Path constraints in semistructured databases

Citation for published version:
Buneman, P, Fan, W & Weinstein, S 2000, 'Path constraints in semistructured databases', *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 146-193. https://doi.org/10.1006/jcss.2000.1710

Digital Object Identifier (DOI):
10.1006/jcss.2000.1710

Published In:
Journal of Computer and System Sciences

Path Constraints in Semistructured Databases

PETER BUNEMAN *

Department of Computer and Information Science, University of Pennsylvania
200 South 33rd Street, Philadelphia, PA 19104-6389
peter@central.cis.upenn.edu

WENFEI FAN †

Department of Computer and Information Sciences, Temple University
1805 N. Broad Street, Philadelphia, PA 19122-2583
fan@joda.cis.temple.edu

AND

SCOTT WEINSTEIN ‡

Department of Philosophy, University of Pennsylvania
433 Logan Hall, Philadelphia, PA 19104-6304
weinstein@linc.cis.upenn.edu

Abstract

We investigate a class of path constraints that is of interest in connection with both semistructured and structured data. In standard database systems, constraints are typically expressed as part of the schema, but in semistructured data there is no explicit schema and path constraints provide a natural alternative. As with structured data, path constraints on semistructured data express integrity constraints associated with the semantics of data and are important in query optimization. We show that in semistructured databases, despite the simple syntax of the constraints, their associated implication problem is r.e. complete and finite implication problem is co-r.e. complete. However, we establish the decidability of the implication and finite implication problems for several fragments of the path constraint language, and demonstrate that these fragments suffice to express important semantic information such as extent constraints, inverse relationships and local database constraints commonly found in object-oriented databases.

*Partly supported by the Army Research Office (DAAH04-95-1-0169) and NSF Grant CCR92-16122.

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

‡Supported by NSF Grant CCR-9403447.

1 INTRODUCTION

Path inclusion constraints have been studied by Abiteboul and Vianu in [5] for semistructured databases. In semistructured databases, the data is unconstrained by any type system or schema and typically has an irregular structure [2, 12]. The study of semistructured data has generated the development of new data models and query languages (e.g., [4, 14, 23, 33, 34]) appropriate to this form of data representation, which already exists in certain scientific data formats. Recently, XML (eXtensible Markup Language [11]) has emerged as a standard for data exchange on the World Wide Web. While a schema may be imposed on an XML document, it is not required, and XML data is usefully treated as semistructured data [20]. Certain kinds of integrity constraints found in object-oriented databases are also common in semistructured databases. Some of these can be expressed as path constraints introduced in [5].

To illustrate the kinds of constraints that we want to capture, let us first investigate the constraints that are commonly placed on object-oriented databases. Consider the following object-oriented schema (expressed in O_2 [6]):

```
class student{
      Name:     string;
      Taking:   set(course);
}

class course{
      CName:     string;
      Enrolled:  set(student);
}

Students:  set(student);
Courses:   set(course);
```

in which we assume that the declarations `Students` and `Courses` define (persistent) entry points into the database. As it stands, this declaration does not provide full information about the intended structure. Given such a database one would often expect the following informally stated constraints to hold:

(a) $\forall s \in \mathit{Students}\ \ \forall c \in s.\mathit{Taking}\ \ (c \in \mathit{Courses})$

(b) $\forall c \in \mathit{Courses}\ \ \forall s \in c.\mathit{Enrolled}\ \ (s \in \mathit{Students})$

That is, 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. We shall call such constraints *extent* constraints. It should be noted that there is a natural analogy between extent constraints and (unary) inclusion dependencies developed for relational databases.

We might also expect an *inverse relationship* to hold between `Taking` and `Enrolled`. Object-oriented databases differ in the ways they enable one to state and enforce extent constraints and inverse relationships. Compare, for example, O_2 [6] and ObjectStore [30].

Let us develop a more formal notation for describing such constraints. In our object-oriented database there are two sets of objects, `Students` and `Courses`. We express this in semistructured data by building a graph with a root node r and a node for each object. Edges connect the root to these object nodes, and these edges are labeled either `Students` or `Courses`. Edges emanating from these nodes indicate attributes or relationships with other objects and are appropriately labeled. For example, a node representing a student object has a single `Name` edge connected to a string node, and multiple `Taking` edges connected to course nodes. See Figure 1 for an example of such a graph.

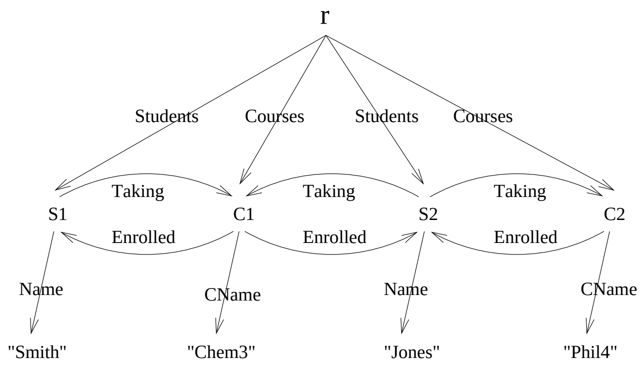

Figure 1: Representation of a student/course database

Using this representation of data we can examine certain kinds of constraints.

Extent Constraints. By taking edge labels as binary predicates, constraints of the form (a) and (b) above can be stated as:

$$\forall c\ (\exists s\ (\mathit{Students}(r,s) \wedge \mathit{Taking}(s,c)) \rightarrow \mathit{Courses}(r,c))$$

$$\forall s\ (\exists c\ (\mathit{Courses}(r,c) \wedge \mathit{Enrolled}(c,s)) \rightarrow \mathit{Students}(r,s))$$

Here r is a constant denoting the root node, and variables c, s range over vertices. The first constraint above states that any vertex that is reached from the root by following a `Students` edge followed by a `Taking` edge can also be reached from the root by following a `Courses` edge. Similarly, the second asserts that any vertex that is reached from the root by following a `Courses` edge followed by an `Enrolled` edge can also be reached from the root by following a `Students` edge.

These constraints are examples of "word constraints" studied in [5]; the implication problems for word constraints were shown to be decidable in semistructured databases there. Also studied in [5] was a form of constraints in which paths are represented by regular expressions. We do not consider this general form of constraints here.

Inverse Constraints. These are common in object-oriented databases [17]. With respect to our student/course schema, the inverse relationship between `Taking` and `Enrolled` is expressed as:

$$\forall s\ (\mathit{Students}(r,s) \rightarrow \forall c\ (\mathit{Taking}(s,c) \rightarrow \mathit{Enrolled}(c,s)))$$

$$\forall c\ (\mathit{Courses}(r,c) \rightarrow \forall s\ (\mathit{Enrolled}(c,s) \rightarrow \mathit{Taking}(s,c)))$$

The first constraint above states that for any student s

and any c, if c is reachable from s by following a `Taking` edge, then s is also reachable from c by following an `Enrolled` edge. Similarly, the second constraint asserts that for any course c and any s, if s is reachable from c by following an `Enrolled` edge, then c is also reachable from s by following a `Taking` edge. Such constraints cannot be expressed as word constraints or even by the more general path constraints given in [5].

Local Database Constraints. In database integration it is sometimes desirable to make one database a component of another database, or to build a "database of databases". Suppose, for example, we want to bring together a number of student/course databases as described above. We might write something like:

```
class School-DB{
   DB-identifier:  string;
   Students:set(student); // as defined above
   Courses: set(course);  // as defined above
}
Schools:  set(School-DB);
```

Now we may want certain constraints to hold on components of this database. For example, the "extent constraints" and "inverse constraints" described above now hold on each member of the `Schools` set. Here we refer to a component database such as a member of the set `Schools` as a *local database* and its constraints as *local database constraints*. Extending our graph representation by adding `Schools` edges from a new root node to the roots of local databases, the local extent and inverse constraints are:

$$\forall\, d\, (Schools(r,d) \to \forall c\, (\exists\, s\, (Students(d,s) \wedge Taking(s,c)) \to Courses(d,c)))$$

$$\forall\, d\, (Schools(r,d) \to \forall\, s\, (\exists\, c\, (Courses(d,c) \wedge Enrolled(c,s)) \to Students(d,s)))$$

$$\forall\, s\, (\exists\, d\, (Schools(r,d) \wedge Students(d,s)) \to \forall c\, (Taking(s,c) \to Enrolled(c,s)))$$

$$\forall\, c\, (\exists\, d\, (Schools(r,d) \wedge Courses(d,c)) \to \forall s\, (Enrolled(c,s) \to Taking(s,c)))$$

Again, these cannot be stated as word constraints or by the more general constraints of [5].

These considerations give rise to the question whether there is a natural generalization of the constraints of [5] which will capture these slightly more complicated forms. Here we consider a class of path constraints, P_c, of either the form

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

or the form

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

where $\alpha(x,y)$ ($\beta(x,y)$, $\gamma(x,y)$) represents a path, i.e., a sequence of edge labels, from node x to node y. As demonstrated above, $\alpha(x,y)$ can be expressed as a first-order logic formula with two free variables x and y by treating edge labels as binary predicates. The path constraint language P_c is a mild generalization of the class of word constraints studied in [5].

This class of path constraints can be used to express all the integrity constraints we have so far encountered. These constraints are not only a fundamental part of the semantics of the data, but are also important in query optimization. They have proven useful in a variety of database contexts, ranging from semistructured data such as data on the World Wide Web and in XML documents, to structured data as found in object-oriented databases. In particular, among the numerous proposals for adding structure or semantics to XML documents, several [10, 26, 31, 32] advocate the need for these integrity constraints. In standard database systems, integrity constraints are typically expressed as part of the schema, but in semistructured data there is no explicit schema and path constraints provide a natural alternative.

To illustrate how these constraints might be used in query optimization, consider again the student/course database given in Figure 1. Suppose, for example, we want to find the names of all the courses enrolled by students who are taking the course "Chem3". Without the inverse and extent constraints described above, one would write the query as Q_1 (in OQL syntax [17]):

Q_1
```
select distinct c.CName
from   Courses c,
       c.Enrolled s,
       s.Taking c'
where  c'.CName = "Chem3"
```

Given these inverse and extent constraints, one can show that Q_1 is equivalent to Q_2 given below:

Q_2
```
select distinct c.CName
from   Courses c',
       c'.Enrolled s,
       s.Taking c
where  c'.CName = "Chem3"
```

In other words, given these constraints, one can rewrite Q_1 to Q_2. In most cases, Q_2 is more efficient than Q_1. Indeed, Q_2 complies with the familiar optimization principle originating in relational database theory: performing selections as early as possible.

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, the most important theoretical question in connection with path constraints. In general, we may know that a set of path constraints is satisfied by a database. The question of logical implication is: what other path constraints are necessarily satisfied by the database? To see why logical implication is important, consider the queries Q_1 and Q_2 against the student/course database given above. To show that Q_1 can be rewritten to Q_2, the following constraints of P_c are also needed in addition to the given inverse and extent constraints:

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

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

To use these constraints, we need to show that they necessarily hold if the given extent and inverse constraints hold. That is, they are implied by the given path constraints.

There are two forms of implication problems associated with path constraints. Databases are usually considered to be finite. Logical implication is called *finite implication* for the case in which only finite database instances are permitted. It is also interesting to consider logical implication in the traditional logic framework in which infinite instances are also allowed. Logical implication is called *unrestricted implication*, or simply *implication*, for the case in which both finite database instances and infinite instances are permitted.

In the remainder of the paper, we investigate the implication and finite implication problems associated with path constraints of P_c in the context of semistructured data. Surprisingly, the implication problems for this mild generalization of word constraints are undecidable, whereas the implication problems for word constraints are decidable in PTIME [5]. However, certain restricted cases are decidable, and these cases are sufficient to express at least the constraints we have described above.

Related work. There is a natural analogy between the work on path constraints and inclusion dependency theory developed for relational databases (see, e.g., [3] for an in-depth presentation of inclusion dependency theory). Path constraints specify inclusions among certain sets of objects, and can be viewed as a generalization of inclusion dependencies. Inclusion dependencies have proven useful in semantic specification and query optimization for relational databases. In the same way, path constraints are important in a variety of database contexts, ranging from semistructured data to object-oriented databases.

Another form of constraints defined in terms of navigation paths, called *path functional dependencies*, has been studied by Weddell, et al. [8, 29]. These constraints differ significantly from the path constraints investigated here because they are a generalization of functional dependencies for a restricted type system, while P_c constraints can be viewed as a generalization of inclusion dependencies for both semistructured and structured databases.

Closer to the work reported here is the path inclusion constraint language introduced and investigated by Abiteboul and Vianu in [5]. A constraint in this language is an expression of the form $p \subseteq q$ or $p = q$, where p and q are regular expressions representing paths. In particular, if p and q are simply paths, i.e., sequences of edge labels, the constraint is called a *word constraint*. Such a constraint expresses the inclusion or equality relation between the two sets of nodes reachable along p and q. The decidability of the implication problems for this language was established for semistructured data in [5]. In addition, it was also shown there that word constraint implication is decidable in PTIME. This constraint language differs from the constraint language P_c in expressive power. On the one hand, the language of [5] allows a more general form of path expressions than P_c. On the other hand, it cannot express inverse and local database constraints, whereas these constraints are expressible in P_c.

Recently, the application of integrity constraints to query optimization was also studied by Popa and Tannen in [35]. Among other things, [35] developed an equational theory for query rewriting by using a certain form of constraints. Semantic optimization has also been investigated for semistructured databases in [13, 24] and for structured databases in [18, 19, 25].

Another issue is the interaction between path constraints and types. Structured data, e.g., data in object-oriented databases, is constrained by a schema, in which both types and integrity constraints are specified. In addition, although the XML standard itself does not require any type system, a number of proposals [10, 26, 32] have been developed that roughly correspond to data definition languages. These allow one to constrain the structure of XML data by imposing a type on it. These and other proposals (e.g., [31]) also advocate the need for integrity constraints, which can be expressed as path constraints. The type system or schema definition may also be viewed as imposing a constraint on the data. It is a constraint of a different form. That is, type constraints cannot be expressed as path constraints and *vice versa*. In structured data and possibly in XML documents both forms of constraints are present, and therefore, we need to understand the interaction between them. In general we can no longer expect results developed for semistructured data to hold when a type is imposed on the data. In other words, the imposition of a type can alter the computational complexity of the path constraint implication problem in unexpected ways. Indeed, in [16] we have shown 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, some decidability results on path constraint implication developed for semistructured data break down when some type system is added, and on the other hand, some undecidability results on untyped data also collapse when some type constraint is imposed. This issue was first addressed in [15] and then treated in detail in [16].

Organization. The remainder of the paper is organized as follows. Section 2 formally presents our path constraint language P_c. Section 3 establishes the undecidability of the implication and finite implication problems associated with P_c in the context of semistructured databases. Section 4 identifies several fragments of P_c, and shows that the implication and finite implication problems for each of these fragments are decidable in semistructured databases. It also demonstrates that these fragments suffice to express many important integrity constraints such as extent, inverse and local database constraints. Finally, Section 5 summarizes our results.

2 PATH CONSTRAINTS

In this section, we first present an abstraction of semistructured databases in terms of first-order logic, and then define paths and path constraints of P_c.

2.1 Semistructured Databases

Semistructured data is usually represented as an edge-labeled (rooted) directed graph, e.g., in UnQL [14] and in OEM [4, 34]. See [2, 12] for surveys of semistructured data models. Along the same lines, here we use an abstraction of semistructured databases as (finite) first-order logic structures of a relational signature

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

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

We specify a *σ-structure* G by giving $(|G|,\ r^G,\ E^G)$, where

- $|G|$ is a set called the *universe* (*domain*) of G, and elements of $|G|$ are called the *nodes* (*vertices*) of G;
- r^G is a distinguished element of $|G|$, called the *root node* of G;
- E^G is a finite set of binary relations on $|G|$, each of which is named by a relation symbol of E. For any $K \in E$, we write K^G for the relation in G named by K.

Structure G can be naturally depicted as a rooted edge-labeled directed graph with $|G|$ as the set of vertices, E^G the set of labeled edges and r^G the root. For any $K \in E$ and $a, b \in |G|$, there is an edge labeled K from a to b in the graph if and only if $(a, b) \in K^G$.

It should be mentioned that we do not assume the reachability of all nodes from the root in a σ-structure (graph). However, none of our results or proofs are affected if reachability is enforced.

2.2 Paths

A path, i.e., a sequence of labels, can be represented as a logic formula with two free variables. More specifically, a path is a first-order logic formula $\alpha(x, y)$ of one of the following forms:

- $x = y$, denoted by $\epsilon(x, y)$ and called an *empty path*;
- $K(x, y)$, where $K \in E$; or
- $\exists z(K(x, z) \wedge \beta(z, y))$, 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. We write $\alpha(x,\ y)$ as α when the parameters x and y are clear from the context. In particular, we may replace free variable x or y by r, where r is the constant denoting the root given in signature σ. That is, we use $\alpha(r,\ y)$ or $\alpha(x,\ r)$ to denote a path from or to the root.

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

$$\exists z\, (Students(x,\ z) \wedge Taking(z,\ y))$$

$$\exists z\, (Courses(x,\ z) \wedge \exists w\, (Enrolled(z,\ w) \wedge Taking(w,\ y)))$$

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

- $\beta(x,y)$, if $\alpha = \epsilon$;
- $\exists z\,(K(x,z) \wedge \beta(z,y))$, if $\alpha = K$ for some $K \in E$;
- $\exists u\,(K(x,u) \wedge (\alpha'(u,z) \cdot \beta(z,y)))$, if $\alpha(x,z)$ is of the form $\exists u\,(K(x,u) \wedge \alpha'(u,z))$, where $K \in E$ and α' is a path.

For example, the paths above can be written as:

$$Students \cdot Taking(x,y)$$
$$Courses \cdot Enrolled \cdot Taking(x,y)$$

We use $(\alpha)^m$ to denote the m-time concatenations of α, defined by:

$$(\alpha)^m = \begin{cases} \epsilon & \text{if } m = 0 \\ \alpha \cdot (\alpha)^{m-1} & \text{otherwise} \end{cases}$$

A path ρ is said to be a *proper prefix* of ϱ, denoted by $\rho \prec_p \varrho$, iff there exists a path λ such that $\lambda \neq \epsilon$ and $\varrho = \rho \cdot \lambda$. A path ρ is said to be a *prefix* of ϱ, denoted by $\rho \preceq_p \varrho$, iff $\rho \prec_p \varrho$ or $\rho = \varrho$. Similarly, ρ is said to be a *suffix* of ϱ, denoted by $\rho \preceq_s \varrho$, iff there exists λ such that $\varrho = \lambda \cdot \rho$.

For example, the path $Courses \cdot Enrolled \cdot Taking$ has the following prefixes: the empty path ϵ, $Courses$, $Courses \cdot Enrolled$ and itself. Its suffixes include ϵ, $Taking$, $Enrolled \cdot Taking$ and itself.

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

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

For example, $|Courses \cdot Enrolled \cdot Taking| = 3$ and $|Students \cdot Taking| = 2$.

In particular, a path of the form $\alpha(r,x)$ or $\alpha(x,r)$, i.e., a path from or to the root, can be expressed as a first-order logic formula with at most two *distinct* variables. For example, the path

$$Students \cdot Taking \cdot Enrolled \cdot Taking(r,\, x)$$

can be expressed as:

$$\exists y\ (Taking(y,\, x) \wedge \exists x\ (Enrolled(x,\, y) \wedge \exists y\ (Taking(y,\, x) \wedge Students(r,\, y))))$$

Observe that this logic formula uses only two distinct variables. In general, a path $\alpha(x,\, y)$ can be expressed as a first-order logic formula with at most three distinct variables.

2.3 Path Constraint Language P_c

By using path formulas, the path constraint language P_c is formalized as follows.

Definition 2.1: A *path constraint* φ is an expression of either the *forward* form

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

or the *backward* form

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

where α, β, γ are paths, called the *prefix*, *left tail* and *right tail* of φ, and denoted by $pf(\varphi)$, $lt(\varphi)$ and $rt(\varphi)$, respectively.

A path constraint is called a *forward constraint* if it is of the forward form, and is called a *backward constraint* if it is of the backward form.

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

For example, all the path constraints we have seen in Section 1 are P_c constraints. Among these, the extent and local extent constraints are examples of forward constraints, while the inverse and local inverse constraints are backward constraints. By using path concatenation "$\cdot$", we may represent these constraints in a simpler form. For example, the extent constraints given in Section 1 can be rewritten as:

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

A forward constraint of P_c asserts that for any vertex x that is reached from the root r by following path α and for any vertex y that is reached from x by following path β, y is also reachable from x by following path γ. Similarly, a backward P_c constraint states that for any x that is reached from r by following α and for any y that is reached from x by following β, x is also reachable from y by following γ.

As demonstrated in Section 1, path constraints of P_c are capable of expressing, among other things, extent, inverse and local database constraints.

Next, we identify several special subclasses of P_c.

We call a path constraint φ of P_c a *simple (path) constraint* if $pf(\varphi) = \epsilon$. That is, the prefix of φ is an empty path. More specifically, φ is of either the form

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

or the form

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

The set of all simple path constraints is denoted by P_s.

A proper subclass of simple path constraints, called *word constraints*, was introduced and investigated in [5]. A word constraint can be represented as

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

where β and γ are paths. The set of all word constraints is denoted by P_w.

In other words, a word constraint is a simple forward path constraint of P_c. As demonstrated in Section 1, extent constraints can be expressed as word constraints. However, inverse and local database constraints are not expressible in P_w.

We borrow the standard notions of model and implication from first-order logic [22].

Let G be a σ-structure and φ a P_c constraint. We use $G \models \varphi$ to denote that G *satisfies* φ (i.e., G *is a model of* φ). Let Σ be a set of P_c constraints. We use $G \models \Sigma$ to denote that G *satisfies* Σ (i.e., G *is a model of* Σ). That is, for every $\phi \in \Sigma$, $G \models \phi$.

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

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

As observed by [5], every word constraint (in fact, every simple path constraint) can be expressed by a sentence in two-variable first-order logic (FO^2), the fragment of first-order logic consisting of all relational sentences with at most two distinct variables. Recently, Grädel, Kolaitis and Vardi [27] have 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 follows immediately. In fact, [5] directly established (without reference to the embedding into FO^2) that the implication and finite implication problems for word constraints are in PTIME.

In contrast to word constraints, many path constraints of P_c are not expressible in FO^2.

Example 2.1: Consider the structures G and G' given in Figure 2. It is easy to verify, using the 2-pebble Ehrenfeucht-Fraïssé style game [7, 21, 28], that G and G' are equivalent in FO^2. However, G and G' are distinguished by the path constraint

$$\varphi = \forall\, x\, (K(r,x) \to \forall\, y\, (K(x,y) \to K \cdot K(x,y))),$$

because $G \models \varphi$ but $G' \not\models \varphi$. This shows that φ is not expressible in FO^2. ■

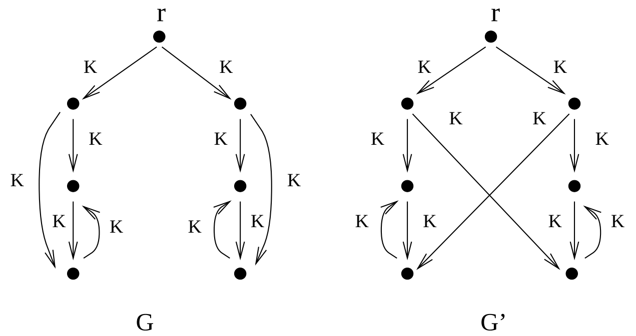

Figure 2: Structures distinguishable by P_c

The central technical problems investigated in this paper are the implication and finite implication problems for P_c, and fragments thereof, in the context of semistructured databases.

3 UNDECIDABLE IMPLICATION PROBLEMS

In this section, we show that despite the simple syntax of P_c, the implication and finite implication problems for P_c are undecidable in the context of semistructured databases.

Theorem 3.1: The implication problem for P_c is r.e. complete, and the finite implication problem for P_c is co-r.e. complete. ■

In fact, these undecidability results also hold for two proper subclasses of P_c. One of the subclasses, P_f, is the set of all the constraints of P_c having the forward form. The other, P_+, is the set

$$\{\varphi \mid \varphi \in P_c,\ lt(\varphi) \neq \epsilon,\ rt(\varphi) \neq \epsilon\},$$

where $lt(\varphi)$ and $rt(\varphi)$ are described in Definition 2.1. The set P_+ is the largest subset of P_c without equality.

For P_+ and P_f we have the following theorems, from which Theorem 3.1 follows immediately.

Theorem 3.2: The implication problem for P_+ is r.e. complete, and the finite implication problem for P_+ is co-r.e. complete. ■

Theorem 3.3: The implication problem for P_f is r.e.

complete, and the finite implication problem for P_f is co-r.e. complete. ■

To prove Theorem 3.2, we consider the satisfiability and finite satisfiability problems corresponding to P_f constraint implication. First recall the following.

Let X be a recursive class of logic sentences. The *satisfiability problem* for X is the problem of determining, given any $\psi \in X$, whether ψ has a model. The *finite satisfiability problem* for X is to determine, given any $\psi \in X$, whether ψ has a finite model.

The (finite) implication problem for P_f corresponds to the (finite) satisfiability problem for the following set:

$$S(P_+) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_+,\ \Sigma \subset P_+,\ \Sigma \text{ is finite}\}.$$

More specifically, to prove Theorem 3.2, it suffices to show that the satisfiability problem for $S(P_+)$ is co-r.e. complete and the finite satisfiability problem for $S(P_+)$ is r.e. complete. The idea of the proof is to show that there exists a conservative reduction from the set of all first-order logic sentences to $S(P_+)$. To do this, we establish a reduction from the halting problem for two-register machines.

Along the same lines, to prove Theorem 3.3 we consider the set

$$S(P_f) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_f,\ \Sigma \subset P_f,\ \Sigma \text{ is finite}\}.$$

We show that there exists a conservative reduction from the set of all first-order logic sentences to $S(P_f)$. Again, this is established by reduction from the halting problem for two-register machines.

We prove Theorems 3.2 and 3.3 in Sections 3.2 and 3.3, respectively. Before we present these proofs, we first recall the definitions of conservative reductions and two-register machines (2-RMs. See, e.g., [1, 9]).

3.1 Conservative Reduction and 2-RM

We first review the notion of conservative reductions. To do so, we borrow the following notations from [1, 9].

Let X be a class of sentences. We write $N(X)$ for the set of all *unsatisfiable* sentences in X, i.e.,

$$N(X) = \{\psi \mid \psi \in X,\ \psi \text{ does not have a model}\},$$

and $F(X)$ for the set of all *finitely satisfiable* sentences in X, i.e.,

$$F(X) = \{\psi \mid \psi \in X,\ \psi \text{ has a finite model}\}.$$

We write FO for the set of all first-order sentences.

Conservative reductions are defined as follows.

Definition 3.1 [9]: Let X and Y be recursive classes of sentences. A *conservative reduction* from X to Y is a recursive function $f : X \to Y$ such that for any $\psi \in X$,

- ψ is satisfiable iff $f(\psi)$ is satisfiable; and
- ψ is finitely satisfiable iff $f(\psi)$ is finitely satisfiable.

A recursive class of sentences X is said to be a *conservative reduction class* if there exists a conservative reduction from FO to X. ■

Recall that the satisfiability problem for FO is well known to be co-r.e. complete, and the finite satisfiability problem for FO is r.e. complete. Hence, if a recursive class of sentences X is a conservative reduction class, then,

- the satisfiability problem for X is co-r.e. complete; and
- the finite satisfiability problem for X is r.e. complete.

As a result, to show Theorems 3.2 and 3.3, it suffices to show that $S(P_+)$ and $S(P_f)$ are conservative reduction classes.

To show that a recursive subset X of FO is a conservative reduction class, it suffices to reduce $N(FO)$ and $F(FO)$ to $N(X)$ and $F(X)$, respectively. This is described by the notion of semi-conservative reductions.

Definition 3.2 [9]: Let X and Y be recursive classes of sentences. A *semi-conservative reduction* from X to Y is a recursive function $f : X \to Y$ such that

- $f(N(X)) \subseteq N(Y)$; and
- $f(F(X)) \subseteq F(Y)$.

■

Lemma 3.4 [9]: If there exists a semi-conservative reduction from FO to a recursive subset X of FO, then X is a conservative reduction class. ■

Hence, to show Theorems 3.2 and 3.3, it suffices to establish the existence of semi-conservative reductions from FO to $S(P_+)$ and $S(P_f)$.

We shall proceed to construct the semi-conservative reductions by making use of the halting problem for two-register machines. Before we present the construction, we first review the notion of two-register machines.

A two-register machine (2-RM) M has two registers $register_1, register_2$, and is programmed by a numbered sequence $I_0, I_1, ..., I_l$ of instructions. Each register contains a natural number. An *instantaneous description*

(ID) of M is (i, m, n), where $i \in [0, l]$, m and n are natural numbers. It indicates that M is ready to execute instruction I_i (or at "state i") with $register_1$ and $register_2$ containing m and n, respectively.

An instruction I_i of M can be either an *addition* or a *subtraction*, which defines a relation $\rightarrow_M$ between IDs, described as follows:

- *addition*: (i, rg, j), where rg is either $register_1$ or $register_2$, and $0 \leq i, j \leq l$. Its semantics is: at state i, M adds 1 to the content of rg, and then goes to state j. Accordingly:

$$(i, m, n) \rightarrow_M \begin{cases} (j, m+1, n) & \text{if } rg = register_1 \\ (j, m, n+1) & \text{otherwise} \end{cases}$$

- *subtraction*: (i, rg, j, k), here rg is either $register_1$ or $register_2$, and $0 \leq i, j, k \leq l$. Its semantics is: at state i, M tests whether the content of rg is 0, and if it is, then goes to state j; otherwise M subtracts 1 from the content of rg and goes to the state k. Accordingly:

$$(i, m, n) \rightarrow_M \begin{cases} (j, 0, n) & \text{if } rg = register_1 \\ & \text{ and } m = 0 \\ (k, m-1, n) & \text{if } rg = register_1 \\ & \text{ and } m \neq 0 \\ (j, m, 0) & \text{if } rg = register_2 \\ & \text{ and } n = 0 \\ (k, m, n-1) & \text{if } rg = register_2 \\ & \text{ and } n \neq 0 \end{cases}$$

The relation $\rightarrow_M$ can be understood as a set of rewrite rules for IDs. We use $\Rightarrow_M$ to denote the reflexive and transitive closure of $\rightarrow_M$. The relation of *M-reachability* $C \Rightarrow_M D$ holds just in case M, started from ID C, reaches ID D by application of zero or more $\rightarrow_M$ rules.

A two-register machine may halt at some states. Without loss of generality, one can assume that a halting state has zeros in both registers. That is, halting IDs have the form $(i,\ 0,\ 0)$, where i is a halting state and $0 \leq i \leq l$.

Recall the following well-known result.

Lemma 3.5 [36]: There exists an effective partial procedure by which, given a sentence in FO, we can test whether it has no model, a finite model, or only infinite models. The procedure terminates in the first two cases, but does not terminate in the last case. ∎

We fix M_L to be a 2-RM with the following behavior (the existence of such a machine follows from the result just quoted. See [1, 9] for further discussion). The 2-RM M_L has two halting states: $(1, 0, 0)$ and $(2, 0, 0)$. For each $\psi \in FO$, let $m(\psi)$ be an appropriate encoding of ψ (a natural number) and $C(\psi)$ be the ID $(0, m(\psi), 0)$ of M_L. Started from $C(\psi)$,

- M_L halts at (1, 0, 0) iff ψ is not satisfiable; and
- M_L halts at (2, 0, 0) iff ψ has a finite model.

In other words, M_L has the following property: for $i = 1, 2$, let

$$H_{M_L,i} = \{\psi \mid \psi \in FO,\ C(\psi) \Rightarrow_{M_L} (i, 0, 0)\}.$$

Then $H_{M_L,1}$ is $N(FO)$ and $H_{M_L,2}$ is $F(FO)$.

If we can encode the description and computations of this 2-RM in terms of path constraints, we can transform certain decision problems regarding FO sentences to the problems for path constraints. More specifically, the idea of the proof of Theorem 3.2 is to encode the description and computations of M_L in terms of P_+ constraints. Using this encoding, we are able to define a recursive function $f : FO \rightarrow S(P_+)$ such that for each $\psi \in FO$,

1. if $\psi \in H_{M_L,1}$, then $f(\psi)$ is not satisfiable; and
2. if $\psi \in H_{M_L,2}$, then $f(\psi)$ has a finite model.

That is, f is a semi-conservative reduction from FO to $S(P_+)$.

We can prove Theorem 3.3 along the same lines.

3.2 Implication Problems for P_+

Next, we prove Theorem 3.2. It suffices to show that $S(P_+)$ is a conservative reduction class. By Lemma 3.4, to establish the conservative reduction class property for $S(P_+)$, it is sufficient to show that there is a semi-conservative reduction from FO to $S(P_+)$.

We establish the existence of the semi-conservative reduction by reduction from the halting problem for 2-RMs. To do this, we first present an encoding of 2-RMs in terms of constraints in P_+, and then prove a reduction property of the encoding. Using this reduction property, we define a semi-conservative reduction from FO to $S(P_+)$.

3.2.1 Encoding

We encode the IDs, the contents of the registers and the instructions of a 2-RM in terms of P_+ constraints.

Let M be a 2-RM. Assume that M is programmed by

$$I_0,\ I_1,\ \ldots,\ I_l.$$

Without loss of generality, we also assume that the set E of binary relation symbols in signature σ includes:

- predicates encoding the states of M:
  - $K_0, K_1, ..., K_l$,
  - $K_0^-, K_1^-, ..., K_l^-$;
- predicates encoding the contents of the registers:
  - R_1^+, R_1^-: to encode the successor and predecessor of the content of $register_1$;
  - R_2^+, R_2^-: to encode the successor and predecessor of the content of $register_2$;
  - E_{01}, E_{01}^-: to indicate that $register_1$ is 0;
  - E_{02}, E_{02}^-: to indicate that $register_2$ is 0;
- predicates distinguishing $register_1$ from $register_2$ and identifying the root r:
  - L_1, L_1^-: to identify $register_1$;
  - L_2, L_2^-: to identify $register_2$; and
  - L_r: to identify the root r.

We should remark that all these predicates are binary. Using these predicates, we intend to construct structures of the form shown in Figure 3 (E_{01}^-, E_{02}^-, L_1^-, L_2^-, R_1^-, R_2^-, K_i^- edges are omitted in the graph). Figure 3 illustrates the encoding of the 2-RM M. It has (at least) two chains from the root node rt. One starts with an edge labeled E_{01} followed by a sequence of R_1^+ edges. The nodes in the chain are denoted by natural numbers and intend to represent the contents of $register_1$ of M. The R_1^+ edges can be viewed as the successor relation on the contents of $register_1$. In addition, there are R_1^- edges (not shown in the graph), which form the inverse relation of R_1^+ edges and can be viewed as the predecessor relation on the contents of $register_1$. The E_{01} edge indicates that $register_1$ has 0. There is also an E_{01}^- edge (not shown in the graph), which is the inverse of E_{01}. To each node in the chain there is an edge labeled L_1 from the root rt. These L_1 edges are used to identify $register_1$. There are also L_1^- edges (not shown in the graph), which are the inverse of L_1 edges. Similarly, the other chain starts with an edge labeled E_{02} followed by a sequence of R_2^+ edges. It encodes the contents of $register_2$. Moreover, for each $i \in [0, l]$, there are K_i edges from the nodes in the chain encoding $register_1$ to the nodes in the chain representing $register_2$. For example, as shown in Figure 3, there is a K_i edge from m to n'. This indicates that an ID of M is (i, m, n). For the ease of encoding, we also have K_i^- edges (not shown in the graph), which form the inverse relation of

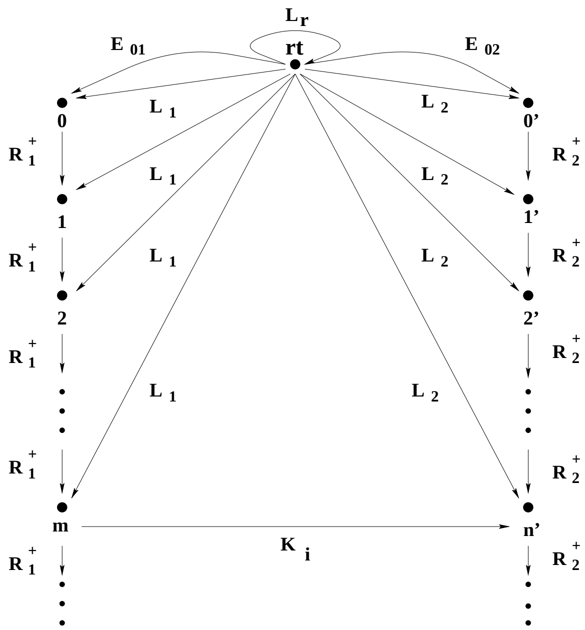

Figure 3: A structure depicting 2-RM encoding

K_i edges. Finally, there is an edge labeled L_r from rt to rt, which is used to identify the root.

The above requirements on the structure encoding the computations of the 2-RM M can be expressed by P_+ constraints. We should remark here that we need not require the structure to consist of only these two chains. Indeed, the structure may have many such chains and others. To prove our results, it suffices that our structure has at least two chains with the properties mentioned above.

We now present the encoding of M in terms of P_+ constraints.

IDs. We encode each ID $C = (i, m, n)$ of M by φ_C:

$$\begin{aligned}&\forall x\, (L_1(r, x) \rightarrow \\ &\quad \forall y\, ((R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n (x, y) \rightarrow K_i(x, y))),\end{aligned}$$

where $(\alpha)^m$ stands for the m-time concatenations of α, as defined in Section 2. It should be noted that φ_C is a forward constraint in P_+ with $pf(\varphi_C) = L_1$, $lt(\varphi_C) = (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n$, and $rt(\varphi_C) = K_i$, where pf, lt and rt are described in Definition 2.1.

Observe that we require the contents of $register_1$ and $register_2$ to be encoded in a single path $lt(\varphi_C)$. This leads to a lack of symmetry in the treatment of the two registers in the encoding. In particular, the content of $register_1$, encoded as $(R_1^-)^m$, is a prefix of $lt(\varphi_C)$, and the content of $register_2$, encoded as $(R_2^+)^n$, is a suffix of $lt(\varphi_C)$.

Registers. We encode the contents of the registers by Φ_N, which is the conjunction of the constraints of P_+ given below.

- Successor, predecessor:

$$\begin{aligned}
\phi_1 &= \forall x\,(L_1(r,x) \to \forall y\,(R_1^+(x,y) \to R_1^-(y,x)))\\
\phi_2 &= \forall x\,(L_1(r,x) \to \forall y\,(R_1^-(x,y) \to R_1^+(y,x)))\\
\phi_3 &= \forall x\,(L_2(r,x) \to \forall y\,(R_2^+(x,y) \to R_2^-(y,x)))\\
\phi_4 &= \forall x\,(L_2(r,x) \to \forall y\,(R_2^-(x,y) \to R_2^+(y,x)))\\
\phi_5 &= \forall x\,(L_1(r,x) \to R_1^+ \cdot L_1^-(x,r))\\
\phi_6 &= \forall x\,(L_2(r,x) \to R_2^+ \cdot L_2^-(x,r))
\end{aligned}$$

  These are backward constraints. Constraints ϕ_1 and ϕ_2 (resp. ϕ_3 and ϕ_4) specify that R_1^+ and R_1^- (resp. R_2^+ and R_2^-) are inverse to each other. Constraints ϕ_5 and ϕ_6 assert that the contents of $register_1$ and $register_2$ always have successors.

- Register identification:

$$\begin{aligned}
\phi_7 &= \forall x\,(L_1 \cdot R_1^+(r,x) \to L_1(r,x))\\
\phi_8 &= \forall x\,(L_1 \cdot R_1^-(r,x) \to L_1(r,x))\\
\phi_9 &= \forall x\,(L_2 \cdot R_2^+(r,x) \to L_2(r,x))\\
\phi_{10} &= \forall x\,(L_2 \cdot R_2^-(r,x) \to L_2(r,x))
\end{aligned}$$

  These are simple forward constraints. They ensure that for each node coding a content of $register_1$, there is always an edge labeled L_1 from the root to it. Similarly, for any node representing a content of $register_2$, there is an edge labeled L_2 from the root to it.

- States: for $i \in [0,l]$,

$$\begin{aligned}
\phi_{11}^i &= \forall x\,(L_1(r,x) \to \forall y\,(K_i(x,y) \to K_i^-(y,x)))\\
\phi_{12}^i &= \forall x\,(L_2(r,x) \to \forall y\,(K_i^-(x,y) \to K_i(y,x)))
\end{aligned}$$

  These are backward constraints. They assert that there is an inverse relationship between K_i and K_i^- for each $i \in [0,l]$.

- Zeros:

$$\begin{aligned}
\phi_{13} &= \forall x\,(L_1(r,x) \to \forall y\,(E_{01}^-(x,y) \to E_{01}(y,x)))\\
\phi_{14} &= \forall x\,(L_1 \cdot E_{01}^-(r,x) \to L_r(r,x))\\
\phi_{15} &= \forall x\,(L_r \cdot E_{01}(r,x) \to E_{01}(r,x))\\
\phi_{16} &= \forall x\,(L_1 \cdot E_{01}^- \cdot E_{02}(r,x) \to E_{02}(r,x))\\
\phi_{17} &= \forall x\,(E_{01}(r,x) \to L_1(r,x))\\
\phi_{18} &= \forall x\,(E_{02}(r,x) \to L_2(r,x))
\end{aligned}$$

  Constraints ϕ_{13}, ϕ_{14} and ϕ_{15} assert that if there is an edge labeled L_1 from the root to a node a and a has an outgoing edge labeled E_{01}^-, then there is an edge labeled E_{01} from the root to a. Constraint ϕ_{16} ensures that if there exists a path $L_1 \cdot E_{01}^- \cdot E_{02}$ from the root to a node b, then there is an E_{02} edge from the root to b. Constraint ϕ_{17} states that there is an edge labeled L_1 from the root to a node coding 0 in $register_1$. Similarly, ϕ_{18} states that there is an edge labeled L_2 from the root to a node coding 0 in $register_2$.

It should be mentioned that the constraints given above enforce stronger properties than necessary. Some of these constraints are not used in the proofs of our results. We retain these constraints to simply the constructions below.

Instructions. For each $i \in [0,l]$, we encode the instruction I_i by ϕ_{I_i} given below. Constraint ϕ_{I_i} describes the relation $\to_M$ presented in Section 3.1.

- Addition:

  For $(i, register_1, j)$, ϕ_{I_i} is

$$\begin{aligned}
\phi_{a_1}^i = \forall x\,(L_1(r,x) \to{}&\\
&\forall y\,(R_1^- \cdot K_i(x,y) \to K_j(x,y))).
\end{aligned}$$

  For $(i, register_2, j)$, ϕ_{I_i} is

$$\begin{aligned}
\phi_{a_2}^i = \forall x\,(L_1(r,x) \to{}&\\
&\forall y\,(K_i \cdot\ R_2^+(x,y) \to K_j(x,y))).
\end{aligned}$$

  Note that $\phi_{a_1}^i$ and $\phi_{a_2}^i$ are forward constraints.

- Subtraction:

  For $(i, register_1, j, k)$, ϕ_{I_i} is $\phi_{s_1}^i = \phi_{s_{1,0}}^i \wedge \phi_{s_{1,n}}^i$, where

$$\begin{aligned}
\phi_{s_{1,0}}^i &= \forall x\,(E_{01}(r,x) \to \forall y\,(K_i(x,y) \to K_j(x,y))),\\
\phi_{s_{1,n}}^i &= \forall x\,(L_1(r,x) \to\\
&\qquad \forall y\,(R_1^+ \cdot K_i(x,y) \to K_k(x,y))).
\end{aligned}$$

  Note that $\phi_{s_{1,0}}^i$ and $\phi_{s_{1,n}}^i$ are forward constraints.

  For $(i, register_2, j, k)$, ϕ_{I_i} is $\phi_{s_2}^i = \phi_{s_{2,0}}^i \wedge \phi_{s_{2,n}}^i$, where

$$\begin{aligned}
\phi_{s_{2,0}}^i &= \forall x\,(E_{02}(r,x) \to \forall y\,(K_i^-(x,y) \to K_j(y,x))),\\
\phi_{s_{2,n}}^i &= \forall x\,(L_1(r,x) \to\\
&\qquad \forall y\,(K_i \cdot R_2^-(x,y) \to K_k(x,y))).
\end{aligned}$$

  Here $\phi_{s_{2,0}}^i$ is a backward constraint and $\phi_{s_{2,n}}^i$ is a forward constraint.

The encoding of the program of M is $\Phi_M = \bigwedge_{i=0}^{l} \phi_{I_i}$. Clearly, Φ_M is a conjunction of path constraints in P_+.

Using the encoding given above, we are able to express the M-reachability problem $C \Rightarrow_M D$ as a logical implication problem for P_+ constraints. More specifically, we show that the encoding above has the following reduction property.

Proposition 3.6: For all IDs C and D of M,

$$C \Rightarrow_M D \quad \text{iff} \quad \Phi_N \wedge \Phi_M \wedge \varphi_C \rightarrow \varphi_D \text{ is valid.}$$

■

Proof: The proof consists of two parts.

(1) Assume $C \Rightarrow_M D$. We show that for each model G of $\Phi_N \wedge \Phi_M \wedge \varphi_C$, $G \models \varphi_D$. To show this, it suffices to show that for each natural number t and each ID C' of M, if C' is reached by M in t steps starting from C (denoted by $C \Rightarrow_M^t C'$), then $G \models \varphi_{C'}$. We prove this claim by induction on t.

Base case: If $t = 0$, then the claim holds since $G \models \varphi_C$.

Inductive step: Assume the claim for t.

Suppose $C \Rightarrow_M^t C_1 \rightarrow_M^{I_i} C'$, where $C_1 = (i, m, n)$, and $C_1 \rightarrow_M^{I_i} C'$ means that C' is reached by executing instruction I_i at C_1. Then by the induction hypothesis, we have $G \models \varphi_{C_1}$. That is

$$\begin{aligned} G \models & \forall x\, (L_1(r, x) \rightarrow \\ & \forall y\, ((R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(x, y) \rightarrow K_i(x, y))). \end{aligned}$$

We argue by contradiction that the claim holds for $t+1$. Suppose $G \not\models \varphi_{C'}$. We show that this assumption leads to a contradiction in each case of I_i, which has six cases in total.

Case 1: $I_i = (i, register_1, j)$. In this case, C' must be $(j, m+1, n)$. By the assumption, there are $a, b \in |G|$ such that

$$\begin{aligned} G \models & L_1(r, a) \wedge \\ & (R_1^-)^{m+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_j(a, b). \end{aligned}$$

Thus there exists $c \in |G|$, such that

$$G \models R_1^-(a, c) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c, b).$$

By ϕ_8 in Φ_N, $G \models L_1(r, c)$. Therefore, by $G \models \varphi_{C_1}$, $G \models K_i(c, b)$. Hence $G \models L_1(r, a) \wedge R_1^-(a, c) \wedge K_i(c, b)$. Thus by $\phi_{a_1}^i$ in Φ_M, we have that $G \models K_j(a, b)$. This contradicts the assumption.

Case 2: $I_i = (i, register_2, j)$. In this case, C' must be $(j, m, n+1)$. By the assumption, there are $a, b \in |G|$ such that

$$\begin{aligned} G \models & L_1(r, a) \wedge \\ & (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{n+1}(a, b) \wedge \neg K_j(a, b). \end{aligned}$$

Hence there exists $c \in |G|$, such that

$$G \models (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, c) \wedge R_2^+(c, b).$$

By $G \models \varphi_{C_1}$, we have $G \models K_i(a, c)$. As a result, we have $G \models L_1(r, a) \wedge K_i(a, c) \wedge R_2^+(c, b)$. Thus by $\phi_{a_2}^i$ in Φ_M, $G \models K_j(a, b)$. This contradicts the assumption.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, C' must be $(j, 0, n)$. By the assumption, there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_j(a, b).$$

Thus by $G \models \varphi_{C_1}$, we have $G \models K_i(a, b)$. In addition, there exists $c \in |G|$, such that $G \models L_1(r, a) \wedge E_{01}^-(a, c)$. By ϕ_{13}, ϕ_{14} and ϕ_{15} in Φ_N, we have $G \models E_{01}(r, a)$. Hence $G \models E_{01}(r, a) \wedge K_i(a, b)$. Thus by $\phi_{s_{1,0}}^i$ in Φ_M, we have $G \models K_j(a, b)$. This contradicts the assumption.

Case 4: $I_i = (i, register_1, j, k)$ and $m = p+1$. In this case, C' must be (k, p, n). By the assumption, there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^p \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a, b) \wedge \neg K_k(a, b).$$

Hence by ϕ_5 in Φ_N, there exists $c \in |G|$, such that

$$G \models L_1(r, a) \wedge R_1^+(a, c).$$

By ϕ_7, ϕ_1 in Φ_N, we have that $G \models L_1(r, c) \wedge R_1^-(c, a)$. Hence $G \models L_1(r, c) \wedge (R_1^-)^{p+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c, b)$. Thus by $G \models \varphi_{C_1}$, we have $G \models K_i(c, b)$. As a result, $G \models L_1(r, a) \wedge R_1^+(a, c) \wedge K_i(c, b)$. Thus by $\phi_{s_{1,n}}^i$ in Φ_M, $G \models K_k(a, b)$. This contradicts the assumption.

Case 5: $I_i = (i, register_2, j, k)$ and $n = 0$. In this case, C' must be $(j, m, 0)$. By the assumption, there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02}(a, b) \wedge \neg K_j(a, b).$$

Thus by $G \models \varphi_{C_1}$, we have $G \models K_i(a, b)$. By ϕ_{11}^i in Φ_N, $G \models K_i^-(b, a)$. Moreover, there exist $c, d \in |G|$, such that $G \models (R_1^-)^m(a, d) \wedge E_{01}^-(d, c) \wedge E_{02}(c, b)$. By $G \models L_1(r, a)$ and ϕ_8 in Φ_N, we have $G \models L_1(r, d)$. Thus by ϕ_{16} in Φ_N, we have $G \models E_{02}(r, b)$. As a result, $G \models E_{02}(r, b) \wedge K_i^-(b, a)$. Thus by $\phi_{s_{2,0}}^i$ in Φ_M, we have $G \models K_j(a, b)$. This contradicts the assumption.

Case 6: $I_i = (i, register_2, j, k)$ and $n = p+1$. In this case, C' must be (k, m, p). By the assumption, there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^p(a, b) \wedge \neg K_k(a, b).$$

Hence there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a, c) \wedge E_{01}^- \cdot E_{02}(c, d) \wedge (R_2^+)^p(d, b).$$

By ϕ_8 in Φ_N, we have $G \models L_1(r, c)$. By ϕ_{16} in Φ_N, $G \models E_{02}(r, d)$. By ϕ_{18} in Φ_N, $G \models L_2(r, d)$. By ϕ_9 in

Φ_N, $G \models L_2(r,b)$. Therefore, by ϕ_6 in Φ_N, there exists $e \in |G|$, such that $G \models R_2^+(b,e)$. Hence

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{p+1}(a,e).$$

By $G \models \varphi_{C_1}$, we have $G \models K_i(a,e)$. By ϕ_3 in Φ_N and $G \models R_2^+(b,e)$, we have $G \models R_2^-(e,b)$. As a result, we have $G \models L_1(r,a) \wedge K_i(a,e) \wedge R_2^-(e,b)$. Thus by $\phi^i_{s_{2,n}}$ in Φ_M, we have $G \models K_k(a,b)$. This contradicts the assumption.

Hence the claim holds for $t+1$ for all the cases of I_i.

(2) Conversely, assume that $C \not\Rightarrow_M D$. We show that $\Phi_N \wedge \Phi_M \wedge \varphi_C \to \varphi_D$ is not valid. To show this, we construct a σ-structure G such that $G \models \Phi_N \wedge \Phi_M \wedge \varphi_C$ and $G \models \neg\varphi_D$.

The structure G has the form shown in Figure 3. It is defined as follows. The universe of G consists of a distinguished node rt, which is the interpretation of the constant r in G, and two distinct infinite chains of natural numbers. More specifically, let $\mathbb{N}$ denote the set of all natural numbers, then

$$|G| = \{rt\} \cup \mathbb{N} \cup \{i' \mid i \in \mathbb{N}\}.$$

The binary relations in G are populated as follows (the superscript G is omitted in the relation names):

$$\begin{array}{ll}
L_r & = \{(rt,rt)\} \\
E_{01} & = \{(rt,0)\} \\
E_{01}^- & = \{(0,rt)\} \\
E_{02} & = \{(rt,0')\} \\
E_{02}^- & = \{(0',rt)\} \\
L_1 & = \{(rt,i) \mid i \in \mathbb{N}\} \\
L_1^- & = \{(i,rt) \mid i \in \mathbb{N}\} \\
L_2 & = \{(rt,i') \mid i \in \mathbb{N}\} \\
L_2^- & = \{(i',rt) \mid i \in \mathbb{N}\} \\
R_1^+ & = \{(i,i+1) \mid i \in \mathbb{N}\} \\
R_1^- & = \{(i+1,i) \mid i \in \mathbb{N}\} \\
R_2^+ & = \{(i',(i+1)') \mid i \in \mathbb{N}\} \\
R_2^- & = \{((i+1)',i') \mid i \in \mathbb{N}\} \\
K_i & = \{(m,n') \mid C \Rightarrow_M (i,m,n)\} \\
K_i^- & = \{(n',m) \mid (m,n') \in K_i\}
\end{array}$$

It is easy to verify the following. First, $G \models \Phi_N$. This is immediate from the construction of G. Second, $G \models \varphi_C \wedge \neg\varphi_D$, because $C \Rightarrow_M C$, $C \not\Rightarrow_M D$ and by the definition of K_i. Finally, $G \models \Phi_M$. To see this, first observe the following simple facts.

Fact 1: $G \models K_i(m,n')$ iff $C \Rightarrow_M (i,m,n)$.

Fact 2: If $C \Rightarrow_M (i,m,n) \to_M^{I_i} C'$, then $C \Rightarrow_M C'$. Moreover, C' is determined by the relation $\to_M$ described in Section 3.1.

Using these facts, we can verify that $G \models \Phi_M$ by contradiction. More specifically, suppose $G \not\models \Phi_M$. Then there is $i \in [0,l]$ such that $G \not\models \phi_{I_i}$. Here I_i has six cases. For each of these cases, the assumption contradicts the facts above. As an example, consider the case in which I_i is $(i, register_1, j)$. Then there must be $m, n' \in |G|$, such that $G \models K_i(m,n') \wedge \neg K_j(m+1,n')$. By Fact 1, $C \Rightarrow_M (i,m,n')$. In addition, by Fact 2, we have $C \Rightarrow_M (j,m+1,n')$. Thus again by Fact 1, $G \models K_j(m+1,n')$. This contradicts the assumption. The proofs for the other cases are similar.

Therefore, if $C \not\Rightarrow_M D$, then $\Phi_N \wedge \Phi_M \wedge \varphi_C \wedge \neg\varphi_D$ is satisfiable. ∎

3.2.2 Semi-conservative reduction

Taking advantage of the reduction property established above, we define a recursive function $f : FO \to S(P_+)$ by:

$$f(\psi) \quad \mapsto \quad \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)},$$

where $C(\psi)$ is the ID $(0, m(\psi), 0)$ of the 2-RM M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 3.1.

The proposition below shows that f is indeed a semi-conservative reduction from FO to $S(P_+)$.

Proposition 3.7: Let M_L be the 2-RM described in Section 3.1. For each $\psi \in FO$,

1. $\psi \in H_{M_L,1}$ iff $f(\psi)$ is not satisfiable; and
2. if $\psi \in H_{M_L,2}$, then $f(\psi)$ has a finite model. ∎

Proof: Recall $H_{M_L,1} = N(FO)$ and $H_{M_L,2} = F(FO)$ from Section 3.1.

(1) By Proposition 3.6, we have $C(\psi) \Rightarrow_{M_L} (1,0,0)$ iff $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \to \varphi_{(1,0,0)}$ is valid. In other words, $C(\psi) \Rightarrow_{M_L} (1,0,0)$ iff $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$ is not satisfiable. Since $\psi \in H_{M_L,1}$ iff $C(\psi) \Rightarrow_{M_L} (1,0,0)$, we have that $\psi \in H_{M_L,1}$ iff $f(\psi)$ is not satisfiable.

(2) We show that if $\psi \in H_{M_L,2}$, then $f(\psi)$ has a finite model.

First note that if $\psi \in H_{M_L,2}$, then the computation of M_L with initial ID $C(\psi)$ is finite. Therefore, the set

$$SID_{C(\psi)} \;=\; \{(i,m,n) \mid C(\psi) \Rightarrow_{M_L} (i,m,n)\}$$

is finite. Hence there is a natural number p, such that for each $(i,m,n) \in SID_{C(\psi)}$, $m+2 \le p$ and $n+2 \le p$.

Now we construct a finite σ-structure H satisfying $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$. The universe of H has

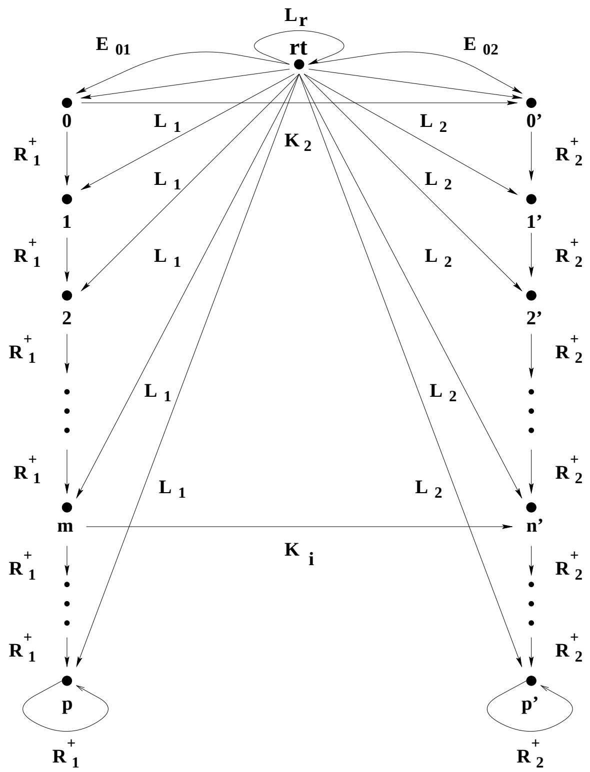

Figure 4: The structure H in Proposition 3.7

$2p+1$ nodes. More specifically,

$$|H| \;=\; \{rt, 1, 2, ..., p\} \;\cup\; \{1', 2', ..., p'\},$$

where rt is the interpretation of the constant r in H.

The binary relations $L_r, E_{01}, E_{02}, E_{01}^-, E_{02}^-, K_i$ and K_i^- in H are exactly the same as those in the σ-structure G given in the proof of Proposition 3.6. The binary relations L_1, L_1^-, L_2, L_2^-, R_1^+, R_1^-, R_2^+ and R_2^- are populated in H as follows (the superscript H is omitted in the relation names):

$$\begin{array}{rcl}
R_1^+ &=& \{(i, i+1) \mid 0 \leq i < p\} \cup \{(p,p)\} \\
R_1^- &=& \{(i+1, i) \mid 0 \leq i < p\} \cup \{(p,p)\} \\
R_2^+ &=& \{(i', (i+1)') \mid 0 \leq i < p\} \cup \{(p', p')\} \\
R_2^- &=& \{((i+1)', i') \mid 0 \leq i < p\} \cup \{(p', p')\}
\end{array}$$

$$\begin{array}{rcl}
L_1 &=& \{(rt, i) \mid 0 \leq i \leq p\} \\
L_1^- &=& \{(i, rt) \mid 0 \leq i \leq p\} \\
L_2 &=& \{(rt, i') \mid 0 \leq i \leq p\} \\
L_2^- &=& \{(i', rt) \mid 0 \leq i \leq p\}
\end{array}$$

See Figure 4 for the structure H (E_{01}^-, E_{02}^-, L_1^-, L_2^-, R_1^-, R_2^-, K_i^- edges are omitted in the graph). Note that the relations K_i and K_i^- in H are well-defined, since if $C(\psi) \Rightarrow_{M_L} (i, m, n)$, then $m < p-1$ and $n < p-1$.

We now show that $H \models \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

First, by $C(\psi) \Rightarrow_{M_L} C(\psi)$ and $C(\psi) \not\Rightarrow_{M_L} (1,0,0)$, we have that $H \models \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

Second, it is easy to verify that $H \models \Phi_N$. It should be mentioned that it is to ensure $H \models \phi_5 \wedge \phi_6$ that we require $H \models R_1^+(p,p) \wedge R_2^+(p',p')$.

Finally, we show that $H \models \Phi_M$. Since $\psi \in H_{M_L,2}$, it is straightforward to verify the following simple fact.

Fact 3: If $C(\psi) \Rightarrow_{M_L} (i, m, n)$, then $m < p-1$ and $n < p-1$.

In addition, Facts 1 and 2 given in the proof of Proposition 3.6 also hold here. Therefore, the argument for showing $G \models \Phi_M$ in the proof of Proposition 3.6, together with Fact 3 given above, proves $H \models \Phi_M$. This verifies that the structure H is indeed a finite model of $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$. ■

As an immediate result of Lemma 3.4 and Proposition 3.7, we have the following corollary, from which Theorem 3.2 follows immediately.

Corollary 3.8: The set $S(P_+)$ is a conservative reduction class. ■

3.3 Implication Problems for P_f

We next establish Theorem 3.3. As in the proof of Theorem 3.2, we show that the set $S(P_f)$ is a conservative reduction class. To do this, we first present an encoding of 2-RMs with constraints in P_f, and then define a semi-conservative reduction from FO to $S(P_f)$.

3.3.1 Encoding

We encode 2-RMs in terms of P_f constraints. Recall that P_f allows the left tail and right tail of a constraint to be empty path ϵ. In other words, equality is allowed in P_f.

Let M be a 2-RM. Assume that the set E of binary relation symbols in signature σ is the same as the one described in Section 3.2.1, except that the predicates L_r and K_i^- for $i \in [0, l]$ are no longer required here. We define the encoding as follows.

IDs. The encoding of each ID C of M, φ_C, is the same as the one given in Section 3.2.1. Note that φ_C is in P_f.

Registers. We encode the contents of the registers by Φ_N^f, which is the conjunction of the constraints of P_f given below.

- Successor, predecessor:

$$\begin{aligned}
\phi_1 &= \forall x\,(L_1(r,x) \to \forall y\,(R_1^+ \cdot R_1^-(x,y) \to \epsilon(x,y)))\\
\phi_2 &= \forall x\,(L_1(r,x) \to \forall y\,(R_1^- \cdot R_1^+(x,y) \to \epsilon(x,y)))\\
\phi_3 &= \forall x\,(L_2(r,x) \to \forall y\,(R_2^+ \cdot R_2^-(x,y) \to \epsilon(x,y)))\\
\phi_4 &= \forall x\,(L_2(r,x) \to \forall y\,(R_2^- \cdot R_2^+(x,y) \to \epsilon(x,y)))\\
\phi_5 &= \forall x\,(L_1(r,x) \to \forall y\,(\epsilon(x,y) \to R_1^+ \cdot R_1^-(x,y)))\\
\phi_6 &= \forall x\,(L_2(r,x) \to \forall y\,(\epsilon(x,y) \to R_2^+ \cdot R_2^-(x,y)))
\end{aligned}$$

  Constraints ϕ_1 and ϕ_2 (resp. ϕ_3 and ϕ_4) assert an inverse relationship between R_1^+ and R_1^- (resp. R_2^+ and R_2^-). It should be noted that since equality is allowed in P_f, ϕ_1 and ϕ_2 (resp. ϕ_3 and ϕ_4) enforce a node representing a content of $register_1$ (resp. $register_2$) to be unique. Constraints ϕ_5 and ϕ_6 state that R_1^+ and R_2^+ edges form "infinite" chains.

- Register identification: ϕ_7, ϕ_8, ϕ_9 and ϕ_{10} are the same as given in Section 3.2.1.

- Zeros:

$$\begin{aligned}
\phi_{11} &= \forall x\,(L_1 \cdot E_{01}^-(r,x) \to \epsilon(r,x))\\
\phi_{12} &= \forall x\,(L_1(r,x) \to\\
&\qquad \forall y\,(E_{01}^-(x,y) \to E_{01}^- \cdot E_{01} \cdot E_{01}^-(x,y)))\\
\phi_{13} &= \forall x\,(L_1(r,x) \to \forall y\,(E_{01}^- \cdot E_{01}(x,y) \to \epsilon(x,y)))\\
\phi_{14} &= \forall x\,(L_1 \cdot E_{01}^- \cdot E_{02}(r,x) \to E_{02}(r,x))\\
\phi_{15} &= \forall x\,(E_{02}(r,x) \to \forall y\,(\epsilon(x,y) \to E_{02}^- \cdot E_{02}(x,y)))\\
\phi_{16} &= \forall x\,(E_{02}(r,x) \to L_2(r,x))
\end{aligned}$$

  Constraints ϕ_{11}, ϕ_{12} and ϕ_{13} assert that if there is an edge labeled L_1 from the root to a node a and a has an outgoing E_{01}^- edge, then there is an E_{01} edge from the root to a. Constraint ϕ_{14} states that if there exists a path $L_1 \cdot E_{01}^- \cdot E_{02}$ from the root to a node b, then there is an E_{02} edge from the root to b. Constraint ϕ_{15} asserts that if there is an E_{02} edge from the root to a node c, then there exists a node d such that there is an E_{02}^- edge from c to d and there is an E_{02} edge from d to c. Finally, ϕ_{16} states that there is an edge labeled L_2 from the root to a node representing 0 in $register_2$.

Instructions. The encoding of instruction I_i, ϕ_{I_i}, is the same as the one given in Section 3.2.1, except that here $\phi^i_{s_{2,0}}$ is

$$\forall x\,(L_1(r,x) \to \forall y\,(K_i \cdot E_{02}^- \cdot E_{02}(x,y) \to K_j(x,y))).$$

The encoding of the program of M is $\Phi_M^f = \bigwedge_{i=0}^{l} \phi_{I_i}$. It is clear that Φ_M^f is a conjunction of constraints in P_f.

Analogous to Proposition 3.6, we show that the encoding above has the following reduction property.

Proposition 3.9: For all IDs C and D of M,

$$C \Rightarrow_M D \quad \text{iff} \quad \Phi_N^f \wedge \Phi_M^f \wedge \varphi_C \to \varphi_D \text{ is valid.}$$

■

Proof: The proof is similar to that of Proposition 3.6.

(1) Assume that $C \Rightarrow_M D$. We prove by induction on step t that for each ID C' of M and each model G of $\Phi_N^f \wedge \Phi_M^f \wedge \varphi_C$, if $C \Rightarrow_M^t C'$ then $G \models \varphi_{C'}$. This can be shown in basically the same way as for Proposition 3.6, except for the following cases in the inductive step.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, C' must be $(j, 0, n)$. Suppose, for a contradiction, that there are $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a,b) \wedge \neg K_j(a,b).$$

Then by $G \models \varphi_{C_1}$, we have $G \models K_i(a,b)$. In addition, there exists $e \in |G|$, such that $G \models L_1(r,a) \wedge E_{01}^-(a,e)$. By ϕ_{12} in Φ_N^f, there exist $c, d \in |G|$, such that

$$G \models L_1(r,a) \wedge E_{01}^-(a,c) \wedge E_{01}(c,d).$$

Thus by ϕ_{13} in Φ_N^f, we have $G \models \epsilon(a,d)$. As a result, $G \models L_1(r,a) \wedge E_{01}^-(a,c) \wedge E_{01}(c,a)$. By ϕ_{11} in Φ_N^f and $G \models L_1(r,a) \wedge E_{01}^-(a,c)$, we have $G \models \epsilon(r,c)$. Thus $G \models E_{01}(r,a)$. Hence $G \models E_{01}(r,a) \wedge K_i(a,b)$. Thus by $\phi^i_{s_{1,0}}$ in Φ_M^f, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 4: $I_i = (i, register_1, j, k)$ and $m = p+1$. In this case, C' must be (k, p, n). Suppose, for a contradiction, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^p \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(a,b) \wedge \neg K_k(a,b).$$

Then by ϕ_5 in Φ_N^f, there exists node $c \in |G|$, such that $G \models L_1(r,a) \wedge R_1^+(a,c) \wedge R_1^-(c,a)$. By ϕ_7 in Φ_N^f, we have $G \models L_1(r,c) \wedge R_1^-(c,a)$. As a result,

$$G \models L_1(r,c) \wedge (R_1^-)^{p+1} \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(c,b).$$

Thus by $G \models \varphi_{C_1}$, $G \models K_i(c,b)$. Therefore, we have that $G \models L_1(r,a) \wedge R_1^+(a,c) \wedge K_i(c,b)$. Thus by $\phi^i_{s_{1,n}}$ in Φ_M^f, we have $G \models K_k(a,b)$. This contradicts the assumption.

Case 5: $I_i = (i, register_2, j, k)$ and $n = 0$. In this case, C' must be $(j, m, 0)$. Suppose, for a contradiction, that there exist $a, b \in |G|$, such that

$$G \models L_1(r,a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02}(a,b) \wedge \neg K_j(a,b).$$

Then by $G \models \varphi_{C_1}$, we have $G \models K_i(a,b)$. Moreover, there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a,d) \wedge E_{01}^-(d,c) \wedge E_{02}(c,b).$$

By $G \models L_1(r, a)$ and ϕ_8 in Φ_N^f, we have $G \models L_1(r, d)$. Thus by ϕ_{14} in Φ_N^f, we have $G \models E_{02}(r, b)$. By ϕ_{15} in Φ_N^f, there is $e \in |G|$, such that $G \models E_{02}^-(b, e) \wedge E_{02}(e, b)$. Hence $G \models L_1(r, a) \wedge K_i(a, b) \wedge E_{02}^-(b, e) \wedge E_{02}(e, b)$. Thus by $\phi_{s_{2,0}}^i$ in Φ_M^f, we have $G \models K_j(a, b)$. This contradicts the assumption.

Case 6: $I_i = (i, register_2, j, k)$ and $n = p + 1$. In this case, C' must be (k, m, p). Suppose, for a contradiction, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^p(a, b) \wedge \neg K_k(a, b).$$

Then there exist $c, d \in |G|$, such that

$$G \models (R_1^-)^m(a, c) \wedge E_{01}^- \cdot E_{02}(c, d) \wedge (R_2^+)^p(d, b).$$

By ϕ_8 in Φ_N^f, we have $G \models L_1(r, c)$. By ϕ_{14} in Φ_N^f, $G \models E_{02}(r, d)$. By ϕ_{16} in Φ_N^f, $G \models L_2(r, d)$. By ϕ_9 in Φ_N^f, $G \models L_2(r, b)$. Therefore, by ϕ_6 in Φ_N^f, there exists $e \in |G|$, such that $G \models R_2^+(b, e) \wedge R_2^-(e, b)$. Therefore,

$$G \models L_1(r, a) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^{p+1}(a, e).$$

By $G \models \varphi_{C_1}$, $G \models K_i(a, e)$. As a result, we have that $G \models L_1(r, a) \wedge K_i(a, e) \wedge R_2^-(e, b)$. Thus by $\phi_{s_{2,n}}^i$ in Φ_M^f, $G \models K_k(a, b)$. This contradicts the assumption.

(2) Conversely, assume that $C \not\Rightarrow_M D$. It is easy to verify that the σ-structure G (without L_r and K_i^- edges) constructed in the proof of Proposition 3.6 is a model of $\Phi_N^f \wedge \Phi_M^f \wedge \varphi_C \wedge \neg\varphi_D$. ■

3.3.2 Semi-conservative reduction

We define a recursive function $g : FO \to S(P_f)$ by:

$$g(\psi) \quad \mapsto \quad \Phi_N^f \wedge \Phi_M^f \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)},$$

where $C(\psi)$ is the ID $(0, m(\psi), 0)$ of the 2-RM M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 3.1.

Proposition 3.10 below shows that the function g is indeed a semi-conservative reduction from FO to $S(P_f)$.

Proposition 3.10: Let M_L be the 2-RM described in Section 3.1. For each $\psi \in FO$,

1. $\psi \in H_{M_L,1}$ iff $g(\psi)$ is not satisfiable; and
2. if $\psi \in H_{M_L,2}$, then $g(\psi)$ has a finite model.

■

Proof: The proof is similar to the proof of Proposition 3.7, except that here in the structure H shown in Figure 4, there are no L_r and K_i^- edges. ■

From Proposition 3.10 and Lemma 3.4 follows the corollary below. As a result, Theorem 3.3 follows.

Corollary 3.11: The set $S(P_f)$ is a conservative reduction class. ■

4 DECIDABLE RESTRICTED IMPLICATION

The undecidability results established in the last section suggest that we search for fragments of P_c which possess decidable implication problems, and yet retain sufficient expressive power of the full language. This section identifies several fragments of P_c which share the following properties. First, they each properly contain the set of word constraints. Second, each of them fails to be included in two-variable first-order logic. Third, they allow the formulation of many interesting semantic relations. And finally, the implication and finite implication problems for each of them are decidable in the context of semistructured databases.

We begin by introducing these fragments of P_c, and then establish the decidability of their associated implication and finite implication problems. Finally, we investigate a mild generalization of P_c, $P_c^\wedge$.

4.1 Decidable Fragments of P_c

We describe three fragments of P_c and demonstrate their expressive power.

4.1.1 Prefix restricted implication for P_c

The implication problems for simple path constraints, which are known to be decidable, can be viewed as a restricted form of the implication problems for P_c. More specifically, the implication problems for P_s are the implication problems for P_c under the following restriction: in any finite subset of P_c in the implication problems, the prefix of each constraint is the empty path.

By replacing this prefix restriction with a weaker one, we define the prefix restricted implication problems for P_c as follows.

Definition 4.1: A *prefix restricted subset of* P_c is a finite subset of P_c in which the prefixes of all the constraints have the same length.

The *prefix restricted (finite) implication problem for* P_c is the problem to determine, given any prefix restricted subset $\Sigma \cup \{\varphi\}$ of P_c, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). ■

Obviously, the (finite) implication problem for word constraints is a special case of the prefix restricted (finite) implication problem for P_c. Moreover, in contrast to word constraint implication, prefix restricted implication cannot be stated in two-variable first-order logic (FO^2). A convenient argument for this is that $\{\varphi\}$, where φ is the constraint given in Example 2.1, is a prefix restricted subset of P_c. However, φ is not expressible in FO^2.

Many cases of integrity constraint implication commonly found in databases are instances of the prefix restricted implication problem for P_c. Among these are implications for inverse constraints and local database constraints. As an example, consider the set Σ consisting of the following local inverse constraints in the school databases described in Section 1:

$$\begin{array}{l} \forall\, s\ (Schools \cdot Students(r, s) \to \\ \qquad \forall\, c\ (Taking(s, c) \to Enrolled(c, s))) \\ \forall\, c\ (Schools \cdot Courses(r, c) \to \\ \qquad \forall\, s\ (Enrolled(c, s) \to Taking(s, c))) \end{array}$$

and the constraint φ:

$$\begin{array}{l} \forall\, s_1\ (Schools \cdot Students(r, s_1) \to \\ \qquad \forall\, s_2\ (\epsilon(s_1, s_2) \to Taking \cdot Enrolled(s_1, s_2))). \end{array}$$

The question whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$) is an instance of the prefix restricted (finite) implication problem for P_c.

4.1.2 Sublanguage P_β

Some cases of path constraint implication canvassed earlier are not instances of the prefix restricted implication. For example, recall the two extent constraints and the two inverse constraints for student/course databases given in Section 1:

$$\begin{array}{l} \forall\, c\ (Students \cdot Taking(r, c) \to Courses(r, c)) \\ \forall\, s\ (Courses \cdot Enrolled(r, s) \to Students(r, s)) \end{array}$$

$$\begin{array}{l} \forall\, s\ (Students(r, s) \to \\ \qquad \forall\, c\ (Taking(s, c) \to Enrolled(c, s))) \\ \forall\, c\ (Courses(r, c) \to \\ \qquad \forall\, s\ (Enrolled(c, s) \to Taking(s, c))) \end{array}$$

The set consisting of these constraints is not a prefix restricted subset of P_c.

The constraints in the last example, however, are in the sublanguage P_β of P_c defined below. Recall the notations $lt(\varphi)$ and $pf(\varphi)$ for a P_c constraint φ described in Definition 2.1.

Definition 4.2: A *β-restricted path constraint* φ is a constraint in P_c with $|lt(\varphi)| \leq 1$. That is, either $lt(\varphi)$ is ϵ, or $lt(\varphi) = K$ for some $K \in E$.

The sublanguage P_β is defined to be the class of P_c constraints φ such that either $|pf(\varphi)| = 0$ or $|lt(\varphi)| \leq 1$. In other words, P_β consists of all simple path constraints and all β-restricted path constraints.

The *(finite) implication problem for P_β* is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of P_β, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). ∎

Note that the class of word constraints is a proper subset of P_β. In addition, not all constraints in P_β are expressible in FO^2. Indeed, the constraint φ given in Example 2.1 is in P_β, but is not in FO^2.

4.1.3 Extended implication for P_β

Recall the local extent constraints given in Section 1:

$$\begin{array}{l} \forall\, d\ (Schools(r, d) \to \\ \qquad \forall\, c\ (Students \cdot Taking(d, c) \to Courses(d, c))) \\ \forall\, d\ (Schools(r, d) \to \\ \qquad \forall\, s\ (Courses \cdot Enrolled(d, s) \to Students(d, s))) \end{array}$$

Consider the set consisting of these local extent constraints and the local inverse constraints given in Section 4.1.1. This set is neither a prefix restricted subset of P_c nor a subset of P_β. However, the constraints in this set share the following property: all of them are constraints in student/course databases as shown in Figure 1 augmented with a common prefix `Schools`. In general, when represented in a global environment, path constraints in a local database are augmented with a common prefix. This example motivates the following extension of P_β.

Definition 4.3: Let α be a path and φ be a constraint in P_β. The *extension of φ with prefix α*, denoted by $\delta(\varphi, \alpha)$, is the constraint defined either by

$$\forall\, x\, (\alpha \cdot pf(\varphi)(r, x) \to \forall\, y\, (lt(\varphi)(x, y) \to rt(\varphi)(x, y)))$$

when φ is of the forward form, or by

$$\forall\, x\, (\alpha \cdot pf(\varphi)(r, x) \to \forall\, y\, (lt(\varphi)(x, y) \to rt(\varphi)(y, x)))$$

when φ is of the backward form. Here $\cdot$ is the path concatenation operator, and pf, lt and rt are defined in Definition 2.1.

Let α be a path and Σ be a finite subset of P_β. The *extension of Σ with prefix α* is the subset of P_c defined by $\{\delta(\varphi, \alpha) \mid \varphi \in \Sigma\}$. Such a set is called a *prefix extended subset of P_β*.

The *extended (finite) implication problem for P_β* is the problem of determining, given any prefix extended subset $\Sigma \cup \{\varphi\}$ of P_β, whether $\Sigma \models \varphi$ ($\Sigma \models_f \varphi$). ■

For instance, the set described in the last example is a prefix extended subset of P_β.

Note that the (finite) implication problem for P_β is a special case of the extended (finite) implication problem for P_β, namely, when the prefix α described in Definition 4.3 is the empty path ϵ. As an immediate result, implications of word constraints are special cases of extended implications of P_β constraints. In addition, extended implications of P_β constraints cannot be stated in FO^2.

4.2 Decidability of Prefix Restricted Implication

In this section, we show the following:

Theorem 4.1: The prefix restricted implication and finite implication problems for P_c are decidable. ■

The idea of the proof is to show that the satisfiability and finite satisfiability problems for the set S_p:

$$\{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a prefix restricted subset of } P_c\}$$

are decidable. That is, we show that it is decidable to determine, given any $\psi \in S_p$, whether there is a (finite) σ-structure such that $G \models \psi$.

To show that S_p possesses decidable satisfiability problems, let us recall the following notion from [9].

Definition 4.4 [9]: A class X of logic sentences has the *small model property* for satisfiability iff there exists a recursive function s such that for each $\psi \in X$, if ψ is satisfiable, then ψ has a finite model of size at most $s(|\psi|)$, where $|\psi|$ stands for the length of ψ. ■

If a class X of logic sentences has the small model property, then the satisfiability and finite satisfiability problems for X coincide and are decidable. In fact, for any $\psi \in X$, one can determine whether ψ is satisfiable in $s(|\psi|)$-space, where s is the recursive function described in Definition 4.4. Therefore, to show the decidability of the satisfiability and finite satisfiability problems for S_p, it suffices to establish the small model property for S_p. To do this, we use a path label criterion to characterize whether a σ-structure satisfies a sentence of S_p. More specifically, given a structure G and a sentence ψ of S_p, we label each node of G with paths in ψ. The path label of G, $LB(G, \psi)$, is the collection of the labels of all the nodes in G. This path label has the following properties:

- for any σ-structure H, if $LB(H, \psi) = LB(G, \psi)$, then $H \models \psi$ iff $G \models \psi$; and
- there is a σ-structure H of size at most $2^{2^{2|\psi|}}$, such that $LB(H, \psi) = LB(G, \psi)$.

As a result, if ψ is satisfiable, then it has a model of size at most $2^{2^{2|\psi|}}$.

We next define the path labels and show that they have the properties described above.

4.2.1 Path labels

Let $G = (|G|, r^G, E^G)$ and $\psi \in S_p$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$. To define path labels, we need the following notations:

$$Paths_\alpha(\psi) = \{pf(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\}$$
$$Paths_\beta(\psi) = \{lt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\}$$
$$Paths^+_\gamma(\psi) = \{rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}, \ \phi \text{ has the forward form}\}$$
$$Paths^-_\gamma(\psi) = \{-rt(\phi) \mid \phi \in \Sigma \cup \{\varphi\}, \ \phi \text{ has the backward form}\}$$
$$Paths_{(\beta,\gamma)}(\psi) = Paths_\beta(\psi) \cup Paths^+_\gamma(\psi) \cup Paths^-_\gamma(\psi)$$

Here the notation $-\rho$ denotes the pair $(-, \rho)$. We use this notation merely to distinguish the occurrence of a path as the right tail of a backward constraint as opposed to a forward constraint. The notations pf, lt and rt are described in Definition 2.1.

For each node a in $|G|$, we define a path label using paths in $Paths_\alpha(\psi)$ and $Paths_{(\beta,\gamma)}(\psi)$. This label consists of a pair of sets. Its first component is the set of paths in $Paths_\alpha(\psi)$ from r^G to a. That is,

$$lb_\alpha(a, G, \psi) = \{\rho \mid \rho \in Paths_\alpha(\psi), \ G \models \rho(r^G, a)\}.$$

The second component is a collection of sets of paths in $Paths_{(\beta,\gamma)}(\psi)$. Each set consists of the paths between the node a and some node in $|G|$. More specifically, for each $b \in |G|$, let:

$$lbs_\beta(a, b, G, \psi) = \{\rho \mid \rho \in Paths_\beta(\psi), \ G \models \rho(a, b)\}$$
$$lbs_\gamma(a, b, G, \psi) = \{\rho \mid \rho \in Paths^+_\gamma(\psi), \ G \models \rho(a, b)\} \cup \{-\rho \mid -\rho \in Paths^-_\gamma(\psi), \ G \models \rho(b, a)\}$$

We define $lbs_{(\beta,\gamma)}(a, b, G, \psi)$ to be

$$lbs_\beta(a, b, G, \psi) \ \cup \ lbs_\gamma(a, b, G, \psi).$$

The second component of the label is defined by:

$$lb_{(\beta,\gamma)}(a, G, \psi) = \{lbs_{(\beta,\gamma)}(a, b, G, \psi) \mid b \in |G|\}$$

More precisely, we define the *label of node a in G w.r.t. ψ*, denoted by $lb(a, G, \psi)$ to be

- $(\emptyset,\ \emptyset)$, if $lb_\alpha(a, G, \psi) = \emptyset$; or
- $(lb_\alpha(a, G, \psi),\ lb_{(\beta,\gamma)}(a, G, \psi))$, otherwise.

The *label of* G *w.r.t.* ψ is defined by

$$LB(G, \psi) \ = \ \{lb(a, G, \psi) \mid a \in |G|\}.$$

Every label $l \in LB(G, \psi)$ is a pair of sets. We refer to the first component of l as $lb_\alpha(l)$, and the second as $lb_{(\beta,\gamma)}(l)$. In addition, we use the following notations:

$$\begin{array}{rcl} LB_\alpha(G, \psi) & = & \{lb_\alpha(l) \mid l \in LB(G, \psi)\} \\ LB_{(\beta,\gamma)}(G, \psi) & = & \{lb_{(\beta,\gamma)}(l) \mid l \in LB(G, \psi)\} \end{array}$$

Let us examine the cardinality of $LB(G, \psi)$. We use the notation $card(S)$ to denote the cardinality of a set S. It is easy to verify that

$$\begin{array}{rcl} card(Paths_\alpha(\psi)) & \leq & |\psi|, \\ card(Paths_{(\beta,\gamma)}(\psi)) & \leq & |\psi|. \end{array}$$

Note that for any $l \in LB(G, \psi)$, $lb_\alpha(l)$ is a subset of $Paths_\alpha(\psi)$ and $lb_{(\beta,\gamma)}(l)$ is a subset of the power set of $Paths_{(\beta,\gamma)}(\psi)$. Therefore,

$$card(LB(G, \psi)) \ \leq \ 2^{|\psi| + 2^{|\psi|}}.$$

In particular, if ψ involves simple constraints only, i.e., $\Sigma \cup \{\varphi\}$ is a subset of P_s, then $Paths_\alpha(\psi) = \{\epsilon\}$. In this case, it is easy to verify that $card(LB(G, \psi))$ is at most 2. More specifically, $LB(G, \psi) \subseteq \{(\emptyset, \emptyset),\ lb(r^G,\ G, \psi)\}$.

We shall use $s_\alpha(\psi)$ to denote the prefix length of φ. That is, $s_\alpha(\psi) = |pf(\varphi)|$. Note that the prefixes of all the constraints in $\Sigma \cup \{\varphi\}$ have the same length.

The lemma below shows that $LB(G, \psi)$ characterizes whether $G \models \psi$. This lemma can be easily verified by contradiction.

Lemma 4.2: For any σ-structures G, H, and any sentence $\psi \in S_p$, if $LB(G, \psi) = LB(H, \psi)$, then $G \models \psi$ iff $H \models \psi$. ∎

4.2.2 The small model property

Next, we establish the small model property for S_p. By Lemma 4.2, it suffices to show the following.

Proposition 4.3: For each σ-structure G and each sentence ψ in S_p, there is a σ-structure H, such that

1. the size of H is at most $2^{2^{2|\psi|}}$; and
2. $LB(H, \psi) = LB(G, \psi)$.

∎

For if the proposition holds, then every satisfiable sentence ψ in S_p has a model of size at most $2^{2^{2|\psi|}}$. That is, S_p has the small model property.

The idea of the proof of Proposition 4.3 is as follows. Let G be a σ-structure and ψ a sentence in S_p. We first construct a graph G_α that includes precisely one node a_l representing $lb_\alpha(l)$ for each $l \in LB(G, \psi)$. We then construct a graph G_l for each $l \in LB(G, \psi)$, such that the root of G_l represents $lb_{(\beta,\gamma)}(l)$. Finally, we glue to each node a_l the root of the corresponding graph G_l. This yields the σ-structure H described in Proposition 4.3.

The implementation of the idea requires two lemmas and the following notation.

Definition 4.5: Let G be a σ-structure, m be a natural number and $a \in |G|$. The *m-neighborhood of a in* G is the structure $G(a) = (|G(a)|, r^{G(a)}, E^{G(a)})$, where

- $|G(a)| \ = \ \{c \mid c \in |G|$, there is path ρ, $|\rho| \leq m$ and either $G \models \rho(a, c)$ or $G \models \rho(c, a)\}$;
- $r^{G(a)} = a$; and
- for all $b, c \in |G(a)|$ and any $K \in E$, $G(a) \models K(b, c)$ iff $G \models K(b, c)$.

That is, $G(a)$ is the restriction of G to $|G(a)|$ with a as the new root. ∎

Given a σ-structure G and a sentence ψ in S_p, the first lemma below proves the existence of a σ-structure G_α which has the following properties.

- $LB_\alpha(G_\alpha, \psi) = LB_\alpha(G, \psi)$. In addition, for each $l \in LB(G, \psi)$, there is a distinguished node a_l in $|G_\alpha|$ such that $lb_\alpha(a_l, G_\alpha, \psi) = lb_\alpha(l)$.
- For each $a \in |G_\alpha|$, if $lb_\alpha(a, G_\alpha, \psi) \neq \emptyset$, then a does not have any outgoing edge. That is, for each $K \in E$ and $b \in |G_\alpha|$, $G_\alpha \models \neg K(a, b)$.

We shall proceed to construct the σ-structure H described in Proposition 4.3, such that in H, G_α is the $s_\alpha(\psi)$-neighborhood of r^H. This will ensure that

$$LB_\alpha(H, \psi) = LB_\alpha(G, \psi).$$

Lemma 4.4: For each σ-structure G and $\psi \in S_p$, there is a σ-structure $G_\alpha = (|G_\alpha|, r^{G_\alpha}, E^{G_\alpha})$, such that

1. the size of G_α is at most $|\psi| + 2^{|\psi| + 2^{|\psi|}}$;
2. there is a subset L_α of $|G_\alpha|$, such that

(a) there exists a bijection $f : LB(G,\psi) \rightarrow L_\alpha$, such that $lb_\alpha(l) = lb_\alpha(f(l), G_\alpha, \psi)$ for each label $l \in LB(G,\psi)$; and in addition, for every $K \in E$ and $b \in |G_\alpha|$, $G_\alpha \models \neg K(f(l), b)$;

(b) for each $b \in |G_\alpha| \setminus L_\alpha$, $lb_\alpha(b, G_\alpha, \psi) = \emptyset$.

■

Proof: Let $I_\alpha(\psi) = \{\rho \mid \varrho \in Paths_\alpha(\psi),\ \rho \prec_p \varrho\}$. Here $\rho \prec_p \varrho$ stands for that ρ is a proper prefix of ϱ, as defined in Section 2. We construct G_α using $LB(G,\psi)$ and $I_\alpha(\psi)$ as follows. For each $\rho \in I_\alpha(\psi)$, let a_ρ be a distinguished node, and for each $l \in LB(G,\psi)$, let a_l be a distinguished node. Let

- $L_\alpha = \{a_l \mid l \in LB(G,\psi)\}$;
- $|G_\alpha| = L_\alpha \cup \{a_\rho \mid \rho \in I_\alpha(\psi)\}$;
- $r^{G_\alpha} = \begin{cases} a_\epsilon & \text{if } s_\alpha(\psi) \geq 1 \\ a_{lb(r^G, G, \psi)} & \text{otherwise;} \end{cases}$
- for all $a, b \in |G_\alpha|$ and $K \in E$, $G_\alpha \models K(a,b)$ iff there exists $\rho \in I_\alpha(\psi)$, such that $a = a_\rho$ (i.e., $a \notin L_\alpha$), and one of the following conditions is satisfied:
  - there exists $\varrho \in I_\alpha(\psi)$, such that $b = a_\varrho$ (i.e., $b \notin L_\alpha$), and $\varrho = \rho \cdot K$; or
  - there exists $l \in LB(G,\psi)$, such that $b = a_l$ (i.e., $b \in L_\alpha$), and there exists $\varrho \in lb_\alpha(l)$, such that $\varrho = \rho \cdot K$.

It should be noted that when $s_\alpha(\psi) = 0$, i.e., when ψ involves simple constraints only, $I_\alpha(\psi) = \emptyset$ and $|G_\alpha|$ consists of r^{G_α} and at most another node. This is because in this case, $LB(G,\psi) \subseteq \{(\emptyset, \emptyset),\ lb(r^G, G, \psi)\}$. Here r^{G_α} represents the label of the root r^G if G, i.e., $r^{G_\alpha} = a_{lb(r^G, G, \psi)}$. The other node, if it exists, is $a_{(\emptyset,\emptyset)}$.

The structure G_α is basically a rooted acyclic directed graph. It has the following properties.

- The restriction of G_α to $\{a_\rho \mid \rho \in I_\alpha(\psi)\}$ is a tree of height $s_\alpha(\psi) - 1$. For each node a_ρ in the tree, there is a single path ρ from the root r^{G_α} to a_ρ.
- At level $s_\alpha(\psi)$, there are $card(LB(G,\psi))$ many nodes. Each of these nodes is uniquely marked with a label in $LB(G,\psi)$. In addition, it does not have any outgoing edge, and all its incoming edges are from leaves of the tree mentioned above.

We now verify that G_α indeed meets all the requirements of the lemma.

(1) *The size of* G_α. Let $size(A)$ denote the size of a structure A. It is easy to verify that

$$\begin{aligned} card(I_\alpha(\psi)) &\leq |\psi|, \\ card(L_\alpha) &= card(LB(G,\psi)). \end{aligned}$$

Therefore, $size(G_\alpha)$ is at most $|\psi| + 2^{|\psi| + 2^{|\psi|}}$. In particular, when $s_\alpha(\psi) = 0$, $size(G_\alpha)$ is at most 2.

(2) *Properties of* L_α. The bijection f from $LB(G,\psi)$ to L_α can be defined by: $l \mapsto a_l$. To verify the other properties of L_α, first observe the following:

Claim: For any $\varrho \in I_\alpha(\psi)$, $|\varrho| < s_\alpha(\psi)$ and

$$\{\rho \mid \rho \text{ is a path, } G_\alpha \models \rho(r^{G_\alpha}, a_\varrho)\} = \{\varrho\}.$$

This claim can be verified by a straightforward induction on $|\varrho|$. By this claim and the definition of G_α, it is easy to verify the second statement of the lemma. ■

The next lemma deals with $LB_{(\beta,\gamma)}(G,\psi)$. More specifically, given a label l in $LB(G,\psi)$, it constructs a σ-structure $G_l = (|G_l|, r^{G_l}, E^{G_l})$ such that

$$lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l).$$

We shall construct the structure H described in Proposition 4.3 such that for each l in $LB(G,\psi)$, G_l is part of H, and moreover,

$$lb_{(\beta,\gamma)}(r^{G_l}, H, \psi) = lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi).$$

Lemma 4.5: Let G be a σ-structure and $\psi \in S_p$. For each $l \in LB(G,\psi)$, there is a σ-structure G_l, such that

1. the size of G_l is at most $2^{|\psi|}$; and
2. $lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l)$.

■

Proof: We give a filtration argument. Since l is in $LB(G,\psi)$, there exists $a \in |G|$ such that $lb(a, G, \psi) = l$. Let

$$\begin{aligned} I^+(\psi) &= \{\rho \mid \varrho \in Paths_\beta(\psi) \cup Paths^+_\gamma(\psi),\ \rho \preceq_p \varrho\}, \\ I^-(\psi) &= \{-\rho \mid -\varrho \in Paths^-_\gamma(\psi),\ \rho \preceq_s \varrho\}, \\ I(\psi) &= I^+(\psi) \cup I^-(\psi). \end{aligned}$$

Here $\rho \preceq_p \varrho$ ($\rho \preceq_s \varrho$) means that ρ is a prefix (suffix) of ϱ, as defined in Section 2. It is easy to verify that $card(I(\psi)) \leq |\psi|$.

We define a function g from $|G|$ to the power set of $I(\psi)$ such that for any $b \in |G|$,

$$\begin{aligned} g(b) \mapsto\ & \{\rho \mid \rho \in I^+(\psi),\ G \models \rho(a,b)\} \cup \\ & \{-\rho \mid -\rho \in I^-(\psi),\ G \models \rho(b,a)\}. \end{aligned}$$

Clearly, the action of g induces an equivalence relation $\sim$ on $|G|$:

$$b \sim b' \quad \text{iff} \quad g(b) = g(b').$$

We denote the equivalence class of b with respect to $\sim$ as $[b]$. We proceed to construct a σ-structure G_l whose nodes are these equivalence classes.

- $|G_l| = \{[b] \mid b \in |G|\}$;
- $r^{G_l} = [a]$;
- for each $K \in E$ and $o_1, o_2 \in |G_l|$, $G_l \models K(o_1, o_2)$ iff there exist $b_1, b_2 \in |G|$, such that $[b_1] = o_1$, $[b_2] = o_2$, and $G \models K(b_1, b_2)$.

Obviously, the size of G_l is no larger than the cardinality of the power set of $I(\psi)$, and therefore, is at most $2^{|\psi|}$. In addition, it can be verified by a straightforward induction on $|\rho|$ and $|\varrho|$ that for any $\rho \in I^+(\psi)$, $-\varrho \in I^-(\psi)$ and $b \in |G|$,

$$G \models \rho(a, b) \quad \text{iff} \quad G_l \models \rho(r^{G_l}, [b]),$$
$$G \models \varrho(b, a) \quad \text{iff} \quad G_l \models \varrho([b], r^{G_l}).$$

From these follows that $lb_{(\beta,\gamma)}(r^{G_l}, G_l, \psi) = lb_{(\beta,\gamma)}(l)$. ■

Finally, we prove Proposition 4.3. As mentioned earlier, given a σ-structure G and a sentence ψ in S_p, we define the structure H described in Proposition 4.3 such that

- the structure G_α described in Lemma 4.4 is the $s_\alpha(\psi)$-neighborhood of r^H in H;
- for each $l \in LB(G, \psi)$, G_l in Lemma 4.5 is part of H such that
  - $r^{G_l} = f(l)$, where f is the function specified in Lemma 4.4,
  - $lb_{(\beta,\gamma)}(r^{G_l}, H, \psi) = lb_{(\beta,\gamma)}(l)$, and
  - $lb_\alpha(r^{G_l}, H, \psi) = lb_\alpha(l)$.

Proof of Proposition 4.3: Given a σ-structure G and $\psi \in S_p$, let G_α be the σ-structure specified in Lemma 4.4, and for each $l \in LB(G, \psi)$, let G_l be the structure specified in Lemma 4.5. Without loss of generality, assume that $|G_l| \cap |G_\alpha| = \emptyset$ and $|G_l| \cap |G_{l'}| = \emptyset$ if $l \neq l'$. Using these, we now construct a σ-structure $H = (|H|, r^H, E^H)$, as follows.

- $|H| = |G_\alpha| \cup \bigcup_{l \in LB(G,\psi)} (|G_l| \ \setminus \ \{r^{G_l}\})$;
- $r^H = r^{G_\alpha}$;

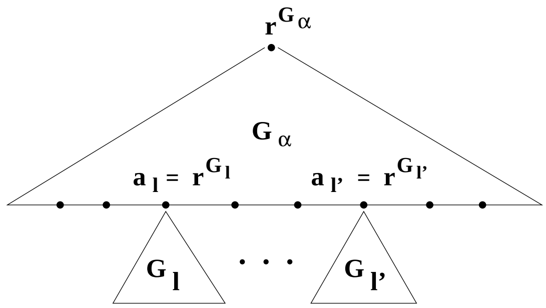

Figure 5: The structure H in Proposition 4.3

- For all $a, b \in |H|$ and each $K \in E$, $H \models K(a, b)$ iff one of the following conditions is satisfied:
  - $a, b \in |G_\alpha|$ and $G_\alpha \models K(a, b)$;
  - There are $l \in LB(G, \psi)$, $a, b \in |G_l|$ such that $G_l \models K(a, b)$;
  - Let L_α be the subset of $|G_\alpha|$ and f be the function specified in Lemma 4.4. For some $l \in LB(G, \psi)$,
    - $a = f(l)$, $b \in |G_l|$ and $G_l \models K(r^{G_l}, b)$; or
    - $b = f(l)$, $a \in |G_l|$ and $G_l \models K(a, r^{G_l})$; or
    - $a = b = f(l)$ and $G_l \models K(r^{G_l}, r^{G_l})$.

Intuitively, H is built from G_α and G_l's by identifying $f(l)$ with r^{G_l} for each $l \in LB(G, \psi)$. See Figure 5 for the structure H.

We now show that H is indeed the structure desired.

(1) *The size of H.* Obviously, $size(H)$ is no larger than

$$size(G_\alpha) + \sum_{l \in LB(G,\psi)} size(G_l) - card(LB(G, \psi)).$$

By Lemmas 4.4 and 4.5, it can be shown that $size(H)$ is no larger than $2^{2^{2^{|\psi|}}}$. Note that when $s_\alpha(\psi) = 0$, $size(H)$ is at most $2^{|\psi|}$.

(2) $LB(H, \psi) = LB(G, \psi)$. By Lemmas 4.4, 4.5 and the definition of H, it is easy to verify the following:

Claim: Let L_α be the set and f the function specified in Lemma 4.4. They have the following properties.

1. For each $a \in |H| \setminus L_\alpha$, $lb(a, H, \psi) = (\emptyset, \ \emptyset)$.
2. For each $l \in LB(G, \psi)$, $lb(f(l), H, \psi) = l$.

By the claim, $LB(G, \psi) \subseteq LB(H, \psi)$. In addition, by Lemma 4.4, f is a bijection between $LB(G, \psi)$ and L_α. Therefore, $LB(H, \psi) = LB(G, \psi)$. It should be noted that the proof of the claim uses the restriction on prefixes described in Definition 4.1. ■

4.3 Decidability of Implication Problems for P_β

We now establish the following:

Theorem 4.6: The implication and finite implication problems for P_β are decidable. ∎

In the same way as in the proof of Theorem 4.1, we show Theorem 4.6 by establishing the small model property for the set:

$$S(P_\beta) = \{\bigwedge \Sigma \wedge \neg\varphi \mid \varphi \in P_\beta,\ \Sigma \subset P_\beta,\ \Sigma \text{ is finite}\}.$$

To do this, we give a filtration argument. Given a satisfiable sentence ψ in $S(P_\beta)$, we find the set of paths in ψ and use a path labeling mechanism similar to the one employed in the proof of Theorem 4.1. More specifically, let G be a model of ψ. We use the paths in ψ to label each node of G, and therefore, obtain the label of G with respect to ψ. The cardinality of this label is determined only by $|\psi|$, the length of ψ. We then construct a σ-structure H, such that H and G have the same label with respect to ψ, and moreover, $H \models \psi$. In addition, each node of H has a unique path label. The size of H is, therefore, bounded by the cardinality of the label of G with respect to ψ, which is at most $2^{|\psi|}$. Thus the small model property is established.

We first define the path labels, called *relative path labels*. Using the path labels, we then establish the small model property for $S(P_\beta)$.

4.3.1 Relative path labels

Let ψ be a satisfiable sentence of $S(P_\beta)$, where ψ is $\bigwedge \Sigma \wedge \neg\varphi$. We use the following to denote paths in ψ:

$$\begin{aligned} Paths_{(\alpha,\beta)}(\psi) &= \{pf(\phi) \mid \phi \in \Sigma \cup \{\varphi\}\} \cup \\ &\quad\ \{lt(\phi) \mid \phi \in \Sigma \cup \{\varphi\},\ \phi \in P_s\} \\ I_{(\alpha,\beta)}(\psi) &= \{\rho \mid \varrho \in Paths_{(\alpha,\beta)}(\psi),\ \rho \preceq_p \varrho\} \end{aligned}$$

$$I(\varphi) = \begin{cases} \{\rho \mid \rho \preceq_p rt(\varphi)\} & \text{if } \varphi \text{ has forward form} \\ \{\rho \mid \rho \preceq_s rt(\varphi)\} & \text{if } \varphi \text{ has backward form} \end{cases}$$

Here $\rho \preceq_p \varrho$ ($\rho \preceq_s \varrho$) means that ρ is a prefix (suffix) of ϱ, as defined in Section 2.

Let G be a model of ψ, $G = (|G|, r^G, E^G)$, and (a,b) be a pair of nodes in $|G|$ such that

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

if φ is a forward constraint, and

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

if φ is a backward constraint. This pair is referred to as a *witness of* $\neg\varphi$ *in* G.

For each $c \in |G|$, we label c with a pair. The first component of the pair is

$$ls_{(\alpha,\beta)}(c, G, \psi) = \{\rho \mid \rho \in I_{(\alpha,\beta)}(\psi),\ G \models \rho(r^G, c)\}.$$

The second component, $ls_\varphi(c, a, G, \psi)$, is defined to be

- $\{\rho \mid \rho \in I(\varphi),\ G \models \rho(a,c)\}$ if φ is a forward constraint, and
- $\{\rho \mid \rho \in I(\varphi),\ G \models \rho(c,a)\}$ if φ is a backward constraint.

The *path label of node* c *in* G *relative to* ψ *and* a is defined to be:

$$ls(c, G, \psi, a) = (ls_{(\alpha,\beta)}(c, G, \psi),\ ls_\varphi(c, a, G, \psi))$$

The *path label of* G *relative to* ψ *and* a is defined to be:

$$LS(G, \psi, a) = \{ls(c, G, \psi, a) \mid c \in |G|\}$$

We now examine the cardinality of $LS(G, \psi, a)$. It is easy to verify that $card(I_{(\alpha,\beta)}(\psi)) + card(I(\varphi)) \leq |\psi|$. Note that for each $c \in |G|$, $ls_{(\alpha,\beta)}(c, G, \psi) \subseteq I_{(\alpha,\beta)}(\psi)$ and $ls_\varphi(c, a, G, \psi) \subseteq I(\varphi)$. Hence $card(LS(G, \psi, a))$ is at most $2^{|\psi|}$.

The notion of relative path labels differs from the one described in Section 4.2.1 in the following respects. First, relative path labels are defined for models of satisfiable sentences in $S(P_\beta)$, rather than for arbitrary σ-structures. Second, the relative path label of a node a in a structure involves only the paths between a and two fixed nodes in the structure, namely, the root node and a node in a witness of $\neg\varphi$, whereas the one given in Section 4.2.1 contains paths connecting all pairs of nodes in the structure. As a result, a relative path label has a much smaller cardinality. Third, a relative path label does not characterize whether a σ-structure is a model of a sentence in $S(P_\beta)$, but based on it we are able to construct a filtration argument to establish the small model property for $S(P_\beta)$.

4.3.2 The small model property

Using relative path labels we show the following:

Proposition 4.7: Every satisfiable sentence ψ of $S(P_\beta)$ has a model of size at most $2^{|\psi|}$. ∎

Proof: Let ψ be a satisfiable sentence in $S(P_\beta)$, where $\psi = \bigwedge \Sigma \wedge \neg\varphi$, and $\Sigma \cup \{\varphi\}$ is a finite subset of P_β. Since ψ is satisfiable, there is a σ-structure $G = (|G|, r^G, E^G)$ such that $G \models \psi$. It follows that there exist a, b in $|G|$ such that (a,b) is a witness of $\neg\varphi$ in G. Consider

$LS(G, \psi, a)$. As in the proof of Lemma 4.5, we define an equivalence relation $\sim$ on $|G|$ by:

$$b \sim b' \quad \text{iff} \quad ls(b, G, \psi, a) = ls(b', G, \psi, a).$$

For each $b \in |G|$ we denote the equivalence class of b with respect to $\sim$ as $[b]$. By taking these equivalence classes as nodes, we proceed to construct a σ-structure H as follows:

- $|H| = \{[b] \mid b \in |G|\}$;
- $r^H = [r^G]$;
- for each $K \in E$ and $o_1, o_2 \in |H|$, $H \models K(o_1, o_2)$ iff there exist $b_1, b_2 \in |G|$, such that $[b_1] = o_1$, $[b_2] = o_2$, and $G \models K(b_1, b_2)$.

We next show that $H \models \psi$, and moreover, the size of H is at most $2^{|\psi|}$.

(1) *The size of H.* Since $size(H) = card(LS(G, \psi, a))$, $size(H)$ is at most $2^{|\psi|}$.

(2) $H \models \psi$. It suffices to show the following claims.

Claim 1: For any path ρ and $c, d \in |G|$, if $G \models \rho(c, d)$, then $H \models \rho([c], [d])$.

Claim 2: For each $c \in |G|$,

$$ls(c, G, \psi, a) = ls([c], H, \psi, [a]).$$

Claim 1 can be easily verified by induction on $|\rho|$. Similarly, Claim 2 can be verified by showing that for any $\rho \in I_{(\alpha,\beta)}(\psi)$, $\varrho \in I(\varphi)$ and $c \in |G|$,

$$\rho \in ls_{(\alpha,\beta)}(c, G, \psi) \quad \text{iff} \quad \rho \in ls_{(\alpha,\beta)}([c], H, \psi),$$
$$\varrho \in ls_{\varphi}(c, a, G, \psi) \quad \text{iff} \quad \varrho \in ls_{\varphi}([c], [a], H, \psi).$$

Again, these can be shown by induction on $|\rho|$ and $|\varrho|$.

Using these claims, we prove $H \models \psi$ as follows.

We first show that $H \models \Sigma$. Suppose, for a contradiction, that there exists $\phi \in \Sigma$ such that $H \models \neg\phi$. Without loss of generality, assume that ϕ is a forward constraint (the argument for the backward case is analogous). Then there exist $c, d \in |H|$, such that

$$H \models pf(\phi)(r^H, c) \wedge lt(\phi)(c, d) \wedge \neg rt(\phi)(c, d).$$

We have two cases to consider.

Case 1: ϕ is a simple constraint. That is, $pf(\phi) = \epsilon$ and $c = r^H$. In this case, we have $lt(\phi) \in ls_{(\alpha,\beta)}(d, H, \psi)$ and $H \models \neg rt(\phi)(r^H, d)$. By the definition of H, there exists $d_1 \in |G|$, such that $[d_1] = d$. By Claim 2, $ls(d_1, G, \psi, a) = ls(d, H, \psi, [a])$. By the definition of ls, we have $ls_{(\alpha,\beta)}(d_1, G, \psi) = ls_{(\alpha,\beta)}(d, H, \psi)$. Hence $lt(\phi) \in ls_{(\alpha,\beta)}(d_1, G, \psi)$. That is, $G \models lt(\phi)(r^G, d_1)$. Since $G \models \phi$, we have that $G \models rt(\phi)(r^G, d_1)$. By Claim 1, we have $H \models rt(\phi)(r^H, d)$. This contradicts the assumption.

Case 2: ϕ is a β-restricted constraint, i.e., $|lt(\phi)| \leq 1$.

If $|lt(\phi)| = 0$, then $c = d$. Thus by the assumption, $pf(\phi) \in ls_{(\alpha,\beta)}(c, H, \psi)$ and $H \models \neg rt(\phi)(c, c)$. By the definition of H, there exists $c_1 \in |G|$, such that $[c_1] = c$. By Claim 2, $ls_{(\alpha,\beta)}(c_1, G, \psi) = ls_{(\alpha,\beta)}(c, H, \psi)$. Thus $pf(\phi) \in ls_{(\alpha,\beta)}(c_1, G, \psi)$. That is, $G \models pf(\phi)(r^G, c_1)$. By $G \models \phi$, $G \models rt(\phi)(c_1, c_1)$. Thus by Claim 1, we have $H \models rt(\phi)(c, c)$. This contradicts the assumption.

If $|lt(\phi)| = 1$, then $lt(\phi) = K$ for some $K \in E$. By the assumption, we have $pf(\phi) \in ls_{(\alpha,\beta)}(c, H, \psi)$ and $H \models K(c, d) \wedge \neg rt(\phi)(c, d)$. By the definition of H, there exist $c_1, d_1 \in |G|$, such that $[c_1] = c$, $[d_1] = d$ and moreover, $G \models K(c_1, d_1)$. By Claim 2, we have that $ls_{(\alpha,\beta)}(c_1, G, \psi) = ls_{(\alpha,\beta)}(c, H, \psi)$. As a result, we have $G \models pf(\phi)(r^G, c_1)$. By $G \models \phi$, $G \models rt(\phi)(c_1, d_1)$. Thus by Claim 1, we have $H \models rt(\phi)(c, d)$. Again, this contradicts the assumption.

We next show that $H \models \neg\varphi$. Since (a, b) is a witness of $\neg\varphi$ in G, $G \models pf(\varphi)(r^G, a) \wedge lt(\varphi)(a, b)$. By Claim 1,

$$H \models pf(\varphi)(r^H, [a]) \wedge lt(\varphi)([a], [b]).$$

By Claim 2, $ls_{\varphi}(b, a, G, \psi) = ls_{\varphi}([b], [a], H, \psi)$. Hence when φ is a forward constraint, by $G \models \neg rt(\varphi)(a, b)$, we have that $H \models \neg rt(\varphi)([a], [b])$; and when φ is a backward constraint, by $G \models \neg rt(\varphi)(b, a)$, we have that $H \models \neg rt(\varphi)([b], [a])$. Therefore, $H \models \neg\varphi$. ∎

4.4 Decidability of Extended Implication for P_β

Next, we prove the following:

Theorem 4.8: The extended implication and finite implication problems for P_β are decidable. ∎

We prove the theorem by reduction to the implication problems for P_β, whose decidability is established by Theorem 4.6.

Let Pts be the set of all paths, and let $S_e(P_\beta)$ be

$$\{\bigwedge \Sigma \wedge \neg\varphi \mid \Sigma \cup \{\varphi\} \text{ is a prefix extended subset of } P_\beta\}.$$

Recall the set $S(P_\beta)$ defined in Section 4.3. We define the *prefix extension function from $S(P_\beta)$ to $S_e(P_\beta)$* to be the mapping $f : S(P_\beta) \times Pts \to S_e(P_\beta)$, such that

$$f(\bigwedge \Sigma \wedge \neg\varphi, \alpha) \mapsto \bigwedge_{\phi \in \Sigma} \delta(\phi, \alpha) \wedge \neg\delta(\varphi, \alpha),$$

where δ is described in Definition 4.3.

To prove Theorem 4.8, it suffices to show:

Proposition 4.9: Let ψ be a sentence in $S(P_\beta)$, α a path, and f the prefix extension function from $S(P_\beta)$ to $S_e(P_\beta)$. Then

1. ψ is satisfiable iff $f(\psi, \alpha)$ is satisfiable;
2. ψ is finitely satisfiable iff $f(\psi, \alpha)$ is finitely satisfiable. In addition, if ψ has a finite model of size N, then $f(\psi, \alpha)$ has a model of size $N + |\alpha|$.

■

For if Proposition 4.9 holds, then $S_e(P_\beta)$ has the small model property for satisfiability. More specifically, given $\phi \in S_e(P_\beta)$, we can determine a path α and $\psi \in S(P_\beta)$ in linear time, such that $\phi = f(\psi, \alpha)$. In addition, $|\phi| \geq |\psi| + |\alpha|$. If ϕ is satisfiable, then by Proposition 4.9, so is ψ. By Proposition 4.7, ψ has a model of size at most $2^{|\psi|}$. Thus again by Proposition 4.9, ϕ has a model of size at most $2^{|\psi|} + |\alpha|$, which is no larger than $2^{|\phi|}$. Therefore, $S_e(P_\beta)$ has the small model property and it follows that the extended implication and finite implication problems for P_β are decidable.

Proof of Proposition 4.9: We only prove (2) of the proposition. The proof of (1) is similar.

Let $\psi = \bigwedge \Sigma \wedge \neg\varphi$. Note that if $|\alpha| = 0$, then $f(\psi, \alpha) = \psi$. Obviously, the proposition holds in this case. Hence in the sequel, we assume that $|\alpha| \geq 1$.

Assume that ψ has a finite model $G = (|G|, r^G, E^G)$. We show that $f(\psi, \alpha)$ has a model $H = (|H|, r^H, E^H)$, and moreover, the size of H, $size(H)$, is $size(G) + |\alpha|$.

Let $R_\alpha = \{\rho \mid \rho \text{ is a path}, \rho \prec_p \alpha\}$, where $\rho \prec_p \alpha$ means that ρ is a proper prefix of α. We construct H as follows. For each $\rho \in R_\alpha$, let c_ρ be a distinct node which is not in $|G|$. Let

- $|H| = |G| \cup \{c_\rho \mid \rho \in R_\alpha\}$;
- $r^H = c_\epsilon$;
- 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 $\rho \in R_\alpha$, such that $a = c_\rho$ and $b = c_{\rho \cdot K}$ and $\rho \cdot K \in R_\alpha$; or
  - there exists $\rho \in R_\alpha$, such that $\alpha = \rho \cdot K$ and $a = c_\rho$ and $b = r^G$; or
  - $a, b \in |G|$ and $G \models K(a, b)$.

Obviously, $size(H) = size(G) + |\alpha|$. In addition, it is straightforward to verify that $H \models f(\psi, \alpha)$.

Conversely, suppose that $f(\psi, \alpha)$ has a finite model $G = (|G|, r^G, E^G)$. We construct a finite model of ψ.

Without loss of generality, assume that φ is a forward constraint (the proof for the backward case is analogous). Since $G \models \neg\delta(\varphi, \alpha)$, 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 be the largest natural number in the following set: $\{|pf(\phi)| + |lt(\phi)| + |rt(\phi)| \mid \phi \in \Sigma \cup \{\varphi\}\}$. Let $G(a)$ be the m-neighborhood of a in G, as described in Definition 4.5. Clearly, $G(a)$ is a finite σ-structure. We next prove that $G(a) \models \psi$.

We first show $G(a) \models \neg\varphi$. By $|pf(\varphi)| + |lt(\varphi)| < m$ and $|pf(\varphi)| + |rt(\varphi)| < m$, we have that $b \in |G(a)|$ and $c \in |G(a)|$. Thus by the definition of $G(a)$, we have

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

That is, $G(a) \models \neg\varphi$.

Second, we show by contradiction that for any $\phi \in \Sigma$, $G(a) \models \phi$. Suppose that there exists $\phi \in \Sigma$ such that $G(a) \models \neg\phi$. Without loss of generality, assume that ϕ is a forward constraint (the proof for the backward case is analogous). Then there exist $d, e \in |G(a)|$ such that

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

Thus by the definition of $G(a)$, we have

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

That is, $G \models \neg\delta(\phi, \alpha)$. This contradicts the assumption that $G \models f(\psi, \alpha)$. ■

4.5 Conjunctive Path Constraints

We next show that the complexity results established above also hold for an extension of path constraints. This extension is defined as follows.

Definition 4.6: A *conjunctive path constraint* ϕ is an expression of either the *forward* form

$$\forall x \, (\bigwedge_{\alpha \in A} \alpha(r, x) \rightarrow \forall y \, (\bigwedge_{\beta \in B} \beta(x, y) \rightarrow \gamma(x, y))),$$

or the *backward* form

$$\forall x \, (\bigwedge_{\alpha \in A} \alpha(r, x) \rightarrow \forall y \, (\bigwedge_{\beta \in B} \beta(x, y) \rightarrow \gamma(y, x))),$$

where A, B are non-empty finite sets of paths, and are denoted by $pf(\phi)$ and $lt(\phi)$, respectively. Here γ is a path, denoted by $rt(\phi)$. The set of all conjunctive path constraints is denoted by $P_c^\wedge$. ■

As an example, consider the following conjunctive path constraints:

$$\forall x\,(dept(r,x) \to \forall y\,(ta(x,y) \to student(x,y)))$$
$$\forall x\,(dept(r,x) \to \forall y\,(ta(x,y) \to employee(x,y)))$$
$$\forall x\,(dept(r,x) \to \forall y\,((student(x,y) \wedge employee(x,y)) \to ta(x,y)))$$

Abusing object-oriented database terms, these $P_c^\wedge$ constraints assert:

- TA of a department is a "subclass" of both Student and Employee of the department; and
- the "extent" of TA is the intersection of the "extents" of Student and Employee.

Obviously, P_c is a subclass of $P_c^\wedge$. Therefore, the corollary below follows from Theorem 3.1 immediately.

Corollary 4.10: The implication problem for $P_c^\wedge$ is r.e. complete, and the finite implication problem for $P_c^\wedge$ is co-r.e. complete. ∎

Below we define fragments of $P_c^\wedge$ analogous to the fragments of P_c discussed above.

Definition 4.7: A finite subset Σ of $P_c^\wedge$ is called a *prefix restricted subset* of $P_c^\wedge$ iff for all ϕ, ψ in Σ, all the paths in $pf(\phi) \cup pf(\psi)$ have the same length.

The *prefix restricted (finite) implication problem for* $P_c^\wedge$ is the problem to determine, given any finite prefix restricted subset $\Sigma \cup \{\phi\}$ of $P_c^\wedge$, whether all the (finite) models of Σ are also models of ϕ. ∎

Definition 4.8: A *simple conjunctive path constraint* ϕ is a constraint of $P_c^\wedge$ with $pf(\phi) = \{\epsilon\}$.

A *β-restricted conjunctive path constraint* ϕ is a constraint of $P_c^\wedge$ such that for each $\beta \in lt(\phi)$, $|\beta| \leq 1$.

The sublanguage $P_\beta^\wedge$ is defined to be the class of $P_c^\wedge$ constraints ϕ such that either for any $\alpha \in pf(\phi)$, $|\alpha| = 0$, or for any $\beta \in lt(\phi)$, $|\beta| \leq 1$. That is, $P_c^\wedge$ is the set of all simple conjunctive path constraints and all β-restricted conjunctive path constraints. ∎

Definition 4.9: Let ρ be a path and ϕ be a constraint in $P_\beta^\wedge$. The *extension of ϕ with prefix ρ*, denoted by $\delta(\phi, \rho)$, is the constraint in $P_c^\wedge$ defined either by

$$\forall x\,(\bigwedge_{\alpha \in pf(\phi)} \rho \cdot \alpha(r,x) \to \forall y\,(\bigwedge_{\beta \in lt(\phi)} \beta(x,y) \to rt(\phi)(x,y)))$$

when ϕ is of the forward form, or by

$$\forall x\,(\bigwedge_{\alpha \in pf(\phi)} \rho \cdot \alpha(r,x) \to \forall y\,(\bigwedge_{\beta \in lt(\phi)} \beta(x,y) \to rt(\phi)(y,x)))$$

when ϕ is of the backward form.

Let ρ be a path and Σ a finite subset of $P_\beta^\wedge$. The *extension of Σ with prefix ρ* is the subset of $P_c^\wedge$ defined by $\{\delta(\phi,\rho) \mid \phi \in \Sigma\}$. Such a set is called a *prefix extended subset of $P_\beta^\wedge$*.

The *extended (finite) implication problem for $P_\beta^\wedge$* is the problem of determining, given any prefix extended subset $\Sigma \cup \{\phi\}$ of $P_\beta^\wedge$, whether all the (finite) models of Σ are also models of ϕ. ∎

On semistructured data we have the following, which are analogous to Theorems 4.1, 4.6 and 4.8.

Theorem 4.11: The following problems are decidable:

- The prefix restricted implication and finite implication problems for $P_c^\wedge$.
- The implication and finite implication problems for $P_\beta^\wedge$.
- The extended implication and finite implication problems for $P_\beta^\wedge$.

∎

With slight modification, the proofs of Theorems 4.1, 4.6 and 4.8 are applicable to Theorem 4.11.

With thanks to an anonymous referee, we observe that the arguments for these theorems can even be used to establish the decidability of certain extensions of the decidable fragments of P_c and $P_c^\wedge$. For example, the proof of Theorem 4.1 yields a stronger result: the satisfiability of any Boolean combination of constraints in prefix restricted subsets of P_c is decidable. More specifically, let Σ be a a prefix restricted subset of P_c. We define a set $B(\Sigma)$ of logic sentences as follows:

- $\Sigma \subseteq B(\Sigma)$;
- if $\varphi \in B(\Sigma)$, then so is $\neg\varphi$;
- if φ and ϕ are in $B(\Sigma)$, then so are $\varphi \wedge \phi$ and $\varphi \vee \phi$.

The *(finite) satisfiability problem for Boolean combinations of constraints in prefix restricted subsets of P_c* is the problem to determine, given any prefix restricted subset Σ of P_c and any $\varphi \in B(\Sigma)$, whether φ has a (finite) model.

With slight modification, the argument for Theorem 4.1 can be used to prove the following:

Proposition 4.12: The satisfiability and finite satisfiability problems for Boolean combinations of constraints in prefix restricted subsets of P_c are decidable. ∎

5 CONCLUSIONS

We have introduced a class of path constraints, P_c, and investigated its associated implication and finite implication problems. These path constraints capture many natural integrity constraints that commonly arise in both structured and semistructured databases. They are not only a fundamental part of the semantics of the data; they are also useful in query optimization. The importance of these constraints was also emphasized in several XML proposals (e.g., [10, 26, 31, 32]). Due to the recent popularity of the World Wide Web and the success of the XML standard [11], these constraints have found a wide range of applications.

In the context of semistructured data, we have shown that, despite the simple syntax of the language P_c, its associated implication problem is r.e. complete and its finite implication problem is co-r.e. complete. These results are rather surprising since P_c is a mild generalization of word constraints introduced and studied in [5], for which the implication and finite implication problems are in PTIME. In light of these undecidability results, we have also identified several fragments of P_c which suffice to express many interesting semantic relations such as extent, inverse and local database constraints, and properly contain the class of word constraints. We have established the decidability of the implication and finite implication problems associated with each of these fragments.

Another issue of equal importance is the interaction between path and type constraints. Although the XML standard itself does not require any schema or type system, a number of proposals have been developed that allow one to constrain the structure of XML data by imposing a schema or a type constraint on it. These and other proposals also advocate the need for certain integrity constraints, which can be expressed as P_c constraints. It is likely that future XML proposals will involve both forms of constraints, and it is therefore appropriate to understand the interaction between them. It would be tempting to directly apply the complexity results developed for semistructured data to typed data. However, we have shown in [15, 16] that path constraints interact with type constraints. More specifically, a number of decidability and undecidability results have been established there which demonstrate that adding a type system may in some cases simplify reasoning about path constraints, and in other cases make it harder. A full treatment of these results will appear in a future publication.

ACKNOWLEDGEMENTS

We thank Leonid Libkin, Val Tannen and Victor Vianu for valuable comments and discussions. We also thank the referees whose detailed suggestions have substantially improved this paper.

REFERENCES

[1] S. O. Aanderaa, On the decision problem for formulas in which all disjunctions are binary, *Proc. 2nd Scandinavian Logic Symp.*, 1971, pp. 1-18.

[2] S. Abiteboul, Querying semi-structured data, *Proc. 6th Int. Conf. on Database Theory*, 1997, pp. 1-18.

[3] S. Abiteboul, R. Hull, and V. Vianu, "Foundations of Databases", Addison-Wesly, 1995.

[4] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Weiner, The lorel query language for semistructured data, *J. Digital Libraries* **1**(1), 1997, pp. 68-88.

[5] S. Abiteboul and V. Vianu, Regular path queries with constraints, *Proc. 16th ACM Symp. on Principles of Database Systems*, 1997, pp. 122-133.

[6] F. Bancilhon, C. Delobel, and P. Kanellakis (ed.), "Building an object-oriented database system: the story of O2", Morgan Kaufmann, San Mateo, California, 1992.

[7] J. Barwise, On Moschovakis closure ordinals, *J. Symbolic Logic* **42**, 1977, pp. 292-296.

[8] M. F. van Bommel and G. E. Weddell, Reasoning about equations and functional dependencies on complex objects, *IEEE Trans. on Knowledge Data Engrg.* **6**(3), 1994, pp. 455-469.

[9] E. Börger, E. Grädel, and Y. Gurevich, "The classical decision problem", Springer-Verlag, 1997.

[10] 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`.

[11] 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/TR/REC-xml`.

[12] P. Buneman, Semistructured data, Tutorial in *Proc. 16th ACM Symp. on Principles of Database Systems*, 1997, pp. 117-121.

[13] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu, Adding structure to unstructured data, *Proc. 6th Int. Conf. on Database Theory*, 1997, pp. 336-350.

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

[15] P. Buneman, W. Fan, and S. Weinstein, Path constraints on semistructured and structured data, *Proc. 17th ACM Symp. on Principles of Database Systems*, 1998, pp. 129-138.

[16] P. Buneman, W. Fan, and S. Weinstein, Interaction between path and type constraints, *Proc. 18th ACM Symp. on Principles of Database Systems*, 1999, pp. 56-67. Full paper available as Technical Report MS-CIS-98-16, Department of Computer and Information Science, University of Pennsylvania, 1998.

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

[18] U. S. Chakravarthy, J. Grant, and J. Minker, Foundations of semantic query optimization for deductive databases, J. Minker (ed.), "Foundations of Deductive Databases and Logic Programming", Morgan Kaufmann, San Mateo, California, 1988, pp. 243-273.

[19] S. Cluet and C. Delobel, A general framework for the optimization of object-oriented queries, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1992, pp. 383-392.

[20] 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`.

[21] H.-D. Ebbinghaus and J. Flum, "Finite Model Theory", Springer, 1995.

[22] H. B. Enderton, "A mathematical introduction to logic", Academic Press, 1972.

[23] M. Fernandez, D. Florescu, J. Kang, A. Levy, and D. Suciu, Catching the boat with Strudel: experience with a Web-site management system, *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 1998, pp. 414-425.

[24] M. Fernandez and D. Suciu, Optimizing regular expressions using graph schemas, *Proc. Int. Conf. on Data Engineering*, 1998, pp. 14-23.

[25] D. Florescu, L. Raschid, and P. Valduriez, A methodology for query reformulation in CIS using semantic knowledge, *Int. J. Cooperative Information Systems* **5**(4), 1996, pp. 431-468.

[26] 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`.

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

[28] N. Immerman, Upper and lower bounds for first order expressibility, *J. Comput. System Sci.* **25**(1), 1982, pp. 76-98.

[29] M. Ito and G. E. Weddell, Implication problems for functional constraints on databases supporting complex objects, *J. Comput. System Sci.* **50**(1), 1995, pp. 165-187.

[30] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb, The ObjectStore Database system, *Communications of the ACM* **34**(10), 1991, pp. 51-63.

[31] 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`.

[32] 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. `http://www.w3.org/TR/1998/NOTE-XML-data`.

[33] A. O. Mendelzon, G. A. Mihaila, and T. Milo, Querying the World Wide Web, *J. Digital Libraries* **1**(1), 1997, pp. 54-67.

[34] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, Object exchange across heterogeneous information sources, *Proc. 11th Int. Conf. on Data Engineering*, 1995, pp. 251-260.

[35] L. Popa and V. Tannen, An equational chase for path-conjunctive queries, constraints, and views, *Proc. 7th Int. Conf. on Database Theory*, 1999, pp. 39-57.

[36] H. Wang, Dominoes and the $\forall\exists\forall$-case of the decision problem, *Proc. of Symp. on Mathematical Theory of Automata*, Brooklyn Polytechnic Institute, 1962, pp. 23-55.