April 1997

Some Undecidable Implication Problems for Path Constraints

Peter Buneman
University of Pennsylvania

Wenfei Fan
University of Pennsylvania

Scott Weinstein
University of Pennsylvania, weinstein@cis.upenn.edu

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-97-14.

Some Undecidable Implication Problems for Path Constraints

Abstract

We present a class of path constraints of interest in connection with both structured and semi-structured databases, and investigate their associated implication problems. These path constraints are capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization. We show that, despite the simple syntax of the constraints, the implication problem for the constraints is r.e. complete and the finite implication problem for the constraints is co-r.e. complete. Indeed, we establish the existence of a conservative reduction of the set of all first-order sentences to the path constraint language.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-97-14.

Some Undecidable Implication Problems for Path Constraints

Peter Buneman[*] peter@central.cis.upenn.edu

Wenfei Fan[†] wfan@saul.cis.upenn.edu

Scott Weinstein[‡] weinstein@linc.cis.upenn.edu

Department of Computer and Information Science
University of Pennsylvania

April 1997

Abstract

We present a class of path constraints of interest in connection with both structured and semistructured databases, and investigate their associated implication problems. These path constraints are capable of expressing natural integrity constraints that are not only a fundamental part of the semantics of the data, but are also important in query optimization. We show that, despite the simple syntax of the constraints, the implication problem for the constraints is r.e. complete and the finite implication problem for the constraints is co-r.e. complete. Indeed, we establish the existence of a conservative reduction of the set of all first-order sentences to the path constraint language.

1 Introduction

Path inclusion constraints have been studied in [5] in the context of semistructured data.

Consider the following object-oriented schema:

```
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 we would expect the following informally stated constraints to hold:

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

[†]Supported by an IRCS graduate fellowship.

[‡]Supported by NSF Grant CCR-9403447.

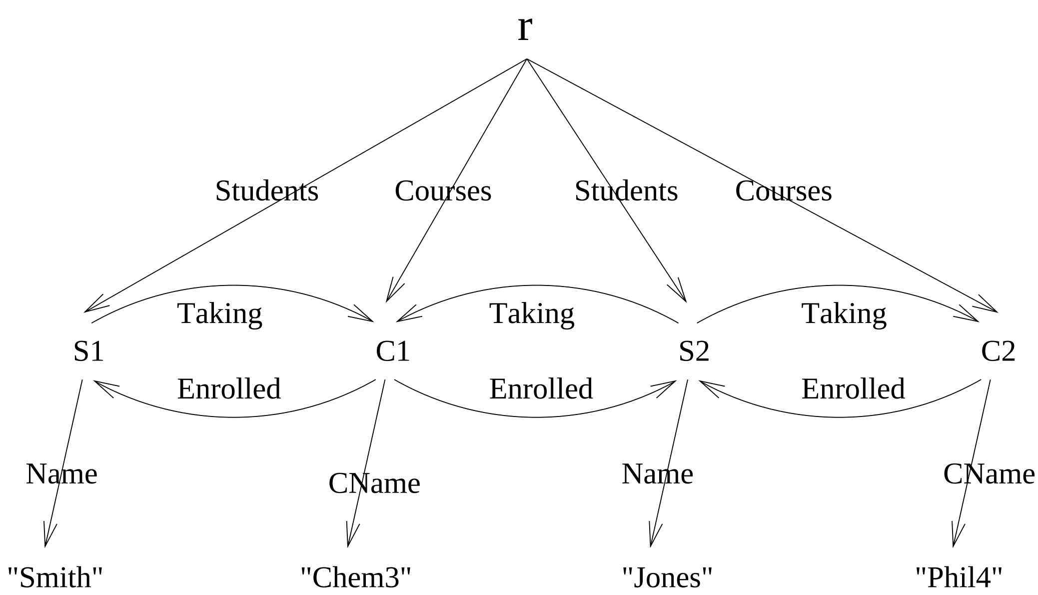

Figure 1: Representation of a student/course database

(a) $\forall\, s \in Students \;\; \forall\, c \in s.Taking \;\; (c \in Courses)$

(b) $\forall\, c \in Courses \;\; \forall\, s \in c.Enrolled \;\; (s \in Students)$

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.

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 [21]. The presence of such constraints is important both for database and for query optimization.

Let us develop a more formal notation for describing such constraints. To do this we borrow an idea that has been exploited in semistructured data models (e.g., [26, 11, 4, 25, 22]) of regarding semistructured data as an edge-labeled graph. The database consists of two sets, and we express this by a root node r with edges emanating from it that are labeled either `Students` or `Courses`. These connect to nodes that respectively represent students and courses which have edges emanating from them that respectively describe the structure of students and courses. For example a student 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.

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:

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

These constraints are examples of “word constraints” studied in [5]; the implication problems for word constraints were shown to be decidable there.

Inverse Constraints. These are common in object-oriented databases [13]. With respect to our student/course schema, the inverse between `Taking` and `Enrolled` is expressed as:

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

Such constraints 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 wanted 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" 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 the new root node to the roots of local databases, the local extent constraints are:

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

Again, these cannot be stated as path 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 of either the form

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

or the form

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

where $\alpha(x,y)$ ($\beta(x,y)$, $\gamma(x,y)$) represents a path from node x to node y.

This class of path constraints can be used to express all the constraints we have so far encountered. For semistructured data, in particular, this class of constraints is useful not only for optimizing navigational queries, but also for inferring structure (see [10, 24, 25] on this subject). Surprisingly, the implication problems for this mild generalization of word constraints are undecidable. However, certain restricted cases are decidable in semistructured databases, and these cases are sufficient to express at least the constraints we have described above. In this paper, we establish the undecidability of the implication problems for this class of path constraints. We defer to another paper [12] a full treatment of the decidability results.

Related work

The idea of representing data as an edge-labeled graph and using paths to specify navigational queries dates back to the early 1980s [23, 30]. Recently, the idea has been exploited and adapted to a variety of new database applications, ranging from querying object-oriented databases (e.g., XSQL [20], OQL-doc [3]) to querying semistructured data (e.g., UnQL [11], Lorel [4], WebSQL [22], STRUQL [16]).

There has also been work in constraint languages defined in terms of paths for structured data [28, 14, 8, 18, 29, 19] as well as for semistructured data [5]. A class of functional constraints, called *path functional dependencies*, was proposed in [28, 14]. The axiomatizability and decidability of its associated unrestricted implication problem were established in [28, 8, 18]. This constraint language generalizes functional dependencies in the relational data model for semantic and object-oriented data models. It differs significantly from ours, which is a generalization of (unary) inclusion dependencies in the relational model for both structured and semistructured data.

In [29, 14], a class of constraints for specifying range restrictions associated with paths, called *specialization constraints*, was proposed for object-oriented data models. The axiomatizability and decidability of its associated implication problems were established in [19]. A specialization constraint asserts a type condition which is often specified by a schema. The central difference between specialization constraints and our path constraints is that specialization constraints are type constraints for a graph representing a schema, whereas our path constraints specify inclusion relations for a graph representing data. In particular, specialization constraints must be associated with a schema, whereas our path constraints are defined for both structured and semistructured data.

Closer to our work is the path inclusion constraint language introduced and investigated 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 denoting paths in a graph representing semistructured data. In particular, if p and q are simply sequences of labels, the constraint is called a *word constraint*. A constraint $p \subseteq q$ ($p = q$) expresses the inclusion (equality) relation between the two sets of nodes reachable via p and q. The decidability of the implication problems for the language was established in [5]. In addition, [5] also showed that word constraint implication is decidable in PTIME. This constraint language differs from ours in expressive power. On the one hand, the constraint language of [5] allows a more general form of path expressions than ours. On the other hand, it does not capture inverse constraints and local database constraints mentioned above, whereas these constraints can be expressed in our language. In short, our constraints language is a generalization of the class of word constraints given in [5].

The rest of the paper is organized as follows. Section 2 formally presents our path constraint language and identifies two of its fragments. Section 3 and 4 show that for each of the two fragments, the implication problem is r.e. complete and the finite implication problem is co-r.e. complete, and therefore establish the undecidability of the implication problems for our path constraint language.

2 Path Constraint Language P

In this section, we formalize the path constraints language, P. We first present an abstraction of semistructured databases, and define the language P in terms of first-order logic. We then describe the implication problems for P and state the main results of the paper.

We assume the standard notions of sentences, models and implication used in first-order logic [15].

2.1 Abstraction of semistructured data

Semistructured data is usually represented as an edge-labeled (rooted) directed graph, e.g., in UnQL [11] and in OEM [26, 4, 25]. See [2] for a survey of semistructured data models. Along the same lines, here we use an abstraction of semistructured databases as (finite) first-order logic structures of signature

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

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

2.2 Path constraints

A path, i.e., a sequence of labels, can be represented as a logic formula with two free variables.

Definition 2.1: A path is a formula $\alpha(x, y)$ having one of the following forms:

- $x = y$, denoted $\epsilon(x, y)$ 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. ■

The path constraint language P is formalized as follows.

Definition 2.2: A *path constraint* φ is an expression of either the *forward* form

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

or the *backward* form

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

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

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

For example, all the path constraints presented in the last section are constraints in the set P.

Next, we identify several special subclasses of P.

We call a path constraint φ in P a *simple path constraint* if $pf(\varphi) = \epsilon$. That is, 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* and denoted by P_w, was introduced and investigated in [5]. A word constraint can be represented as

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

where β and γ are paths.

2.3 Path constraint implication

We next describe the implication problems for path constraints in P.

We borrow the standard notations of models and implication from first-order logic [15]. For a σ-structure G and a constraint φ in P, we use $G \models \varphi$ to denote that G *satisfies* φ (i.e., G is a model of φ). For any finite subset $\Sigma \cup \{\varphi\}$ of P, we use $\Sigma \models \varphi$ to denote that Σ *implies* φ. That is, for every 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 structure G, if $G \models \Sigma$, then $G \models \varphi$.

The *implication problem* for P is the problem of determining, given any finite set $\Sigma \cup \{\varphi\}$ of sentences in P, whether $\Sigma \models \varphi$. The *finite implication problem* for P is the problem of determining, given any finite subset $\Sigma \cup \{\varphi\}$ of P, 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, [17] has shown that the satisfiability problem for FO^2 is NEXPTIME-complete by establishing that any satisfiable FO^2 sentence has a model of size exponential in the length of the sentence. The decidability of the implication and finite implication problems for word constraints (and for simple constraints) follows immediately. In fact, [5] directly establishes (without reference to the embedding into FO^2) that the implication problems for word constraints are in PTIME.

In contrast to word constraint implication, both the implication and the finite implication problems for P are undecidable. These undecidability results, which are the main results of the paper, are stated as follows.

Theorem 2.1: The implication problem for P is r.e. complete, and the finite implication problem for P is co-r.e. complete. ■

In fact, these results hold true for two proper subclasses of P. One of the subclasses, P_f, is the set of all the constraints of the forward form in P. The other, P_+, is the set

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

Note that P_+ is the largest subset of P without equality.

For P_+ and P_f we have the following theorems, from which Theorem 2.1 follows immediately.

Theorem 2.2: The implication problem for P_+ is r.e. complete, and the finite implication problem for P_+ is co-r.e. complete. ■

Theorem 2.3: The implication problem for P_f is r.e. complete, and the finite implication problem for P_f is co-r.e. complete. ■

We prove Theorem 2.2 and 2.3 in the next two sections.

3 The Implication Problems for P_+

In this section, we establish the undecidability of the implication and finite implication problems for P_+.

3.1 Preliminaries

We first recall the definitions of two-register machines and conservative reduction classes from [1, 9].

3.1.1 Two-register machines

A two-register machine (2-RM) M consists of 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 of M is 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 $register_2$, $0 \leq i, j \leq l$. The semantics of the addition instruction 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), where rg is either $register_1$ or $register_2$, $0 \le i,j,k \le l$. The semantics of the subtraction instruction 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 $\rightarrow_M$ rewrite rules.

We will need the following definitions from [1, 9].

Definition 3.1 [1, 9]: Let X be a class of sentences. We write

- $N(X)$ for the set of all *unsatisfiable* sentences in X; that is,

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

- $F(X)$ for the set of all *finitely satisfiable* sentences in X; that is,

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

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

Recall the following well-known result [27]:

> There is 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 two-register machine with the following behavior (the existence of such a machine follows from the result just quoted. See [9] for additional discussions). The two-register machine 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 $C(\psi)$ 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;
- 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 $F(FO)$.

Here halting of M_L means that the ID sequence becomes constant when reaching a stop state. This stop condition can be assumed without loss of generality [9].

3.1.2 Conservative reductions

Recall the following definitions from [1, 9].

Definition 3.2 [9]: Let X and Y be recursive classes of sentences. A *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.

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 a *conservative reduction class* if there exists a conservative reduction from FO to X. ∎

The *(finite) satisfiability problem* for a recursive class of sentences X is the problem of determining, given any $\psi \in X$, whether ψ has a (finite) model.

Obviously, 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.

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. More precisely, we define the notion of semi-conservative reductions as follows.

Definition 3.3 [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 3.1 [9]: If there exists a semi-conservative reduction from FO to a recursive subset X of FO, then X is a conservative reduction class. ∎

3.2 Reduction from the halting problem for 2-RMs

We next show Theorem 2.2. It suffices to show that the set

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

is a *conservative reduction class*. To establish the conservative reduction class property for $S(P_+)$, by Lemma 3.1, it suffices to show that there is a semi-conservative reduction from FO to $S(P_+)$.

We establish the existence of such a semi-conservative reduction by reduction from the halting problem for two-register machines. We first present an encoding of two-register machines by sentences 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

Let M be a two-register machine. Assume that M is programmed by

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

We also assume that the set E of binary relations in the signature σ includes:

- the predicates encoding the states of M:
  - $K_0, K_1, ..., K_l$,
  - $K_0^-, K_1^-, ..., K_l^-$;
- the predicates encoding the contents of the registers (natural numbers):
  - R_1^+, R_1^-: to encode the successor and the predecessor of the contents of $register_1$;
  - R_2^+, R_2^-: to encode the successor and the predecessor of the contents of $register_2$;
  - E_{01}, E_{01}^-: to indicate that the content of $register_1$ is 0;
  - E_{02}, E_{02}^-: to indicate that the content of $register_2$ is 0;
- the predicates distinguishing $register_1$ from $register_2$:
  - 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 now define the encoding as follows.

Registers

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

- Successor, predecessor:
  - $\phi_1 = \forall xy(L_1(r,x) \wedge R_1^+(x,y) \rightarrow R_1^-(y,x))$
  - $\phi_2 = \forall xy(L_1(r,x) \wedge R_1^-(x,y) \rightarrow R_1^+(y,x))$
  - $\phi_3 = \forall xy(L_2(r,x) \wedge R_2^+(x,y) \rightarrow R_2^-(y,x))$
  - $\phi_4 = \forall xy(L_2(r,x) \wedge R_2^-(x,y) \rightarrow R_2^+(y,x))$

    (ϕ_1, ϕ_2, ϕ_3 and ϕ_4 are constraints of the backward form.)
  - $\phi_5 = \forall x(L_1(r,x) \rightarrow \exists y(R_1^+(x,y) \wedge L_1^-(y,r)))$.
  - $\phi_6 = \forall x(L_2(r,x) \rightarrow \exists y(R_2^+(x,y) \wedge L_2^-(y,r)))$.

    (ϕ_5 and ϕ_6 are simple constraints of the backward form.)
- Register identification:
  - $\phi_7 = \forall x(\exists y(L_1(r,y) \wedge R_1^+(y,x)) \rightarrow L_1(r,x))$
  - $\phi_8 = \forall x(\exists y(L_1(r,y) \wedge R_1^-(y,x)) \rightarrow L_1(r,x))$
  - $\phi_9 = \forall x(\exists y(L_2(r,y) \wedge R_2^+(y,x)) \rightarrow L_2(r,x))$
  - $\phi_{10} = \forall x(\exists y(L_2(r,y) \wedge R_2^-(y,x)) \rightarrow L_2(r,x))$

    (ϕ_7, ϕ_8, ϕ_9 and ϕ_{10} are simple constraints of the forward form.)

- States: for $i \in [0, l]$,
  - $\phi^i_{11} = \forall xy(L_1(r,x) \wedge K_i(x,y) \to K_i^-(y,x))$
  - $\phi^i_{12} = \forall xy(L_2(r,x) \wedge K_i^-(x,y) \to K_i(y,x))$

    (ϕ^i_{11} and ϕ^i_{12} are constraints of the backward form.)

- Zeros:
  - $\phi_{13} = \forall xy(L_1(r,x) \wedge E_{01}^-(x,y) \to E_{01}(y,x))$
  - $\phi_{14} = \forall xy(L_1(r,x) \wedge E_{01}^-(x,y) \to L_r(r,y))$
  - $\phi_{15} = \forall xy(L_r(r,x) \wedge E_{01}(x,y) \to E_{01}(r,y))$
  - $\phi_{16} = \forall x(\exists z(L_1(r,z) \wedge \exists y(E_{01}^-(z,y) \wedge E_{02}(y,x))) \to E_{02}(r,x))$

    (ϕ_{13} is a constraint of the backward form, ϕ_{14}, ϕ_{15} and ϕ_{16} are simple constraints of the forward form.)
  - $\phi_{17} = \forall x(E_{01}(r,x) \to L_1(r,x))$
  - $\phi_{18} = \forall x(E_{02}(r,x) \to L_2(r,x))$

    (ϕ_{17} and ϕ_{18} are simple constraints of the forward form.)

IDs

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

$$\varphi_C = \forall xy(L_1(r,x) \wedge (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n(x,y) \to K_i(x,y)),$$

where

- $\alpha \cdot \beta$ (abbreviation for $\alpha(x,z) \cdot \beta(z,y)$) stands for the concatenation of paths $\alpha(x,z)$ and $\beta(z,y)$, which is defined by:

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

- $(\alpha)^m$ stands for the m-time concatenation of α, which is defined by:

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

It is easy to see that the constraint φ_C is in P_+. It has the forward form with $pf(\varphi_C) = L_1$, $lt(\varphi_C) = (R_1^-)^m \cdot E_{01}^- \cdot E_{02} \cdot (R_2^+)^n$, and $rt(\varphi_C) = K_i$.

Instructions

For each $i \in [0, l]$, we encode the instruction I_i by ϕ_{I_i} given below.

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

$$\phi^i_{a_1} = \forall xy(L_1(r,x) \wedge \exists x'(R_1^-(x,x') \wedge K_i(x',y)) \to K_j(x,y)).$$

    Here $\phi^i_{a_1}$ is a constraint of the forward form with $pf(\phi^i_{a_1}) = L_1$, $lt(\phi^i_{a_1}) = R_1^- \cdot K_i$ and $rt(\phi^i_{a_1}) = K_j$.

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

$$\phi^i_{a_2} = \forall xy(L_1(r,x) \wedge \exists y'(K_i(x,y') \wedge R_2^+(y',y)) \to K_j(x,y)).$$

Here $\phi^i_{a_2}$ is a constraint of the forward form with $pf(\phi^i_{a_2}) = L_1$, $lt(\phi^i_{a_2}) = K_i \cdot R_2^+$ and $rt(\phi^i_{a_2}) = K_j$.

- Subtraction:

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

$$\phi^i_{s_1} = \phi^i_{s_{1,0}} \wedge \phi^i_{s_{1,n}},$$

where

$$\phi^i_{s_{1,0}} = \forall xy(E_{01}(r,x) \wedge K_i(x,y) \to K_j(x,y)),$$

and

$$\phi^i_{s_{1,n}} = \forall xy(L_1(r,x) \wedge \exists x'(R_1^+(x,x') \wedge K_i(x',y)) \to K_k(x,y)).$$

Here $\phi^i_{s_1}$ is a conjunction of two constraints having the forward form. The first conjunct is a constraint with $pf(\phi^i_{s_{1,0}}) = E_{01}$, $lt(\phi^i_{s_{1,0}}) = K_i$ and $rt(\phi^i_{s_{1,0}}) = K_j$. In the second conjunct, $pf(\phi^i_{s_{1,n}}) = L_1$, $lt(\phi^i_{s_{1,n}}) = R_1^+ \cdot K_i$ and $rt(\phi^i_{s_{1,n}}) = K_k$.

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

$$\phi^i_{s_2} = \phi^i_{s_{2,0}} \wedge \phi^i_{s_{2,n}},$$

where

$$\phi^i_{s_{2,0}} = \forall xy(E_{02}(r,y) \wedge K_i^-(y,x) \to K_j(x,y)),$$

and

$$\phi^i_{s_{2,n}} = \forall xy(L_1(r,x) \wedge \exists y'(K_i(x,y') \wedge R_2^-(y',y)) \to K_k(x,y)).$$

Here $\phi^i_{s_2}$ is a conjunction of two constraints. The first conjunct is a constraint of the backward form with $pf(\phi^i_{s_{2,0}}) = L_1$, $lt(\phi^i_{s_{2,0}}) = K_i^-$ and $rt(\phi^i_{s_{2,0}}) = K_j$. The second conjunct is a constraint of the forward form with $pf(\phi^i_{s_{2,n}}) = L_1$, $lt(\phi^i_{s_{2,n}}) = K_i \cdot R_2^-$ and $rt(\phi^i_{s_{2,n}}) = K_k$.

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_+.

3.2.2 Reduction property

Now we show that the encoding above has the following reduction property.

Proposition 3.2: Let M be a two-register machine. For all IDs C and D of M,

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

■

Proof:

(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 E of M, if E is reached by M in t steps starting from C (denoted $C \Rightarrow^t_M E$), then $G \models \varphi_E$. We prove this 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 that $C \Rightarrow_M^t C_1 \rightarrow_M^{I_i} E$, where $C_1 = (i, m, n)$, and $C_1 \rightarrow_M^{I_i} E$ stands for that E is reached by executing instruction I_i at C_1. Then by the induction hypothesis, we have $G \models \varphi_{C_1}$. That is

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

We show that the claim holds for $t+1$ for each case of I_i, which has six cases in total.

Case 1: $I_i = (i, register_1, j)$. In this case, E must be $(j, m+1, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

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

then 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, we have $G \models L_1(r,c)$. Thus 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 $G \models K_j(a,b)$. This contradicts the assumption.

Case 2: $I_i = (i, register_2, j)$. In this case, E must be $(j, m, n+1)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

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

then 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 $G \models K_i(a,c)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,c) \wedge R_2^+(c,b).$$

Thus by $\phi_{a_2}^i$ in Φ_M, we have $G \models K_j(a,b)$. This contradicts the assumption.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, E must be $(j, 0, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

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

then 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, E must be (k, p, n).

Suppose, for *reductio*, 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, 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 $G \models L_1(r,c) \wedge R_1^-(c,a)$. 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 $G \models \varphi_{C_1}$, $G \models K_i(c,b)$. Hence

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

Thus by $\phi^i_{s_{1,n}}$ in Φ_M, 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, E must be $(j, m, 0)$.
Suppose, for *reductio*, 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 $G \models K_i(a,b)$. By ϕ^i_{11} 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, $G \models E_{02}(r,b)$. Hence

$$G \models E_{02}(r,b) \wedge K_i^-(b,a).$$

Thus by $\phi^i_{s_{2,0}}$ 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, E must be (k, m, p).
Suppose, for *reductio*, 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, 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 $G \models R_2^+(b,e)$, $G \models R_2^-(e,b)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,e) \wedge R_2^-(e,b).$$

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 $C \not\Rightarrow_M D$. We show that $\Phi_N \wedge \Phi_M \wedge \varphi_C \to \varphi_D$ is not valid. That is, we show that $\Phi_N \wedge \Phi_M \wedge \varphi_C \wedge \neg\varphi_D$ is satisfiable. To show this, we construct a structure G such that $G \models \Phi_N \wedge \Phi_M \wedge \varphi_C$ and $G \models \neg\varphi_D$.

The structure G 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:

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

See Figure 2 for the structure G ($E_{01}^-, E_{02}^-, L_1^-, L_2^-, R_1^-, R_2^-, K_i^-$ edges are omitted in the graph). It is easy to verify the following claims.
Claim 1: $G \models \varphi_C \wedge \neg\varphi_D$.
This is because $C \Rightarrow_M C$ and $C \not\Rightarrow_M D$.

Claim 2: $G \models \Phi_N$.
This is immediate from the construction of G.

Claim 3: $G \models \Phi_M$.
Claim 3 follows from the simple facts given below.

- *Fact 1*: $G \models K_i(m, n')$ iff $C \Rightarrow_M (i, m, n)$.
- *Fact 2*: If $C \Rightarrow_M (i, m, n) \rightarrow_M^{I_i} E$, then $C \Rightarrow_M E$. Moreover, E must satisfy the following conditions.
  - If $I_i = (i, register_1, j)$, then $E = (j, m+1, n)$.
  - If $I_i = (i, register_2, j)$, then $E = (j, m, n+1)$.
  - If $I_i = (i, register_1, j, k)$ and $m = 0$, then $E = (j, 0, n)$.
  - If $I_i = (i, register_1, j, k)$ and $m = p+1$, then $E = (j, p, n)$.
  - If $I_i = (i, register_2, j, k)$ and $n = 0$, then $E = (j, m, 0)$.
  - If $I_i = (i, register_2, j, k)$ and $n = p+1$, then $E = (j, m, p)$.
- *Fact 3*: If $G \not\models \Phi_M$, i.e., there exists I_i for some $i \in [0, l]$ such that $G \not\models \phi_{I_i}$, then there exist $m, n' \in |G|$, such that
  - if $I_i = (i, register_1, j)$, then $G \models K_i(m, n') \wedge \neg K_j(m+1, n')$,
  - if $I_i = (i, register_2, j)$, then $G \models K_i(m, n') \wedge \neg K_j(m, (n+1)')$,
  - if $I_i = (i, register_1, j, k)$, then either
    * $G \models K_i(0, n') \wedge \neg K_j(0, n')$, where $m = 0$, or
    * $G \models K_i(p+1, n') \wedge \neg K_k(p, n')$, where $m = p+1$,
  - if $I_i = (i, register_2, j, k)$, then either
    * $G \models K_i(m, 0') \wedge \neg K_j(m, 0')$, where $n = 0$, or

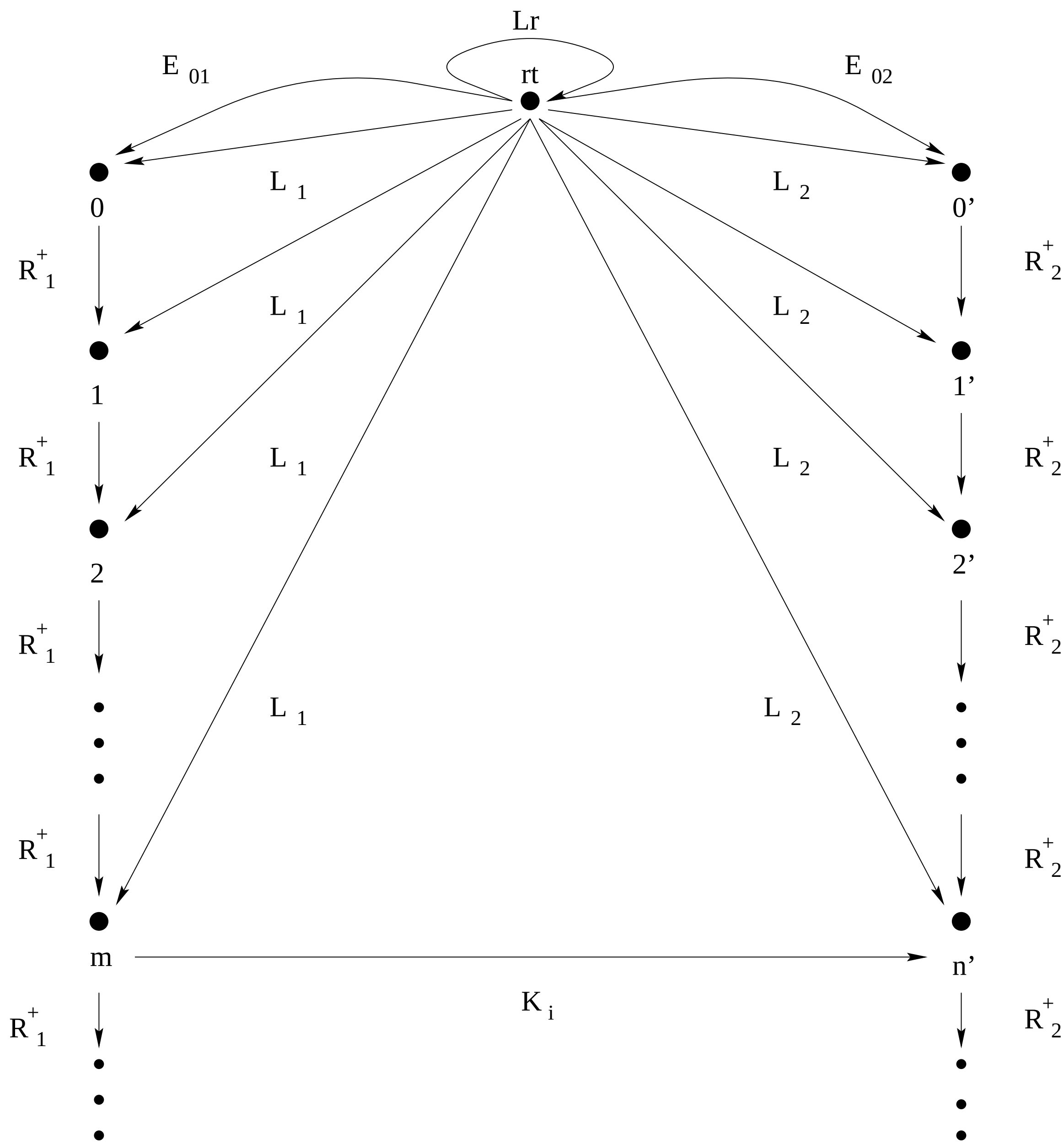

Figure 2: The structure G in Proposition 3.2

* $G \models K_i(m, (p+1)') \wedge \neg K_k(m, p')$, where $n = p + 1$.

Using these facts, Claim 3 can be easily verified by *reductio*. More specifically, suppose that $G \not\models \Phi_M$. 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 $I_i = (i, register_1, j)$. By Fact 3, there exist $m, n' \in |G|$, 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, $C \Rightarrow_M (j, m+1, n')$. 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.
This completes the proof of Proposition 3.2. ■

3.2.3 Semi-conservative reduction

Taking advantage of the reduction property established above, we now 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 M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 3.1.1.

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

Proposition 3.3: Let M_L be the two-register machine described in Section 3.1.1. For each first-order sentence ψ,

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:
(1) By Proposition 3.2, we have that

$$C(\psi) \Rightarrow_{M_L} (1,0,0) \quad \text{iff} \quad \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \to \varphi_{(1,0,0)} \text{ is valid.}$$

Therefore,

$$C(\psi) \Rightarrow_{M_L} (1,0,0) \quad \text{iff} \quad \Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)} \text{ is not satisfiable.}$$

Notice that $\psi \in H_{M_L,1}$ iff $C(\psi) \Rightarrow_{M_L} (1,0,0)$. Therefore, $\psi \in H_{M_L,1}$ iff $f(\psi)$ is not satisfiable. This completes the proof of claim 1.

(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 $(i,m,n) \in SID_{C(\psi)}$, $m+2 \leq p$ and $n+2 \leq p$.
Now we construct a finite model H for $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$. The universe of H has $2p+1$ 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 structure G given in the proof of Proposition 3.2. 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:

$$\begin{array}{lllllll}
R^+_1 & = & \{(i, i+1) \mid 0 \leq i < p\} \cup \{(p,p)\} & & & \\
R^-_1 & = & \{(i+1, i) \mid 0 \leq i < p\} \cup \{(p,p)\} & & & \\
R^+_2 & = & \{(i', (i+1)') \mid 0 \leq i < p\} \cup \{(p',p')\} & & & \\
R^-_2 & = & \{((i+1)', i') \mid 0 \leq i < p\} \cup \{(p',p')\} & & & \\
L_1 & = & \{(rt, i) \mid 0 \leq i \leq p\} & L^-_1 & = & \{(i, rt) \mid 0 \leq i \leq p\} \\
L_2 & = & \{(rt, i') \mid 0 \leq i \leq p\} & L^-_2 & = & \{(i', rt) \mid 0 \leq i \leq p\}
\end{array}$$

See Figure 3 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 $n < p-1$. In addition, it is easy to verify that H is well-defined.

We now show that H is indeed a model of $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

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 $H \models \Phi_N$. Note here 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 $H \models \Phi_M$. It is straightforward to verify the following simple facts.

- *Fact 4:* If $C(\psi) \Rightarrow_{M_L} (i, m, n)$, then $m < p-1$ and $n < p-1$.
- *Fact 5:* If $(i, m, n) \rightarrow^{I_i}_{M_L} (j, m_1, n_1)$, then $m_1 \leq m+1$ and $n_1 \leq n+1$. As a result of Fact 4, $m_1 < p$ and $n_1 < p$.

Consequently, Facts 1, 2 and 3 for showing $G \models \Phi_M$ in the proof of Proposition 3.2 are also true here. Therefore, the argument for showing $G \models \Phi_M$ in the proof of Proposition 3.2, together with Facts 4 and 5 above, proves $H \models \Phi_M$.

Hence H is indeed a finite model of $\Phi_N \wedge \Phi_M \wedge \varphi_{C(\psi)} \wedge \neg\varphi_{(1,0,0)}$.

This completes the proof of Proposition 3.3. ■

Corollary 3.4: The function f defined above is a reduction from FO to $S(P_+)$. ■

Proof: By the definition of M_L, for each $\psi \in FO$, ψ is satisfiable iff $\psi \notin H_{M_L,1}$. As shown in the proof of Proposition 3.3, $\psi \notin H_{M_L,1}$ iff $f(\psi)$ is satisfiable. Therefore, f is a reduction from FO to $S(P_+)$. ■

As an immediate result of Proposition 3.3 and Lemma 3.1, we have the following corollary.

Corollary 3.5: The set $S(P_+)$ is a conservative reduction class. ■

From Corollary 3.5, Theorem 2.2 follows immediately.

4 The Implication Problems for P_f

In this section, we establish Theorem 2.3. As in the proof of Theorem 2.2, we show that the set

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

is a *conservative reduction class*. To do this, we first present an encoding of two-register machines by sentences in P_f, and then prove a reduction property of the encoding. Using this reduction property, we define a semi-conservative reduction from FO to $S(P_f)$.

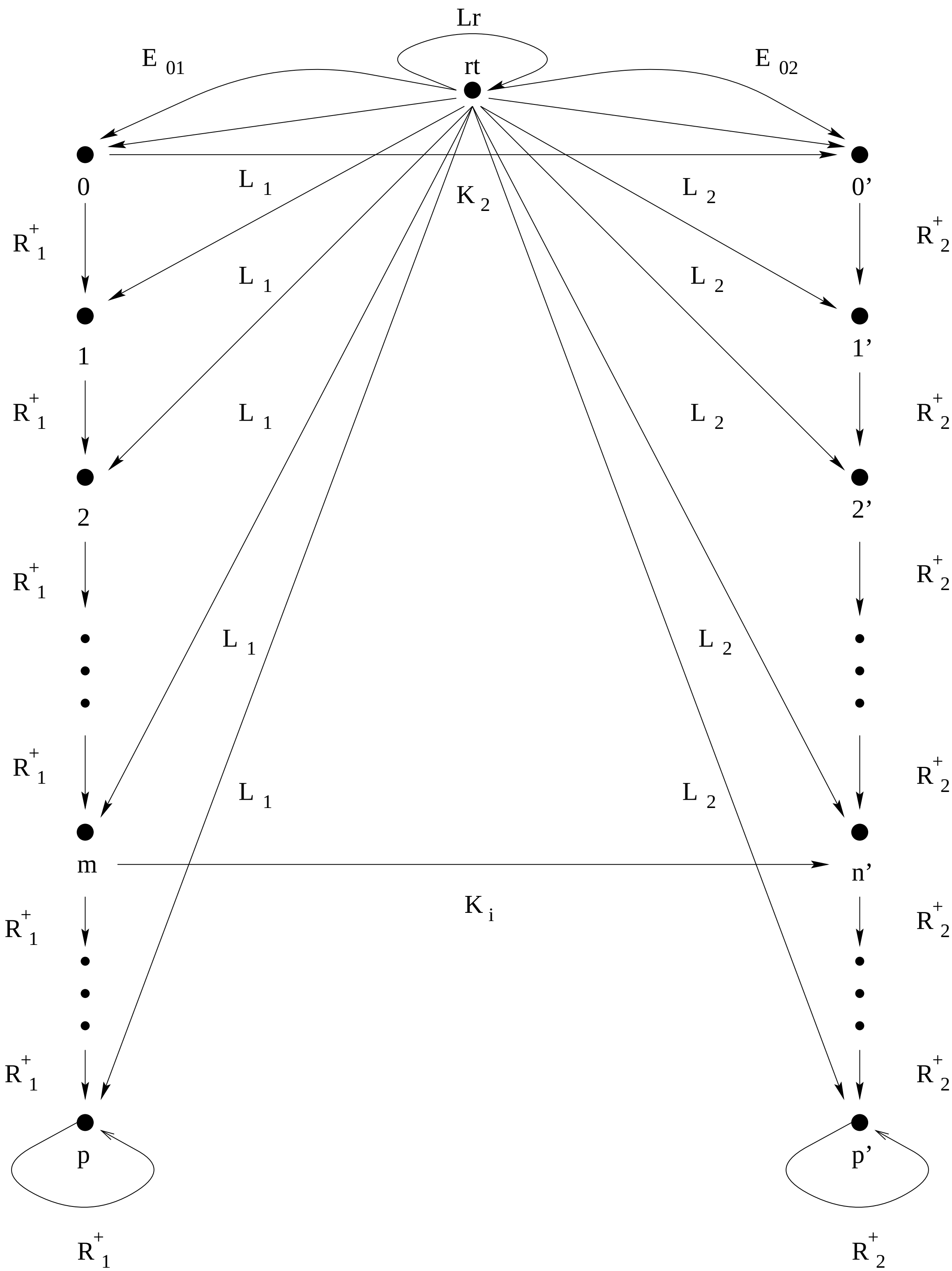

Figure 3: The structure H in Proposition 3.3

4.1 Encoding

Let M be a two-register machine, as described in Section 3.2.1. Assume that the set E of binary relations in the 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.

Registers

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

- Successor, predecessor:
  - $\phi_1 = \forall xy(L_1(r,x) \wedge \exists z(R_1^+(x,z) \wedge R_1^-(z,y)) \to \epsilon(x,y))$
    ($pf(\phi_1) = L_1$, $lt(\phi_1) = R_1^+ \cdot R_1^-$ and $rt(\phi_1) = \epsilon$.)
  - $\phi_2 = \forall xy(L_1(r,x) \wedge \exists z(R_1^-(x,z) \wedge R_1^+(z,y)) \to \epsilon(x,y))$
  - $\phi_3 = \forall xy(L_2(r,x) \wedge \exists z(R_2^+(x,z) \wedge R_2^-(z,y)) \to \epsilon(x,y))$
  - $\phi_4 = \forall xy(L_2(r,x) \wedge \exists z(R_2^-(x,z) \wedge R_2^+(z,y)) \to \epsilon(x,y))$
  - $\phi_5 = \forall xy(L_1(r,x) \wedge \epsilon(x,y) \to \exists z(R_1^+(x,z) \wedge R_1^-(z,y)))$
    ($pf(\phi_5) = L_1$, $lt(\phi_5) = \epsilon$ and $rt(\phi_5) = R_1^+ \cdot R_1^-$.)
  - $\phi_6 = \forall xy(L_2(r,x) \wedge \epsilon(x,y) \to \exists z(R_2^+(x,z) \wedge R_2^-(z,y)))$
- Register identification: ϕ_7, ϕ_8, ϕ_9 and ϕ_{10} are the same as given in Section 3.2.1.
- Zeros:
  - $\phi_{11} = \forall x(\exists y(L_1(r,y) \wedge E_{01}^-(y,x)) \to \epsilon(r,x))$
    (ϕ_{11} is a simple path constraint with $lt(\phi_{11}) = L_1 \cdot E_{01}^-$ and $rt(\phi_{11}) = \epsilon$.)
  - $\phi_{12} = \forall xy(L_1(r,x) \wedge E_{01}^-(x,y) \to \exists z(E_{01}^-(x,z) \wedge \exists z'(E_{01}(z,z') \wedge E_{01}^-(z',y))))$
    ($pf(\phi_{12}) = L_1$, $lt(\phi_{12}) = E_{01}^-$ and $rt(\phi_{12}) = E_{01}^- \cdot E_{01} \cdot E_{01}^-$.)
  - $\phi_{13} = \forall xy(L_1(r,x) \wedge \exists z(E_{01}^-(x,z) \wedge E_{01}(z,y)) \to \epsilon(x,y))$
    ($pf(\phi_{13}) = L_1$, $lt(\phi_{13}) = E_{01}^- \cdot E_{01}$ and $rt(\phi_{13}) = \epsilon$.)
  - $\phi_{14} = \forall x(\exists y(L_1(r,y) \wedge \exists z(E_{01}^-(y,z) \wedge E_{02}(z,x))) \to E_{02}(r,x))$
    (ϕ_{14} is a simple path constraint with $lt(\phi_{14}) = L_1 \cdot E_{01}^- \cdot E_{02}$ and $rt(\phi_{14}) = E_{02}$.)
  - $\phi_{15} = \forall xy(E_{02}(r,x) \wedge \epsilon(x,y) \to \exists z(E_{02}^-(x,z) \wedge E_{02}(z,y)))$
    ($pf(\phi_{15}) = E_{02}$, $lt(\phi_{15}) = \epsilon$ and $rt(\phi_{15}) = E_{02}^- \cdot E_{02}$.)
  - $\phi_{16} = \forall x(E_{02}(r,x) \to L_2(r,x))$

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.

Instructions

The encoding of each instruction I_i, ϕ_{I_i}, is the same as the one given in Section 3.2.1, except

$$\phi^i_{s_{2,0}} = \forall xy(L_1(r,x) \wedge \exists z(K_i(x,z) \wedge E_{02}^- \cdot E_{02}(z,y)) \to K_j(x,y)).$$

Here $pf(\phi^i_{s_{2,0}}) = L_1$, $lt(\phi^i_{s_{2,0}}) = K_i \cdot E_{02}^- \cdot E_{02}$, and $rt(\phi^i_{s_{2,0}}) = K_j$.

The encoding of the program of M is $\Phi^f_M = \bigwedge_{i=0}^{l} \phi_{I_i}$.

It is clear that Φ^f_M is a conjunction of path constraints in P_f.

4.2 Reduction property

Analogous to Proposition 3.2, we establish the reduction property of the encoding above as follows.

Proposition 4.1: Let M be a two-register machine. For all IDs C and D of M, we have that

$$C \Rightarrow_M D \quad \text{iff} \quad \Phi^f_N \wedge \Phi^f_M \wedge \varphi_C \to \varphi_D \text{ is valid.}$$

■

Proof:

(1) Assume that $C \Rightarrow_M D$. As in the proof of Proposition 3.2, we prove by induction on step t that for each ID E of M and each model G of $\Phi^f_N \wedge \Phi^f_M \wedge \varphi_C$, if $C \Rightarrow^t_M E$ then $G \models \varphi_E$. This can be shown in basically the same way as for Proposition 3.2, except for the following cases in the inductive step.

Case 3: $I_i = (i, register_1, j, k)$ and $m = 0$. In this case, E must be $(j, 0, n)$.

Suppose, for *reductio*, that there exist $a, b \in |G|$, such that

$$G \models L_1(r, a) \wedge E^-_{01} \cdot E_{02} \cdot (R^+_2)^n(a, b) \wedge \neg K_j(a, b).$$

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^f_N, 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^f_N, we have $G \models \epsilon(a, d)$. Hence

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

By $G \models L_1(r, a) \wedge E^-_{01}(a, c)$ and ϕ_{11} in Φ^f_N, 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^f_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, E must be (k, p, n).

Suppose, for *reductio*, 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^f_N, there exists $c \in |G|$, such that

$$G \models L_1(r, a) \wedge R^+_1(a, c) \wedge R^-_1(c, a).$$

By ϕ_7 in Φ^f_N, we have $G \models L_1(r, c) \wedge R^-_1(c, a)$. 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 $G \models K_i(c,b)$. Hence

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

Thus by $\phi^i_{s_{1,n}}$ in Φ^f_M, 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, E must be $(j, m, 0)$. Suppose, for *reductio*, 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^f_N, we have $G \models L_1(r,d)$. Thus by ϕ_{14} in Φ^f_N, we have

$$G \models E_{02}(r,b).$$

By ϕ_{15} in Φ^f_N, there exists $e \in |G|$, 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 $\phi^i_{s_{2,0}}$ in Φ^f_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, E must be (k, m, p). Suppose, for *reductio*, 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^f_N, we have $G \models L_1(r,c)$. By ϕ_{14} in Φ^f_N, $G \models E_{02}(r,d)$. By ϕ_{16} in Φ^f_N, $G \models L_2(r,d)$. By ϕ_9 in Φ^f_N, $G \models L_2(r,b)$. Therefore, by ϕ_6 in Φ^f_N, there exists $e \in |G|$, such that

$$G \models R_2^+(b,e) \wedge R_2^-(e,b).$$

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 $G \models K_i(a,e)$. Hence

$$G \models L_1(r,a) \wedge K_i(a,e) \wedge R_2^-(e,b).$$

Thus by $\phi^i_{s_{2,n}}$ in Φ^f_M, we have $G \models K_k(a,b)$. This contradicts the assumption.

(2) Conversely, assume that $C \not\Rightarrow_M D$. It is easy to verify that the structure G (without L_r and K_i^- edges) constructed in the proof of Proposition 3.2 is a model of $\Phi^f_N \wedge \Phi^f_M \wedge \varphi_C \wedge \neg\varphi_D$. ∎

4.3 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 M_L with an appropriate encoding $m(\psi)$ of ψ, as described in Section 3.1.1.

Proposition 4.2 below shows that the function g is indeed a semi-conservative reduction from FO to $S(P_f)$.

Proposition 4.2: Let M_L be the two-register machine described in Section 3.1.1. For each first-order sentence ψ,

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 the same as the proof of Proposition 3.3, except that here in the structure H shown in Figure 3, there are no L_r and K_i^- edges. ■

Corollary 4.3: The function g defined above is a reduction from FO to $S(P_f)$. ■

Corollary 4.4: The set $S(P_f)$ is a conservative reduction class. ■

From Corollary 4.4, Theorem 2.3 follows immediately.

5 Conclusions

We have presented a class of path constraints, P. These constraints are important in both structured and semistructured data for specifying natural integrity constraints. They are not only a fundamental part of the semantics of the data; they are also important in query optimization. For example, the familiar inverse constraints that occur in object-oriented databases can be stated as path constraints of P.

For semistructured data, we have shown that, despite the simple syntax of the language P, its associated implication problem is r.e. complete and its finite implication problem is co-r.e. complete. Indeed, we have established these undecidability results for two fragments of P. One of the fragments is the largest subset of P without equality. The other is the set of path constraints of the forward form in P.

These undecidability results motivate our search for decidable fragments of P. In [12], we establish the decidability of the implication problems for several fragments of P, which retain sufficient expressive power to capture important semantic information such as inverse constraints and local database constraints commonly found in object-oriented databases.

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

References

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

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

[3] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Siméon. "Querying documents in object databases". *Journal of Digital Libraries*, 1(1), 1997.

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

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

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

[7] J. Barwise. "On Moschovakis closure ordinals". *Journal of Symbolic Logic*, 42:292-296, 1977.

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

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

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

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

[12] P. Buneman, W. Fan, and S. Weinstein. "The decidability of some restricted implication problems for path constraints". Technical Report MS-CIS-97-15, Department of Computer and Information Science, University of Pennsylvania, 1997.

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

[14] N. Coburn and G. E. Weddell. "Path constraints for graph-based data model: Towards s unified theory of typing constraints, equations and functional dependencies". In *Proc. 2nd International Conf. on Deductive and Object-Oriented Databases*, pp. 312-331, 1991.

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

[16] M. Fernandez, D. Florescu, A. Levy, and D. Suciu. "A query language and processor for a web-site management system". In *Workshop on Management of Semistructured Data*, 1997.

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

[18] M. Ito and G. E. Weddell. "Implication problems for functional constraints on databases supporting complex objects". *Journal of Computer and System Science*, 50: 165-187, 1995.

[19] M. Ito, G. E. Weddell, and N. Coburn. "On specialization constraints over complex objects". Technical Report CS-91-62, Department of Computer Science, University of Waterloo, 1991.

[20] M. Kifer, W. Kim, and Y. Sagiv. "Querying object-oriented databases". In *Proc. ACM SIGMOD International Conf. on Management of Data*, pp. 393-402, 1992.

[21] C. Lamb, G. Landis, J. Orenstein, and D. Weinreb. "The ObjectStore Database system". *Comm. ACM*, 34(10): 51-63, October 1991.

[22] A. O. Mendelzon, G. A. Mihaila, and T. Milo. "Querying the World Wide Web". In *Proc. PDIS*, pp. 80-91, 1996.

[23] J. Mylopoulos, P. Bernstein, and H. Wong. "A language facility for designing database-intensive applications". *ACM Trans. on Database Sys.*, 5(2), June 1980.

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

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

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

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

[28] G. E. Weddell. "Reasoning about functional dependencies generalized for semantic data models." *ACM Trans. on Database Sys.*, 17(1): 32-64, March 1992.

[29] G. E. Weddell and N. Coburn. "A theory of specialization constraints for complex objects." In *Proc. of 3rd International Conf. on Database Theory*, pp. 229-244, December 1990.

[30] C. Zaniolo. "The database language GEM". In *Proc. ACM SIGMOD International Conf. on Management of Data*, pp. 423-434, 1983.