name> ...  </name>
              <age> ...  </age>
      </person>
 </bib>
```

It is an example of semistructured data and could be expressed in a number of other data formats. Typical path constraints on this graph describe an inverse relationship between `author` and `wrote`. This can be expressed as:

$$\forall\, x\, (book(r,\, x) \to \forall\, y\, (author(x,\, y) \to wrote(y, x)))$$
$$\forall\, x\, (person(r,\, x) \to \forall\, y\, (wrote(x,\, y) \to author(y,\, x)))$$

Here $r$ denotes the root of the graph, variables $x$ and $y$ range over vertices, and the predicates denote edge labels. A path in the graph is a sequence of edge labels, which can be expressed as a formula $\alpha(x,\, y)$ denoting that $\alpha$ is a sequence of edge labels from vertex $x$ to $y$. For example, $book \cdot author(r,\, x)$ is a path from root $r$ to some vertex $x$ in Figure 1.

Note that we have introduced these constraints *before* any mention of a type system. These are the kind of constraints that have been studied in [4, 9, 10, 11].

In addition, we may also want to impose a *type* on the document. For example, a type specified in XML-Data would be:

```
 <elementType   id = "person">
   <attribute   name="SSN"/>
   <attribute   name="wrote"      range= "#book"/>
```

```
    <element     type="#name"/>
    <element     type="#age"       occurs="optional"/>
  </elementType>

  <elementType   id = "book">
    <attribute   name="ISBN"/>
    <attribute   name="author"     range="#person"/>
    <attribute   name="ref"        range="#book"/>
    <element     type="#title"/>
    <element     type="#year"      occurs="optional"/>
  </elementType>

  <elementType   id = "name">
    <string/>
  </elementType>

  <elementType   id = "age">
    <string/>
  </elementType>

  <elementType   id = "title">
    <string/>
  </elementType>

  <elementType   id = "year">
    <string/>
  </elementType>
```

Types also constrain the data, but in a very different fashion. We are therefore interested in the interaction between these two forms of constraints.

**Word and path constraints.** A class of constraints, called *word constraints*, was introduced and studied in [4][1]. Referring to Figure 1, typical word constraints are:

$$\begin{array}{l} \forall x\,(book \cdot author(r,\,x) \to person(r,\,x)) \\ \forall x\,(person \cdot wrote(r,\,x) \to book(r,\,x)) \\ \forall x\,(book \cdot ref(r,\,x) \to book(r,\,x)) \end{array}$$

Suppose Figure 1 represents a bibliography database at University of Penn. Let us refer to this database as Penn-bib. Abusing object-oriented database terms, the word constraints above assert that an author of a book in Penn-bib must be in the database "extent" of `person` in Penn-bib, a book written by a person in Penn-bib must occur in Penn-bib "extent" of `book`, etc. These are typical integrity constraints and were called *extent constraints* in [11]. It was

[1]Also considered in [4] was a form of constraints in which paths are represented by regular expressions.

shown in [4] that in the context of semistructured data, the implication and finite implication problems for word constraints are decidable in PTIME.

The class of path constraints studied in [9, 10, 11], $P_c$, is a mild generalization of word constraints. The inverse constraints above are in $P_c$ but are not word constraints. As another example, consider Penn-bib again. This database may have links to external resources, such as bibliography databases at MIT and Warner. Call them MIT-bib and Warner-bib, respectively. These databases can be viewed as components of Penn-bib, and therefore, are called *local databases* of Penn-bib. In our graph representation, this can be depicted by adding two edges emanating from the root node $r$ of Penn-bib that are labeled with `MIT`, `Warner`, and lead to MIT-bib and Warner-bib, respectively. It is natural to expect the extent and inverse constraints above to hold on these local databases. For example, the extent and inverse constraints on MIT-bib are:

$$\begin{array}{l} \forall x\,(MIT(r,\,x) \to \forall y\,(book \cdot author(x,\,y) \to person(x,\,y))) \\ \forall x\,(MIT(r,\,x) \to \forall y\,(person \cdot wrote(x,\,y) \to book(x,\,y))) \\ \\ \forall x\,(MIT \cdot book(r,\,x) \to \forall y\,(author(x,\,y) \to wrote(y,x))) \\ \forall x\,(MIT \cdot person(r,\,x) \to \forall y\,(wrote(x,\,y) \to author(y,\,x))) \end{array}$$

Constraints on local databases are called *local database constraints*. Again, these are $P_c$ constraints but are not examples of word constraints. As demonstrated in [11], $P_c$ constraints are capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization. They are useful for, among others, specifying and querying XML documents.

In [9, 11], it was shown that in the context of semistructured data, the implication and finite implication problems for $P_c$ are undecidable. However, several decidable fragments of $P_c$ were identified [10, 11]. Each of these fragments properly contains the class of word constraints, and is capable of expressing extent, inverse and local database constraints.

**Main results.** This paper investigates the interaction between $P_c$ constraints and three type systems: $\mathcal{M}^+$, $\mathcal{M}^+_f$ and $\mathcal{M}$. Similar to the object-oriented models studied in [2, 3, 12, 16], $\mathcal{M}^+$ supports classes, records, sets and recursive structures. The model $\mathcal{M}^+_f$ is the same as $\mathcal{M}^+$ except that when infinite instances are considered, it requires that sets are finite. The model $\mathcal{M}$ is a restriction of $\mathcal{M}^+$: it does not allow sets. In the contexts of a semistructured data model and these object-oriented models, we investigate the implication and finite implication problems for $P_c$ and for two fragments of $P_c$, called $P_w(\alpha)$ and the class of local extent constraints. The complexity results on these problems are summarized in Table 1. These results demonstrate that path constraint implication has wildly different complexities in the context of untyped data as opposed to typed data. In particular, the following are established:

- On the one hand, the implication and finite implication problems for $P_c$ are undecidable in the context of untyped data. However, when the type system $\mathcal{M}$ is added, these problems become not only decidable in cubic-time, but also finitely axiomatizable.

| | *The implication and finite implication for $P_w(\alpha)$* | *The implication and finite implication for local extent constraints* | *The implication and finite implication for $P_c$* |
|---|---|---|---|
| Semistructured data model | Undecidable | Decidable (PTIME) | Undecidable |
| Object-oriented model $\mathcal{M}$ | Decidable (cubic-time) | Decidable (cubic-time) | Decidable (cubic-time) |
| Object-oriented model $\mathcal{M}^+$ | Undecidable | Undecidable | Undecidable |
| Object-oriented model $\mathcal{M}_f^+$ | Undecidable | Undecidable | Undecidable |

Table 1: The main results of the paper

- On the other hand, the implication and finite implication problems for local extent constraints are decidable in PTIME in the untyped context. However, when the type system $\mathcal{M}^+$ or $\mathcal{M}_f^+$ is imposed, these problems become undecidable.

It should be mentioned that the undecidability of the implication and finite implication problems for $P_c$ was first shown in [9]. These results are strengthened in this paper by establishing the undecidability of the implication and finite implication problems for $P_w(\alpha)$, which is a "small" fragment of $P_c$.

**Organization.** The rest of the paper is organized as follows. Section 2 reviews the formal definitions of $P_c$ constraints and word constraints, and describes some implication problems associated with $P_c$ constraints, namely, the implication and finite implication problems for $P_c$, $P_w(\alpha)$, and local extent constraints. Section 3 presents a semistructured data model and three object-oriented models: $\mathcal{M}^+$, $\mathcal{M}_f^+$ and $\mathcal{M}$. Type constraints of these object-oriented models are described. Section 4 shows that undecidability results established for semistructured data may collapse when a type system is imposed, by investigating the implication and finite implication problems for $P_w(\alpha)$. More specifically, it first strengthens the undecidability result reported in [9, 11] by establishing the undecidability of the implication and finite implication problems for $P_w(\alpha)$ on untyped data. It then establishes the decidability of the implication and finite implication problems for $P_c$ in the context of $\mathcal{M}$. In addition, it also shows that the implication and finite implication problems for $P_w(\alpha)$ remain undecidable in the contexts of $\mathcal{M}^+$ and $\mathcal{M}_f^+$. Section 5 demonstrates that adding a type system does not necessarily "help" in constraint implication problems, by investigating the implication and finite implication problems for local extent constraints. More specifically, it shows that on untyped data, these problems are decidable in PTIME. However, when the type system $\mathcal{M}^+$ or $\mathcal{M}_f^+$ is imposed, these problems become undecidable. Finally, Section 6 summarizes the main results of the paper and identifies directions for further work.

## 2 Path constraints

We first review the definition of the path constraint language $P_c$ introduced in [11]. We then describe two fragments of $P_c$ and their associated implication and finite implication problems. In Sections 4 and 5, we shall show that these problems have wildly different complexities in the context of untyped data as opposed to typed data.

### 2.1 Path constraint language $P_c$

The vocabulary of the constraint language is specified by a relational signature

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

where $r$ is a constant and $E$ is a finite set of binary relation symbols. A $\sigma$-structure $(|G|,\ r^G,\ E^G)$ can be depicted as an edge-labeled, rooted, directed graph, in which $|G|$ is the set of vertices, $r^G$ the root, and $E^G$ the set of labeled edges. For example, the graph in Figure 1 can be viewed as such a structure (referred to as $G_0$).

A path is a finite sequence of labels of $E$. Following [11], we define a *path* to be a formula $\alpha(x,y)$ which has one of the following forms:

- $x = y$, denoted $\epsilon(x,y)$ and called the *empty path*;
- $\exists z(K(x,z) \wedge \beta(z,y))$, where $K \in E$ and $\beta(z,y)$ is a path.

Here the free variables $x$ and $y$ denote the tail and head nodes of the path, respectively. Intuitively, if $x$ and $y$ are vertices in a $\sigma$-structure $G$, $\alpha(x,y)$ is true in $G$ just when there is a path from $x$ to $y$ whose sequence of edge labels is $\alpha$. We write $\alpha(x,y)$ as $\alpha$ when the parameters $x$ and $y$ are clear from the context.

The *concatenation* of paths $\alpha(x,z)$ and $\beta(z,y)$, denoted $\alpha(x,z) \cdot \beta(z,y)$ or simply $\alpha \cdot \beta$, is the path

- $\beta(x,y)$, if $\alpha = \epsilon$;
- $\exists\, u\, (K(x,u) \wedge (\alpha'(u,z) \cdot \beta(z,y)))$, if $\alpha(x,z)$ is of the form $\exists\, u\, (K(x,u) \wedge \alpha'(u,z))$.

A path $\alpha$ is said to be a *prefix* of $\beta$, denoted $\alpha \preceq_p \beta$, if there exists $\gamma$, such that $\beta = \alpha \cdot \gamma$. Similarly, $\alpha$ is said to be a *suffix* of $\beta$, denoted $\alpha \preceq_s \beta$, if there exists $\gamma$, such that $\beta = \gamma \cdot \alpha$.

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

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

Referring to $G_0$ given in Figure 1, $person \cdot wrote \cdot ref(r,\ x)$ is a path, and there is node $x$ such that this path formula is true in $G_0$. The prefixes of the path are $\epsilon$, $person$, $person \cdot wrote$ and itself. The length of the path is 3.

Formally, $P_c$ constraints can be defined as follows.

**Definition 2.1 [11]:** A *path constraint* $\varphi$ is an expression of either the *forward* form

$$\forall x\,(\alpha(r,x) \to \forall y\,(\beta(x,y) \to \gamma(x,y))),$$

or the *backward* form

$$\forall x\,(\alpha(r,x) \to \forall y\,(\beta(x,y) \to \gamma(y,x))).$$

Here $\alpha, \beta, \gamma$ are paths. The path $\alpha$ is called the *prefix* of $\varphi$, denoted by $pf(\varphi)$. The paths $\beta$ and $\gamma$ are denoted by $lt(\varphi)$ and $rt(\varphi)$, respectively.

The set of all path constraints is denoted by $P_c$. ■

For example, all the integrity constraints encountered in Section 1 are in $P_c$. These include extent, inverse and local database constraints.

A proper subclass of $P_c$ was introduced and investigated in [4]:

**Definition 2.2 [4]:** A *word constraint* is an expression of the form

$$\forall x\,(\beta(r,x) \to \gamma(r,x)),$$

where $\beta$ and $\gamma$ are paths.

The set of all word constraints is denoted by $P_w$. ■

In other words, a word constraint is a forward constraint of $P_c$ with its prefix being the empty path $\epsilon$. For example, the extent constraints given in Section 1 are word constraints, while the inverse and local database constraints are not.

## 2.2 Implication problems

To take advantage of path constraints, it is important to be able to reason about them. This gives rise to the question of logical implication of path constraints. Below we describe implication and finite implication of $P_c$ constraints. These notions will be refined in different database contexts in Section 3.

We assume the standard notions of model and implication from first-order logic [14]. Let $G$ be a structure and $\varphi$ be a $P_c$ constraint. We use $G \models \varphi$ to denote that $G$ *satisfies* $\varphi$ (i.e., $G$ *is a model of* $\varphi$). Let $\Sigma$ be a finite set of $P_c$ constraints. We use $G \models \Sigma$ to denote that $G$ *satisfies* $\Sigma$ (i.e., $G$ *is a model of* $\Sigma$). That is, for every $\phi \in \Sigma$, $G \models \phi$.

The *implication problem for* $P_c$ is the problem to determine, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_c$, whether every model of $\Sigma$ also satisfies $\varphi$. Similarly, the *finite implication problem for* $P_c$ is the problem to determine whether every finite model of $\Sigma$ also satisfies $\varphi$.

For example, let $\Sigma$ be the set consisting of all the $P_c$ constraints given in Section 1, and $\varphi$ be the constraint

$$\forall x\,(MIT(r,\, x) \to \forall y\,(book \cdot ref(x,\, y) \to book(x,\, y))).$$

The question whether every (finite) model of $\Sigma$ also satisfies $\varphi$ is an instance of the (finite) implication problem for $P_c$. As shown in [11], $P_c$ constraint implication is useful for, among others, query optimization.

In Section 4, we shall show that the implication and finite implication problems for $P_c$ are undecidable in the context of untyped data. In contrast, these problems are not only decidable in cubic-time but also finitely axiomatizable in the context of an object-oriented model.

In fact, in Section 4, we show that these undecidability results on untyped data also hold for a fragment of $P_c$. This "small" fragment of $P_c$ is an even milder generalization of $P_w$, the class of word constraints introduced in [4] and described above.

We present the fragment as follows. Let $\alpha$ be a path. For each $\psi \in P_w$, where $\psi$ is $\forall x\,(\beta(r, x) \rightarrow \gamma(r, x))$, let

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

The fragment is defined by

$$P_w(\alpha) = P_w \;\cup\; \{\delta(\psi, \alpha) \mid \psi \in P_w\}.$$

The *(finite) implication problem for* $P_w(\alpha)$ is the problem to determine, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\alpha)$, whether every (finite) model of $\Sigma$ also satisfies $\varphi$.

The (finite) implication problem for $P_w(\alpha)$ studies (finite) implication of extent and local extent constraints. For example, consider Penn-bib described in Section 1. This database has local databases MIT-bib, Warner-bib, etc. Let $\Sigma$ be the set consisting of the extent constraints on Penn-bib and the two local extent constraints on MIT-bib given in Section 1. Let $\varphi$ be the constraint given above, which is also a local extent constraint on MIT-bib. Then $\Sigma \cup \{\varphi\}$ is a finite subset of $P_w(MIT)$. The question whether every (finite) model of $\Sigma$ also satisfies $\varphi$ is an instance of the (finite) implication problem for $P_w(MIT)$.

We shall show that in the untyped context, these problems are undecidable. These undecidability results are rather surprising since $P_w(\alpha)$ generalizes $P_w$ in such a mild way. As shown by [4], the implication and finite implication problems for $P_w$ are decidable in PTIME.

In light of these undecidability results, we consider a special case of the (finite) implication problem for $P_c$ constraints, namely, the (finite) implication problem for local extent constraints. To illustrate this, consider the two local extent constraints on MIT-bib given in Section 1. Suppose we want to know whether every model of these constraints also satisfies the constraint $\varphi$ given above. In addition, we consider this implication in the presence of constraints on other local databases, such as the following on Warner-bib:

$$\forall x\,(Warner(r, x) \rightarrow \forall y\,(book \cdot author(x, y) \rightarrow person(x, y)))$$
$$\forall x\,(Warner(r, x) \rightarrow \forall y\,(person \cdot wrote(x, y) \rightarrow book(x, y)))$$

$$\forall x\,(Warner \cdot book(r,\,x) \to \forall y\,(author(x,\,y) \to wrote(y,x)))$$
$$\forall x\,(Warner \cdot person(r,\,x) \to \forall y\,(wrote(x,\,y) \to author(y,\,x)))$$

More precisely, let $\Sigma$ be the set consisting of the two local extent constraints on MIT-bib and the constraints on Warner-bib given above. The question whether every (finite) model of $\Sigma$ also satisfies $\varphi$ is an instance of the (finite) implication problem for local extent constraints.

In general, when represented in a global environment, constraints on a local database are augmented with a common prefix. For example, the constraints on MIT-bib are represented with common prefix $MIT$ in Penn-bib. Because of this, we use the following notion to describe local extent constraints.

**Definition 2.3:** Let $\alpha$ be a path and $K$ a binary relation symbol. A constraint $\varphi$ of $P_c$ is said to be *bounded by* $\alpha$ *and* $K$ if it is of the form

$$\forall x\,(\alpha \cdot K(r,x) \to \forall y\,(\beta(x,y) \to \gamma(x,y))),$$

where $\beta \neq \epsilon$ and $K \not\preceq_p \beta$ (i.e., $K$ is not a prefix of $\beta$).

A *subset* $\Sigma$ *of* $P_c$ *with prefix bounded by* $\alpha$ *and* $K$ is a finite subset of $P_c$ such that for each $\varphi \in \Sigma$, either $\varphi$ is bounded by $\alpha$ and $K$, or for some path $\alpha'$, $pf(\varphi) = \alpha \cdot \alpha'$ and $K \not\preceq_p \alpha'$. In addition, if $\alpha' = \epsilon$, then $\varphi$ is of the form

$$\forall x\,(\alpha(r,x) \to \forall y\,(\epsilon(x,y) \to K(x,y))).$$

Here $pf(\varphi)$ denotes the prefix of $\varphi$, as described in Definition 2.1. ■

Intuitively, let $DB$ be a database and $DB_l$ be a local database connected to $DB$ by path $\alpha \cdot K$. Constraints bounded by $\alpha$ and $K$ can be viewed as local extent constraints on $DB_l$. A subset of $P_c$ with prefix bounded by $\alpha$ and $K$ consists of such local extent constraints and constraints on other local databases connected to $DB$ by some path $\alpha{\cdot}\alpha'$, where $K \not\preceq_p \alpha'$. Such a set can be partitioned into $\Sigma_1$ and $\Sigma_2$, where $\Sigma_1$ consists of local extent constraints on $DB_l$, and $\Sigma_2$ contains constraints on other local databases. We are interested in (finite) implication of local extent constraints on $DB_l$ (i.e., constraints in $\Sigma_1$) in the presence of constraints on other local databases (i.e., constraints in $\Sigma_2$). This is formalized in the following definition.

**Definition 2.4:** The *(finite) implication problem for local extent constraints* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_c$ with prefix bounded by $\alpha$ and $K$, where $\varphi$ is a constraint bounded by $\alpha$ and $K$, whether every (finite) model of $\Sigma$ also satisfies $\varphi$. ■

In Section 5, we shall show that in the untyped context, constraints on other local databases (e.g., constraints in $\Sigma_2$) do not interact with implication and finite implication of local extent constraints on $DB_l$ (e.g., constraints in $\Sigma_1$). However, this may no longer be true in the typed context. As a result, the implication and finite implication problems for local extent constraints are decidable in PTIME in the context of semistructured data. In contrast, these problems become undecidable in the context of some object-oriented models.

## 3 Semistructured data vs structured data

We next consider semistructured data versus structured data. More specifically, we investigate a semistructured data model and three object-oriented models. For each of these models, we present an abstraction of databases in terms of first-order logic. In Sections 4 and 5, we use these abstractions to study path constraint implication in these models.

### 3.1 Semistructured data model

Semistructured data is characterized as having irregular structure and missing schema. That is, data whose structure is not constrained by a schema. Examples of such data can be found on the World Wide Web, in biological databases and after data integration. In particular, documents of XML [7] are usually viewed as semistructured data [13].

As observed by [1, 8], semistructured data is best modeled as a rooted, edge-labeled, directed graph, unconstrained by any type system or schema. Along the same lines, we use an abstraction of semistructured databases as (finite) $\sigma$-structures. Here $\sigma$ is a signature of the form $(r, E)$ as described in Section 2, in which $r$ denotes the root and $E$ denotes the edge labels.

Below we refine the notion of path constraint implication in the context of semistructured data. We use $\Sigma \models \varphi$ to denote that $\Sigma$ *implies* $\varphi$. That is, for every $\sigma$-structure $G$, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_f \varphi$ to denote that $\Sigma$ *finitely implies* $\varphi$. That is, for every finite $\sigma$-structure $G$, if $G \models \Sigma$, then $G \models \varphi$.

*In the context of semistructured data, the (finite) implication problem for $P_c$* is the problem to determine, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_c$, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$).

Similarly, the implication and finite implication problems for $P_w(\alpha)$ and local extent constraints can be formalized in the context of semistructured data.

### 3.2 Object-oriented model $\mathcal{M}^+$

Next, we consider structured data, by which we mean data constrained by a schema. Such data can be found for instance in object-oriented databases. In addition, as mentioned in Section 1, there are applications in which data usually considered to be semistructured, such as XML data, is further constrained by a schema.

We first study databases in a generic object-oriented model, $\mathcal{M}^+$, Similar to the models studied in [2, 3, 12, 16], $\mathcal{M}^+$ supports classes, records, sets and recursive structures. We characterize schemas in $\mathcal{M}^+$ in terms of type constraints. In Sections 4 and 5, we investigate interaction between these type constraints and path constraints.

#### Schemas and instances

We describe schemas and instances of $\mathcal{M}^+$ as follows. Assume a fixed countable set of labels, $\mathcal{L}$, and a fixed finite set of *atomic types* (e.g., *int* and *string*), $\mathcal{B}$.

**Definition 3.1:** Let $\mathcal{C}$ be a finite set of *classes*. The set of *types over* $\mathcal{C}$, $Types^{\mathcal{C}}$, is defined by:

$$\tau ::= b \ | \ C \ | \ \{\tau\} \ | \ [l_1 : \tau_1, \ldots, l_n : \tau_n]$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. The notations $\{\tau\}$ and $[l_1 : \tau_1, \ldots, l_n : \tau_n]$ represent *set type* and *record type*, respectively. ■

**Definition 3.2:** A *schema* $\Delta$ in $\mathcal{M}^+$ is a triple $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ is a finite set of classes,
- $\nu$ is a mapping: $\mathcal{C} \rightarrow Types^{\mathcal{C}}$ such that for each $C \in \mathcal{C}$, $\nu(C) \notin \mathcal{B} \cup \mathcal{C}$, and
- $DBtype \in Types^{\mathcal{C}} \setminus (\mathcal{B} \cup \mathcal{C})$. ■

Here we assume that every database of a schema has a unique (persistent) entry point, and *DBtype* in the schema specifies the type of the entry point.

**Example 3.1:** The XML document given in Figure 1 can be specified by a $\mathcal{M}^+$ schema $\Delta_x = (\mathcal{C}, \nu, DBtype)$, as follows (optional attributes are specified as sets):

- $\mathcal{C}$ consists of *Book* and *Person*;
- $\nu$ maps *Book* and *Person* to record types:

$$\begin{array}{rcl} Person & \mapsto & [name : string,\ SSN : string,\ age : \{int\},\ wrote : \{Book\}] \\ Book & \mapsto & [title : string,\ ISBN : string,\ year : \{string\},\ ref : \{Book\}, \\ & & \ author : \{Person\}] \end{array}$$

- *DBtype* is $[persons : \{Person\},\ books : \{Book\}]$. ■

**Definition 3.3:** A *database instance* of schema $(\mathcal{C}, \nu, DBtype)$ is a triple $(\pi, \mu, d)$, where

- $\pi$ is an *oid* (object identity) *assignment* that maps each $C \in \mathcal{C}$ to a finite set of oids, $\pi(C)$, such that for all $C, C' \in \mathcal{C}$, $\pi(C) \cap \pi(C') = \emptyset$ if $C \neq C'$;
- for each $C \in \mathcal{C}$, $\mu$ maps each oid in $\pi(C)$ to a value in $[\![\nu(C)]\!]_\pi$, where

$$\begin{array}{rcl} [\![b]\!]_\pi & = & D_b, \\ [\![C]\!]_\pi & = & \pi(C), \\ [\![\{\tau\}]\!]_\pi & = & \{V \mid V \subseteq [\![\tau]\!]_\pi,\ V \text{ is finite}\}, \\ [\![[l_1 : \tau_1, ..., l_n : \tau_n]]\!]_\pi & = & \{[l_1 : v_1, ..., l_n : v_n] \mid v_i \in [\![\tau_i]\!]_\pi,\ i \in [1, n]\}; \end{array}$$

  here $D_b$ denotes the domain of atomic type $b$;
- $d$ is a value in $[\![DBtype]\!]_\pi$, which represents the (persistent) entry point into the database instance.

We denote the set of all database instances of schema $\Delta$ by $\mathcal{I}(\Delta)$. ■

## Type constraints

We next present an abstraction of databases in $\mathcal{M}^+$. Structured data can be viewed as semistructured data further constrained by a schema. Along the same lines of the abstraction of semistructured data given above, we represent a structured database as a first-order logic structure satisfying a certain type constraint. Such a structure can also be depicted as an edge-labeled, rooted, directed graph, which has a certain "shape" specified by the type constraint. This abstraction simplifies the analysis of the interaction between path constraints and the type system.

To do this, we first define the first-order signature determined by a schema.

Given a schema $\Delta = (\mathcal{C}, \nu, \mathit{DBtype})$, we define *the set of binary relation symbols*, $E(\Delta)$, and *the set of unary relation symbols*, $T(\Delta)$, as follows:

- $\mathit{DBtype} \in T(\Delta)$ and $\mathcal{C} \subseteq T(\Delta)$;
- For each $\tau \in T(\Delta)$,
  - if $\tau = \{\tau'\}$ (or for some $C \in \mathcal{C}$, $\nu(C) = \{\tau'\}$), then $\tau'$ is in $T(\Delta)$ and $*$ is in $E(\Delta)$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or for some $C \in \mathcal{C}$, $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, $\tau_i$ is in $T(\Delta)$ and $l_i$ is in $E(\Delta)$.

Note here we use the distinguished binary relation $*$ to denote the set membership relation.

**Definition 3.4:** The *signature determined by schema* $\Delta$ is

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

where $r$ is a constant symbol (denoting the root), $E(\Delta)$ is the finite set of binary relation symbols (denoting the edge labels) and $T(\Delta)$ is the finite set of unary relation symbols (denoting the sorts or types) defined above. ■

As an example, the signature determined by the schema given in Example 3.1 is $(r, E, T)$, where

- $r$ is a constant, which in each instance $(\pi, \mu, d)$ of the schema intends to name $d$;
- $E$ includes *persons*, *books*, *name*, *SSN*, *wrote*, *age*, *title*, *ISBN*, *year*, *ref*, *author* and $*$;
- $T$ includes *Person*, *Book*, *string*, $\{\mathit{int}\}$, $\{\mathit{string}\}$, $\{\mathit{Book}\}$, $\{\mathit{Person}\}$ and *DBtype*.

We represent an instance $I$ of a schema $\Delta$ as a (finite) $\sigma(\Delta)$-structure $G$ satisfying a certain type constraint. More specifically, assume that $\Delta = (\mathcal{C}, \nu, \mathit{DBtype})$, $I = (\pi, \mu, d)$ and $G = (|G|, r^G, E^G, T^G)$. We use $|G|$, $r^G$, $E^G$ and $T^G$ to represent data entities, the entry point $d$, record labels and set membership, and the types of the data entities, respectively. This structure must satisfy the *type constraint imposed by* $\Delta$, $\Phi(\Delta)$, which specifies restrictions on the edges going out of vertices of different types.

Based on the definition of database instances in $\mathcal{M}^+$, we give $\Phi(\Delta)$ as follows.

- Every element of $|G|$ has a unique type in $T(\Delta)$. In particular, $r^G$ has *DBtype*.
- If an element $a$ of $|G|$ has type $\tau$, then $a$ must satisfy the constraint imposed by $\tau$:
  - If $\tau$ is an atomic type $b$, then $a$ has no outgoing edge.
  - If $\tau = \{\tau'\}$, or $\tau$ is a class type $C$ and $\nu(C)$ is $\{\tau'\}$, then all the outgoing edges of $a$ are labeled with $*$ and lead to elements of type $\tau'$.
    In addition, if $\tau = \{\tau'\}$, then for each $b \in |G|$ such that $b$ also has type $\tau$, $a = b$ iff for any $c \in |G|$, $G \models *(a, c) \leftrightarrow *(b, c)$.
  - If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, or $\tau$ is a class type $C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $a$ has exactly $n$ outgoing edges. These edges are labeled with $l_1$, ..., $l_n$, respectively. In addition, for each $i \in [1, n]$, if $G \models l_i(a, o)$ for some $o \in |G|$, then $o$ has type $\tau_i$.
    Moreover, if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then for each $b \in |G|$ having type $\tau$, $a = b$ iff for any $i \in [1, n]$ and $c \in |G|$, $G \models l_i(a, c) \leftrightarrow l_i(b, c)$.

Formally, the type constraint imposed by a schema can be formulated as a sentence in first-order logic [14]. To simplify the description, below we use the counting quantifier $\exists\,!$, whose semantics is described as follows: structure $G$ satisfies $\exists\,!x\,\psi(x)$ if and only if there exists a unique element $a$ of $G$ such that $G \models \psi(a)$ (see, e.g., [5] for detailed discussions of counting quantifiers). It should be noted that $\exists\,!$ is definable in first-order logic.

**Definition 3.5:** Let $\Delta$ be a schema in $\mathcal{M}^+$. For each $\tau$ in $T(\Delta)$, the *constraint determined by* $\tau$ is the sentence $\forall\, x\, \phi_\tau(x)$ defined as follows:

- if $\tau = b$, then $\phi_\tau(x)$ is
$$\tau(x) \to \forall\, y\, (\bigwedge_{l \in E(\Delta)} \neg l(x, y));$$
- if $\tau = \{\tau'\}$ or for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{\tau'\}$, then $\phi_\tau(x)$ is
$$\tau(x) \to \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{*\}} \neg l(x, y)) \;\wedge \forall\, y\, (*(x, y) \to \tau'(y));$$
  in addition, if $\tau = \{\tau'\}$, then $\phi_\tau(x)$ also has the following conjunct:
$$\forall\, y\, (\tau(y) \wedge \forall\, z\, (*(x,\, z) \leftrightarrow *(y,\, z)) \;\to\; x = y);$$
- if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ or for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\phi_\tau(x)$ is
$$\tau(x) \to \forall\, y\, (\bigwedge_{l \in E(\Delta) \setminus \{l_1, \ldots, l_n\}} \neg l(x, y)) \;\wedge\; \bigwedge_{i \in [1, n]} (\exists\,!\, y\, l_i(x, y) \;\wedge \forall\, y\, (l_i(x, y) \to \tau_i(y))).$$

in addition, if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\phi_\tau(x)$ also has the following conjunct:

$$\forall y\,(\tau(y) \wedge \bigwedge_{i \in [1,n]} \forall z\,(l_i(x, z) \leftrightarrow l_i(y, z)) \rightarrow x = y).$$

The *type constraint determined by schema* $\Delta$ is the sentence $\Phi(\Delta)$ defined by:

$$DBtype(r) \wedge \bigwedge_{\tau \in T(\Delta)} \forall x\, \phi_\tau(x) \wedge \forall x\,( \bigvee_{\tau \in T(\Delta)} \tau(x) \wedge \bigwedge_{\tau \in T(\Delta)} (\tau(x) \rightarrow \bigwedge_{\tau' \in T(\Delta) \setminus \{\tau\}} \neg\tau'(x)))$$

■

**Definition 3.6:** An *abstract database of a schema* $\Delta$ is a finite $\sigma(\Delta)$-structure $G$ such that $G \models \Phi(\Delta)$. We denote the set of all abstract databases of $\Delta$ by $\mathcal{U}_f(\Delta)$.

We use $\mathcal{U}(\Delta)$ to denote the set of all $\sigma(\Delta)$-structures satisfying $\Phi(\Delta)$. ■

## Path constraints revisited

Next, we refine the definitions of paths and path constraints in the context of $\mathcal{M}^+$, and justify the abstraction of databases given above by considering path constraint satisfiability.

Why do we need to refine these definitions? Given the signature $(r, E(\Delta), T(\Delta))$ determined by a schema $\Delta$, one could define paths and path constraints using binary predicates in $E(\Delta)$ in the same way as in Section 2. These definitions, however, are somewhat too coarse in the context of the object-oriented model. Because of the type constraint $\Phi(\Delta)$, some paths are not meaningful in structures of $\mathcal{U}(\Delta)$. That is, there exists path $\alpha(x, y)$ defined in this way such that for all $G \in \mathcal{U}(\Delta)$ and all $a, b \in |G|$, $G \not\models \alpha(a, b)$. Such paths are said to be *undefined over* $\Delta$. A path constraint containing undefined paths is satisfied by either all the structures in $\mathcal{U}(\Delta)$ or by none of them. We are not interested in such constraints.

We use $Paths(\Delta)$ to denote the set of *paths over* $\Delta$. That is, $\alpha \in Paths(\Delta)$ iff there is $G \in \mathcal{U}(\Delta)$ such that $G \models \exists x\ \alpha(r, x)$. In addition, we assume that over any schema $\Delta$ in $\mathcal{M}^+$, $P_c$ constraints are defined in terms of paths in $Paths(\Delta)$. Formally, these notions are defined as follows.

**Definition 3.7:** Let a $\mathcal{M}^+$ schema $\Delta$ be $(\mathcal{C}, \nu, DBtype)$. The set of *paths over schema* $\Delta$, $Paths(\Delta)$, and *the type of path* $\alpha$ *in* $Paths(\Delta)$, $type(\alpha)$, are defined inductively as follows:

- the empty path $\epsilon$ is in $Paths(\Delta)$ and $type(\epsilon) = DBtype$;
- for any $\alpha \in Paths(\Delta)$, where $type(\alpha) = \tau$,
  - if $\tau = \{\tau'\}$ or for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{\tau'\}$, then $\alpha \cdot *$ is a path in $Paths(\Delta)$ and $type(\alpha \cdot *) = \tau'$;
  - if $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ or there exists class $C$ in $\mathcal{C}$ such that $\tau = C$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then for each $i \in [1, n]$, $\alpha \cdot l_i$ is in $Paths(\Delta)$ and $type(\alpha \cdot l_i) = \tau_i$.

■

As in Section 2, a path over a schema can be represented by a first-order logic formula $\alpha(x, y)$, where $x$ and $y$ denote the tail and head nodes of the path, respectively. In the same way, the notions of path concatenation, prefix, suffix, and length can be defined.

The set of *paths definable over schema* $\Delta$ is defined by

$$Pts^d(\Delta) = Paths(\Delta) \cup \{\alpha \mid \text{there is } \beta \in Paths(\Delta),\ \alpha \preceq_s \beta\},$$

where $\alpha \preceq_s \beta$ denotes that $\alpha$ is a suffix of $\beta$, as described in Section 2.

Using these notions, we define path constraints in the context of $\mathcal{M}^+$ as follows..

**Definition 3.8:** A *path constraint* $\varphi$ *over schema* $\Delta$ is an expression of either the *forward* form

$$\forall x\, (\alpha(r,\, x) \to \forall y\, (\beta(x,\, y) \to \gamma(x,\, y))),$$

or the *backward* form

$$\forall x\, (\alpha(r,\, x) \to \forall y\, (\beta(x,\, y) \to \gamma(y,\, x))),$$

where $\alpha, \beta, \gamma$ are paths in $Pts^d(\Delta)$. In addition,

- if $\varphi$ is of the forward form, then $\alpha \cdot \beta \in Paths(\Delta)$, $\alpha \cdot \gamma \in Paths(\Delta)$, and moreover, $type(\alpha \cdot \beta) = type(\alpha \cdot \gamma)$;
- if $\varphi$ is of the backward form, then $\alpha \in Paths(\Delta)$, $\alpha \cdot \beta \cdot \gamma \in Paths(\Delta)$, and moreover, $type(\alpha) = type(\alpha \cdot \beta \cdot \gamma)$.

The path $\alpha$ is called the *prefix* of $\varphi$. The paths $\alpha$, $\beta$ and $\gamma$ are denoted by $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$, respectively.

We denote the set of all path constraints over $\Delta$ by $P_c(\Delta)$. ■

**Definition 3.9:** A *word constraint* $\varphi$ *over schema* $\Delta$ is a sentence of the form

$$\forall x\ (\alpha(r, x) \to \beta(r, x)),$$

where $\alpha$ and $\beta$ are in $Paths(\Delta)$, and $type(\alpha) = type(\beta)$. We denote $\alpha$, $\beta$ as $lt(\varphi)$ and $rt(\varphi)$, respectively.

We use $P_w(\Delta)$ to denote the set of all word constraints over $\Delta$. ■

When $\Delta$ is understood from the context, we write $P_c(\Delta)$ and $P_w(\Delta)$ simply as $P_c$ and $P_w$, respectively.

In the context of $\mathcal{M}^+$, we refine the definition of $P_w(\alpha)$ described in Section 2.2 as follows. Let $\Delta$ be a schema in $\mathcal{M}^+$. The fragment $P_w(\alpha)$ *over* $\Delta$ is defined by

$$P_w(\Delta,\, \alpha) = P_w(\Delta) \cup \{\delta(\psi,\, \alpha) \mid \psi \in P_w(\Delta),\ \delta(\psi,\, \alpha) \in P_c(\Delta)\}.$$

Here $\delta$ is the function defined in Section 2.2. Similarly, local extent constraints described in Definition 2.3 can be refined in the context of $\mathcal{M}^+$.

In the typed context, path constraint implication is restricted by a schema. More specifically, let $\Delta$ be a schema in $\mathcal{M}^+$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_c(\Delta)$. We use $\Sigma \models_\Delta \varphi$ to denote that $\Sigma$ *implies* $\varphi$ *over* $\Delta$. That is, for every $G \in \mathcal{U}(\Delta)$, if $G \models \Sigma$ then $G \models \varphi$. Similarly, we use $\Sigma \models_{(f,\Delta)} \varphi$ to denote that $\Sigma$ *finitely implies* $\varphi$ *over* $\Delta$. That is, for every $G \in \mathcal{U}_f(\Delta)$, if $G \models \Sigma$ then $G \models \varphi$.

Let $\Delta$ be a schema in $\mathcal{M}^+$. *The (finite) implication problem for* $P_c$ *over* $\Delta$ is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$, whether $\Sigma \models_\Delta \varphi$ ($\Sigma \models_{(f,\Delta)} \varphi$). *The (finite) implication problem for* $P_c$ *in the context of* $\mathcal{M}^+$ is the problem to determine, given any schema $\Delta$ in $\mathcal{M}^+$, whether the (finite) implication problem for $P_c$ over $\Delta$ is decidable.

Similarly, in the context of $\mathcal{M}^+$, the implication and finite implication problems for $P_w(\alpha)$ and local extent constraints can also be formalized.

**Example 3.2:** The following are $P_c$ constraints over the schema $\Delta_x$ given in Example 3.1:

$$\forall x\,(\mathit{persons} \cdot * \cdot \mathit{wrote} \cdot *(r,\, x) \;\rightarrow\; \mathit{books} \cdot *(r,\, x))$$
$$\forall x\,(\mathit{books} \cdot * \cdot \mathit{author} \cdot *(r,\, x) \;\rightarrow\; \mathit{persons} \cdot *(r,\, x))$$

$$\forall x\,(\mathit{persons} \cdot *(r,\, x) \;\rightarrow\; \forall y\,(\mathit{wrote} \cdot *(x,\, y) \rightarrow \mathit{author} \cdot *(y,\, x)))$$
$$\forall x\,(\mathit{books} \cdot *(r,\, x) \;\rightarrow\; \forall y\,(\mathit{author} \cdot *(x,\, y) \rightarrow \mathit{wrote} \cdot *(y,\, x)))$$

In particular, the first two are word constraints over $\Delta_x$. Note that these constraints are presented here in a slightly different way from Section 1.

In an instance $(\pi,\, \mu,\, d)$ of $\Delta_x$, these constraints are interpreted as:

$$\forall x\,(\exists p\,(p \in d.\mathit{persons} \wedge x \in p.\mathit{wrote}) \;\rightarrow\; x \in d.\mathit{books})$$
$$\forall x\,(\exists b\,(b \in d.\mathit{books} \wedge x \in b.\mathit{author}) \;\rightarrow\; x \in d.\mathit{persons})$$

$$\forall x\,(x \in d.\mathit{persons} \;\rightarrow\; \forall y\,(y \in x.\mathit{wrote} \rightarrow x \in y.\mathit{author}))$$
$$\forall x\,(x \in d.\mathit{books} \;\rightarrow\; \forall y\,(y \in x.\mathit{author} \rightarrow x \in y.\mathit{wrote}))$$

Here $v.l$ stands for the projection of record $v$ at attribute $l$, and $v \in s$ means that $v$ is a element of set $s$. ■

As illustrated by the example above, path constraints over a schema $\Delta$ can be naturally interpreted in database instances of $\Delta$. Likewise, the notion "$I \models \varphi$" can also be defined for an instance $I$ of $\Delta$ and a constraint $\varphi$ of $P_c(\Delta)$.

The lemma below justifies the abstraction of structured databases defined above. It reveals the agreement between databases and their abstraction with respect to path constraints.

**Lemma 3.1:** Let $\Delta$ be a schema in $\mathcal{M}^+$. For each $I \in \mathcal{I}(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$, such that

$$(\dagger) \qquad \text{for any } \varphi \in P_c(\Delta),\ I \models \varphi \text{ iff } G \models \varphi.$$

Similarly, for each $G \in \mathcal{U}_f(\Delta)$, there is $I \in \mathcal{I}(\Delta)$, such that (†) holds. ■

**Proof:** Let $\Delta = (\mathcal{C},\ \nu,\ DBtype)$.

(1) Given $I \in \mathcal{I}(\Delta)$, we construct $G \in \mathcal{U}_f(\Delta)$, such that for each $\varphi \in P_c(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

Let $I = (\pi,\ \mu,\ d)$. Then we define $V$ to be the smallest set satisfying the following:

1. $d \in V$;
2. for every $v \in V$,
   - if $v$ is a set (or $v$ is an object and $\mu(v)$ is a set), then every element of $v$ (or $\mu(v)$) is in $V$;
   - if $v$ is a record (or $v$ is an object and $\mu(v)$ is a record), then every attribute of $v$ (or $\mu(v)$) is in $V$.

For every $v \in V$, let $o(v)$ be a distinct node. Let $G = (|G|,\ r^G,\ E^G,\ T^G)$, where

- $|G| = \{o(v) \mid v \in V\}$;
- $r^G = o(d)$;
- for each $o(v) \in |G|$ and $\tau \in T(\Delta)$, $G \models \tau^G(o(v))$ iff $v$ is of type $\tau$; here $\tau^G$ denotes the unary relation in $G$ named by $\tau$;
- for all $o(v), o(v') \in |G|$,
  - for each $l \in \mathcal{L} \cap E(\Delta)$, $G \models l(o(v),\ o(v'))$ iff $v' = v.l$ (or $v' = \mu(v).l$ if $v$ is an object);
  - $G \models *(o(v),\ o(v'))$ iff $v' \in v$ (or $v' \in \mu(v)$ if $v$ is an object).

Then it is straightforward to verify the following:

- $G \in \mathcal{U}_f(\Delta)$; that is, $G$ is a finite $\sigma(\Delta)$-structure and $G \models \Phi(\Delta)$;
- for each $\varphi \in P_c(\Delta)$, $G \models \varphi$ iff $I \models \varphi$. This can be easily verified by *reductio*.

(2) Given $G = (|G|,\ r^G,\ E^G,\ T^G)$ in $\mathcal{U}_f(\Delta)$, we define $I = (\pi,\ \mu,\ d)$ in $\mathcal{I}(\Delta)$, such that for every $\varphi \in P_c(\Delta)$, $I \models \varphi$ iff $G \models \varphi$.

To simplify the discussion, we assume that for every base type $b$, its domain $D_b$ is infinite. By this assumption, there exists an injective mapping $g_b : b^G \to D_b$, where $b^G$ is the unary relation in $G$ denoting the sort $b$.

For every $C \in \mathcal{C}$, let $\pi(C) = C^G$, where $C^G$ is the unary relation in $G$ denoting the class type $C$. We then define a mapping $f : |G| \to \bigcup_{\tau \in T(\Delta)} [\![\tau]\!]_\pi$ as follows: For each $o \in |G|$,

- if $o \in C^G$ for some $C \in \mathcal{C}$, then let $f(o) = o$;

- if $o \in b^G$ for some base type $b$, then let $f(o) = g(o)$;
- if $o \in \tau^G$ and $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then let $f(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)]$, where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$;
- if $o \in \tau^G$ and $\tau = \{\tau'\}$, then let $f(o) = \{f(o') \mid o' \in |G|,\ G \models *(o, o')\}$.

Note that $f$ is well-defined and is an injection, since $G$ is finite and $G \models \Phi(\Delta)$. Now let

- $d = f(r^G)$;
- for each $C \in \mathcal{C}$ and each $o \in \pi(C)$,
  - if $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then $\mu(o) = [l_1 : f(o_1), \ldots, l_n : f(o_n)]$, where for $i \in [1, n]$, $o_i \in |G|$ and $G \models l_i(o, o_i)$.
  - if $\nu(C) = \{\tau\}$, then $\mu(o) = \{f(o') \mid o' \in |G|,\ G \models *(o, o')\}$.

Again, this is well-defined. Moreover, it is easy to verify that $I \in \mathcal{I}(\Delta)$, and $G \models \varphi$ iff $I \models \varphi$. ■

From Lemma 3.1 follows immediately the corollary below.

**Corollary 3.2:** Let $\Delta$ be any schema in $\mathcal{M}^+$ and $\Sigma \cup \{\varphi\}$ be any finite subset of $P_c(\Delta)$. Then there is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$ if and only if there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$. ■

## 3.3 Object-oriented model $\mathcal{M}^+_f$

To further explore the impact of type systems on path constraint implication, we consider another object-oriented model, $\mathcal{M}^+_f$. The model $\mathcal{M}^+_f$ is a mild variation of $\mathcal{M}^+$. The difference between these two is that $\mathcal{M}^+$ supports set construct, whereas $\mathcal{M}^+_f$ supports finite set construct. As a result, when infinite instances are considered, $\mathcal{M}^+$ allows infinite sets, while sets in $\mathcal{M}^+_f$ must be finite.

Syntactically, types, schemas and instances in $\mathcal{M}^+_f$ are defined in the same way as in $\mathcal{M}^+$. Given a schema $\Delta$ in $\mathcal{M}^+_f$, the notions of $E(\Delta)$, $T(\Delta)$, $\sigma(\Delta)$, and type constraint $\Phi(\Delta)$ can be defined as above. Along the same lines, the notions of abstract databases of $\Delta$ and $\mathcal{U}_f(\Delta)$ can also be defined. However, the definition of $\mathcal{U}(\Delta)$ is different. Here $\mathcal{U}(\Delta)$ denotes the set of all the $\sigma(\Delta)$-structures that satisfy $\Phi(\Delta)$ and respect *the finite set rule*. That is, for each $G \in \mathcal{U}(\Delta)$ and $\tau \in T(\Delta)$, if $\tau$ is a set type, or $\tau$ is a class and $\nu(\tau)$ is a set type, then for each $o \in \tau^G$, there are only finitely many $o'$ in $G$ such that $G \models *(o, o')$. As a result, each node in $G$ has finitely many outgoing edges. In contrast, in $\mathcal{M}^+$, the structures in $\mathcal{U}(\Delta)$ are not required to respect the finite set rule. That is, nodes representing sets in such structures are allowed to have infinitely many outgoing edges.

As in $\mathcal{M}^+$, given a schema $\Delta$ in $\mathcal{M}_f^+$, the notions of $Paths(\Delta)$, $P_c(\Delta)$, $P_w(\Delta)$, $P_w(\Delta, \alpha)$, $\models_\Delta$, $\models_{(f, \Delta)}$, and the implication and finite implication problems for $P_c$, $P_w(\alpha)$, and local extent constraints over $\Delta$ can be defined. Similarly, the implication and finite implication problems for $P_c$, $P_w(\alpha)$, and local extent constraints in the context of $\mathcal{M}_f^+$ can also be defined. In addition, it can be shown that Lemma 3.1 also holds in $\mathcal{M}_f^+$.

The following should be noted.

- For any schema $\Delta$ in $\mathcal{M}^+$, $\mathcal{U}(\Delta)$ is definable in first-order logic. As a result, if the implication problem and the finite implication problem for $P_c$ coincide in $\mathcal{M}^+$, then both problems are decidable. However, in Section 4, we shall show that in $\mathcal{M}^+$, these problems are different.
- In contrast, for a schema $\Delta$ in $\mathcal{M}_f^+$, if $T(\Delta)$ contains set types, then $\mathcal{U}(\Delta)$ may not be definable in first-order logic. As a result, the equivalence of the implication problem and the finite implication problem for $P_c$ in $\mathcal{M}_f^+$ does not necessarily lead to the decidability of these problems. In Section 4, we shall show that over some schemas in $\mathcal{M}_f^+$, it is indeed the case that the implication problem and the finite implication problem for $P_c$ are equivalent, but these problems are undecidable.

## 3.4 Object-oriented model $\mathcal{M}$

We also consider a restriction of $\mathcal{M}^+$, $\mathcal{M}$. The model $\mathcal{M}$ supports classes, records and recursive structures. However, it does not allow sets. In addition, a record in $\mathcal{M}$ consists of values of atomic types and oids only. More specifically, let $\mathcal{C}$ be some finite set of classes. The set of *types over $\mathcal{C}$ in $\mathcal{M}$* is defined by:

$$\begin{array}{rcl} t & ::= & b \mid C \\ \tau & ::= & t \mid [l_1 : t_1, \ldots, l_n : t_n] \end{array}$$

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$.

The notions of schemas and instances in $\mathcal{M}$ can be defined in the same way as in $\mathcal{M}^+$. Databases of $\mathcal{M}$ are comparable to feature structures [20], which have proven useful for representing linguistic data.

Given a schema $\Delta$ in $\mathcal{M}$, the notions of $E(\Delta)$, $T(\Delta)$, $\sigma(\Delta)$, and type constraint $\Phi(\Delta)$ are defined in the same way as in $\mathcal{M}^+$, except that set types are not considered here. Similarly, the notions of $\mathcal{U}_f(\Delta)$, $\mathcal{U}(\Delta)$, $Paths(\Delta)$ and $P_c(\Delta)$ can also be defined. Using $\mathcal{U}_f(\Delta)$ and $\mathcal{U}(\Delta)$, we can define the implication and finite implication problems for $P_c$ in the context of $\mathcal{M}$ in the same way as in $\mathcal{M}^+$. In addition, Lemma 3.1 also holds in the context of $\mathcal{M}$.

It should be noted that over any schema $\Delta$ in $\mathcal{M}$, $\mathcal{U}(\Delta)$ is definable in first-order logic.

## 4 Implication problems for $P_w(\alpha)$

This section shows that an undecidability result established for semistructured data collapses when a type of $\mathcal{M}$ is imposed on the data, by investigating the implication and finite implication problems for $P_w(\alpha)$. More specifically, we prove the following:

**Theorem 4.1:** In the context of semistructured data, the implication and finite implication problems for $P_w(\alpha)$ are undecidable. ■

**Theorem 4.2:** In the context of the object-oriented data model $\mathcal{M}$, the implication and finite implication problems for $P_c$ are decidable in cubic-time and are finitely axiomatizable. ■

Recall that $P_w(\alpha)$ is a "small" fragment of $P_c$. Therefore, the corollary below follows immediately from Theorem 4.2.

**Corollary 4.3:** In the context of $\mathcal{M}$, the implication and finite implication problems for $P_w(\alpha)$ are decidable in cubic-time and are finitely axiomatizable. ■

Likewise, Theorem 4.1 strengthens the following undecidability results reported in [9, 11].

**Theorem 4.4 [9, 11]:** In the context of semistructured databases, both the implication and the finite implication problems for $P_c$ are undecidable. ■

In this section, we also investigate the implication and finite implication problems for $P_w(\alpha)$ in the contexts of $\mathcal{M}^+$ and $\mathcal{M}^+_f$.

**Theorem 4.5:** In the context of $\mathcal{M}^+$, the implication and finite implication problems for $P_w(\alpha)$ are undecidable. ■

**Theorem 4.6:** In the context of $\mathcal{M}^+_f$, the implication and finite implication problems for $P_w(\alpha)$ are undecidable. ■

As immediate corollaries of Theorems 4.5 and 4.6, we have the following:

**Corollary 4.7:** In the context of $\mathcal{M}^+$, the implication and finite implication problems for $P_c$ are undecidable. ■

**Corollary 4.8:** In the context of $\mathcal{M}^+_f$, the implication and finite implication problems for $P_c$ are undecidable. ■

### 4.1 Undecidability on untyped data

We first prove Theorem 4.1 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.

### 4.1.1 The word problem for (finite) monoids

Recall the following notions from [2].

**Definition 4.1:** 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. ■

**Definition 4.2:** Let $\Gamma$ be a finite alphabet. The *free monoid generated by* $\Gamma$ is $(\Gamma^*, \cdot, \epsilon)$, where

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

■

**Definition 4.3:** Let $\Gamma$ be a finite alphabet. An *equation* (over $\Gamma$) is a pair $(\alpha, \beta)$ of strings in $\Gamma^*$.

Let a finite set of equations

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

and a *test equation* $\theta$ be $(\alpha, \beta)$, where $\alpha, \beta \in \Gamma^*$. Then $\Theta \models \theta$ ($\Theta \models_f \theta$) if for every (finite) monoid $(M, \circ, 1)$ and every homomorphism $h : \Gamma^* \to M$, 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 $\Theta$ and $\theta$, whether $\Theta \models \theta$ ($\Theta \models_f \theta$). ■

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

**Theorem 4.9:** Both the word problem for monoids and the word problem for finite monoids are undecidable. ■

### 4.1.2 Reduction from the word problem for (finite) monoids

Next, we prove Theorem 4.1. In fact, we show that the undecidability results also hold when $\alpha$ is a binary relation symbol $K$, i.e., $|\alpha| = 1$. More specifically, we consider the implication and finite implication problems for $P_w(K)$, where $K$ is a binary relation symbol $K$.

**Theorem 4.10:** In the context of semistructured data, the implication and finite implication problems for $P_w(K)$ are undecidable. ■

We prove Theorem 4.10 by first presenting an encoding of the word problem for (finite) monoids in terms of the (finite) implication problem for $P_w(K)$, and then showing that the encoding is indeed a reduction.

Let $\Gamma_0$ be a finite alphabet and $\Theta_0$ be a finite set of equations (over $\Gamma_0$). Without loss of generality, assume

$$\begin{aligned} \Gamma_0 &= \{l_j \mid j \in [1,m],\ l_i \neq l_j \text{ if } i \neq j\}, \\ \Theta_0 &= \{(\alpha_i, \beta_i) \mid \alpha_i, \beta_i \in \Gamma_0^*,\ i \in [1,n]\}. \end{aligned}$$

Then we define a first-order logic signature

$$\sigma_0 = (r,\ \Gamma_0 \cup \{K\}),$$

where $K \notin \Gamma_0$, $r$ is a constant symbol, and $\Gamma_0 \cup \{K\}$ is a set of binary relation symbols. Note here that each symbol in $\Gamma_0$ is a binary relation symbol in $\sigma_0$. Therefore, every $\alpha \in \Gamma_0^*$ can be represented as a path formula, also denoted $\alpha$. In addition, we use $\cdot$ to denote the concatenation operator for both paths and strings.

Next, we encode $\Theta_0$ in terms of $\Sigma_1 \subseteq P_w$ and $\Sigma_2 \subseteq P_w(K)$.

1. $\Sigma_1$ consists of the following constraints:

$$\begin{aligned} \forall x\, (\epsilon(r,x) &\rightarrow K(r,x)) \\ \forall x\, (K \cdot l_j(r,x) &\rightarrow K(r,x)) \end{aligned}$$

   for every $j \in [1,m]$.

2. $\Sigma_2$ consists of the following constraints:

$$\begin{aligned} \forall x\, (K(r,x) &\rightarrow \forall y\, (\alpha_i(x,y) \rightarrow \beta_i(x,y))) \\ \forall x\, (K(r,x) &\rightarrow \forall y\, (\beta_i(x,y) \rightarrow \alpha_i(x,y))) \end{aligned}$$

   for every $(\alpha_i,\ \beta_i) \in \Theta_0$.

Let $(\alpha,\ \beta)$ be a test equation, where $\alpha$ and $\beta$ are arbitrary strings in $\Gamma_0^*$. We encode such a pair of strings as a pair of constraints in $P_w$:

$$\begin{aligned} \varphi_{(\alpha,\beta)} &= \forall x\, (\alpha(r,x) \rightarrow \beta(r,x)) \\ \varphi_{(\beta,\alpha)} &= \forall x\, (\beta(r,x) \rightarrow \alpha(r,x)) \end{aligned}$$

We reduce the word problem for monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models \varphi_{(\alpha,\beta)} \wedge \varphi_{(\beta,\alpha)},$$

and analogously, reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models_f \varphi_{(\alpha,\beta)} \wedge \varphi_{(\beta,\alpha)}.$$

Obviously, $\Sigma_1 \cup \Sigma_2 \cup \{\varphi_{(\alpha,\beta)}, \varphi_{(\beta,\alpha)}\}$ is a subset of $P_w(K)$.

Before we show that this encoding is indeed a reduction, we first identify a basic property of $\Sigma_1$.

**Lemma 4.11:** For every $\sigma_0$-structure $G$, if $G \models \Sigma_1$, then for every $\alpha \in \Gamma_0^*$ and $o \in |G|$ such that $G \models \alpha(r^G,\, o)$, we have $G \models K(r^G,\, o)$.

In addition, for all $o, o' \in |G|$ such that $G \models K(r^G,\, o') \wedge \alpha(o',\, o)$, we have $G \models K(r^G,\, o)$. ■

**Proof:** By a straightforward induction on $|\alpha|$. ■

Finally, we show that the encoding given above is indeed a reduction from the word problem for (finite) monoids. It suffices to show the following lemma.

**Lemma 4.12:** In the context of semistructured data, for all $\alpha$ and $\beta$ in $\Gamma_0^*$,

$$\Theta_0 \models (\alpha,\, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models \forall x\, (\alpha(r,x) \to \beta(r,x)) \ \wedge\ \forall x\, (\beta(r,x) \to \alpha(r,x)), \qquad \text{(a)}$$

$$\Theta_0 \models_f (\alpha,\, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_f \forall x\, (\alpha(r,x) \to \beta(r,x)) \ \wedge\ \forall x\, (\beta(r,x) \to \alpha(r,x)). \qquad \text{(b)}$$

■

**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 $\Sigma_1 \cup \Sigma_2 \not\models_f \forall x\, (\alpha(r,x) \to \beta(r,x))$. That is, we show that there exists a finite $\sigma_0$-structure $G$, such that $G \models \Sigma_1 \cup \Sigma_2$, but $G \not\models \forall x\, (\alpha(r,x) \to \beta(r,x))$.

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 $h : \Gamma_0^* \to M$ such that for any $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Based on $M$ and $h$, we define an equivalence relation $\approx$ on $\Gamma_0^*$ as follows:

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

For every $\rho \in \Gamma_0^*$, let $\widehat{\rho}$ be the equivalence class of $\rho$ with respect to $\approx$. Let

$$C_{\Theta_0} = \{\widehat{\rho} \mid \rho \in \Gamma_0^*\}.$$

Using these notations, we construct a 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

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

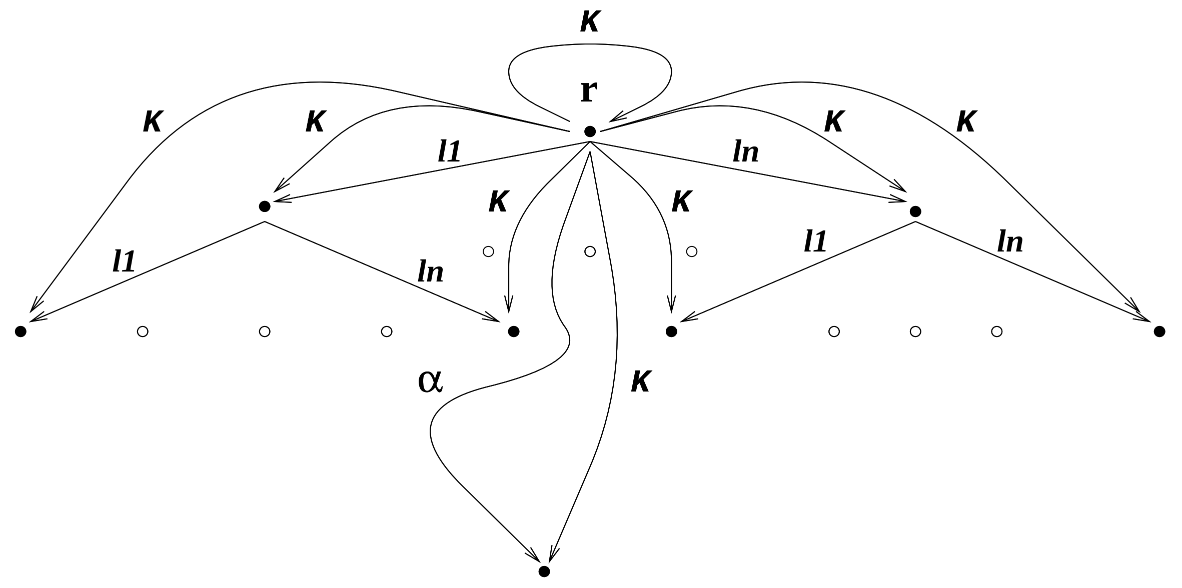

Figure 2: The structure $G$ in the proof of Lemma 4.12

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

(3) The binary relations are populated as follows: For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models K(o(\widehat{\epsilon}),\, o(\widehat{\rho}))$. In addition, for each $j \in [1, m]$, let $G \models l_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot l_j}))$.

The structure $G$ is shown in Figure 2.

By the construction of $G$, it is easy to see that for every $\rho \in \Gamma_0^*$ and $j \in [1, m]$, $o(\widehat{\rho \cdot l_j})$ is the unique node such that $G \models l_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot l_j}))$. This is because $h$ is a homomorphism, and as a result, if $\rho_1 \approx \rho_2$, then

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

Using this property of $G$, it is also easy to verify the following claims.

*Claim 1:* $G$ is finite.

To show this, it is sufficient to show that $C_{\Theta_0}$ is finite. Consider 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, $C_{\Theta_0}$ is also finite.

*Claim 2:* $G \models \Sigma_1$.

This is immediate from the construction of $G$.

*Claim 3:* $G \models \Sigma_2$.

First, by assumption, $\alpha_i \approx \beta_i$ for any $i \in [1, n]$. In addition, for every $\rho \in \Gamma_0^*$,

$$h(\rho \cdot \alpha_i) = h(\rho \cdot \beta_i),$$

because $h$ is a homomorphism. Therefore, $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. That is,

$$\widehat{\rho \cdot \alpha_i} = \widehat{\rho \cdot \beta_i}.$$

Second, by the construction of $G$, for any $o \in |G|$, $o = o(\widehat{\rho})$ for some $\rho \in \Gamma_0^*$. Moreover, by the property of $G$ described above, for each $\varrho \in \Gamma_0^*$, it can be shown by a straightforward induction on $|\varrho|$ that there is a unique $o' \in |G|$, such that

$$G \models \varrho(o(\widehat{\rho}),\, o').$$

In addition,

$$o' = o(\widehat{\rho \cdot \varrho}).$$

Therefore, for each $o(\widehat{\rho})$ such that $G \models K(o(\widehat{\epsilon}),\, o(\widehat{\rho}))$, $o(\widehat{\rho \cdot \alpha_i})$ is the unique node in $|G|$ such that

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

Similarly, we have $G \models \beta_i(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \beta_i}))$. By $o(\widehat{\rho \cdot \alpha_i}) = o(\widehat{\rho \cdot \beta_i})$, we have

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

Therefore, for each $i \in [1, n]$,

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

Similarly, it can be shown that

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

Therefore, $G \models \Sigma_2$.

*Claim 4:* $G \not\models \forall\, x\, (\alpha(r, x) \to \beta(r, x))$.

As in Claim 3, we can show that $G \models \alpha(o(\widehat{\epsilon}),\, o(\widehat{\alpha}))$, and $G \models \beta(o(\widehat{\epsilon}),\, o(\widehat{\beta}))$. In addition, $o(\widehat{\beta})$ is the unique node in $|G|$ such that $G \models \beta(o(\widehat{\epsilon}),\, o(\widehat{\beta}))$. By assumption, we have $\alpha \not\approx \beta$. Thus

$$\widehat{\alpha} \neq \widehat{\beta}.$$

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

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

That is, $G \not\models \forall\, x\, (\alpha(r, x) \to \beta(r, x))$.

(*only if*) Suppose that $\Sigma_1 \cup \Sigma_2 \not\models_f \forall x\,(\alpha(r,x) \to \beta(r,x)) \;\wedge\; \forall x\,(\beta(r,x) \to \alpha(r,x))$. We show that

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

More specifically, we define a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : \Gamma_0^* \to M$ such that for any $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on $\Gamma_0^*$. By assumption, there exists a finite $\sigma_0$-structure $G$, such that $G \models \Sigma_1 \cup \Sigma_2$, but

$$G \not\models \forall x\,(\alpha(r,x) \to \beta(r,x)) \;\wedge\; \forall x\,(\beta(r,x) \to \alpha(r,x)).$$

Without loss of generality, assume that there is $o \in |G|$ such that

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

Based on $G$, we define an equivalence relation $\sim$ on $\Gamma_0^*$ as follows:

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

Then by $G \models \Sigma_2$, for any $i \in [1,n]$, we have $\alpha_i \sim \beta_i$. By $G \models \Sigma_1$, $G \models K(r^G,\, r^G)$. In addition, by $G \models \alpha(r^G,\, o) \wedge \neg\beta(r^G, o)$, we have

$$G \not\models \forall x\,(K(r,x) \to \forall y\,(\alpha(x,y) \to \beta(x,y))).$$

Therefore, $\alpha \not\sim \beta$.

For every $\rho \in \Gamma_0^*$, let $[\rho]$ denote the equivalence class of $\rho$ with respect to $\sim$. Then clearly, for any $i \in [1,n]$,

$$[\alpha_i] = [\beta_i],$$

but

$$[\alpha] \neq [\beta].$$

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in \Gamma_0^*\}.$$

An important property of $M$ is described as follows.

*Claim 5:* $M$ is finite.

To show this, for every $\rho \in \Gamma_0^*$, let

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

In addition, let

$$S_G = \{S_\rho \mid \rho \in \Gamma_0^*\}.$$

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

*Fact:* For all $\rho, \varrho \in \Gamma_0^*$, $\rho \sim \varrho$ iff $S_\rho = S_\varrho$.

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

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

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

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

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

$$G \not\models \forall x(K(r, x) \to \forall y\, (\rho(x, y) \to \varrho(x, y))).$$

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

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

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

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

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

Using the fact 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 6:* $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in \Gamma_0^*$ 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

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

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

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

By Lemma 4.11 and $G \models K(r^G,\, o) \wedge \rho_1(o,\, o')$, we have

$$G \models K(r^G,\, o').$$

Thus 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,

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

Similarly, we can show that

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

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

*Claim 7:* $\circ$ is associative.

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

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

*Claim 8:* $[\epsilon]$ is the identity for $\circ$. This is because for any $[\rho] \in M$,

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

By these claims, $(M,\, \circ,\, [\epsilon])$ is a finite monoid.

Finally, we define $h : \Gamma_0^* \rightarrow M$ 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 any $i \in [1, n]$, by $[\alpha_i] = [\beta_i]$, $h(\alpha_i) = h(\beta_i)$. Moreover, by $[\alpha] \neq [\beta]$, we have $h(\alpha) \neq h(\beta)$. Therefore,

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

This completes the proof of Lemma 4.12. ■

## 4.2 The collapse of the undecidability in $\mathcal{M}$

In contrast to Theorem 4.4, we next show that in the context of $\mathcal{M}$, path constraint implication is not only decidable in cubic-time, but is also finitely axiomatizable. More specifically, we establish Theorem 4.2 by first presenting a finite axiomatization for implication and finite implication of $P_c$ constraints, and then providing a cubic-time algorithm for testing path constraint implication.

### 4.2.1 A finite axiomatization

Let $\mathcal{I}_r$ be the set consisting of the following inference rules:

- Reflexivity:

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

- Transitivity:

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

- Right-congruence:

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

- Commutativity:

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

- Forward-to-word:

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

- Word-to-forward:

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

- Backward-to-word:

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

- Word-to-backward:

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

Let $\Delta$ be a schema in $\mathcal{M}$ and $\Sigma \cup \{\varphi\}$ be a finite subset of $P_c(\Delta)$. We use $\Sigma \vdash_{\mathcal{I}_r} \varphi$ to denote that $\varphi$ is provable from $\Sigma$ using $\mathcal{I}_r$.

The theorem below shows that $\mathcal{I}_r$ is indeed a finite axiomatization of path constraints.

**Theorem 4.13:** Let $\Delta$ be any schema in $\mathcal{M}$. For every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta)$,

$$\Sigma \models_\Delta \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_r} \varphi,$$
$$\Sigma \models_{(f,\Delta)} \varphi \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_r} \varphi.$$

■

As an immediate result, over any schema $\Delta$ in $\mathcal{M}$, the implication and finite implication problems for $P_c(\Delta)$ coincide and are decidable. To see this, let $\Sigma \cup \{\varphi\}$ be any finite subset of $P_c(\Delta)$. If $\Sigma \models_\Delta \varphi$, then obviously $\Sigma \models_{(f,\Delta)} \varphi$. Conversely, if $\Sigma \models_{(f,\Delta)} \varphi$, then $\Sigma \vdash_{\mathcal{I}_r} \varphi$. By the soundness of $\mathcal{I}_r$ for implication, we have $\Sigma \models_\Delta \varphi$. Thus these two problems coincide. In addition, since $\mathcal{U}(\Delta)$ is definable in first-order logic, the equivalence of these problems leads to the decidability of both problems.

The collapse of the undecidability is due to the following lemma, which can be proved by a straightforward induction on the length of $\alpha$ and by using $\Phi(\Delta)$. On untyped data, this lemma does not hold in general.

**Lemma 4.14:** Let $\Delta$ be an arbitrary schema in $\mathcal{M}$, and $G \in \mathcal{U}(\Delta)$. Then for every $\alpha$ in $Paths(\Delta)$, there is a unique $o \in |G|$, such that $G \models \alpha(r^G, o)$. ■

Because of Lemma 4.14, the following lemmas hold in the context of $\mathcal{M}$.

**Lemma 4.15:** Let $\Delta$ be a schema in $\mathcal{M}$, $\varphi$ be a forward constraint of $P_c(\Delta)$:

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

and $\psi$ be a word constraint in $P_c(\Delta)$:

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

Then for any $G \in \mathcal{U}(\Delta)$, $G \models \varphi$ iff $G \models \psi$. ■

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

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

Thus there exists $a \in |G|$ such that $G \models \alpha(r^G, a) \land \beta(a, b)$. 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

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

Thus $G \models \neg\varphi$.

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

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

By Lemma 4.14, $a$ 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 4.16:** Let $\Delta$ be a schema in $\mathcal{M}$, $\varphi$ be a backward constraint of $P_c(\Delta)$:

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

and $\psi$ be a word constraint in $P_c(\Delta)$:

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

Then for any $G \in \mathcal{U}(\Delta)$, $G \models \varphi$ iff $G \models \psi$. ∎

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

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

By Lemma 4.14, there exists $b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. Clearly, $G \models \neg\gamma(b, a)$ since otherwise $G \models \alpha \cdot \beta \cdot \gamma(r^G, a)$. Hence there are $a, b \in |G|$ such that

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

Thus $G \models \neg\varphi$.

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

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

By Lemma 4.14, $a$ is the unique node such that $G \models \alpha(r^G, a)$, and $b$ is the unique node such that $G \models \beta(a, b)$. Therefore, $G \models \neg\alpha \cdot \beta \cdot \gamma(r^G, a)$ 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 $G \models \neg\psi$. ∎

Using these lemmas, we show Theorem 4.13 as follows.

**Proof of Theorem 4.13:** Soundness of $\mathcal{I}_r$ can be verified by induction on the lengths of $\mathcal{I}_r$-proofs. For the proof of completeness, it suffices to show

*Claim 1:* Let $\Delta$ be a schema in $\mathcal{M}$, $\Sigma \cup \{\varphi\}$ be any finite subset of $P_c(\Delta)$ 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 $G \in \mathcal{U}(\Delta)$ such that

1. $G \models \Sigma$,

2. for every path $\rho$ 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
   $$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(x,\, y)));$$
   - if $G \models \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(y,\, x)))$, then
   $$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(pf(\varphi)(r,\, x) \to \forall y\,(lt(\varphi)(x,\, y) \to \rho(y,\, x))).$$

Here the notations $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$ are described in Definition 2.1.

For if Claim 1 holds and $\Sigma \models_{\Delta} \varphi$, then by $G \models \Sigma$, we have $G \models \varphi$. In addition, by $G \in \mathcal{U}_f(\Delta)$, if $\Sigma \models_{(f,\,\Delta)} \varphi$, then we also have $G \models \varphi$. Thus again by Claim 1, $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

We next show Claim 1. Let $\Delta = (\mathcal{C},\, \nu,\, DBtype)$. We define the structure $G$ described in Claim 1 in two steps: we first define the $k$-neighborhood of $G$, $G_k$, and then construct $G$ from $G_k$. Here *the $k$-neighborhood of $G$* is the substructure $G_k$ of $G$ with its universe

$$|G_k| = \{o \mid o \in |G|,\, G \models \alpha(r^G,\, o) \text{ for some } \alpha \in Paths(\Delta) \text{ with } |\alpha| \leq k\}.$$

To construct $G_k$, we define the following:

- $Paths^k(\Delta) = \{\alpha \mid \alpha \in Paths(\Delta),\ |\alpha| \leq k\}$.
- An equivalence relation $\approx$ on $Paths^k(\Delta)$ defined by
  $$\alpha \approx \beta \ \text{ iff } \ \Sigma \vdash_{\mathcal{I}_r} \forall x\ (\alpha(r,\, x) \to \beta(r,\, x)).$$
  It should be noted that by Commutativity in $\mathcal{I}_r$, $\Sigma \vdash_{\mathcal{I}_r} \forall x\ (\alpha(r,\, x) \to \beta(r,\, x))$ iff $\Sigma \vdash_{\mathcal{I}_r} \forall x\ (\beta(r,\, x) \to \alpha(r,\, x))$.
- $\widehat{\alpha}$ denoting the equivalence class of $\alpha$ with respect to $\approx$, and $\mathcal{A} = \{\widehat{\alpha} \mid \alpha \in Paths^k(\Delta)\}$.
- $type(\widehat{\alpha}) = type(\alpha)$, where $type(\alpha)$ is the type of path $\alpha$ determined by $\Delta$. This is well-defined since if $\alpha$ and $\beta$ are in the same equivalence class, then by the definition of $P_w(\Delta)$, $type(\alpha) = type(\beta)$.

Using these notions, we define $G_k = (|G_k|, r^{G_k}, E^{G_k}, T^{G_k})$ as follows.

- For each $\widehat{\alpha} \in \mathcal{A}$, let $o(\widehat{\alpha})$ be a distinct node and let $|G_k| = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A}\}$.
- Let $r^{G_k} = o(\widehat{\epsilon})$.
- For each $\tau \in T(\Delta)$, let $\tau^{G_k} = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A},\ type(\widehat{\alpha}) = \tau\}$.
- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $i \in [1, n]$, let $G_k \models l_i(o(\widehat{\alpha}),\, o(\widehat{\beta \cdot l_i}))$. Note that this is well-defined by Transitivity and Right-congruence in $\mathcal{I}_r$.

Based on $G_k$, we define $G$ as follows. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node. Let $G = (|G|, r^G, E^G, T^G)$, where

- $|G| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$;
- $r^G = r^{G_k}$;
- for each $\tau \in T(\Delta)$, $\tau^G = \tau^{G_k} \cup \{o(\tau)\}$;
- for each $l \in E(\Delta)$, if $G_k \models l(o, o')$, then $G \models l(o, o')$. Moreover,
  - for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $type(\widehat{\alpha})$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and for some $i \in [1, n]$, $o(\widehat{\alpha})$ does not have any outgoing edge labeled with $l_i$, then let $G \models l_i(o(\widehat{\alpha}), o(\tau_i))$;
  - for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $\tau$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, let $G \models l_i(o(\tau), o(\tau_i))$.

We now show that $G$ is indeed the structure described in Claim 1.

1. $G \in \mathcal{U}_f(\Delta)$.

It is easy to verify that $|G|$ is finite and $G \models \Phi(\Delta)$.

2. $G \models \Sigma$.

We first show the following claim.

*Claim 2:* For every $\alpha \in Paths^k(\Delta)$, $G \models \alpha(r^G, o(\widehat{\alpha}))$.

As an immediate result of Claim 2 and Lemma 4.14, $o(\widehat{\alpha})$ is the unique node in $G$ such that $G \models \alpha(r^G, o(\widehat{\alpha}))$.

We show Claim 2 by induction on $|\alpha|$.

*Base case:* $\alpha = \epsilon$.

Recall that $r^G = o(\widehat{\epsilon})$. Obviously, $G \models \epsilon(r^G, o(\widehat{\epsilon}))$.

*Inductive step:* Assume Claim 2 for $\alpha$. We next show that Claim 2 also holds for $\alpha \cdot l$, where $\alpha \cdot l \in Paths^k(\Delta)$.

By induction hypothesis, $G \models \alpha(r^G, o(\widehat{\alpha}))$. Since $\alpha \cdot l \in Paths^k(\Delta)$, $|\alpha \cdot l| \leq k$. Hence $|\alpha| < k$. By $\alpha \in \widehat{\alpha}$ and the definition of $G$, we have

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

That is, $G \models \alpha \cdot l(r^G, o(\widehat{\alpha \cdot l}))$.

Hence Claim 2 holds.

Using Claim 2, we show $G \models \Sigma$.

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 $\psi$ is a forward constraint $\forall x\, (\alpha(r, x) \to \forall y\, (\beta(x, y) \to \gamma(x, y)))$, 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 4.14 and Claim 2, we have $a = o(\widehat{\alpha})$ and $b = o(\widehat{\alpha \cdot \beta})$. By Forward-to-word in $\mathcal{I}_r$, we have

$$\alpha \cdot \beta \approx \alpha \cdot \gamma.$$

Therefore, again by Claim 2, we have $G \models \alpha \cdot \gamma(r^G, o(\widehat{\alpha \cdot \beta}))$. By Lemma 4.14, we have

$$G \models \gamma(o(\widehat{\alpha}), o(\widehat{\alpha \cdot \beta})).$$

This contradicts the assumption.

If $\psi$ is a backward constraint $\forall x\, (\alpha(r, x) \to \forall y\, (\beta(x, y) \to \gamma(y, x)))$, 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 4.14 and Claim 2, we have $a = o(\widehat{\alpha})$ and $b = o(\widehat{\alpha \cdot \beta})$. By Backward-to-word in $\mathcal{I}_r$, we have

$$\alpha \approx \alpha \cdot \beta \cdot \gamma.$$

Therefore, again by Claim 2, we have $G \models \alpha \cdot \beta \cdot \gamma(r^G, o(\widehat{\alpha}))$. By Lemma 4.14, we have

$$G \models \gamma(o(\widehat{\alpha \cdot \beta}), o(\widehat{\alpha})).$$

This again contradicts the assumption.

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

3. $G$ has the property described by (2) of Claim 1.

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

If $G \models \forall x\, (pf(\varphi)(r, x) \to \forall y\, (lt(\varphi)(x, y) \to \rho(x, y)))$, then by Lemma 4.15,

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

By Claim 2 and Lemma 4.14, we have

$$G \models pf(\varphi) \cdot \rho(r^G, o(\widehat{pf(\varphi) \cdot lt(\varphi)})),$$

and moreover, $pf(\varphi) \cdot lt(\varphi) \approx pf(\varphi) \cdot \rho$. Thus by the definition of $\approx$ and Commutativity in $\mathcal{I}_r$, we have

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

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

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

If $G \models \forall x\,(pf(\varphi)(r, x) \to \forall y\,(lt(\varphi)(x, y) \to \rho(y, x)))$, then by Lemma 4.16, we have

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

By Claim 2 and lemma 4.14, we have

$$G \models pf(\varphi) \cdot lt(\varphi) \cdot \rho(r^G, o(\widehat{pf(\varphi)})),$$

and moreover, $pf(\varphi) \approx pf(\varphi) \cdot lt(\varphi) \cdot \rho$. Thus by the definition of $\approx$ and Commutativity in $\mathcal{I}_r$, we have

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

By Word-to-backward in $\mathcal{I}_r$, we have

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

This completes the proof of Claim 1, and therefore, the proof of Theorem 4.13. ■

### 4.2.2 An algorithm

Next, we present an algorithm for testing path constraint implication in the context of $\mathcal{M}$. Let $\Delta$ be a schema in $\mathcal{M}$. This algorithm takes as input a finite subset $\Sigma$ of $P_c(\Delta)$ and a path $\alpha \cdot \beta$ in $Paths(\Delta)$. It computes a pseudo model $G$ of $\Sigma$ having the following properties: $G \models \Sigma$ and there are $a, b \in |G|$ such that $G \models \alpha(r^G, a) \wedge \beta(a, b)$. In addition, for any path $\gamma$,

$$\begin{aligned} G &\models \gamma(a, b) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \to \forall y\,(\beta(x, y) \to \gamma(x, y))), \\ G &\models \gamma(b, a) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r, x) \to \forall y\,(\beta(x, y) \to \gamma(y, x))). \end{aligned}$$

By Theorem 4.13, this algorithm can be used for testing implication and finite implication of constraints of $P_c(\Delta)$ in the context of $\mathcal{M}$.

Before we present the algorithm, we first define the following. Let $\Delta$ be a schema in $\mathcal{M}$, $\Sigma$ be a finite subset of $P_c(\Delta)$ and $\alpha \cdot \beta \in Paths(\Delta)$. We define

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

Here $\rho \preceq_p \varrho$ denotes that $\rho$ is a prefix of $\varrho$, as described in Section 2.

Using these notions, we give the algorithm (Algorithm 4.1) in Table 2. The procedure $merge(a, b)$ used in Algorithm 4.1 is given in Table 3.

The following should be noted about Algorithm 4.1.

*Remark 1:* Algorithm 4.1 is independent of any particular schema. Although it is required that input constraints and path are defined over some schema in $\mathcal{M}$, no particular information

**Algorithm 4.1:**

*Input:* a finite subset $\Sigma$ of $P_c(\Delta)$ and a path $\alpha \cdot \beta \in Paths(\Delta)$
*Output:* the structure $G$ described above

```
1. $E_\phi$ := the set of edge labels appearing in either $\alpha \cdot \beta$ or some path in constraints of $\Sigma$;

2. $Rules := \Sigma$;

3. $G := (|G|, r^G, E^G_\phi)$, where
    • $|G| = \{o(\rho) \mid \rho \in CloPts(\Sigma, \alpha \cdot \beta),\ o(\rho)$ is a distinct node$\}$,
    • $r^G = o(\epsilon)$,
    • $E^G_\phi$ is populated in such a way that $G \models l(o(\rho), o(\varrho))$ iff $\varrho = \rho \cdot l$;

4. for each $\psi \in \Sigma$ do:
    (1) if $\psi = \forall x\,(\rho(r, x) \to \forall y\,(\varrho(x, y) \to \zeta(x, y)))$ then
          (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \to \forall y\,(\varrho(x, y) \to \zeta(x, y)))\}$;
          (ii) $merge(o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta})$,
               where $o_{\rho\cdot\varrho}, o_{\rho\cdot\zeta} \in |G|$ such that $G \models \rho \cdot \varrho(r^G, o_{\rho\cdot\varrho}) \wedge \rho \cdot \zeta(r^G, o_{\rho\cdot\zeta})$;
    (2) if $\psi = \forall x\,(\rho(r, x) \to \forall y\,(\varrho(x, y) \to \zeta(y, x)))$ then
          (i) $Rules := Rules \setminus \{\forall x\,(\rho(r, x) \to \forall y\,(\varrho(x, y) \to \zeta(y, x)))\}$;
          (ii) $merge(o_\rho, o_{\rho\cdot\varrho\cdot\zeta})$,
               where $o_\rho, o_{\rho\cdot\varrho\cdot\zeta} \in |G|$ such that $G \models \rho(r^G, o_\rho) \wedge \rho \cdot \varrho \cdot \zeta(r^G, o_{\rho\cdot\varrho\cdot\zeta})$;

5. output $G$.
```

Table 2: An algorithm for testing path constraint implication in $\mathcal{M}$

```
procedure  $merge(a, b)$

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

Table 3: Procedure *merge* used in Algorithm 4.1

about the schema is used by the algorithm. As a result, this algorithm can be used in the context of any schema in $\mathcal{M}$.

*Remark 2:* Let $\Delta$ be a schema in $\mathcal{M}$, and an input of the algorithm be a finite subset $\Sigma$ of $P_c(\Delta)$ and a path $\alpha \cdot \beta \in Paths(\Delta)$. Then the structure $G$ computed by the algorithm may not be in $\mathcal{U}(\Delta)$. However, $G$ can be naturally extended to a structure $H \in \mathcal{U}_f(\Delta)$, as follows. Let $H = (|H|, r^H, E^H, T^H)$, where

- $|H| = |G| \cup \{o(\tau) \mid \tau \in T(\Delta),\ o(\tau) \text{ is a distinct node}\}$;
- $r^H = r^G$;
- for each $\tau \in T(\Delta)$,

$$\tau^H = \{o(\tau)\} \cup \{o \mid o \in |G|,\ \rho \in Paths(\Delta),\ type(\rho) = \tau,\ G \models \rho(r^G, o)\};$$

- for each $l \in E(\Delta)$, if $G \models l(o, o')$, then $H \models l(o, o')$. Moreover,
  - for each $o \in |G|$, if there is $\rho \in Paths(\Delta)$ such that $G \models \rho(r^G, o)$ and $type(\rho) = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $type(\rho)$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), and in addition, for some $i \in [1, n]$, $o$ does not have an outgoing edge labeled $l_i$, then let $H \models l_i(o, o(\tau_i))$;
  - for each type $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or $\tau$ is some class $C \in \mathcal{C}$ and $\nu(C) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$), then for each $i \in [1, n]$, let $H \models l_i(o(\tau), o(\tau_i))$.

It is easy to verify that $H \in \mathcal{U}(\Delta)$ as long as $\Sigma \subseteq P_c(\Delta)$ and $\alpha \cdot \beta \in Paths(\Delta)$. In addition, it is easy to verify that $H$ is finite, and therefore, $H \in \mathcal{U}_f(\Delta)$.

We call the structure $H$ defined above *the extension of* $G$ *with respect to* $\Delta$.

*Remark 3:* The rationale behind the procedure *merge* is Commutativity, Transitivity and Right-congruence in $\mathcal{I}_r$.

*Remark 4:* The rationale behind step 4 (1) (i) and 4 (2) (i) of Algorithm 4.1 is Lemma 4.14. Let $\Delta$ be a schema in $\mathcal{M}$ and $G \in \mathcal{U}(\Delta)$. For any path $\rho \in Paths(\Delta)$, there exists a unique $o \in |G|$ such that $G \models \rho(r^G, o)$. As a result, every constraint in $\Sigma$ can be applied at most once by the algorithm. It is because of this property that Algorithm 4.1 has low complexity.

Next, we analyze the complexity of the algorithm. Let $n_E$ be the cardinality of $E_\phi$, $n_C$ the cardinality of $CloPts(\Sigma, \alpha \cdot \beta)$, $n_G$ the size of $|G|$, $n$ the length of $\Sigma$ and $\alpha \cdot \beta$, and $n_\Sigma$ the cardinality of $\Sigma$. Then the following should be noted.

- $n_E \leq n$, $n_C \leq n$, $n_G \leq n$ and $n_\Sigma \leq n$.
- Step 4 is executed $n_\Sigma$ 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 $E_\phi$, this can be done in $O(|\rho|)$ time, i.e., $O(n)$ time.
- 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 4.1 runs in $O(n^3)$ time. In addition, when implemented using appropriate data structures, this algorithm runs in $O(n^2)$ time.

The proposition below shows that Algorithm 4.1 is correct.

**Proposition 4.17:** Let $\Delta$ be a schema in $\mathcal{M}$. Given a finite subset $\Sigma$ of $P_c(\Delta)$ and path $\alpha \cdot \beta \in \mathit{Paths}(\Delta)$, Algorithm 4.1 computes a structure $G$ having the following property: $G \models \Sigma$, and there are $a, b \in |G|$ such that $G \models \alpha(r^G,\, a) \wedge \beta(a,\, b)$. Moreover, for any path $\gamma$,

$$\begin{aligned} G &\models \gamma(a,\, b) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(x,\, y))),\\ G &\models \gamma(b,\, a) \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(y,\, x))). \end{aligned}$$

■

**Proof:** Step 4 of Algorithm 4.1 ensures that $G \models \Sigma$, because of Lemma 4.14. In addition, step 3 ensures that there are $a, b \in |G|$, such that $G \models \alpha(r^G,\, a) \wedge \beta(a,\, b)$. Let $H$ denote the extension of $G$ with respect to $\Delta$. Then it is easy to verify that $H \models \alpha(r^G,\, a) \wedge \beta(a,\, b)$ and $H \models \Sigma$. Thus if $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(x,\, y)))$, then by Theorem 4.13, $H \models \gamma(a,\, b)$. By the definition of $H$, it is easy to verify that

$$G \models \gamma(a,\, b).$$

Similarly, if $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(y,\, x)))$, then $H \models \gamma(b,\, a)$. Again by the definition of $H$, it is easy to verify that

$$G \models \gamma(b,\, a).$$

Conversely, by an induction on the number of steps in the construction of $G$ by the algorithm, we can show that for all paths $\rho$ and $\varrho$, if there exists node $o \in |G|$ such that $G \models \rho(r^G,\, o) \wedge \varrho(r^G,\, o)$, then $\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\rho(r,\, x) \to \varrho(r,\, x))$. Indeed, each step of the construction in fact corresponds to applications of some rules in $\mathcal{I}_r$. For example, step 4 (1) corresponds to an application of Forward-to-word, step 4 (2) corresponds to an application of Backward-to-word, and *merge* corresponds to applications of Transitivity, Right-congruence and Commutativity in $\mathcal{I}_r$. As a result, if $G \models \gamma(a, b)$, then by Word-to-forward in $\mathcal{I}_r$, it can be verified that

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(x,\, y))).$$

Similarly, if $G \models \gamma(b, a)$, then by Word-to-backward in $\mathcal{I}_r$, it can be verified that

$$\Sigma \vdash_{\mathcal{I}_r} \forall x\,(\alpha(r,\, x) \to \forall y\,(\beta(x,\, y) \to \gamma(y,\, x))).$$

■

From Proposition 4.17, Algorithm 4.1 and Theorem 4.13, Theorem 4.2 follows immediately.

## 4.3 Undecidability in the context of $\mathcal{M}^+$

To further explore the impact of type systems, we next show Theorem 4.5. That is, in the context of $\mathcal{M}^+$ (i.e., when the set construct is allowed), the implication and finite implication problems for $P_w(\alpha)$ remain undecidable.

Along the same lines as the proof of Theorem 4.1, we show Theorem 4.5 by reduction from the word problem for (finite) monoids. We first present an encoding of the word problem for (finite) monoids in terms of the (finite) implication problem for $P_w(\alpha)$ in $\mathcal{M}^+$, and then show that the encoding is indeed a reduction.

We begin with the definition of a schema $\Delta_0$. Recall the alphabet $\Gamma_0$ described in Section 4.1. Using $\Gamma_0$, we define

$$\Delta_0 = (\mathcal{C},\ \nu,\ DBtype),$$

where

- $\mathcal{C} = \{C, C_s\}$,
- $\nu$ is defined by:

$$\begin{array}{rcl} C & \mapsto & [l_1 : C,\ \ldots,\ l_m : C] \\ C_s & \mapsto & \{C\} \end{array}$$

- $DBtype = [a : C,\ b : C_s]$, where $a, b \notin \Gamma_0$.

Note here that each symbol in $\Gamma_0$ is a record label of $C$, and thus is a binary relation symbol in $E(\Delta_0)$. Moreover, every $\alpha$ in $\Gamma_0^*$ can be represented as a path over $\Delta_0$, also denoted $\alpha$. In addition, it is straightforward to verify the following lemma, using the type constraint $\Phi(\Delta_0)$.

**Lemma 4.18:** For each $G \in \mathcal{U}(\Delta_0)$ and every $\alpha \in \Gamma_0^*$, $G$ has the following properties.

1. There is a unique $o \in |G|$ such that $G \models a \cdot \alpha(r^G, o)$. This node is denoted by $o_\alpha$.
2. For every $o \in |G|$, if $G \models C^G(o)$, then there is a unique $o' \in |G|$, such that $G \models \alpha(o, o')$. ■

Next, we encode equations over $\Gamma_0$. Recall the finite set $\Theta_0$ of equations described in Section 4.1. We encode $\Theta_0$ in terms of two finite sets of constraints $\Sigma_1 \subseteq P_w(\Delta_0)$ and $\Sigma_2 \subseteq P_w(\Delta_0,\ b \cdot *)$ (the notion of $P_w(\Delta,\ \alpha)$ is defined in Section 3.2).

1. The constraint

$$\forall x\, (a(r, x) \rightarrow b \cdot *(r, x))$$

is in $\Sigma_1$. Moreover, for each $j \in [1, m]$, the following constraint is in $\Sigma_1$:

$$\forall x\, (b \cdot * \cdot l_j(r, x) \rightarrow b \cdot *(r, x))$$

In addition, $\Sigma_1$ consists of only those constraints defined above.

2. For each $(\alpha_i,\ \beta_i) \in \Theta_0$, the following constraint is in $\Sigma_2$:

$$\forall\, x\, (b \cdot *(r, x) \to \forall\, y\, (\alpha_i(x, y) \to \beta_i(x, y)))$$

   In addition, $\Sigma_2$ consists of only those constraints defined above.

We encode a test equation $(\alpha,\ \beta)$ over $\Gamma_0$ as a constraint in $P_w(\Delta_0)$:

$$\varphi_{(\alpha,\beta)} = \forall\, x (a \cdot \alpha(r, x) \to a \cdot \beta(r, x)).$$

Obviously, $\Sigma_1 \cup \Sigma_2 \cup \{\varphi_{(\alpha,\beta)}\}$ is a subset of $P_w(\Delta_0,\ b \cdot *)$.

We reduce the word problem for monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models_{\Delta_0} \varphi_{(\alpha,\beta)},$$

and analogously, reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma_1 \cup \Sigma_2 \models_{(f,\,\Delta_0)} \varphi_{(\alpha,\beta)}.$$

Before we show that this is indeed a reduction, we first identify some basic properties of $\mathcal{U}(\Delta_0)$ and $\Sigma_1$, which can be easily verified by using Lemma 4.18.

**Lemma 4.19:** For any $G \in \mathcal{U}(\Delta_0)$ and $\alpha, \beta \in \Gamma_0^*$, if

$$G \models \forall\, x\, (b \cdot *(r, x) \to \forall\, y\, (\alpha(x, y) \to \beta(x, y))),$$

then

$$G \models \forall\, x\, (b \cdot *(r, x) \to \forall\, y\, (\beta(x, y) \to \alpha(x, y))). \tag{a}$$

Similarly, if

$$G \models \forall\, x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x)),$$

then

$$G \models \forall\, x\, (a \cdot \beta(r, x) \to a \cdot \alpha(r, x)). \tag{b}$$

■

**Proof:** We show (a) only. The proof of (b) is similar.

By Lemma 4.18, for each $o \in |G|$ such that $G \models b \cdot *(r^G, o)$, there is a unique node $o_1 \in |G|$ such that

$$G \models \alpha(o, o_1).$$

Similarly, there is a unique node $o_2 \in |G|$ such that

$$G \models \beta(o, o_2).$$

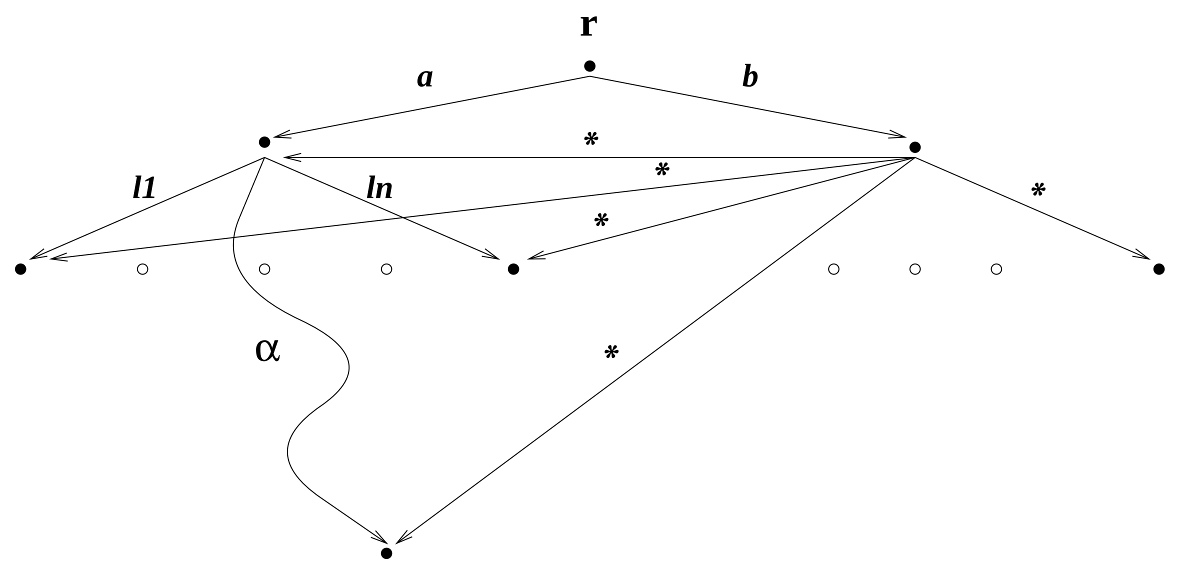

Figure 3: A structure in $\mathcal{U}(\Delta_0)$ that satisfies $\Sigma_1$

By $G \models \forall x\,(b \cdot *(r, x) \to \forall y\,(\alpha(x, y) \to \beta(x, y)))$, we have

$$o_1 = o_2.$$

Hence $G \models \forall x\,(b \cdot *(r, x) \to \forall y\,(\beta(x, y) \to \alpha(x, y)))$. ■

**Lemma 4.20:** For every $G \in \mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$, then for every $\alpha \in \Gamma_0^*$,

$$G \models b \cdot *(r^G, o_\alpha),$$

where $o_\alpha$ is the unique node in $|G|$ such that $G \models a \cdot \alpha(r^G, o_\alpha)$. In addition, for all $o, o' \in |G|$ such that $G \models b \cdot *(r^G,\, o') \wedge \alpha(o', o)$, we have

$$G \models b \cdot *(r^G, o).$$

■

**Proof:** By a straightforward induction on $|\alpha|$. ■

By Lemma 4.20, the structures in $\mathcal{U}(\Delta_0)$ that satisfy $\Sigma_1$ have the form shown in Figure 3.

**Lemma 4.21:** For every $G \in \mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$ and

$$G \models \forall x\,(b \cdot *(r, x) \to \forall y\,(\alpha(x, y) \to \beta(x, y)))$$

for some $\alpha, \beta \in \Gamma_0^*$, then

$$G \models \forall x\,(a \cdot \alpha(r, x) \to a \cdot \beta(r, x)).$$

■

**Proof:** By Lemma 4.18, there is a unique node $o \in |G|$ such that

$$G \models a(r^G, o).$$

Denote this node by $o_a$. Again by Lemma 4.18, there is a unique node $o_\alpha \in |G|$ such that

$$G \models a \cdot \alpha(r^G, o_\alpha),$$

and there is a unique node $o_\beta \in |G|$ such that

$$G \models a \cdot \beta(r^G, o_\beta).$$

Therefore,

$$G \models \alpha(o_a,\, o_\alpha) \wedge \beta(o_a,\, o_\beta).$$

By Lemma 4.20,

$$G \models b \cdot *(r^G,\, o_a).$$

As a result, if $G \models \forall x\, (b \cdot *(r, x) \to \forall y\, (\alpha(x, y) \to \beta(x, y)))$, then

$$o_\alpha = o_\beta.$$

Hence $G \models \forall x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x))$. ∎

Finally, we show that the encoding given above is indeed a reduction from the word problem for (finite) monoids. It suffices to show the following lemma.

**Lemma 4.22:** In the context of $\mathcal{M}^+$, for all $\alpha$ and $\beta$ in $\Gamma_0^*$,

$$\Theta_0 \models (\alpha,\, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_{\Delta_0} \forall x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x)), \tag{a}$$

$$\Theta_0 \models_f (\alpha,\, \beta) \quad \text{iff} \quad \Sigma_1 \cup \Sigma_2 \models_{(f,\, \Delta_0)} \forall x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x)). \tag{b}$$

∎

**Proof:** The proof is similar to that of Lemma 4.12. 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 there exists structure $G \in \mathcal{U}_f(\Delta_0)$, such that $G \models \Sigma_1 \cup \Sigma_2$, but $G \not\models \forall x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x))$.

By $\Theta_0 \not\models_f (\alpha,\, \beta)$, there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : \Gamma_0^* \to M$ such that for any $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Using $M$ and $h$, we define an equivalence relation $\approx$ in the same way as in the proof of Lemma 4.12. In addition, for each $\rho \in \Gamma_0^*$, let $\widehat{\rho}$ be defined as in the proof of Lemma 4.12. Similarly, we define $C_{\Theta_0}$.

Using $C_{\Theta_0}$, we construct structure $G = (|G|,\, r^G,\, E^G,\, T^G)$ as follows.

(1) $|G|$.

For each $\widehat{\rho} \in C_{\Theta_0}$, let $o(\widehat{\rho})$ be a distinct node. In addition, let $o_r$ and $o_b$ be two distinct nodes. Then we define

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

(2) $r^G = o_r$.

(3) The unary relations are populated as follows.

- $C^G = \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}$.
- ${C_s}^G = \{o_b\}$.
- $DBtype^G = \{o_r\}$.

(4) The binary relations are populated as follows.

- $G \models a(o_r,\, o(\widehat{\epsilon}))$.
- $G \models b(o_r,\, o_b)$.
- For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models *(o_b,\, o(\widehat{\rho}))$.
  In addition, for each $j \in [1, m]$, let $G \models l_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot l_j}))$.

By the construction of $G$, it is easy to verify the following claims.

*Claim 1:* $G \in \mathcal{U}_f(\Delta_0)$.

To see this, first note that if $\rho \approx \varrho$, then $\rho \cdot l_j \approx \varrho \cdot l_j$ for all $\rho$ and $\varrho$ in $\Gamma_0^*$ and $j \in [1, m]$. This is because $h$ is a homomorphism, and as a result, if $h(\rho) = h(\varrho)$, then

$$\begin{array}{rcl} h(\rho \cdot l_j) & = & h(\rho) \circ h(l_j) \\ & = & h(\varrho) \circ h(l_j) \\ & = & h(\varrho \cdot l_j). \end{array}$$

Second, as in the proof of Lemma 4.12, it can be shown that $C_{\Theta_0}$ is finite. Therefore, $G$ is finite.

*Claim 2:* $G \models \Sigma_1$.

This is immediate from the construction of $G$.

*Claim 3:* $G \models \Sigma_2$.

First, by assumption, for every $i \in [1, n]$, $\alpha_i \approx \beta_i$. In addition, for every $\rho \in \Gamma_0^*$,

$$h(\rho \cdot \alpha_i) = h(\rho \cdot \beta_i).$$

This is because $h$ is a homomorphism. Therefore, $\rho \cdot \alpha_i \approx \rho \cdot \beta_i$. That is,

$$\widehat{\rho \cdot \alpha_i} = \widehat{\rho \cdot \beta_i}.$$

Second, by the construction of $G$, for any $o \in |G|$, if $G \models b \cdot *(o_r,\, o)$, then $o = o(\widehat{\rho})$ for some $\rho \in \Gamma_0^*$. Moreover, by Lemma 4.18, for each $\varrho \in \Gamma_0^*$, there is a unique $o' \in |G|$, such that

$$G \models \varrho(o(\widehat{\rho}),\, o').$$

By a straightforward induction on $|\varrho|$, it can be shown that

$$o' = o(\widehat{\rho \cdot \varrho}).$$

Therefore, for each $o(\widehat{\rho})$ such that $G \models b \cdot *(o_r,\, o(\widehat{\rho}))$, $o(\widehat{\rho \cdot \alpha_i})$ is the unique node in $|G|$ such that

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

Similarly, we have $G \models \beta_i(o(\widehat{\rho}),\, o(\widehat{\rho \cdot \beta_i}))$. By $o(\widehat{\rho \cdot \alpha_i}) = o(\widehat{\rho \cdot \beta_i})$, we have

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

Therefore, for each $i \in [1, n]$,

$$G \models \forall\, x\, (b \cdot *(r, x) \to \forall\, y\, (\alpha_i(x, y) \to \beta_i(x, y))).$$

That is, $G \models \Sigma_2$.

*Claim 4:* $G \not\models \forall\, x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x))$.

As in Claim 3, we can show that

$$G \models a \cdot \alpha(o_r,\, o(\widehat{\alpha})),$$

and

$$G \models a \cdot \beta(o_r,\, o(\widehat{\beta})).$$

In addition, $o(\widehat{\beta})$ is the unique node in $|G|$ such that $G \models a \cdot \beta(o_r,\, o(\widehat{\beta}))$. By assumption, we have $\alpha \not\approx \beta$. Thus $\widehat{\alpha} \neq \widehat{\beta}$. Hence

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

Therefore,

$$G \models a \cdot \alpha(o_r,\, o(\widehat{\alpha})) \wedge \neg a \cdot \beta(o_r,\, o(\widehat{\alpha})).$$

That is, $G \not\models \forall\, x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x))$.

(*only if*) Suppose that there exists $G \in \mathcal{U}_f(\Delta_0)$, such that $G \models \Sigma_1 \cup \Sigma_2$, but

$$G \not\models \forall\, x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x)).$$

Then we show that $\Theta_0 \not\models_f (\alpha,\, \beta)$. That is, we show that there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : \Gamma_0^* \to M$ such that for any $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on $\Gamma_0^*$ as follows:

$$\rho \sim \varrho \quad \text{iff} \quad G \models \forall\, x (b \cdot *(r, x) \to \forall\, y\, (\rho(x, y) \to \varrho(x, y))).$$

Note that by Lemma 4.19, it can be easily verified that $\sim$ is indeed an equivalence relation.

By $G \models \Sigma_2$ and Lemma 4.19 (a), for any $i \in [1, n]$, we have

$$\alpha_i \sim \beta_i.$$

In addition, because $G \not\models \forall x\, (a \cdot \alpha(r, x) \to a \cdot \beta(r, x))$, by Lemma 4.19 (b) and Lemma 4.21, we have

$$G \not\models \forall x\, (b \cdot *(r, x) \to \forall y\, (\alpha(x, y) \to \beta(x, y))).$$

Therefore,

$$\alpha \not\sim \beta.$$

As in the proof of Lemma 4.12, for every $\rho \in \Gamma_0^*$, let $[\rho]$ be the equivalence class of $\rho$ with respect to $\sim$. Then clearly, for any $i \in [1, n]$, $[\alpha_i] = [\beta_i]$. But $[\alpha] \neq [\beta]$.

Using the notion of $\sim$, we define

$$M = \{[\rho] \mid \rho \in \Gamma_0^*\}.$$

As in the proof of Lemma 4.12, we can show the following claim.

*Claim 5:* $M$ is finite.

To show this, for every $\rho \in \Gamma_0^*$, let

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

In addition, let

$$S_G = \{S_\rho \mid \rho \in \Gamma_0^*\}.$$

By Lemma 4.18, $S_\rho$ is a finite function from $|G|$ to $|G|$. Since $|G|$ is finite, there are finitely many such functions. Therefore, $S_G$ is finite. Moreover, it is easy to verify that for all $\rho, \varrho \in \Gamma_0^*$,

$$\rho \sim \varrho \quad \text{iff} \quad S_\rho = S_\varrho.$$

Consider function $g : M \to S_G$ defined by

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

Clearly, $g$ 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

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

As in the proof of Lemma 4.12, it is easy to verify the following claims.

*Claim 6:* $\circ$ is well-defined.

To see this, for all $\rho_1, \rho_2, \varrho_1, \varrho_2 \in \Gamma_0^*$ such that $\rho_1 \sim \rho_2$ and $\varrho_1 \sim \varrho_2$, we show that

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

By lemma 4.18, for every $o \in |G|$ such that $G \models b \cdot *(r^G, o)$, there exists a unique $o_1 \in |G|$ such that

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

In addition, there is a unique $o' \in |G|$ such that

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

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

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

By Lemma 4.20 and $G \models b \cdot *(r^G, o) \wedge \rho_1(o, o')$, we have

$$G \models b \cdot *(r^G, o').$$

Thus 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,

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

That is, $\rho_1 \cdot \varrho_1 \sim \rho_2 \cdot \varrho_2$.

*Claim 7:* $\circ$ is associative.

The proof is the same as found in the proof of Lemma 4.12.

*Claim 8:* $[\epsilon]$ is the identity for $\circ$.

The proof is the same as found in the proof of Lemma 4.12.

By these claims, $(M, \circ, [\epsilon])$ is a finite monoid.

Finally, we define $h : \Gamma_0^* \to M$ by

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

As in the proof of Lemma 4.12, we can show that $h$ is a homomorphism, and moreover, for any $i \in [1, n]$,

$$\begin{array}{rcl} h(\alpha_i) & = & h(\beta_i), \\ h(\alpha) & \neq & h(\beta). \end{array}$$

Therefore,

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

This completes the proof of Lemma 4.22. ■

## 4.4 Undecidability in the context of $\mathcal{M}_f^+$

Next, we show Theorem 4.6. That is, in the context of $\mathcal{M}_f^+$ (i.e., when the finite set construct is allowed), the implication and finite implication problems for $P_w(\alpha)$ are also undecidable.

Recall the schema $\Delta_0$ defined in Section 4.3. Note that $\Delta_0$ is also a schema in $\mathcal{M}_f^+$. Therefore, we can use the proof of Lemma 4.22 (b) to establish the undecidability of the finite implication problem for $P_w(\alpha)$ in the context of $\mathcal{M}_f^+$. That is, by reduction from the word problem for finite monoids. To establish the undecidability of the implication problem for $P_w(\alpha)$, it suffices to show that in $\mathcal{M}_f^+$, the finite implication problem and the implication problem for $P_c$ over $\Delta_0$ coincide. More specifically, we show the following lemma.

**Lemma 4.23:** Let $\Delta_0$ be the schema defined in Section 4.3. In the context of $\mathcal{M}_f^+$, for any finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta_0)$, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta_0)$, then it has a model in $\mathcal{U}_f(\Delta_0)$. ■

For if the lemma holds, then for every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta_0)$,

$$\Sigma \models_{\Delta_0} \varphi \quad \text{iff} \quad \Sigma \models_{(f,\Delta_0)} \varphi.$$

To see this, first note that if $\Sigma \models_{\Delta_0} \varphi$, then $\Sigma \models_{(f,\Delta_0)} \varphi$. Conversely, if $\Sigma \not\models_{\Delta_0} \varphi$, then by Lemma 4.23, $\Sigma \not\models_{(f,\Delta_0)} \varphi$. Therefore, the implication problem for $P_w(\alpha)$ over $\Delta_0$ is undecidable if and only if the finite implication problem for $P_w(\alpha)$ over $\Delta_0$ is undecidable.

It should be noted here that the equivalence of the implication problem and the finite implication problem for $P_c$ over $\Delta_0$ in $\mathcal{M}_f^+$ does not lead to the decidability of these problems. This is because these problems are considered in connection with schema $\Delta_0$, and $\mathcal{U}(\Delta_0)$ in $\mathcal{M}_f^+$ is not definable in first-order logic.

Below we show Lemma 4.23.

**Proof of Lemma 4.23:** Given $\Sigma \cup \{\varphi\} \subset P_c(\Delta_0)$ and a model $G$ of $\bigwedge \Sigma \wedge \neg\varphi$ in $\mathcal{U}(\Delta_0)$, we construct a finite structure $G'$ such that $G' \in \mathcal{U}_f(\Delta_0)$ and $G' \models \bigwedge \Sigma \wedge \neg\varphi$.

Recall the notion of $k$-neighborhood defined in the proof of Theorem 4.13. Also recall the notations $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$ described in Definition 2.1. Given $\Sigma$ and $\varphi$ as described above, let $k = max\{|pf(\psi)| + |lt(\psi)| + |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\} + 1$. In addition, let $G_k$ be the $k$-neighborhood of $G$. Then we construct $G'$ as follows. For each $\tau \in T(\Delta_0)$, let $o(\tau)$ be a distinct node, and let $G' = (|G'|, r^{G'}, E^{G'}, T^{G'})$, where

- $|G'| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta_0)\}$,
- $r^{G'} = r^{G_k}$,
- for each $\tau \in T(\Delta_0)$, $\tau^{G'} = (\tau^G \cap |G_k|) \cup \{o(\tau)\}$,
- $E^{G'}$ is $E^{G_k}$ augmented with the following:
  - for each $o \in \tau^G \cap |G_k|$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or for some class $A$, $\tau = A$ and $\nu(A) = [l_1 : \tau_1, ..., l_n : \tau_n]$), and if for some $i \in [1, n]$ and any $o' \in |G_k|$, $G_k \not\models l_i(o, o')$, then let $G' \models l_i(o, o(\tau_i))$;

– for each type $\tau \in T(\Delta_0)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$ (or for some class $A$, $\tau = A$ and $\nu(A) = [l_1 : \tau_1, ..., l_n : \tau_n]$), then for each $i \in [1, n]$, let $G \models l_i(o(\tau), o(\tau_i))$.

We now show that $G'$ is indeed the structure desired.

(1) $G' \in \mathcal{U}_f(\Delta_0)$.

Since $G \in \mathcal{U}(\Delta_0)$, each node in $|G|$ has finitely many outgoing edges. Hence by the definition of $G_k$, $|G_k|$ is finite. In addition, $T(\Delta_0)$ is finite. Therefore, by the construction of $G'$, $|G'|$ is finite. In addition, it can be easily verified that $G' \models \Phi(\Delta_0)$.

(2) $G' \models \bigwedge \Sigma \wedge \neg\varphi$.

The following can be easily verified by *reductio*:

*Claim:* $G \models \bigwedge \Sigma \wedge \neg\varphi$ iff $G_k \models \bigwedge \Sigma \wedge \neg\varphi$. In addition, if $G_k$ is the $k$-neighborhood of $G'$, then $G' \models \bigwedge \Sigma \wedge \neg\varphi$ iff $G_k \models \bigwedge \Sigma \wedge \neg\varphi$.

By the claim, it suffices to show that $G_k$ is also the $k$-neighborhood of $G'$. To do this, assume for *reductio* that there exist $o(\tau) \in |G'|$ and $\alpha \in Paths(\Delta_0)$ such that $|\alpha| \leq k$ and $G' \models \alpha(r, o(\tau))$. Without loss of generality, assume that $\alpha$ has the shortest length among such paths. Then by the construction of $G'$, there is $o \in |G_k|$, such that

- $\alpha = \alpha' \cdot l$ and $G' \models \alpha'(r^{G'}, o) \wedge l(o, o(\tau))$;
- there is $\tau \in T(\Delta_0)$ such that $\tau = [l : \tau, ...]$ (or $\tau = A$ and $\nu(A) = [l : \tau, ...]$ for some class $A$), $o \in \tau^G$, and for any $o' \in |G_k|$, $G_k \not\models l(o, o')$; and
- $G_k \models \alpha'(r^{G'}, o)$. This is because for each $\tau \in T(\Delta_0)$, $o(\tau)$ does not have any outgoing edge to any node of $|G_k|$.

By $G \in \mathcal{U}(\Delta_0)$, there is $o' \in |G|$ such that $G \models l(o, o')$. By the argument above, $o' \notin |G_k|$. Hence by the definition of $k$-neighborhood, there is no path $\beta \in Paths(\Delta_0)$ such that $|\beta| < k$ and $G \models \beta(r, o) \wedge l(o, o')$. Therefore, $\alpha'$ must have a length of at least $k$. That is, $|\alpha| > k$. This contradicts the assumption. Hence $G_k$ is indeed the $k$-neighborhood of $G'$.

Therefore, $G'$ is indeed the structure desired. This proves Lemma 4.23. ■

One may wonder why Lemma 4.22 (a) does not hold in the context of $\mathcal{M}^+_f$. As shown by Lemma 4.20, for any structure $G$ in $\mathcal{U}(\Delta_0)$, if $G \models \Sigma_1$, then $G$ has the form shown in Figure 3. Let $o_b$ be the node in $G$ such that $G \models b(r^G, o_b)$. The node $o_b$ represents a set, and for any $o \in |G|$, if $o \notin \{r^G, o_b\}$, then

$$G \models *(o_b, o).$$

That is, all the nodes of $G$ except $r^G$ and $o_b$ are elements of $o_b$. Therefore, if $G$ is infinite, then $o_b$ represents an infinite set. This is not allowed in $\mathcal{M}^+_f$. In other words, such structures cannot be in $\mathcal{U}(\Delta_0)$. However, this is not the case in the context of $\mathcal{M}^+$.

## 5 Implication problems for local extent constraints

In light of Theorems 4.1 and 4.2, one is tempted to think that adding structure will simplify reasoning about path constraints. However, this is not always the case. This section shows that a decidability result developed for semistructured data breaks down when a type of $\mathcal{M}^+$ or $\mathcal{M}_f^+$ is imposed on the data, by investigating the implication and finite implication problems for local extent constraints.

**Theorem 5.1:** In the context of semistructured data, the implication and finite implication problems for local extent constraints are decidable in PTIME. ■

**Theorem 5.2:** In the context of the object-oriented data model $\mathcal{M}^+$, the implication and finite implication problems for local extent constraints are undecidable. ■

**Theorem 5.3:** In the context of the object-oriented data model $\mathcal{M}_f^+$, the implication and finite implication problems for local extent constraints are undecidable. ■

It should be noted that the (finite) implication problem for local extent constraints is a special case of the (finite) implication problem for $P_c$. Therefore, as an immediate corollary of Theorem 4.2, we have:

**Corollary 5.4:** In the context of the object-oriented data model $\mathcal{M}$, the implication and finite implication problems for local extent constraints are decidable in cubic-time and finitely axiomatizable. ■

### 5.1 Decidability on untyped data

We first show Theorem 5.1. The idea of the proof is by reduction to word constraint implication. It has been shown in [4] that in the context of semistructured data, the implication and finite implication problems for $P_w$ are decidable in PTIME:

**Lemma 5.5 [4]:** In the context of semistructured data, the implication and finite implication problems for $P_w$ are decidable in PTIME. ■

We first define a function $f$ that is used in the further construction of the reduction. Let $\alpha$ be a path and $\varphi$ be a $P_c$ constraint. Then $f(\varphi, \alpha)$ is defined to be the $P_c$ constraint

- $\forall x\,(\alpha \cdot \rho(r, x) \to \forall y\,(\beta(x, y) \to \gamma(x, y)))$, if $\varphi$ is $\forall x\,(\rho(r, x) \to \forall y\,(\beta(x, y) \to \gamma(x, y)))$ (i.e., it is a forward constraint); or
- $\forall x\,(\alpha \cdot \rho(r, x) \to \forall y\,(\beta(x, y) \to \gamma(y, x)))$, if $\varphi$ is $\forall x\,(\rho(r, x) \to \forall y\,(\beta(x, y) \to \gamma(y, x)))$ (i.e., it is a backward constraint).

Recall the definition of the (finite) implication problem for local extent constraints from Definition 2.4. Let $\Sigma \cup \{\varphi\}$ be a finite subset of $P_c$ with prefix bounded by path $\alpha$ and binary relation symbol $K$, where $\varphi$ is also bounded by $\alpha$ and $K$. By Definition 2.3, $\Sigma$ can be partitioned into $\Sigma_K$ and $\Sigma_r$ such that

$$\begin{aligned} \Sigma_K &= \{\phi \mid \phi \in \Sigma,\ \phi \text{ is bounded by } \alpha \text{ and } K\}, \\ \Sigma_r &= \Sigma \setminus \Sigma_K. \end{aligned}$$

In addition, for each $\phi \in \Sigma_K \cup \{\varphi\}$, $\phi$ is a forward constraint and the prefix of $\phi$, $pf(\phi)$, is $\alpha \cdot K$. For each $\psi \in \Sigma_r$, $pf(\psi)$ is of the form $\alpha \cdot \alpha'$, where $\alpha'$ is a path such that $K \not\preceq_p \alpha'$, i.e., $K$ is not a prefix of $\alpha'$.

The reduction is defined in two steps. First, using $f$ and $\alpha$, we define a function $g_1$ such that for every $\phi \in \Sigma \cup \{\varphi\}$, $\phi = f(g_1(\phi),\ \alpha)$. That is, $g_1$ removes $\alpha$ from the prefix of $\phi$. Let

$$\begin{aligned} \varphi^1 &= g_1(\varphi), \\ \Sigma_K^1 &= \{g_1(\phi) \mid \phi \in \Sigma_K\}, \\ \Sigma_r^1 &= \{g_1(\psi) \mid \psi \in \Sigma_r\}. \end{aligned}$$

Second, using $f$ and $K$, we define another function $g_2$ such that for all $\phi \in \Sigma_K^1 \cup \{\varphi^1\}$, $\phi = f(g_2(\phi),\ K)$. That is, $g_2$ further removes $K$ from the prefix of $\phi$. Now let

$$\begin{aligned} \varphi^2 &= g_2(\varphi^1), \\ \Sigma_K^2 &= \{g_2(\phi) \mid \phi \in \Sigma_K^1\}. \end{aligned}$$

Clearly, $\Sigma_K^2 \subseteq P_w$ and $\varphi^2 \in P_w$. The functions $g_1$ and $g_2$ establish a reduction:

**Lemma 5.6:** In the context of semistructured data,

$$\Sigma \models \varphi \quad \text{iff} \quad \Sigma_K^1 \cup \Sigma_r^1 \models \varphi^1 \quad \text{iff} \quad \Sigma_K^2 \models \varphi^2, \tag{a}$$
$$\Sigma \models_f \varphi \quad \text{iff} \quad \Sigma_K^1 \cup \Sigma_r^1 \models_f \varphi^1 \quad \text{iff} \quad \Sigma_K^2 \models_f \varphi^2. \tag{b}$$

■

This lemma suffices to show Theorem 5.1. For if it holds, then the (finite) implication problem for local extent constraints is reduced to the (finite) implication problem for $P_w$. Note that given $\Sigma$ and $\varphi$, $\alpha$ and $K$ can be determined in linear-time. In addition, the functions $g_1$ and $g_2$ are computable in linear-time. Therefore, the PTIME decidability of the (finite) implication problem for local extent constraints follows from the PTIME decidability of the (finite) implication problem for $P_w$.

Finally, we prove Lemma 5.6.

**Proof of Lemma 5.6:** We show (b) only. The proof of (a) is similar and simpler. The proof consists of two parts.

*Part I:* We first show that $\Sigma \models_f \varphi$ iff $\Sigma_K^1 \cup \Sigma_r^1 \models_f \varphi^1$. It suffices to prove that for each path $\alpha$ and each finite subset $\Sigma \cup \{\varphi\}$ of $P_c$,

$$\bigwedge \Sigma \wedge \neg\varphi \text{ has a finite model} \quad \text{iff} \quad \bigwedge_{\phi \in \Sigma} f(\phi,\ \alpha) \wedge \neg f(\varphi,\ \alpha) \text{ has a finite model.}$$

To simplify the discussion, assume that all the constraints in $\Sigma \cup \{\varphi\}$ are of the forward form. The proof for the general case is similar.

(*if*) Suppose that $\bigwedge_{\phi \in \Sigma} f(\phi, \alpha) \wedge \neg f(\varphi, \alpha)$ has a finite model $G = (|G|, r^G, E^G)$. Then we construct a finite model of $\bigwedge \Sigma \wedge \neg\varphi$. That is, we construct a finite $\sigma$-structure $H$ such that $H \models \bigwedge \Sigma \wedge \neg\varphi$. Here $\sigma$ is the signature defined in Section 2.

Recall the notations $pf$, $lt$ and $rt$ described in Definition 2.1. Since $G \models \neg f(\varphi, \alpha)$, i.e.,

$$G \models \exists\, x\, y\ (\alpha \cdot pf(\varphi)(r^G, x) \wedge lt(\varphi)(x, y) \wedge \neg rt(\varphi)(x, y)),$$

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

$$G \models \alpha(r^G, a) \wedge pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

Let $m = max\{|pf(\phi)| + |lt(\phi)|,\ |pf(\phi)| + |rt(\phi)| \mid \phi \in \Sigma \cup \{\varphi\}\} + 1$. Then using $a$, $m$ and $G$, we define $H = (|H|, r^H, E^H)$ as follows.

- $|H| = \{o \mid o \in |G|,\ G \models \rho(a, o),\ \rho$ is a path and $|\rho| \leq m\}$.
- $r^H = a$.
- For every $K \in E$ and all $o, o' \in |H|$, $H \models K(o, o')$ iff $G \models K(o, o')$.

It is easy to verify the following.

(1) $H$ is finite. This is because $|G|$ is finite and $|H| \subseteq |G|$.

(2) $H \models \neg\varphi$.

Since $|pf(\varphi)| + |lt(\varphi)| < m$, we have that $b \in |H|$ and $c \in |H|$. Thus by the definition of $H$, we have

$$H \models pf(\varphi)(a, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

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

(3) $H \models \Sigma$.

We show this by *reductio*. Suppose that there exists $\phi \in \Sigma$, such that $H \models \neg\phi$. Then there exist $d, e \in |H|$, such that

$$H \models pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

Since $|pf(\phi)| + |lt(\phi)| < m$ and $|pf(\phi)| + |rt(\phi)| < m$, by the definition of $H$, we have that

$$G \models \alpha(r^G, a) \wedge pf(\phi)(a, d) \wedge lt(\phi)(d, e) \wedge \neg rt(\phi)(d, e).$$

That is, $G \models \neg f(\phi, \alpha)$. This contradicts the assumption that $G \models \{f(\phi, \alpha) \mid \phi \in \Sigma\}$.

(*only if*) Suppose that $\bigwedge \Sigma \wedge \neg\varphi$ has a finite model $G = (|G|, r^G, E^G)$. Then we show that $\bigwedge_{\phi \in \Sigma} f(\phi, \alpha) \wedge \neg f(\varphi, \alpha)$ has a finite model $H = (|H|, r^H, E^H)$, as follows. Let

$$L_\alpha = \{\rho \mid \rho \text{ is a path, } \rho \preceq_p \alpha,\ \rho \neq \alpha\}.$$

Here $\rho \preceq_p \alpha$ denote that $\rho$ is a prefix of $\alpha$, as described in Section 2. For each $\rho \in L_\alpha$, let $c_\rho$ be a distinguished node not in $|G|$. Let

- $|H| = |G| \cup \{c_\rho \mid \rho \in L_\alpha\}$;
- $r^H = c_\epsilon$;
- For all $a, b \in |H|$ and each $K \in E$, $H \models K(a, b)$ iff one of the following conditions is satisfied:
  - there exists $\rho \in L_\alpha$, such that $a = c_\rho$ and $b = c_{\rho\cdot K}$ and $\rho \cdot K \in L_\alpha$; or
  - there exists $\rho \in L_\alpha$, such that $\alpha = \rho \cdot K$ and $a = c_\rho$ and $b = r^G$; or
  - $a, b \in |G|$ and $G \models K(a, b)$.

It is easy to verify the following.

(1) $H$ is finite. This is because both $|G|$ and $L_\alpha$ are finite.

(2) $H \models \neg f(\varphi, \alpha)$.

To show this, we first observe the following simple facts, which are immediate from the construction of $H$.

*Fact 1:* $r^G$ is the unique node in $|H|$ such that $H \models \alpha(c_\epsilon, r^G)$. In addition, for all $\rho, \varrho \in L_\alpha$ and $K \in E$, $H \models K(c_\rho, c_\varrho)$ iff $\varrho = \rho \cdot K$.

*Fact 2:* For each $\rho \in L_\alpha$, $K \in E$,

- for each $a \in |G| \setminus \{r^G\}$, we have $H \models \neg K(c_\rho, a) \wedge \neg K(a, c_\rho)$;
- if $\alpha \neq \rho \cdot K$, then $H \models \neg K(c_\rho, r^G) \wedge \neg K(r^G, c_\rho)$;
- if $\alpha = \rho \cdot K$, then $H \models K(c_\rho, r^G) \wedge \neg K(r^G, c_\rho)$.

Using these facts, we show that $H \models \neg f(\varphi, \alpha)$. By $G \models \neg\varphi$, there exist $b, c \in |G|$, such that

$$G \models pf(\varphi)(r^G, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

By Facts 1 and 2,

$$H \models \alpha(c_\epsilon, r^G) \wedge pf(\varphi)(r^G, b) \wedge lt(\varphi)(b, c) \wedge \neg rt(\varphi)(b, c).$$

That is, $H \models \neg f(\varphi,\, \alpha)$.

(3) $H \models \{f(\phi,\, \alpha) \mid \phi \in \Sigma\}$.

Suppose for *reductio* that there exists $\phi \in \Sigma$ such that $H \models \neg f(\phi,\, \alpha)$. Then there exist $a, b, c \in |H|$, such that

$$H \models \alpha(c_\epsilon,\, a) \wedge pf(\phi)(a,\, b) \wedge lt(\phi)(b,\, c) \wedge \neg rt(\phi)(b,\, c).$$

By Fact 1, $a = r^G$. By Fact 2, $b, c \in |G|$, and moreover, by the construction of $H$, we have

$$G \models pf(\phi)(a,\, b) \wedge lt(\phi)(b,\, c) \wedge \neg rt(\phi)(b,\, c).$$

That is, $G \models \neg\phi$. This contradicts the assumption that $G \models \Sigma$.

*Part II:* We next show that $\Sigma_K^1 \cup \Sigma_r^1 \models_f \varphi^1$ iff $\Sigma_K^2 \models_f \varphi^2$.

The argument of Part I suffices to show that if $\Sigma_K^2 \models_f \varphi^2$ then $\Sigma_K^1 \models_f \varphi^1$. Therefore, $\Sigma_K^1 \cup \Sigma_r^1 \models_f \varphi^1$.

Conversely, suppose that $\bigwedge \Sigma_K^2 \wedge \neg\varphi^2$ has a finite model $G = (|G|,\, r^G,\, E^G)$. Then we construct from $G$ a finite model $H$ of $\bigwedge \Sigma_K^1 \wedge \bigwedge \Sigma_r^1 \wedge \neg\varphi^1$.

Let $H = (|H|,\, r^H,\, E^H)$, where

- $|H| = |G| \cup \{r^H\}$ and $r^H \notin |G|$;
- $E^G$ is $E^G$ augmented with two edges labeled $K$, by letting $H \models K(r^H,\, r^G) \wedge K(r^H,\, r^H)$. In other words, $E^H = E^G \cup \{K(r^H,\, r^G),\; K(r^H,\, r^H)\}$.

The structure $H$ is shown in Figure 4. Clearly, $H$ is finite since $G$ is finite.

Below we show that $H$ is a model of $\bigwedge \Sigma_K^1 \wedge \bigwedge \Sigma_r^1 \wedge \neg\varphi^1$.

(1) $H \models \Sigma_r^1$.

For each $\phi \in \Sigma_r^1$, by Definition 2.3, $K$ is not a prefix of $pf(\phi)$.

If $pf(\phi) \neq \epsilon$, then by the construction of $H$, there are no $o, o' \in |H|$, such that

$$H \models pf(\phi)(r^H,\, o) \wedge lt(\phi)(o,\, o').$$

Therefore, $H \models \phi$.

If $pf(\phi) = \epsilon$, then by Definition 2.3, $\phi$ is

$$\forall x\, (\epsilon(r, x) \to K(r, x)).$$

Clearly, $H \models \phi$ since $(r^H, r^H) \in K^H$.

Therefore, $H \models \Sigma_r^1$.

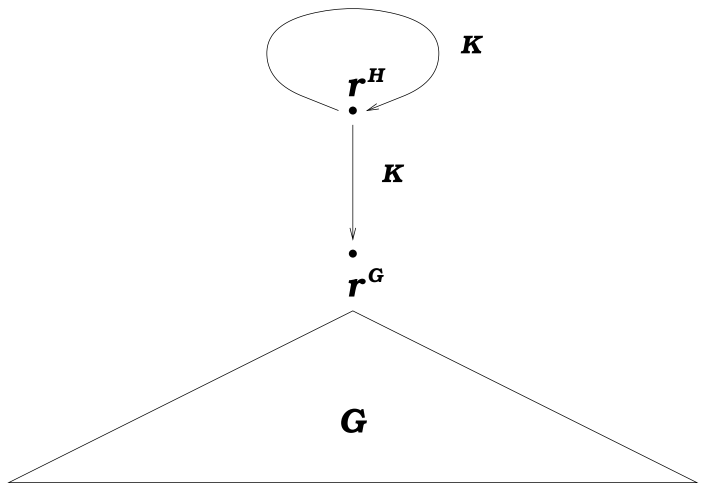

Figure 4: The structure $H$ in the proof of Lemma 5.6

(2) $H \models \Sigma^1_K$.

Suppose, for *reductio*, that there is $\phi \in \Sigma^1_K$ such that $H \models \neg\phi$. Since $\phi$ is bounded by $K$, by Definition 2.3, $\phi$ is of the form

$$\forall x\,(K(r,\,x) \to \forall y\,(\beta(x,\,y) \to \gamma(x,\,y))),$$

where $K \not\preceq_p \beta$ and $\beta \neq \epsilon$. Thus by the definition of $H$, if $H \models \neg\phi$, then there exists $o \in |G|$, such that

$$H \models K(r^H,\,r^G) \wedge \beta(r^G,\,o) \wedge \neg\gamma(r^G,\,o).$$

In addition, we have

$$G \models \beta(r^G,\,o) \wedge \neg\gamma(r^G,\,o).$$

That is, $G \models \neg g_2(\phi)$. This contracts the assumption that $G$ is a model of $\bigwedge \Sigma^2_K \wedge \neg\varphi^2$. Therefore, $H \models \phi$. Hence $H \models \Sigma^1_K$.

(3) $H \models \neg\varphi^1$.

Note that $\varphi^1$ is bounded by $K$. Hence $\varphi^1$ is of the form

$$\forall x\,(K(r,\,x) \to \forall y\,(\beta(x,\,y) \to \gamma(x,\,y))),$$

where $K \not\preceq_p \beta$ and $\beta \neq \epsilon$. In addition, $\varphi^2$ is of the form $\forall x\,(\beta(r,\,x) \to \gamma(r,\,x))$. By $G \models \neg\varphi^2$, there is $o \in |G|$, such that $G \models \beta(r^G,\,o) \wedge \neg\gamma(r^G,\,o)$. By the definition of $H$,

$$H \models K(r^H,\,r^G) \wedge \beta(r^G,\,o) \wedge \neg\gamma(r^G,\,o).$$

That is, $H \models \neg\varphi^1$.

This completes the proof of Lemma 5.6. ■

The following should be noted.

(1) The proof above is not applicable in the context of $\mathcal{M}$, $\mathcal{M}^+$ or $\mathcal{M}^+_f$. More specifically, for any schema $\Delta$ in these models, the structure $H$ shown in Figure 4 is not in $\mathcal{U}(\Delta)$, due to the type constraint $\Phi(\Delta)$.

(2) Theorem 5.1 does not conflict with the proof of Theorem 4.1. Recall $\Sigma_1$, $\Sigma_2$, $\varphi_{(\alpha,\beta)}$ and $\varphi_{(\beta,\alpha)}$ defined in Section 4.1. The set $\Sigma_1 \cup \Sigma_2 \cup \{\varphi_{(\alpha,\beta)}, \varphi_{(\beta,\alpha)}\}$ is not a prefix bounded subset of $P_c$.

## 5.2 The breakdown of the decidability in $\mathcal{M}^+$

Next, we show that the decidability result established above breaks down in the context of $\mathcal{M}^+$. More specifically, we prove Theorem 5.2 by reduction from the word problem for (finite) monoids.

Recall the alphabet $\Gamma_0$ described in Section 4.1. Using $\Gamma_0$, we define a schema $\Delta_1$ in $\mathcal{M}^+$ as follows:

$$\Delta_1 = (\mathcal{C},\ \nu,\ DBtype),$$

where

- $\mathcal{C} = \{C, C_s, C_l\}$,
- $\nu$ is defined by:

$$\begin{array}{lcl} C & \mapsto & [l_1 : C,\ \ldots,\ l_m : C] \\ C_s & \mapsto & \{C\} \\ C_l & \mapsto & [a : C,\ b : C_s,\ K : C_l] \end{array}$$

  where $a, b, l, K \notin \Gamma_0$.
- $DBtype = [l : C_l]$.

Note here that each letter in $\Gamma_0$ is a record label of $C$, and thus is in $E(\Delta_1)$. Hence every $\alpha \in \Gamma_0^*$ can be represented as a path formula, also denoted $\alpha$.

Next, we encode equations over $\Gamma_0$. Recall the finite set $\Theta_0$ of equations described in Section 4.1. We encode $\Theta_0$ in terms of a finite set $\Sigma$, which consists of the following constraints of $P_c$:

1. $\forall x\, (l \cdot K(r,x) \to \forall y\, (a(x,y) \to b \cdot *(x,y)))$;
2. for each $j \in [1,m]$, $\forall x\, (l \cdot K(r,x) \to \forall y\, (b \cdot * \cdot l_j(x,y) \to b \cdot *(x,y)))$;
3. for each $(\alpha_i,\ \beta_i) \in \Theta_0$, $\forall x\, (l \cdot b \cdot *(r,x) \to \forall y\, (\alpha_i(x,y) \to \beta_i(x,y)))$;
4. $\forall x\, (l(r,x) \to \forall y\, (\epsilon(x,y) \to K(x,y)))$.

We encode a test equation $(\alpha,\ \beta)$ over $\Gamma_0$ by the constraint

$$\varphi_{(\alpha,\beta)} = \forall\, x\,(l \cdot K(r,x) \to \forall\, y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y))).$$

By Definition 2.3, it is easy to see that $\Sigma \cup \{\varphi_{(\alpha,\beta)}\}$ is a subset of $P_c$ with prefix bounded by $l$ and $K$. More specifically, this set can be partitioned into $\Sigma_r$ and $\Sigma_K$:

- $\Sigma_K$ consists of $\varphi_{(\alpha,\beta)}$ as well as those defined in (1) and (2). All of these constraints are bounded by $l$ and $K$.
- $\Sigma_r$ consists of the constraints specified in (3) and (4), which are not bounded by $l$ and $K$. In addition, for any $\phi \in \Sigma_r$, the prefix of $\phi$, $pf(\phi)$, is either $l \cdot b \cdot *$ or $l$. In particular, if $pf(\phi) = l$, then $\phi$ is the constraint given in (4).

We reduce the word problem for monoids to the problem of determining whether

$$\Sigma \models_{\Delta_1} \varphi_{(\alpha,\beta)},$$

and analogously, reduce the word problem for finite monoids to the problem of determining whether

$$\Sigma \models_{(f,\Delta_1)} \varphi_{(\alpha,\beta)}.$$

As in Section 4.3, it is easy to verify the following lemmas. These lemmas hold in the context of $\mathcal{M}^+$ due to the type constraint $\Phi(\Delta_1)$. In the context of semistructured data, however, these lemmas no longer hold in general.

**Lemma 5.7:** For each $G \in \mathcal{U}(\Delta_1)$, $G$ has the following properties.

1. There is a unique node $o \in |G|$, such that $G \models l(r^G,\ o)$. This node is denoted by $o_l$.
2. There exists a unique node $o_K \in |G|$, such that $G \models l \cdot K(r^G,\ o_K)$. In addition, if $G \models \forall\, x\,(l(r,x) \to \forall\, y\,(\epsilon(x,y) \to K(x,y)))$, then $o_K = o_l$.
3. For every $\alpha \in \Gamma_0^*$, the following statements hold.
   (a) There is a unique $o \in |G|$ such that $G \models a \cdot \alpha(o_l, o)$. This node is denoted by $o_\alpha$.
   (b) For every $o \in |G|$, if $G \models C^G(o)$, then there is a unique $o' \in |G|$, such that $G \models \alpha(o, o')$.

■

**Lemma 5.8:** For every $G \in \mathcal{U}(\Delta_1)$ and all $\alpha, \beta \in \Gamma_0^*$, if

$$G \models \forall\, x\,(l \cdot b \cdot *(r,x) \to \forall\, y\,(\alpha(x,y) \to \beta(x,y))),$$

then

$$G \models \forall\, x\,(l \cdot b \cdot *(r,x) \to \forall\, y\,(\beta(x,y) \to \alpha(x,y))).$$

Similarly, if

$$G \models \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y))),$$

then

$$G \models \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \beta(x,y) \to a \cdot \alpha(x,y))).$$

■

**Lemma 5.9:** For every $G \in \mathcal{U}(\Delta_1)$, if $G \models \Sigma$, then for every $\alpha \in \Gamma_0^*$, we have

$$G \models l \cdot b \cdot *(r^G,\, o_\alpha),$$

where $o_\alpha$ is the unique node in $|G|$ such that $G \models l \cdot a \cdot \alpha(r^G,\, o_\alpha)$, as specified in Lemma 5.7. In addition, for every $o, o' \in |G|$ such that $G \models l \cdot b \cdot *(r^G,\, o') \wedge \alpha(o',\, o)$, we have

$$G \models l \cdot b \cdot *(r^G,\, o).$$

■

**Lemma 5.10:** For every $G \in \mathcal{U}(\Delta_1)$, if $G \models \Sigma$ and

$$G \models \forall x\,(l \cdot b \cdot *(r,x) \to \forall y\,(\alpha(x,y) \to \beta(x,y)))$$

for some $\alpha, \beta \in \Gamma_0^*$, then

$$G \models \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y))).$$

■

Next, we show that the encoding given above is indeed a reduction from the word problem for (finite) monoids.

**Lemma 5.11:** In the context of $\mathcal{M}^+$, for all $\alpha$ and $\beta$ in $\Gamma_0^*$,

$$\Theta_0 \models (\alpha,\, \beta) \;\text{ iff }\; \Sigma \models_{\Delta_1} \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y))), \tag{a}$$

$$\Theta_0 \models_f (\alpha,\, \beta) \;\text{ iff }\; \Sigma \models_{(f,\,\Delta_1)} \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y))). \tag{b}$$

■

**Proof:** The proof is similar to that of Lemma 4.22. We prove (b) only. The proof of (a) is similar and simpler.

(*if*) Suppose that $\Theta_0 \not\models_f (\alpha,\, \beta)$. Then we construct $G \in \mathcal{U}_f(\Delta_1)$, such that $G \models \Sigma$, and $G \not\models \forall x\,(l \cdot K(r,x) \to \forall y\,(a \cdot \alpha(x,y) \to a \cdot \beta(x,y)))$.

By $\Theta_0 \not\models_f (\alpha,\, \beta)$, there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : \Gamma_0^* \to M$ such that for any $i \in [1,n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$. Using $M$ and $h$, we define an equivalence relation $\approx$ in the same way as in the proof of Lemma 4.12. In addition, for each $\rho \in \Gamma_0^*$, let $\widehat{\rho}$ be defined as in the proof of Lemma 4.12. Similarly, we define $C_{\Theta_0}$.

Using $C_{\Theta_0}$, we define $G = (|G|,\, r^G,\, E^G,\, T^G)$ as follows.

(1) $|G|$.

For each $\rho \in \Gamma_0^*$, let $o(\widehat{\rho})$, $o_r$ and $o_b$ be defined as in the proof of Lemma 4.22. In addition, let $o_l$ be a distinct node. Then we define

$$|G| = \{o_r, o_b, o_l\} \cup \{o(\widehat{\rho}) \mid \widehat{\rho} \in C_{\Theta_0}\}.$$

(2) $r^G = o_r$.

(3) The unary relations $C^G$, ${C_s}^G$ and $DBtype^G$ are defined in the same way as in the proof of Lemma 4.22. In addition, let ${C_l}^G = \{o_l\}$.

(4) The binary relations are populated as follows.

- $G \models l(o_r,\, o_l)$.
- $G \models K(o_l,\, o_l)$.
- $G \models a(o_l, o(\widehat{\epsilon}))$.
- $G \models b(o_l, o_b)$.
- For each $\widehat{\rho} \in C_{\Theta_0}$, let $G \models *(o_b,\, o(\widehat{\rho}))$.
  In addition, for each $j \in [1, m]$, let $G \models l_j(o(\widehat{\rho}),\, o(\widehat{\rho \cdot l_j}))$.

The structure $G$ is shown in Figure 5. It should be noted that since $o_l \in {C_l}^G$, it is valid to have $G \models K(o_l,\, o_l)$. However, since $\nu(C_l)$ is a record type, there is no $o \in |G|$ such that $o \neq o_l$ and $G \models K(o_l,\, o_l) \wedge K(o_l,\, o)$.

As in proof of Lemma 4.22, it is easy to verify that

- $G \in \mathcal{U}_f(\Delta_1)$,
- $G \models \Sigma$, and
- $G \not\models \forall x\, (l \cdot K(r, x) \rightarrow \forall y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y)))$.

(*only if*) Suppose that there is $G \in \mathcal{U}_f(\Delta_1)$, such that $G \models \Sigma$, but

$$G \not\models \forall x\, (l \cdot K(r, x) \rightarrow \forall y\, (a \cdot \alpha(x, y) \rightarrow a \cdot \beta(x, y))).$$

Then we show that $\Theta_0 \not\models_f (\alpha,\, \beta)$. That is, we show that there exist a finite monoid $(M,\, \circ,\, 1)$ and a homomorphism $h : \Gamma_0^* \rightarrow M$ such that for any $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$, but $h(\alpha) \neq h(\beta)$.

To do this, we define another equivalence relation on $\Gamma_0^*$ as follows:

$$\rho \sim \varrho \quad \text{iff} \quad G \models \forall x\, (l \cdot b \cdot *(r, x) \rightarrow \forall y\, (\rho(x, y) \rightarrow \varrho(x, y))).$$

As in the proof of Lemma 4.22, it can be shown that for any $i \in [1, n]$, $\alpha_i \sim \beta_i$. In addition, $\alpha \not\sim \beta$.

As in the proof of Lemma 4.22, we define $[\rho]$ for every $\rho \in \Gamma_0^*$, and using the notion of $[\rho]$, define $M$ and $\circ$. In addition, as in the proof of Lemma 4.22, the following can be verified.

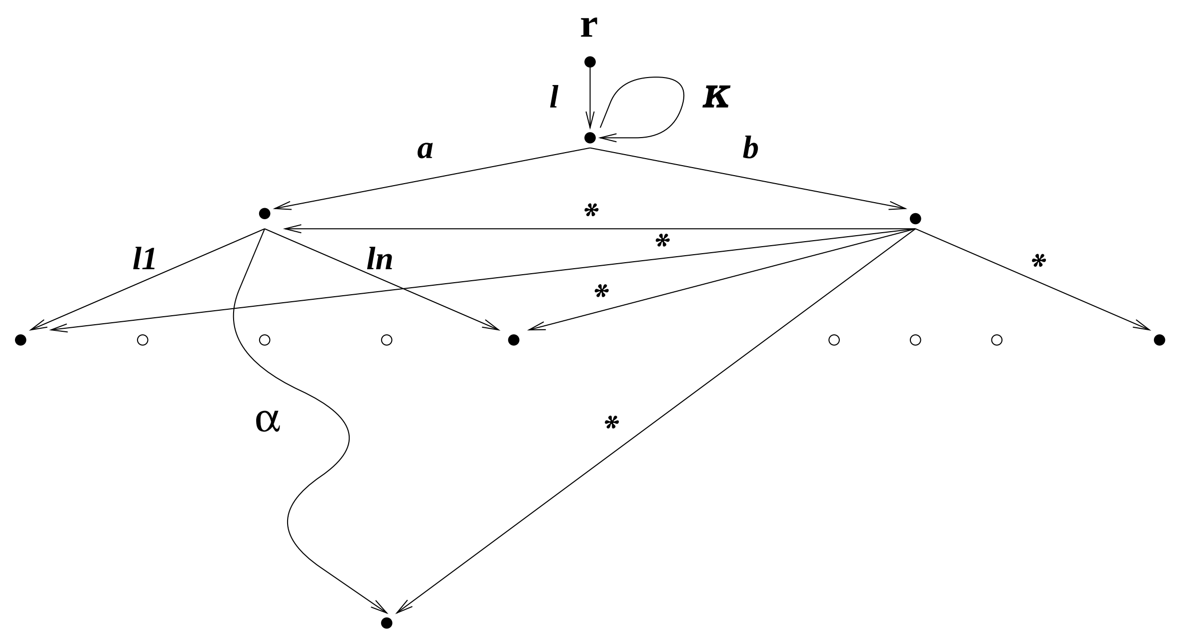

Figure 5: The structure $G$ in the proof of Lemma 5.11

- $M$ is finite.
- $\circ$ is well-defined.
- $\circ$ is associative.
- $[\epsilon]$ is the identity for $\circ$.

Therefore, $(M, \circ, [\epsilon])$ is a finite monoid.

We define $h : \Gamma_0^* \to M$ by

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

Moreover, as in the proof of Lemma 4.22, the following can be verified.

- $h$ is a homomorphism.
- For every $i \in [1, n]$, $h(\alpha_i) = h(\beta_i)$.
- $h(\alpha) \neq h(\beta)$.

Therefore,

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

This completes the proof of Lemma 5.11. ∎

## 5.3 The breakdown of the decidability in $\mathcal{M}_f^+$

Finally, we show Theorem 5.3. That is, when the type system $\mathcal{M}_f^+$ is imposed, Theorem 5.1 also breaks down. In other words, the implication and finite implication problems for local extent constraints also become undecidable in the context of $\mathcal{M}_f^+$.

To see that the finite implication problem for local extent constraints is undecidable in the context of $\mathcal{M}_f^+$, note that the schema $\Delta_1$ defined in Section 5.2 is also a schema of $\mathcal{M}_f^+$. Therefore, the proof of Theorem 5.2 can be used to establish this undecidability.

To show that the implication problem for local extent constraints is also undecidable in the context of $\mathcal{M}_f^+$, the following lemma suffices. The proof of this lemma is similar to that of Lemma 4.23.

**Lemma 5.12:** Let $\Delta_1$ be the schema defined in Section 5.2. In the context of $\mathcal{M}_f^+$, for every finite subset $\Sigma \cup \{\varphi\}$ of $P_c(\Delta_1)$, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta_1)$, then it has a model in $\mathcal{U}_f(\Delta_1)$. ■

By this lemma, in the context of $\mathcal{M}_f^+$, the implication problem and the finite implication problem for local extent constraints over $\Delta_1$ coincide. Therefore, the undecidability of the implication problem for local extent constraints follows from the undecidability of the finite implication problem for local extent constraints.

# 6 Conclusion

Two forms of constraints have been proposed separately for specifying semantics of XML data, namely, type constraints [6, 15, 18] and path constraints [4, 9, 10, 11]. In this paper, we have investigated their interaction. We have demonstrated that adding a type system may in some cases simplify the analysis of path constraint implication, and in other cases make it harder. More specifically, we have studied how $P_c$ constraints introduced in [9, 10, 11] interact with three types systems: $\mathcal{M}^+$, $\mathcal{M}_f^+$ and $\mathcal{M}$. The type system $\mathcal{M}^+$ is an object-oriented model similar to those studied in [2, 3, 12, 16]. It supports classes, records and sets. The type system $\mathcal{M}_f^+$ is the same as $\mathcal{M}^+$ except that it supports finite sets instead of sets. The model $\mathcal{M}$ is a restriction of $\mathcal{M}^+$. Two dozen results have been established in the paper on the interaction between $P_c$ constraints with these type systems (Table 1). In particular, we have shown that the implication and finite implication problems for $P_c$ are undecidable in the context of semistructured data, but become not only decidable in cubic-time but also finitely axiomatizable when a type of $\mathcal{M}$ is added. In addition, we have also shown that the implication and finite implication problems for local extent constraints, which constitute a fragment of $P_c$, are decidable in PTIME in the untyped context. However, when a type of $\mathcal{M}^+$ or $\mathcal{M}_f^+$ is imposed, these problems become undecidable. These results show that in some cases, decidability results established for untyped data break down when a type system is imposed, and in other cases, the reverse can also occur.

One may want to study the interaction between path constraints and more general type systems such as those studied in [6, 15, 18]. However, by Theorems 4.5 and 5.2, anything more complex is almost certain to lead to undecidable results. An extension to this work would be a study of other forms of practical integrity constraints and their interaction with these type systems.

Another significant extension would be a design of path constraint syntax that is conformable with XML and XML DTD [7]. For example, consider the following $P_c$ constraints given in Section 1:

$$\forall x\,(book \cdot author(r,\, x) \rightarrow person(r,\, x))$$
$$\forall x\,(person \cdot wrote(r,\, x) \rightarrow book(r,\, x))$$

$$\forall x\,(book(r,\, x) \rightarrow \forall y\,(author(x,\, y) \rightarrow wrote(y, x)))$$
$$\forall x\,(person(r,\, x) \rightarrow \forall y\,(wrote(x,\, y) \rightarrow author(y,\, x)))$$

In XML syntax, these constraints may be expressed as:

```
<constraint>
     <inclusion path = "book.author"  memberOf ="person" />
</constraint>

<constraint>
     <inclusion path = "person.wrote"  memberOf ="book" />
</constraint>

<constraint>
     <prefix      path = "book" />
     <inverse     path = "author"  inverseOf = "wrote"/>
</constraint>

<constraint>
     <prefix      path = "person" />
     <inverse     path = "wrote"  inverseOf = "author"/>
</constraint>
```

One should be able to describe in this syntax external links such as those studied in [19]. To accomplish this, much more remains to be done.

**Acknowledgements.** We thank Leonid Libkin, Val Tannen and Victor Vianu for valuable 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 and P. C. Kanellakis. "Object identity as a query primitive". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 159-173, 1989.

[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] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Springer, 1997.

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

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

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

[9] P. Buneman, W. Fan, and S. Weinstein. "Some undecidable implication problems for path constraints". Technical Report MS-CIS-97-14, Department of Computer and Information Science, University of Pennsylvania, 1997. Available from `http://www.cis.upenn.edu/~db/langs/97papers.html`.

[10] P. Buneman, W. Fan, and S. Weinstein. "The decidability of some restricted implication problems for path constraints". Technical Report MS-CIS-97-15, Department of Computer and Information Science, University of Pennsylvania, 1997. Available from `http://www.cis.upenn.edu/ ~db/langs/97papers.html`.

[11] 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 also in `http://www.cis.upenn.edu/~db/langs/98papers.html`.

[12] R. G. G. Cattell (ed.). *The object-oriented standard: ODMG-93* (Release 1.2). Morgan Kaufmann, San Mateo, California, 1996.

[13] 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`.

[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] Anthony Kosky. *Transforming databases with recursive data structures*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[17] 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`.

[18] 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. See `http://www.w3.org/TR/1998/NOTE-XML-data`.

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

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