Path Constraints in the Presence of Types

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

October 1997

Abstract

Path constraints have been studied in [3, 8, 9] for semi-structured data. In this paper, we investigate path constraints for structured data. We show that there is interaction between path constraints and type constraints. In other words, results on path constraint implication in semistructured databases may no longer hold in the presence of types. We also investigate the class of word constraints for databases of two practical object-oriented data models. In particular, we present an abstraction of the databases in these models in terms of first-order logic, and establish the decidability of word constraint implication in these models.

1 Introduction

Path constraints and their associated implication problems have been studied in [3, 8, 9] for semistructured data. In these papers, semistructured data is represented as a rooted edge-labeled directed graph, as in other semistructured data models (e.g., OEM [18, 2] and UnQL [7]. See [1] for a survey). Specifically, [8, 9] model semistructured databases as (finite) first-order logic structures of the signature

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

Here r is a constant and E is a finite set of binary relation symbols, which denote the root node and the edge labels in the graph representation of a database, respectively. For example, the graph in Figure 1, which is taken from [9], depicts a school database represented by a structure of the signature

$$(r,\ \{Students, Courses, Taking, Enrolled, Name, CName\}).$$

In this graph model, a path, i.e., a sequence of edge labels, can be represented as a first-order logic formula $\alpha(x, y)$, where x and y indicate the tail and head nodes of the path, respectively. The path constraint language investigated in [8, 9], P, is the class of all the logic formulas of either the form

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

or the form

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

[*]This work was partly supported by the Army Research Office (DAAH04-95-1-0169) and NSF Grant CCR92-16122.
[†]Supported by an IRCS graduate fellowship.
[‡]Supported by NSF Grant CCR-9403447.

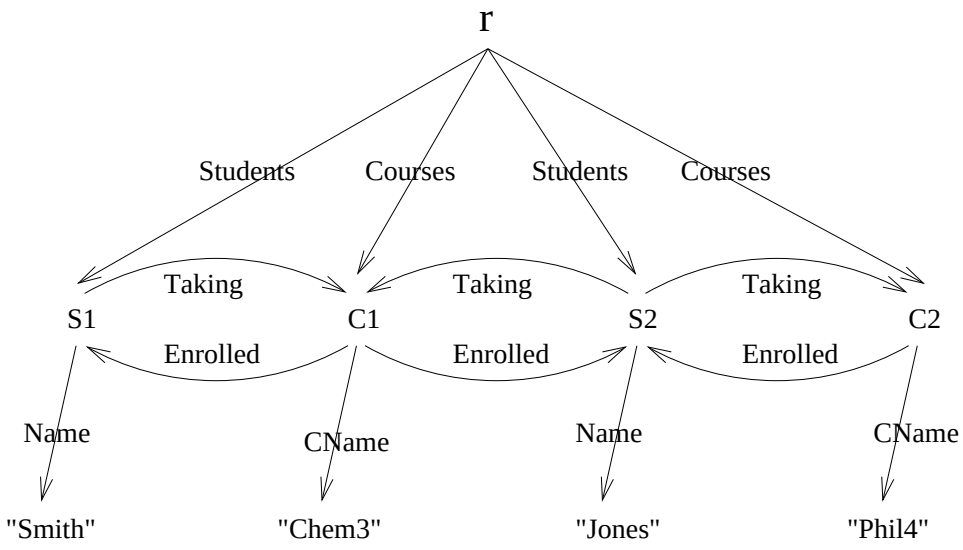

Figure 1: Representation of a school database

where α, β, γ are paths, r is the constant mentioned above, and x, y are variables. A proper subclass of P, called *word constraints*, was introduced and investigated in [3]. A word constraint can be represented as

$$\forall\, y\, (\beta(r, y) \rightarrow \gamma(r, y)),$$

where β and γ are paths. As an example, consider the path constraints below, which are taken from [9]. They are constraints of P for the database depicted in Figure 1.

Extent Constraints. The constraints

$$\forall\, c\, (\exists\, s\, (Students(r, s) \wedge Taking(s, c)) \rightarrow Courses(r, c))$$
$$\forall\, s\, (\exists\, c\, (Courses(r, c) \wedge Enrolled(c, s)) \rightarrow Students(r, s))$$

are examples of word constraints, which state that any course taken by a student must be a course that occurs in the database "extent" of courses, and any student enrolled in a course must be a student that similarly occurs in the database.

Inverse Constraints. The inverse relationship between `Taking` and `Enrolled` is expressed as:

$$\forall\, s\, c\, (Students(r, s) \wedge Taking(s, c) \rightarrow Enrolled(c, s))$$
$$\forall\, c\, s\, (Courses(r, c) \wedge Enrolled(c, s) \rightarrow Taking(s, c))$$

Such constraints are common in object-oriented databases [10].

The ability to reason about path constraints is useful for optimizing query evaluation and for adding structure to semistructured data (see [6, 16, 17] on this subject). In the context of semistructured data, a number of results on path constraint implication have been established. In [8], it is shown that the implication problems for P are undecidable. However, [9] identifies several fragments of P, and shows that each of these fragments properly contains the set of word constraints and possesses decidable implication problems. In [3], it is shown that the implication problems for word constraints are decidable in PTIME.

In the same spirit of [4, 11], the graph data model discussed above can also be used to represent structured data, by which we mean data constrained by a schema. Similarly, path constraints can also be defined for structured data.

There are good reasons for wanting to study path constraints and their associated implication problems for structured data. First, many referential integrity constraints can be expressed as path

constraints. For structured data, checking and maintaining these referential integrity constraints are central to performing updates, optimizing queries and loading databases. Second, these referential integrity constraints also play an important role in database integration and transformation [15, 8]. Third, some fundamental semantic relations commonly found in object-oriented databases can be captured by path constraints. Including these constraints in new data models helps incorporate object-oriented features into these models.

In this paper, we consider the implication problems for path constraints in the context of structured data. What is the difference between path constraint implication in the context of semistructured data as opposed to structured data? In structured databases, path constraint implication is restricted by a schema. More specifically, the implication problem for path constraints over a schema Δ is the problem of determining, given a finite set $\Sigma \cup \{\varphi\}$ of path constraints, whether all the database instances of Δ that satisfy Σ are also models of φ. Here an instance of the schema Δ has a particular structure specified by Δ. In other words, an instance of Δ must satisfy certain type constraints imposed by Δ. In contrast, a semistructured database is free of type constraints.

Here we address the question whether there is interaction between type constraints and path constraints. We show that some results on path constraint implication in semistructured databases no longer hold in the presence of types. For example, consider the implication problems for the path constraint language P described above. In semistructured databases, as established by [8], the implication problems are undecidable. In the typed context, however, the implication problem for P over a schema is decidable as long as the schema does not contain recursive types, i.e., self-referential data structures. This is because in any instance of such a schema, there are only finitely many navigation paths. In other words, the language P over the schema has only finitely many sentences up to equivalence, and therefore, its associated implication problem is decidable.

As another example to illustrate the impact of type constraints, consider the implication problems for word constraint introduced in [3]. A proof of the decidability of word constraint implication in semistructured databases was also presented there. However, we will show that this proof breaks down in the context of an object-oriented data model.

Because of the interaction between type constraints and path constraints, there is need for investigating path constraint implication in the presence of types. In this paper, we focus on the class of word constraints, which is properly contained in every fragment of P studied in [9] that possesses decidable implication problems. We investigate the class of word constraints for databases in two practical object-oriented data models. One of the models has a "generic" type system. The other is an object-oriented model based on ACeDB [19] which, while it is often considered a semistructured model [1, 7], has in fact a separate type system that allows more flexibility than object-oriented types, and is popular with biologists. In the next two sections, we present an abstraction of databases in these models in terms of first-order logic, and establish the decidability of word constraint implication in these models.

2 Word Constraints in a Generic Object-Oriented Model

In this section, we investigate word constraint implication in an object-oriented data model. We first describe the data model, and present an abstraction of the databases in the model in terms of first-order logic. We then formally define word constraints in the model. Finally, we show that in the context of this model, the proof of the decidability of word constraint implication given in [3] breaks down. However, we establish several decidability results on word constraint implication in this context.

2.1 An object-oriented model

We begin with the definitions of database schemas and their instances, and continue with an abstraction of database instances.

The data model

Assume a fixed countable set of labels, $\mathcal{L}$, and a fixed finite set of *base types*, $\mathcal{B}$.

Definition 2.1: Let $\mathcal{C}$ be some finite set of *classes*. The set of *Types over* $\mathcal{C}$, $Types^{\mathcal{C}}$, is defined by the syntax:

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

where $b \in \mathcal{B}$, $C \in \mathcal{C}$, and $l_i \in \mathcal{L}$. The notations $\{t\}$ and $[l_1 : t_1, \ldots, l_n : t_n]$ represent *set type* and *record type*, respectively. We reserve τ to range over $Types^{\mathcal{C}}$. ∎

Definition 2.2: A *schema* is a triple $\Delta = (\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ is a finite set of classes,
- ν is a mapping: $\mathcal{C} \to Types^{\mathcal{C}}$ 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 2.1: An example schema is $(\mathcal{C}, \nu, DBtype)$, where

- $\mathcal{C}$ consists of a single class $Person$,
- ν maps $Person$ to a record type $[name : string, spouse : Person]$, and
- $DBtype$ is $\{Person\}$. ∎

Definition 2.3: A *database instance* of schema $(\mathcal{C}, \nu, DBtype)$ is a triple $I = (\pi, \mu, d)$, where

- π is an *oid assignment* that maps each $C \in \mathcal{C}$ to a finite set of oids, $\pi(C)$, such that for all $C, C' \in \mathcal{C}$,
$$\pi(C) \cap \pi(C') = \emptyset \text{ if } C \neq C';$$
- for each $C \in \mathcal{C}$, μ 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 base 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 Δ by $\mathcal{I}(\Delta)$. ■

Example 2.2: An instance of the schema given in Example 2.1 is $(\pi,\ \mu,\ d)$, where

- $\pi(Person) = \{p_1, p_2, p_3, p_4\}$,
- $\mu : \pi(Person) \rightarrow [\![[name : string,\ spouse : Person]]\!]_\pi$ is defined by:

$$\begin{aligned} \mu(p_1) &\mapsto [name : \text{“Smith”},\ spouse : p_2] \\ \mu(p_2) &\mapsto [name : \text{“Mary”},\ spouse : p_1] \\ \mu(p_3) &\mapsto [name : \text{“Joe”},\ spouse : p_4] \\ \mu(p_4) &\mapsto [name : \text{“Maria”},\ spouse : p_3] \end{aligned}$$

- $d = \{p_1, p_2, p_3, p_4\}$. ■

Abstraction of databases

We next present an abstraction of databases in the object-oriented model. Since structured data can be viewed as semistructured data further constrained by a schema, along the same lines of the abstraction of semistructured databases described in the last section, we represent a structured database as a first-order logic structure satisfying certain type constraint determined by its schema. Such a structure can also be depicted as an edge-labeled rooted directed graph.

We assume the standard notations used in first-order logic [12].

We first define the first-order signature determined by a schema. Two components of the signature are described as follows.

Definition 2.4: Given a schema $\Delta = (\mathcal{C},\ \nu,\ DBtype)$, we define *the set of binary relation symbols* and *the set of types determined by* Δ, denoted $E(\Delta)$ and $T(\Delta)$, respectively, to be the smallest sets having the following properties:

- $DBtype \in T(\Delta)$ and $\mathcal{C} \subseteq T(\Delta)$;
- if $DBtype = \{t\}$ (or for some $C \in \mathcal{C}$, $\nu(C) = \{t\}$), then t is in $T(\Delta)$ and $*$ is in $E(\Delta)$;
- if $DBtype = [l_1 : t_1,\ \ldots,\ l_n : t_n]$ (or for some $C \in \mathcal{C}$, $\nu(C) = [l_1 : t_1,\ \ldots,\ l_n : t_n]$), then for each $i \in [1, n]$, t_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.

Obviously, both $E(\Delta)$ and $T(\Delta)$ are finite. In addition, every type in $T(\Delta)$ except $DBtype$ is either a class type or a base type. That is,

$$T(\Delta) \subseteq \mathcal{C} \cup \mathcal{B} \cup \{DBtype\}.$$

Definition 2.5: The *signature determined by schema* Δ, $\sigma(\Delta)$, is a triple

$$(r,\ E(\Delta),\ R(\Delta)),$$

where r is a constant (denoting the root), $E(\Delta)$ is the finite set of binary relations (denoting the edge labels) defined above, and $R(\Delta)$ is the finite set of unary relations (denoting the sorts) defined by $\{R_\tau \mid \tau \in T(\Delta)\}$. ■

For example, the signature determined by the schema given in Example 2.1 is $(r,\ E,\ R)$, where

- r is a constant, which in each instance $(\pi,\ \mu,\ d)$ of the schema intends to name d;

- $E = \{*, name, spouse\}$; and
- $R = \{R_{DBtype}, R_{Person}, R_{string}\}$.

We next define the type constraint determined by a schema. The type constraint can be formulated as a sentence in two-variable logic with counting [14, 5], C^2. Two-variable logic, FO^2, is the fragment of first-order logic consisting of all relational sentences with at most two distinct variables [13], and C^2 is the extension of FO^2 with counting quantifiers. In particular, 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 $G \models \psi(a)$.

Definition 2.6: Let Δ be a schema. For each τ in $T(\Delta)$, the *constraint determined by* τ is the sentence $\forall\, x\, \phi_\tau(x)$ defined as follows:

- if $\tau = b$, or if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = b$, then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)} \neg l(x, y));$$
- if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{t\}$ (or $\tau = DBtype = \{t\}$), then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)\setminus\{*\}} \neg l(x, y)) \ \wedge \forall\, y\, (*(x, y) \rightarrow R_t(y));$$
- if $\tau = C$ for some $C \in \mathcal{C}$ and $\nu(C) = [l_1 : t_1, \ldots, l_n : t_n]$ (or $\tau = DBtype = [l_1 : t_1, \ \ldots, l_n : t_n]$), then $\phi_\tau(x)$ is
$$R_\tau(x) \rightarrow \forall\, y\, (\bigwedge_{l \in E(\Delta)\setminus\{l_1,\ldots,l_n\}} \neg l(x, y)) \ \wedge \bigwedge_{i \in [1,n]} (\exists!\, y\, l_i(x, y) \ \wedge \forall\, y\, (l_i(x, y) \rightarrow R_{t_i}(y))).$$

The *type constraint determined by schema* Δ is the sentence

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

Note here for simplicity, we assume that for each base type $b \in \mathcal{B}$, the domain of b, D_b, is infinite. If D_b is finite, i.e., the cardinality of D_b is some natural number n, then we define the constraint determined by b to be the following sentence in C^2:

$$\forall\, x\, \phi_b(x) \ \wedge \ \exists^{=n}\, x\, R_b(x).$$

Here $\phi_b(x)$ is the formula given in Definition 2.6 and $\exists^{=n}$ is another counting quantifier. The semantics of $\exists^{=n}$ is described as follows: a structure satisfies $\exists^{=n}\, x\, \psi(x)$ if and only if there are exactly n elements in the structure satisfying ψ. We substitute this constraint for $\forall\, x\, \phi_b(x)$ in $\Phi(\Delta)$.

Using the type constraint defined above, we present an abstraction of databases in the object-oriented model as follows. Its justification will be given later in the paper.

Definition 2.7: An *abstract database of a schema* Δ is a finite structure G of the signature $\sigma(\Delta)$ such that $G \models \Phi(\Delta)$. We denote the set of all abstract databases of a schema Δ by $\mathcal{U}_f(\Delta)$. ■

We use $\mathcal{U}(\Delta)$ to denote the set of all the structures of signature $\sigma(\Delta)$ satisfying the following conditions: for each $G \in \mathcal{U}(\Delta)$,

- $G \models \Phi(\Delta)$; and
- for each set type $\tau \in T(\Delta)$ and each $o \in R^G_\tau$, there are only finitely many o' in G such that $G \models *(o, o')$. That is, each node in G has finitely many outgoing edges.

An example structure is depicted in Figure 2. This structure corresponds to the database instance given in Example 2.2.

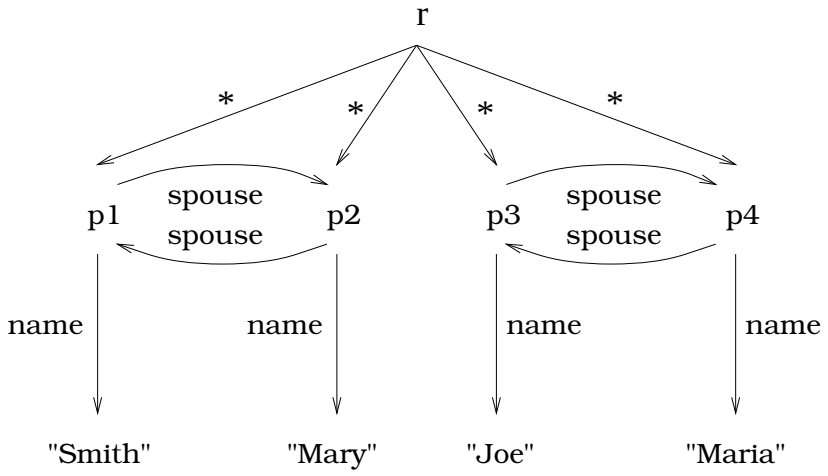

Figure 2: An example of a structure

2.2 Word constraints

In this section, we define word constraints in the object-oriented model, and justify the abstraction of databases given above by considering word constraint satisfiability.

Paths

We first define paths and types of paths over a schema.

Definition 2.8: Given a schema $\Delta = (\mathcal{C}, \nu, DBtype)$, the set of *paths over schema* Δ, $Paths(\Delta)$, and *the type of path* α *in* $Paths(\Delta)$, $type(\alpha)$, are defined inductively as follows:

- the empty path ϵ is in $Paths(\Delta)$ and $type(\epsilon) = DBtype$;
- for any $\alpha \in Paths(\Delta)$, where $type(\alpha) = \tau$,
  - if for some $C \in \mathcal{C}$, $\tau = C$ and $\nu(C) = \{t\}$ (or $\tau = DBtype = \{t\}$), then $\alpha \cdot *$ is a path in $Paths(\Delta)$ and $type(\alpha \cdot *) = t$;
  - if there exists $C \in \mathcal{C}$ such that $\tau = C$ and $\nu(C) = [l_1 : t_1, \ldots, l_n : t_n]$ (or $\tau = DBtype = [l_1 : t_1, \ldots, l_n : t_n]$), then for each $i \in [1, n]$, $\alpha \cdot l_i$ is in $Paths(\Delta)$ and $type(\alpha \cdot l_i) = t_i$. ■

As in semistructured data, path α can be represented by a formula $\alpha(x, y)$, where x and y denote the tail and head nodes of the path, respectively. The formula $\alpha(x, y)$ is defined by:

$$\alpha(x,y) = \begin{cases} x = y & \text{if } \alpha = \epsilon \\ \exists z(\beta(x,z) \wedge *(z,y)) & \text{if } \alpha = \beta \cdot * \\ \exists z(\beta(x,z) \wedge l(z,y)) & \text{if } \alpha = \beta \cdot l \end{cases}$$

Here $\beta(x, z)$ is a formula representing the path β.

In the sequel, we assume that all the paths in $Paths(\Delta)$ are in the form of the formulas defined above.

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 defined by:

$$\alpha(x,z) \cdot \beta(z,y) \;=\; \begin{cases} \beta(x,y) & \text{if } \alpha = \epsilon \\ \alpha'(x,u) \cdot \exists z(*(u,z) \wedge \beta(z,y)) & \text{if } \alpha(x,z) = \exists u(\alpha'(x,u) \wedge *(u,z)) \\ \alpha'(x,u) \cdot \exists z(l(u,z) \wedge \beta(z,y)) & \text{if } \alpha(x,z) = \exists u(\alpha'(x,u) \wedge l(u,z)) \end{cases}$$

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

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

The definition of word constraints

Definition 2.9: A *word constraint φ over schema Δ* is a sentence of the form

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

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

We denote the set of all word constraints over schema Δ as $P_w(\Delta)$. ∎

Obviously, $P_w(\Delta)$ is a language with vocabulary $\sigma(\Delta)$.

We borrow the standard definitions of models and implication from first-order logic [12]. Let G be a structure in $\mathcal{U}(\Delta)$ and φ a constraint in $P_w(\Delta)$. Then we write $G \models \varphi$ if G is a model of φ. Given a finite subset Σ of $P_w(\Delta)$ and $\varphi \in P_w(\Delta)$, we use $\Sigma \models \varphi$ to denote that Σ *implies* φ. That is, for every structure $G \in \mathcal{U}(\Delta)$, if $G \models \Sigma$, then $G \models \varphi$. Similarly, we use $\Sigma \models_f \varphi$ to denote that Σ *finitely implies* φ. That is, for every structure $G \in \mathcal{U}_f(\Delta)$, if $G \models \Sigma$, then $G \models \varphi$.

Example 2.3: The sentences

$$\phi = \forall\, x\, (*(r, x) \rightarrow * \cdot spouse(r, x))$$
$$\varphi = \forall\, x\, (* \cdot spouse(r, x) \rightarrow *(r, x))$$

are word constraints over the schema given in Example 2.1. Let G be the structure given in Figure 2. It is easy to verify that $G \models \phi$ and $G \models \varphi$.

In any instance $(\pi,\ \mu,\ d)$ of the schema, ϕ and φ are interpreted as

$$\forall\, x\, (x \in d \rightarrow \exists\, y\, (y \in d \wedge y.spouse = x)),$$
$$\forall\, x\, (\exists\, y\, (y \in d \wedge y.spouse = x) \rightarrow x \in d),$$

respectively. Here, abusing the type terms, $y.spouse$ stands for the projection of record y at attribute *spouse*, and d is a subset of $\pi(Person)$. The constraint ϕ states: "each person in the set d is the spouse of someone in d", and φ states: "if a person is the spouse of someone in d, then the person is in d". ∎

Justification of the abstraction

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

The agreement between databases and their abstraction with respect to word constraints is revealed by the following lemma, which justifies the abstraction of structured databases defined above.

Lemma 2.1: Let Δ be a schema. For each $I \in \mathcal{I}(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$, such that

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

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) We define a function $f : \mathcal{I}(\Delta) \to \mathcal{U}_f(\Delta)$ such that for each $I \in \mathcal{I}(\Delta)$ and $\varphi \in P_w(\Delta)$, $I \models \varphi$ iff $f(I) \models \varphi$.

Given $I \in \mathcal{I}(\Delta)$, where $I = (\pi, \mu, d)$, let I_B be the set of all the base type values occurring in I. That is, a base type value v is in I_B if and only if either v occurs in d, or there is $C \in \mathcal{C}$ and $o \in \pi(C)$, such that v occurs in $\mu(o)$. Let

$$V = \{d\} \cup I_B \cup \bigcup_{C \in \mathcal{C}} \pi(C).$$

For each $v \in V$, let $o(v)$ be a distinguished node. We then define $f(I)$ to be $G = (|G|, r^G, E^G, R^G)$, where

- $|G| = \{o(v) \mid v \in V\}$;
- $r^G = o(d)$;
- for each $o(v) \in |G|$ and $\tau \in T(\Delta)$, $G \models R^G_\tau(o(v))$ iff v is of type τ;
- for all $o(v), o(v') \in |G|$,
  - $G \models *(o(v), o(v'))$ iff $v' \in v$,
  - for each $l \in \mathcal{L} \cap E(\Delta)$, $G \models l(o(v), o(v'))$ iff $v' = v.l$. Here $v.l$ means the projection of v at attribute l, i.e., the l component of v.

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 $\varphi \in P_w(\Delta)$, $G \models \varphi$ iff $I \models \varphi$. This can be easily verified by *reductio*.

(2) Next, we define $g : \mathcal{U}_f(\Delta) \to \mathcal{I}(\Delta)$ such that for each $G \in \mathcal{U}_f(\Delta)$ and $\varphi \in P_w(\Delta)$, $G \models \varphi$ iff $g(G) \models \varphi$.

Let $G \in \mathcal{U}_f(\Delta)$, where $G = (|G|, r^G, E^G, R^G)$. For each base type $b \in T(\Delta)$, we define an injective mapping $g_b : R^G_b \to D_b$, where R^G_b is the unary relation in G denoting the sort b, and D_b is the domain of b. By the definition of the constraint determined by b given earlier and since G satisfies the constraint, such a mapping always exists. We substitute $g_b(o)$ for each o in R^G_b. We then define $g(G)$ to be $I = (\pi, \mu, d)$, where

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

Note that this is well-defined since $G \models \Phi(\Delta)$. It is easy to verify that $I \in \mathcal{I}(\Delta)$, and $G \models \varphi$ iff $I \models \varphi$.

This proves Lemma 2.1. ∎

From the lemma follows immediately the corollary below.

Corollary 2.2: Let Δ be a schema and $\Sigma \cup \{\varphi\}$ a finite subset of $P_w(\Delta)$. 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 $G \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

Proof: Suppose that there is $I \in \mathcal{I}(\Delta)$ such that $I \models \bigwedge \Sigma \wedge \neg\varphi$. By Lemma 2.1, there is G in $\mathcal{U}_f(\Delta)$, such that for each $\psi \in \Sigma \cup \{\varphi\}$, $I \models \psi$ iff $G \models \psi$. Therefore, $G \models \bigwedge \Sigma \wedge \neg\varphi$.

Conversely, suppose that there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \bigwedge \Sigma \wedge \neg\varphi$. Again by Lemma 2.1, there is $I \in \mathcal{I}(\Delta)$, such that for each $\psi \in \Sigma \cup \{\varphi\}$, $G \models \psi$ iff $I \models \psi$. Therefore, $I \models \bigwedge \Sigma \wedge \neg\varphi$. ∎

2.3 Word constraint implication

In this section, we study the implication and finite implication problems for word constraints in the object-oriented data model. We first describe the problems and show that the proof of the decidability of word constraint implication given in [3] breaks down here. We then prove the decidability of word constraint implication in the context of the object-oriented model. In addition, we show that in two special cases, word constraint implication is decidable in PTIME.

The implication problem

By Corollary 2.2, we can describe word constraint implication as follows.

The *(finite) implication problem for $P_w(\Delta)$ over schema Δ* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$).

As observed by [3], every word constraint can be expressed by a sentence in two-variable logic. Recently, [13] has shown that the satisfiability problem for FO^2 is NEXPTIME-complete by establishing that any satisfiable FO^2 sentence has a model of size exponential in the length of the sentence. The decidability of the implication and finite implication problems for word constraints in semistructured data follows immediately. In fact, [3] directly establishes (without reference to the embedding into FO^2) that the implication problems for word constraints are in PTIME.

In contrast, in the presence of types, implication for word constraints cannot be stated in FO^2. This is because in the (finite) implication problem for $P_w(\Delta)$ over schema Δ, each structure considered must satisfy $\Phi(\Delta)$, which is in C^2 but is not in FO^2.

In the object-oriented model, the proof given in [3] also breaks down. The proof is established by showing that a set of inference rules, $\mathcal{I}_{AV}$, is sound and complete for word constraint implication. This set consists of the following three rules.

- reflexivity:

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

- transitivity:

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

- right-congruence:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \beta(r,x)) \quad \gamma \text{ is a path}}{\forall x\ (\alpha \cdot \gamma(r,x) \rightarrow \beta \cdot \gamma(r,x))}$$

However, the lemma below shows that the proof no longer holds in the context of the object-oriented model.

Lemma 2.3: In the object-oriented model, $\mathcal{I}_{AV}$ is not complete for word constraint implication. ∎

Proof: Consider the constraints ϕ and φ given in Example 2.3. By induction on the length of proof, it can be shown that φ is not provable from ϕ using $\mathcal{I}_{AV}$. More specifically, it can be shown that if φ were provable from ϕ using $\mathcal{I}_{AV}$, then the length of $lt(\varphi)$ would be strictly less than the length of $rt(\varphi)$.

However, by the type constraint imposed by the schema given in Example 2.1, $\{\phi\} \models \varphi$ indeed holds. More specifically, consider an instance I of the schema satisfying ϕ, where $I = (\pi, \mu, d)$. Let $s = \{x.spouse \mid x \in d\}$ and let $|d|$, $|s|$ denote the cardinalities of d and s, respectively. By the type constraint imposed by record type, $|s| \leq |d|$. By $I \models \phi$, $d \subseteq s$. Hence $d = s$, and therefore, $I \models \varphi$. ∎

The decidability of word constraint implication

Next, we show that in the object-oriented model, word constraint implication is indeed decidable.

Proposition 2.4: Over any schema Δ in the object-oriented model, the implication and finite implication problems for $P_w(\Delta)$ are decidable. ∎

The decidability of the finite implication follows from the decidability of the finite satisfiability problem for C^2, which was established by [5], since the type constraints are expressible in C^2 and all the word constraints are in FO^2.

By this result, for the decidability of the implication problem it suffices to show that the implication and finite implication problems coincide. That is, over arbitrary schema Δ and for each finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model in $\mathcal{U}_f(\Delta)$. This is established by the lemma below.

Lemma 2.5: Let Δ be a schema in the object-oriented model. For each finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, if $\bigwedge \Sigma \wedge \neg\varphi$ has a model in $\mathcal{U}(\Delta)$, then it has a model in $\mathcal{U}_f(\Delta)$. ∎

Proof: Given $\Sigma \cup \{\varphi\} \subset P_w(\Delta)$ and model G of $\bigwedge \Sigma \wedge \neg\varphi$ in $\mathcal{U}(\Delta)$, we construct a finite structure G' such that $G' \in \mathcal{U}_f(\Delta)$ and $G' \models \bigwedge \Sigma \wedge \neg\varphi$. To do so, we first define the notion of k-neighborhood of a structure, as follows.

For each structure G in $\mathcal{U}(\Delta)$ and natural number k, *the k-neighborhood of G* is the substructure G_k of G with its universe

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

Given Σ and φ as described above, let

$$k = max\{|lt(\psi)|, |rt(\psi)| \ \mid \ \psi \in \Sigma \cup \{\varphi\}\} \ + \ 1,$$

and let G_k be the k-neighborhood of G. Then we construct G' as follows. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node, and let $G' = (|G'|, r^{G'}, E^{G'}, R^{G'})$, where

- $|G'| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$,
- $r^{G'} = r^{G_k}$,
- for each $\tau \in T(\Delta)$, $R_\tau^{G'} = (R_\tau^G \cap |G_k|) \cup \{o(\tau)\}$,

- $E^{G'}$ is E^{G_k} augmented with the following:
  - for each $o \in R^G_\tau \cap |G_k|$, where $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$, and for each $i \in [1, n]$, if for every $o' \in |G_k|$, $G_k \not\models l_i(o, o')$, then let $G' \models l_i(o, o(\tau_i))$;
  - for any $\tau \in T(\Delta)$, if $\tau = [l_1 : \tau_1, ..., l_n : \tau_n]$, 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) $G' \in \mathcal{U}_f(\Delta)$.

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

(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 $G_k \models \bigwedge \Sigma \wedge \neg\varphi$.

By the claim, it suffices to show that G_k is also the k-neighborhood of G'. To do so, assume for *reductio* that there exist $o(\tau) \in |G'|$ and $\alpha \in Paths(\Delta)$ such that $|\alpha| \leq k$ and $G' \models \alpha(r, o(\tau))$. Without loss of generality, assume that α 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 $G' \models \alpha'(r, o) \wedge l(o, o(\tau))$;
- there is $\tau \in T(\Delta)$ such that $\tau = [l : \tau, ...]$ and $o \in R^G_\tau$, and for any $o' \in |G_k|$, $G_k \not\models l(o, o')$; and
- $G_k \models \alpha'(r, o)$. This is because for each $\tau \in T(\Delta)$, $o(\tau)$ does not have any outgoing edge to any node of $|G_k|$.

By $G \in \mathcal{U}(\Delta)$, 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 $\beta \in Paths(\Delta)$ such that $|\beta| < k$ and $G \models \beta(r, o) \wedge l(o, o')$. Therefore, α' 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 2.5. ∎

The complexity of word constraint implication remains open. However, we show below that in two special cases, word constraint implication is decidable in PTIME.

Word constraint implication over record schema

We next investigate word constraint implication over *record schema*, by which we mean a schema that does not contain any set type.

Proposition 2.6: Over any record schema Δ in the object-oriented model, the implication and finite implication problems for $P_w(\Delta)$ are decidable in PTIME in the size of the implication and the size of the schema. ∎

The proof of the proposition follows closely to the argument given in [3] for the PTIME decidability of word constraint implication in semistructured data. To present the proof, we first introduce a set of inference rules, $\mathcal{I}_r$, over record schema Δ. This set consists of the following rules.

- Reflexivity:

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

- Transitivity:

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

- Right-congruence:

$$\frac{\forall x\ (\alpha(r,x) \rightarrow \beta(r,x)) \quad \alpha \cdot \gamma \in Paths(\Delta) \text{ and } \beta \cdot \gamma \in Paths(\Delta)}{\forall x\ (\alpha \cdot \gamma(r,x) \rightarrow \beta \cdot \gamma(r,x))}$$

- Commutativity:

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

Here for simplicity, we assume that the domain of each base type has at least two elements.

Given a finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, we use $\Sigma \vdash_{\mathcal{I}_r} \varphi$ to denote that there is an $\mathcal{I}_r$-proof of φ from Σ, i.e., φ is provable from Σ using $\mathcal{I}_r$.

The proof of Proposition 2.6 requires the following two lemmas. The second lemma is borrowed from [3]. It involves $\mathcal{I}_{AV}$, the set of inference rules mentioned previously.

Lemma 2.7: Over any record schema Δ, $\mathcal{I}_r$ is sound and complete for finite implication for $P_w(\Delta)$. ∎

Lemma 2.8 [3]: Let Σ be a finite set of word constraints and α a path. The set

$$RewriteTo(\alpha) = \{\beta \mid \Sigma \vdash_{\mathcal{I}_{AV}} \forall x\,(\alpha(r,x) \rightarrow \beta(r,x))\}$$

is a regular language recognized by an nfsa constructible in polynomial time from Σ and α. In particular, whether $\Sigma \vdash_{\mathcal{I}_{AV}} \forall x\,(\alpha(r,x) \rightarrow \beta(r,x))$ can be decided in PTIME. ∎

These two lemmas suffice. To see this, for any record schema Δ and finite subset Σ of $P_w(\Delta)$, let

$$\Sigma' = \Sigma \cup \{\forall x\,(\beta(r,x) \rightarrow \alpha(r,x)) \mid \forall x\,(\alpha(r,x) \rightarrow \beta(r,x)) \in \Sigma\}.$$

It is easy to verify that for each $\varphi \in P_w(\Delta)$, $\Sigma \vdash_{\mathcal{I}_r} \varphi$ if and only if $\Sigma' \vdash_{\mathcal{I}_{AV}} \varphi$ and $\varphi \in P_w(\Delta)$. In addition, it can be verified that whether φ is in $P_w(\Delta)$ can be decided in PTIME in the size of Δ and the size of φ. Hence by Lemma 2.8, whether $\Sigma \vdash_{\mathcal{I}_r} \varphi$ can be decided in PTIME in the size of Δ and the size of $\Sigma \cup \{\varphi\}$. By Lemma 2.7, $\Sigma \models_f \varphi$ iff $\Sigma \vdash_{\mathcal{I}_r} \varphi$. By Lemma 2.5, we also have $\Sigma \models \varphi$ iff $\Sigma \vdash_{\mathcal{I}_r} \varphi$. Therefore, the implication and finite implication problems for $P_w(\Delta)$ are decidable in the size of Δ and the size of $\Sigma \cup \{\varphi\}$.

We next show Lemma 2.7.

Proof of Lemma 2.7: The soundness of $\mathcal{I}_r$ can be verified by a straightforward induction on the length of $\mathcal{I}_r$-proof.

For the proof of the completeness, it suffices to show the following claim.

Claim 1: Given any record schema Δ and finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \Sigma$, and in addition, if $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

First assume that for each base type $b \in T(\Delta)$, the domain of b is infinite. We prove Claim 1 by constructing the structure G desired. Let

$$k = max\{|lt(\psi)|, |rt(\psi)| \mid \psi \in \Sigma \cup \{\varphi\}\} + 1.$$

We first construct the k-neighborhood of G, G_k, and then construct G from G_k.

The construction of G_k. Let

- $Paths^k(\Delta) = \{\alpha \mid \alpha \in Paths(\Delta),\ |\alpha| \leq k\}$;
- $\approx$ be the equivalence relation on $Paths^k(\Delta)$ defined by

$$\alpha \approx \beta \text{ iff } \Sigma \vdash_{\mathcal{I}_r} \forall x\ (\alpha(r,\ x) \rightarrow \beta(r,\ x)) \text{ and } \Sigma \vdash_{\mathcal{I}_r} \forall x\ (\beta(r,\ x) \rightarrow \alpha(r,\ x));$$

- $\widehat{\alpha}$ denote the equivalence class of path α and $\mathcal{A} = \{\widehat{\alpha} \mid \alpha \in Paths^k(\Delta)\}$;
- $type(\widehat{\alpha}) = type(\alpha)$, where $type(\alpha)$ is the type of path α determined by Δ. This is well-defined since if α and β are in the same equivalence class, then by Definition 2.9, $type(\alpha) = type(\beta)$.

We construct 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 $R_\tau^{G_k} = \{o(\widehat{\alpha}) \mid \widehat{\alpha} \in \mathcal{A},\ type(\widehat{\alpha}) = \tau\}$.
- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ 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 $\mathcal{I}_r$.

The construction of G. For each $\tau \in T(\Delta)$, let $o(\tau)$ be a distinct node. Let $G = (|G|, r^G, E^G, R^G)$, where

- $|G| = |G_k| \cup \{o(\tau) \mid \tau \in T(\Delta)\}$;
- $r^G = r^{G_k}$;
- for each $\tau \in T(\Delta)$, $R_\tau^G = R_\tau^{G_k} \cup \{o(\tau)\}$;
- for each label $l \in E(\Delta)$, if $G_k \models l(o,\ o')$, then $G \models l(o,\ o')$. Moreover,
  - for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = [l_1 : \tau_1,\ ...,\ l_n : \tau_n]$ 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 every $\tau \in T(\Delta)$, if τ is of the form $[l_1 : \tau_1,\ ...,\ l_n : \tau_n]$, then for each $i \in [1,\ n]$, let $G \models l_i(o(\tau),\ o(\tau_i))$.

We next show that G is indeed a structure described in Claim 1.

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

Obviously, $|G|$ is finite since $Paths^k(\Delta)$ and $T(\Delta)$ are finite. We next show that $G \models \Phi(\Delta)$. That is, we show that for each $o \in |G|$, if $o \in R_\tau^G$, then $G \models \phi_\tau(o)$. We examine the following cases.

Case 1: $o = o(\tau)$.

By the construction of G, it is obvious $G \models \phi_\tau(o(\tau))$.

Case 2: $o = o(\widehat{\alpha})$.

If $type(\widehat{\alpha}) = b$ for some base type b, then by the construction of G_k, $o(\widehat{\alpha})$ does not have any outgoing edge. Thus $G \models \phi_\tau(o(\widehat{\alpha}))$.

If $\tau = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, we have two cases to consider.

First, if for each $\beta \in \widehat{\alpha}$, $k \leq |\beta|$, then by the construction of G, for each $i \in [1, n]$,

$$G \models l_i(o(\widehat{\alpha}),\ o(\tau_i)),$$

and moreover, these are all the outgoing edges of $o(\widehat{\alpha})$. Clearly, $o(\tau_i) \in R_{\tau_i}^G$. Hence $G \models \phi_\tau(o(\widehat{\alpha}))$.

Second, suppose that there is $\beta \in \widehat{\alpha}$, such that $|\beta| < k$. Then by the construction of G_k, for each $i \in [1, n]$,

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

By Definition 2.8, $type(\widehat{\beta \cdot l_i}) = type(\beta \cdot l_i) = \tau_i$. That is, $o(\widehat{\beta \cdot l_i}) \in R^G_{\tau_i}$. Moreover, by Right-congruence, for each $\gamma \in \widehat{\alpha}$, we have $\beta \cdot l_i \approx \gamma \cdot l_i$. Hence $o(\widehat{\alpha})$ has a unique outgoing edge labeled with l_i. Therefore, $G \models \phi_\tau(o(\widehat{\alpha}))$.

This proves that $G \in \mathcal{U}_f(\Delta)$.

(2) G_k is the k-neighborhood of G.

By the property of record schema and the definition of G, we have the following claim:

Claim 2: For each $\alpha \in Paths^k(\Delta)$, $G \models \alpha(r,\, o(\widehat{\alpha}))$. In addition, if there is $o \in |G|$ such that $G \models \alpha(r,\, o)$, then $o = o(\widehat{\alpha})$.

This claim can be verified by a straightforward induction on $|\alpha|$. This shows that G_k is indeed the k-neighborhood of G.

(3) $G \models \Sigma$.

For each $\psi \in \Sigma$, where $\psi = \forall x\, (\alpha(r, x) \to \beta(r, x))$, we have $\alpha, \beta \in Paths^k(\Delta)$ by the definition of k. By Commutativity, we have $\alpha \approx \beta$. Therefore, $o(\widehat{\alpha}) = o(\widehat{\beta})$. By Claim 2, $o(\widehat{\alpha})$ is the only node in G to which there is an α path from r. Therefore,

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

Hence $G \models \Sigma$.

(4) If $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

Let $\varphi = \forall x\, (\alpha(r, x) \to \beta(r, x))$. By the definition of k, we have that $\alpha, \beta \in Paths^k(\Delta)$. Moreover, by $G \models \varphi$ and Claim 2, $o(\widehat{\alpha}) = o(\widehat{\beta})$. By the construction of G, there must be $\widehat{\alpha} = \widehat{\beta}$. Hence by the definition of $\approx$, we have $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

This shows that if the domain of each base type in $T(\Delta)$ is infinite, then Claim 1 holds.

Now suppose that some base types in $T(\Delta)$ have finite domains (as mentioned previously, we assume that each of these finite domains has at least two elements). We construct a structure G' which has all the properties described in Claim 1 as follows.

Let G be the structure defined above. For each base type $b \in T(\Delta)$ with a finite domain and for all $\widehat{\alpha}$, $\widehat{\beta}$ in $\mathcal{A}$, we identify $o(\widehat{\alpha})$ with $o(\widehat{\beta})$ in $|G|$ if all the following conditions are satisfied:

- $type(\widehat{\alpha}) = type(\widehat{\beta}) = b$;
- if $\widehat{lt(\varphi)} \neq \widehat{rt(\varphi)}$, then none of the following holds:
  - $\widehat{\alpha} = \widehat{lt(\varphi)}$ and $\widehat{\beta} = \widehat{rt(\varphi)}$,
  - $\widehat{\alpha} = \widehat{rt(\varphi)}$ and $\widehat{\beta} = \widehat{lt(\varphi)}$.

In addition, we equalize $o(\tau)$ with $o(\widehat{\alpha})$ for some $\widehat{\alpha} \in \mathcal{A}$ such that $\widehat{\alpha} \neq \widehat{rt(\varphi)}$. If such $\widehat{\alpha}$ does not exist, then let $o(\tau)$ be a distinct node as before.

Let G' be the structure constructed from G by equalizing nodes in $|G|$ as described above. Clearly, $|G'| \subseteq |G|$, and for each base type $b \in T(\Delta)$, if the domain of b is finite, then $R^{G'}_b$ has at most two elements. In addition, by the definition of G', it is easy to verify the following claims.

Claim 3: $G' \models \Phi(\Delta)$.

Claim 4: For each $\alpha \in Paths^k(\Delta)$ and $o \in |G'|$, if $G \models \alpha(r, o)$, then $G' \models \alpha(r, o)$.

Claim 5: If $G' \models \varphi$, then $G \models \varphi$.

These suffice for a proof of Claim 1. For by Claim 3, $G' \in \mathcal{U}_f(\Delta)$. Using Claim 4, it is easy to verify that $G' \models \Sigma$ by *reductio*. By Claim 5, if $G' \models \varphi$, then by the proof above, $\Sigma \vdash_{\mathcal{I}_r} \varphi$.

This completes the proof of Lemma 2.7. ∎

Implication for word constraints having the $*$-form

Next, we consider word constraints of the form:

$$\forall\, x\, (\alpha(r, x) \rightarrow \beta \cdot *(r, x)).$$

We refer to such a constraint as a constraint having *the $*$-form*. Implication $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) is called *$*$-form (finite) implication* if every constraint in $\Sigma \cup \{\varphi\}$ has the $*$-form.

Proposition 2.9: Over any schema Δ in the object-oriented model, the $*$-form implication and finite implication problems for $P_w(\Delta)$ are decidable in PTIME in the size of the implication and the size of the schema. ∎

To show the proposition, let $\mathcal{I}_*$ be the subset of $\mathcal{I}_r$ consisting of Reflexivity, Transitivity and Right-congruence. As in the proof of Proposition 2.6, it suffices to show the following lemma.

Lemma 2.10: Over any schema Δ in the object-oriented model, $\mathcal{I}_*$ is sound and complete for finite implication for $P_w(\Delta)$. ∎

Proof: The proof of the lemma is similar to that of Lemma 2.7.

The soundness of $\mathcal{I}_*$ can be verified by a straightforward induction on the length of $\mathcal{I}_*$-proof.

For the proof of the completeness, it suffices to show the following claim.

Claim 1: Given any schema Δ and finite set $\Sigma \cup \{\varphi\}$ of $*$-form constraints in $P_w(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \Sigma$, and in addition, if $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

We first assume that for each base type $b \in T(\Delta)$, the domain of b is infinite. As in the proof of Lemma 2.7, we define the natural number k. We construct the structure G described in Claim 1 in two steps: we first define G_k and the construct G from G_k.

The construction of G_k. As in the proof of Lemma 2.7, we define $Paths^k(\Delta)$, $\approx$, $\widehat{\alpha}$, $\mathcal{A}$ and $type(\widehat{\alpha})$. In addition, we define a partial order on $\mathcal{A}$ as follows:

$$\widehat{\alpha} \prec \widehat{\beta} \quad \text{iff} \quad \Sigma \vdash_{\mathcal{I}_*} \forall x\, (\alpha(r,\, x) \rightarrow \beta(r,\, x)).$$

Note that this is well-defined by Transitivity in $\mathcal{I}_*$.

Let $G_k = (|G_k|, r^{G_k}, E^{G_k}, R^{G_k})$, where $|G_k|$, r^{G_k} and R^{G_k} are defined in the same way as in the proof of Lemma 2.7. The binary relations in E^{G_k} are populated as follows.

- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ 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 $\mathcal{I}_*$.
- For each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = \{\tau\}$ and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, let $G_k \models *(o(\widehat{\alpha}),\, o(\widehat{\gamma}))$.

The construction of G. The structure G is defined in the same way as in the proof of Lemma 2.7, except the following: for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = \{\tau\}$, then let $G \models *(o(\widehat{\alpha}),\, o(\tau))$.

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

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

It is easy to verify that $|G|$ is finite. We next show that for each $o \in |G|$, if $o \in R^G_\tau$, then $G \models \phi_\tau(o)$. The arguments for the following cases are the same as in the proof of Lemma 2.7.

Case 1: $o = o(\tau)$ and τ is either a base type or a record type.

Case 2: $o = o(\widehat{\alpha})$ and $type(\widehat{\alpha})$ is either a base type or a record type.

We next examine the cases involving set types.

Case 3: $o = o(\tau)$ and $\tau = \{\tau'\}$.

Clearly, $G \models \phi_\tau(o(\tau))$ since $o(\tau)$ does not have any outgoing edge by the construction of G.

Case 4: $o = o(\widehat{\alpha})$ and $type(\widehat{\alpha}) = \{\tau'\}$.

If for each $\beta \in \widehat{\alpha}$, $k \leq |\beta|$, then by the construction of G, $G \models *(o(\widehat{\alpha}), o(\tau'))$. In addition, $o(\widehat{\alpha})$ does not have any other outgoing edge. Clearly, $o(\tau') \in R^G_{\tau'}$. Hence $G \models \phi_\tau(o(\widehat{\alpha}))$ in this case.

Now suppose that there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$. Then by the definition of G, for each γ in $Paths^k(\Delta)$, if $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, then $G \models *(o(\widehat{\alpha}), o(\widehat{\gamma}))$. Moreover, $G \models *(o(\widehat{\alpha}), o(\tau'))$. These are all the outgoing edges from $o(\widehat{\alpha})$. Therefore, $o(\widehat{\alpha})$ has finitely many outgoing edges, which are all labeled with $*$. In addition, clearly $o(\tau') \in R^G_{\tau'}$. Moreover, by $\widehat{\gamma} \prec \widehat{\beta \cdot *}$, we have $type(\widehat{\gamma}) = type(\widehat{\beta \cdot *}) = \tau'$. Hence $o(\widehat{\gamma}) \in R^G_{\tau'}$. Thus $G \models \phi_\tau(o(\widehat{\alpha}))$.

This proves that $G \models \Phi(\Delta)$, and consequently, $G \in \mathcal{U}_f(\Delta)$.

2. $G \models \Sigma$.

It suffices to show the following claim.

Claim 2: For each $\alpha \in Paths^k(\Delta)$, let

$$\begin{aligned} obj(\alpha) &= \{o \mid o \in |G_k|,\ G \models \alpha(r,\ o)\}; \\ inf(\alpha) &= \{o(\widehat{\beta}) \mid \widehat{\beta} \in \mathcal{A},\ \widehat{\beta} \prec \widehat{\alpha}\}. \end{aligned}$$

Then $obj(\alpha) = inf(\alpha)$.

To see this, assume for *reductio* that there is $\psi \in \Sigma$, where $\psi = \forall x\ (\alpha(r,\ x) \rightarrow \beta \cdot *(r,\ x))$, such that $G \not\models \psi$. That is, there is $o \in |G|$, such that $G \models \alpha(r,\ o) \wedge \neg \beta \cdot *(r,\ o)$.

If $o \in |G_k|$, then $o \in obj(\alpha)$. By $\Sigma \vdash_{\mathcal{I}_*} \psi$, we have $\widehat{\alpha} \prec \widehat{\beta \cdot *}$. Hence $inf(\alpha) \subseteq \inf(\beta \cdot *)$. Therefore, by Claim 2, $obj(\alpha) \subseteq obj(\beta \cdot *)$. Hence $o \in obj(\beta \cdot *)$. That is, $G \models \beta \cdot *(r, o)$. This contradicts the assumption.

If $o \in |G| \backslash |G_k|$, i.e., $o = o(\tau)$ for some $\tau \in T(\Delta)$, then by Definition 2.9, $type(\beta \cdot *) = type(\alpha) = \tau$. By Definition 2.8, we have $type(\beta) = \{\tau\}$. Since $o(\widehat{\beta}) \in inf(\beta)$, by Claim 2, $o(\widehat{\beta}) \in obj(\beta)$. That is, $G \models \beta(r,\ o(\widehat{\beta}))$. By the construction of G, $G \models *(o(\widehat{\beta}),\ o(\tau))$. Hence $G \models \beta \cdot *(r,\ o(\tau))$. This contradicts the assumption.

Hence $G \models \Sigma$.

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

Base case: $\alpha = \epsilon$.

Since all the constraints in Σ have the $*$-form, by the definition of $\mathcal{I}_*$, it is easy to see that for each $\beta \in Paths^k$, if $\widehat{\beta} \prec \widehat{\epsilon}$, then $\beta = \epsilon$. Therefore, $inf(\epsilon) = \{o(\widehat{\epsilon})\} = \{r^G\} = obj(\epsilon)$.

Inductive step: Assume Claim 2 for $|\alpha| < m$.

We next show the claim holds for $\alpha \cdot K$, where K is either $*$ or some record label l.

(1) $inf(\alpha \cdot K) \subseteq obj(\alpha \cdot K)$.

Let o be a node in $inf(\alpha \cdot K)$.

If $K \neq *$, then by Definition 2.8, $type(\widehat{\alpha})$ is some record type with field K. In addition, by the definition of inf, there is $\beta \in Paths^k(\Delta)$ such that $o = o(\widehat{\beta})$ and $\widehat{\beta} \prec \widehat{\alpha \cdot K}$. Since all the constraints in Σ have the $*$-form, by the definition of $\mathcal{I}_*$, there must be $\beta' \in Paths^k(\Delta)$ such that

$$\beta = \beta' \cdot K \ \text{ and } \ \widehat{\beta'} \prec \widehat{\alpha}.$$

This can be verified by a straightforward induction on the length of $\mathcal{I}_*$-proof of the constraint $\forall x\,(\beta(r,x) \to \alpha \cdot K(r,x))$ from Σ. Thus $o(\widehat{\beta'}) \in inf(\alpha)$. By the induction hypothesis, we have that $o(\widehat{\beta'}) \in obj(\alpha)$. That is,

$$G \models \alpha(r,\, o(\widehat{\beta'})).$$

Since $|\beta'| < |\beta| < k$ and $type(\beta') = type(\alpha)$, by the definition of G,

$$G \models K(o(\widehat{\beta'}), o(\widehat{\beta' \cdot K})).$$

Therefore, $o(\widehat{\beta}) \in obj(\alpha \cdot K)$. That is, $o \in obj(\alpha \cdot K)$.

If $K = *$, then by Definition 2.8, $type(\widehat{\alpha}) = \{type(\alpha \cdot *)\}$. In addition, there is $\beta \in Paths^k(\Delta)$ such that $o = o(\widehat{\beta})$ and $\widehat{\beta} \prec \widehat{\alpha \cdot *}$. By the induction hypothesis, $o(\widehat{\alpha}) \in inf(\alpha) = obj(\alpha)$. That is,

$$G \models \alpha(r,\, o(\widehat{\alpha})).$$

Since $\widehat{\beta} \prec \widehat{\alpha \cdot *}$, by the construction of G_k,

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

Hence $o(\widehat{\beta}) \in obj(\alpha \cdot *)$. That is, $o \in obj(\alpha \cdot *)$.

Therefore, $inf(\alpha \cdot K) \subseteq obj(\alpha \cdot K)$.

(2) $obj(\alpha \cdot K) \subseteq inf(\alpha \cdot K)$.

For each $o \in obj(\alpha \cdot K)$, there is $o' \in obj(\alpha)$, such that $G \models K(o',\, o)$.

If $K \neq *$, then $type(\alpha)$ is some record type with field K. By the the induction hypothesis, $inf(\alpha) = obj(\alpha)$. Thus $o' \in inf(\alpha)$. Hence there is some $\beta \in Paths^k(\Delta)$, such that $\widehat{\beta} \prec \widehat{\alpha}$ and $o' = o(\widehat{\beta})$. Since $o \in |G_k|$ and $G \models K(o(\widehat{\beta}),\, o)$, by the construction of G_k, there must be $\gamma \in \widehat{\beta}$ such that $|\gamma| < k$ and

$$o(\widehat{\gamma \cdot K}) = o.$$

Since $\widehat{\gamma} = \widehat{\beta}$ and $\widehat{\beta} \prec \widehat{\alpha}$, by Right-congruence,

$$\widehat{\gamma \cdot K} \prec \widehat{\alpha \cdot K}.$$

Hence $o(\widehat{\gamma \cdot K}) \in inf(\alpha \cdot K)$. That is, $o \in inf(\alpha \cdot K)$.

If $K = *$, then $type(\alpha) = \{type(\alpha \cdot *)\}$. By the induction hypothesis, $inf(\alpha) = obj(\alpha)$. Thus $o' \in inf(\alpha)$. Hence there is $\beta \in Paths^k(\Delta)$ such that $\widehat{\beta} \prec \widehat{\alpha}$ and $o' = o(\widehat{\beta})$. By Definition 2.9, $type(\widehat{\beta}) = \{type(\alpha \cdot *)\}$. Since $o \in |G_k|$ and $G \models *(o(\widehat{\beta}),\, o)$, by the construction of G_k, there must be $\gamma \in \widehat{\beta}$ such that $|\gamma| < k$. Hence $\widehat{\gamma \cdot *} \in \mathcal{A}$. In addition, there must be $\rho \in Paths^k(\Delta)$ such that

$$\widehat{\rho} \prec \widehat{\gamma \cdot *} \;\text{ and }\; o(\widehat{\rho}) = o.$$

Since $\gamma \in \widehat{\beta}$, we have $\widehat{\gamma} = \widehat{\beta}$. Since $\widehat{\beta} \prec \widehat{\alpha}$, by Right-congruence, $\widehat{\gamma \cdot *} \prec \widehat{\alpha \cdot *}$. By Transitivity,

$$\widehat{\rho} \prec \widehat{\alpha \cdot *}.$$

Hence $o(\widehat{\rho}) \in inf(\alpha \cdot *)$. That is, $o \in inf(\alpha \cdot *)$.

Therefore, $obj(\alpha \cdot K) \subseteq inf(\alpha \cdot K)$. This proves Claim 2.

3. If $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

Let $\varphi = \forall x\,(\alpha(r,x) \to \beta \cdot *(r,x))$. Since α and $\beta \cdot *$ are in $Paths^k(\Delta)$ and $G \models \varphi$, we have $obj(\alpha) \subseteq obj(\beta \cdot *)$. Hence by Claim 2, we have $inf(\alpha) \subseteq inf(\beta \cdot *)$. Since $o(\widehat{\alpha}) \in inf(\alpha)$, $o(\widehat{\alpha}) \in inf(\beta \cdot *)$. Therefore, $\widehat{\alpha} \prec \widehat{\beta \cdot *}$ by the definition of inf. Hence $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

This shows that if the domain of each base type in $T(\Delta)$ is infinite, then Claim 1 holds.

Now suppose that some base types in $T(\Delta)$ have finite domains (as mentioned previously, we assume that each of these finite domains has at least two elements). We construct a structure G' which has all the properties described in Claim 1 as follows.

Let G be the structure defined above. For each base type $b \in T(\Delta)$ with a finite domain and for all $\widehat{\alpha}$, $\widehat{\beta}$ in $\mathcal{A}$, we identify $o(\widehat{\alpha})$ with $o(\widehat{\beta})$ in $|G|$ if all the following conditions are satisfied:

- $type(\widehat{\alpha}) = type(\widehat{\beta}) = b$;
- if $\widehat{lt(\varphi)} \not\prec \widehat{rt(\varphi)}$, then none of the following holds:
  - $\widehat{lt(\varphi)} \prec \widehat{\alpha}$ and $\widehat{\beta} \prec \widehat{rt(\varphi)}$,
  - $\widehat{lt(\varphi)} \prec \widehat{\beta}$ and $\widehat{\alpha} \prec \widehat{lt(\varphi)}$.

In addition, we equalize $o(\tau)$ with $o(\widehat{\alpha})$ for some $\widehat{\alpha} \in \mathcal{A}$ such that $\widehat{\alpha} \neq \widehat{rt(\varphi)}$. If such $\widehat{\alpha}$ does not exist, then let $o(\tau)$ be a distinct node as before.

Let G' be the structure constructed from G by equalizing nodes in $|G|$ as described above. It is easy to show that Claim 3, 4 and 5 in the proof of Lemma 2.7 also hold here. Thus Claim 1 also holds in this case.

This completes the proof of Lemma 2.10.

■

3 Word Constraints in an ACeDB Model

We next consider word constraint implication in an object-oriented model based on ACeDB [19].

The ACeDB based model does not have an explicit set construct, and in addition, it does not interpret a record type as a function from attributes to corresponding domains. More specifically, a value of a record type $[l_1 : t_1, \ldots, l_n : t_n]$ is a finite subset of

$$(\{l_1\} \times [\![t_1]\!]) \cup \ldots \cup (\{l_n\} \times [\![t_n]\!]),$$

where $[\![t_i]\!]$ denotes the domain of t_i. In graph representation, a node of this record type may have finitely many outgoing edges labeled with l_i for each $i \in [1, n]$.

This ACeDB model is defined in the same way as the object-oriented model given in the last section, except the difference aforementioned. Similarly, the abstraction of the databases and word constraints in the model can be defined, except that the constraint $\forall\, x\, \phi_\tau(x)$ imposed by a record type $\tau = [l_1 : t_1, \ldots, l_n : t_n]$ is now defined by:

$$\phi_\tau(x) \;=\; R_\tau(x) \rightarrow \forall y\, (\bigwedge_{l \in E(\Delta) \setminus \{l_1, \ldots, l_n\}} \neg l(x, y)) \;\wedge\; \bigwedge_{i \in [1,n]} \forall y\, (l_i(x, y) \rightarrow R_{t_i}(y)).$$

Given a schema Δ in the ACeDB model, we assume the definitions of $E(\Delta)$, $T(\Delta)$, $\sigma(\Delta)$, $\Phi(\Delta)$, $P_w(\Delta)$, $\mathcal{U}_f(D)$ and $\mathcal{U}(D)$ used in the object-oriented model defined in the last section. For simplicity, we assume that $P_w(\Delta)$ does not include constraints of the following form (see [3] for an argument for this assumption):

$$\forall\, x\, (\alpha(r, x) \rightarrow \epsilon(r, x)).$$

The proposition below establishes the decidability of word constraint implication in the ACeDB model.

Proposition 3.1: Over any schema Δ in the ACeDB model, the implication and finite implication problems for $P_w(\Delta)$ are decidable in PTIME in the size of the implication and the size of the schema. ■

To prove this proposition, recall $\mathcal{I}_*$, the set of inference rules given in the last section. The lemma below shows that $\mathcal{I}_*$ is also sound and complete for word constraint implication in the ACeDB model.

Lemma 3.2: Over any schema Δ in the ACeDB model, $\mathcal{I}_*$ is sound for both the implication and finite implication problems for $P_w(\Delta)$, and is complete for the finite implication problem for $P_w(\Delta)$. ■

From Lemma 3.2 and Lemma 2.8 follows immediately the PTIME decidability of the finite implication problem for word constraints in the ACeDB model. In addition, by Lemma 3.2, the implication and finite implication problems for word constraints coincide in the ACeDB model. To see this, consider a finite subset $\Sigma \cup \{\varphi\}$ of $P_w(\Delta)$. Obviously, if $\Sigma \models \varphi$, then $\Sigma \models_f \varphi$. Conversely, if $\Sigma \models_f \varphi$ then by the completeness of $\mathcal{I}_*$ for finite implication, $\Sigma \vdash_{\mathcal{I}_*} \varphi$. Since $\mathcal{I}_*$ is also sound for implication, we have $\Sigma \models \varphi$. From this argument also follows the PTIME decidability of the implication problem for word constraints in the ACeDB model.

We next show Lemma 3.2.

Proof of Lemma 3.2: The proof below is similar to that of Lemma 2.10.

The soundness of $\mathcal{I}_*$ can be verified by a straightforward induction on the length of $\mathcal{I}_*$-proof.

For the proof of the completeness, it suffices to show Claim 1 below:

Claim 1: Given any schema Δ in the ACeDB model and finite set $\Sigma \cup \{\varphi\}$ of constraints in $P_w(\Delta)$, there is $G \in \mathcal{U}_f(\Delta)$ such that $G \models \Sigma$, and in addition, if $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

We first assume that for each base type $b \in T(\Delta)$, the domain of b is infinite. As in the proof of Lemma 2.7, we define the natural number k. We construct the structure G described in Claim 1 in two steps: we first define G_k and the construct G from G_k.

The construction of G_k. As in the proof of Lemma 2.10, we define $Paths^k(\Delta)$, $\approx$, $\widehat{\alpha}$, $\mathcal{A}$, $type(\widehat{\alpha})$ and $\prec$. Let $G_k = (|G_k|, r^{G_k}, E^{G_k}, R^{G_k})$, where $|G_k|$, r^{G_k} and R^{G_k} are defined in the same way as in the proof of Lemma 2.7. The binary relations in E^{G_k} are populated as follows: for each $o(\widehat{\alpha})$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$ and there is $\beta \in \widehat{\alpha}$ with $|\beta| < k$, then for each $i \in [1, n]$ and each $\gamma \in \mathcal{A}$ such that $\widehat{\gamma} \prec \widehat{\beta \cdot l_i}$, let

$$G_k \models l_i(o(\widehat{\alpha}),\, o(\widehat{\gamma \cdot l_i})).$$

The construction of G. Let $G = (|G|, r^G, E^G, R^G)$, where $|G|$, r^G and R^G are defined in the same way as in the proof of Lemma 2.7. Let E^G be E^{G_k} augmented as follows: for each $o(\widehat{\alpha}) \in |G_k|$, if $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$, then for each $i \in [1, n]$, let

$$G \models l_i(o(\widehat{\alpha}),\, o(\tau_i)).$$

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

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

It is easy to verify that $|G|$ is finite. We next show that for each $o \in |G|$, if $o \in R^G_\tau$, then $G \models \phi_\tau(o)$. The arguments for the following cases are the same as in the proof of Lemma 2.7.

Case 1: $o = o(\tau)$ and τ is either a base type or a record type.

Case 2: $o = o(\widehat{\alpha})$ and $type(\widehat{\alpha})$ is a base type.

We next examine the following case.

Case 3: $o = o(\widehat{\alpha})$ and $type(\widehat{\alpha}) = [l_1 : \tau_1, \ldots, l_n : \tau_n]$.

If for each $\beta \in \widehat{\alpha}$, $k \leq |\beta|$, then by the construction of G, for each $i \in [1, n]$, $G \models l_i(o(\widehat{\alpha}), o(\tau_i))$. These are all the outgoing edges of $o(\widehat{\alpha})$. Clearly, $o(\tau_i) \in R^G_{\tau_i}$. Hence $G \models \phi_\tau(o(\widehat{\alpha}))$ in this case.

If there is $\beta \in \widehat{\alpha}$ such that $|\beta| < k$, then by the construction of G_k, for each $i \in [1, n]$ and each $\widehat{\gamma} \prec \widehat{\beta \cdot l_i}$,

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

In addition,

$$G \models l_i(o(\widehat{\alpha}),\, o(\tau_i)).$$

These are all the outgoing edges of $o(\widehat{\alpha})$. Clearly, $o(\tau_i) \in R^G_{\tau_i}$. By Definition 2.8, it is easy to see that $type(\widehat{\beta \cdot l_i}) = type(\beta \cdot l_i) = \tau_i$. Moreover, by Definition 2.9, we have that for each $\widehat{\gamma} \prec \widehat{\beta \cdot l_i}$, $type(\widehat{\gamma}) = type(\gamma) = type(\beta \cdot l_i)$. Hence $o(\widehat{\gamma}) \in R^G_{\tau_i}$. Therefore, $G \models \phi_\tau(o(\widehat{\alpha}))$.

This proves that $G \models \Phi(\Delta)$, and therefore, $G \in \mathcal{U}_f(\Delta)$.

2. $G \models \Sigma$.

It suffices to show Claim 2 given in the proof of Lemma 2.10.

To see this, assume for *reductio* that there is $\psi \in \Sigma$, where $\psi = \forall x\ (\alpha(r,\, x) \rightarrow \beta(r,\, x))$, such that $G \not\models \psi$. That is, there is $o \in |G|$, such that $G \models \alpha(r,\, o) \wedge \neg\beta(r,\, o)$.

If $o \in |G_k|$, then $o \in obj(\alpha)$. By $\Sigma \vdash_{\mathcal{I}_*} \psi$, we have $\widehat{\alpha} \prec \widehat{\beta}$. Hence $inf(\alpha) \subseteq \inf(\beta)$. Therefore, by Claim 2, $obj(\alpha) \subseteq obj(\beta)$. Hence $o \in obj(\beta)$. That is, $G \models \beta(r, o)$. This contradicts the assumption.

If $o \in |G| \setminus |G_k|$, i.e., $o = o(\tau)$ for some $\tau \in T(\Delta)$, then $type(\alpha) = \tau$. By the assumption on $P_w(\Delta)$, $1 \leq |\beta|$. Let $\beta = \beta' \cdot l$. Since $\beta \in Paths^k(\Delta)$, $\beta' \in Paths^k(\Delta)$. By Claim 2, $o(\widehat{\beta'}) \in inf(\beta') = obj(\beta')$. Hence

$$G \models \beta'(r,\, o(\widehat{\beta'})).$$

By Definition 2.9, $type(\beta) = type(\alpha) = \tau$. By Definition 2.8, $type(\beta')$ is a record type $[l : \tau, \ldots]$. Hence by the construction of G,

$$G \models l(o(\widehat{\beta'}),\, o(\tau)).$$

Thus $G \models \beta(r, o)$. This contradicts the assumption.

Hence $G \models \Sigma$.

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

Base case: $\alpha = \epsilon$.

By the assumption on $P_w(\Delta)$, it is easy to see that for each $\beta \in Paths^k$, if $\widehat{\beta} \prec \widehat{\epsilon}$, then $\beta = \epsilon$. Therefore, $inf(\epsilon) = \{o(\widehat{\epsilon})\} = \{r^G\} = obj(\epsilon)$.

Inductive step: Assume Claim 2 for $|\alpha| < m$. We next show that the claim holds for $\alpha \cdot l$.

(1) $inf(\alpha \cdot l) \subseteq obj(\alpha \cdot l)$.

For each $o \in inf(\alpha \cdot l)$, there is $\widehat{\beta}$, such that $\widehat{\beta} \prec \widehat{\alpha \cdot l}$ and $o = o(\widehat{\beta})$. By $\alpha \cdot l \in Paths^k(\Delta)$, $\widehat{\alpha} \in \mathcal{A}$. Hence by induction hypothesis, $o(\widehat{\alpha}) \in inf(\alpha) = obj(\alpha)$. That is, $G \models \alpha(r,\, o(\widehat{\alpha}))$. In addition, by Definition 2.8, $type(\widehat{\alpha})$ must be a record type with field l. Since $\widehat{\beta} \prec \widehat{\alpha \cdot l}$, by the construction of G, $G \models l(o(\widehat{\alpha}),\, o(\widehat{\beta}))$. Thus $o \in obj(\alpha \cdot l)$.

Therefore, $inf(\alpha \cdot l) \subseteq obj(\alpha \cdot l)$.

(2) $obj(\alpha \cdot l) \subseteq inf(\alpha \cdot l)$.

For each $o \in obj(\alpha \cdot l)$, there is $o' \in obj(\alpha)$, such that $G \models l(o',\, o)$. By induction hypothesis, $o' \in inf(\alpha)$. Hence there is $\beta \in Paths^k(\Delta)$, such that $\widehat{\beta} \prec \widehat{\alpha}$ and $o' = o(\widehat{\beta})$. Since $o \in |G_k|$ and $G \models l(o(\widehat{\beta}),\, o)$, by the construction of G, there is $\gamma \in \widehat{\beta}$ such that $|\gamma| < k$. Hence $\widehat{\gamma \cdot l} \in \mathcal{A}$. In addition, there must be $\rho \in Paths^k(\Delta)$ such that

$$\widehat{\rho} \prec \widehat{\gamma \cdot l} \quad \text{and} \quad o(\widehat{\rho}) = o.$$

Clearly, $\widehat{\gamma} = \widehat{\beta}$. Since $\widehat{\beta} \prec \widehat{\alpha}$, by Right-congruence, $\widehat{\gamma \cdot l} \prec \widehat{\alpha \cdot l}$. By Transitivity,

$$\widehat{\rho} \prec \widehat{\alpha \cdot l}.$$

Hence $o(\widehat{\rho}) \in inf(\alpha \cdot l)$. That is, $o \in inf(\alpha \cdot l)$.

This proves Claim 2.

3. If $G \models \varphi$, then $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

Let $\varphi = \forall x\,(\alpha(r,x) \rightarrow \beta(r,x))$. Since α and β are in $Paths^k(\Delta)$ and $G \models \varphi$, we have that $obj(\alpha) \subseteq obj(\beta)$. Hence by Claim 2, we have $inf(\alpha) \subseteq inf(\beta)$. Since $o(\widehat{\alpha}) \in inf(\alpha)$, $o(\widehat{\alpha}) \in inf(\beta)$. Therefore, $\widehat{\alpha} \prec \widehat{\beta}$ by the definition of inf. Hence $\Sigma \vdash_{\mathcal{I}_*} \varphi$.

This shows that if the domain of each base type in $T(\Delta)$ is infinite, then Claim 1 holds. As in the proof of Lemma 2.10, it can be shown that Claim 1 also holds if some base types in $T(\Delta)$ have finite domains.

This completes the proof of Lemma 3.2. ∎

4 Conclusions

We have investigated the path constraints introduced and studied in [3, 8, 9] for typed data. The type system or schema definition can be viewed as imposing a type constraint on the data. We have shown that the type constraints interact with the path constraints. As a result, in general we can no longer expect results developed for semistructured data to hold when a type is imposed on the data. Indeed, we have shown that the proof given in [3] for the decidability of word constraint implication in semistructured data breaks down in the presence of type constraints, and only in restricted type systems do we have decidability results on word constraint implication.

In particular, we have investigated word constraint implication in two restricted yet practical object-oriented models: a "generic" object-oriented type system and a type system based on ACeDB [19]. In these models, we have presented abstractions of the databases in terms of first-order logic, and we have established the decidability of word constraint implication.

Acknowledgements. The authors thank Victor Vianu, Val Tannen and Susan Davidson for helpful discussions.

References

[1] S. Abiteboul. "Querying semi-structured data". In *Proc. ICDT*, 1997.

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

[3] S. Abiteboul and V. Vianu. "Regular path queries with constraints", In *Proc. ACM Symp. on Principles of Database Systems*, 1997.

[4] C. Beeri. "A formal approach to object-oriented databases". *IEEE Trans. on Knowledge and Data Engineering*, 5: 353-382, 1990.

[5] E. Börger, E. Grädel, and Y. Gurevich. *The classical decision problem*. Springer, 1997.

[6] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. "Adding structure to unstructured data". In *Proc. ICDT*, 1997.

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

[8] 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.

[9] 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.

[10] R. G. G. Cattell (ed.). *The object-oriented standard: ODMG-93* (Release 1.2). Morgan Kaufmann, San Mateo, California, 1996.

[11] U. Dayal. "Queries and views in an object-oriented data model". In *Proc. 2nd DBPL*, pp. 80-102, 1989.

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

[13] E. Grädel, P. Kolaitis, and M. Vardi. "On the decision problem for two-variable first-order logic". *Bulletin of Symbolic Logic*, 3(1): 53-69, March 1997.

[14] E. Grädel, M. Otto, and E. Rosen. "Two-variable logic with counting is decidable". Preprint, 1996.

[15] Anthony Kosky. *Transforming databases with recursive data structures*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, 1995.

[16] S. Nestorov, S. Abiteboul, and R. Motwani. "Inferring structure in semistructured data". In *Workshop on Management of Semistructured Data*, 1997.

[17] S. Nestorov, J. Ullman, J. Weiner, and S. Chawathe. "Representative objects: Concise representations of semistructured, hierarchical data". In *Proc. Thirteenth International Conf. on Data Engineering*, 1997.

[18] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. "Object exchange across heterogeneous information sources". In *Proc. Eleventh International Conf. on Data Engineering*, pp. 251-260, March 1995.

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