

Models of Path Constraints [★]

Denis Debarbieux, Yves Roos, and Sophie Tison

{debarbie, yroos, tison}@lifl.fr

Laboratoire d'Informatique Fondamentale de Lille,
U.M.R. C.N.R.S. 8022

Universit de Lille 1, 59655 Villeneuve d'Ascq Cedex. France.

Abstract. We model semistructured data as rooted edge-labeled directed graphs. A path inclusion constraint $p \preceq q$ is satisfied by a semistructured data if any nodes reached by the regular query p is also reached by the regular query q . In this paper we define the notion of an exact model: an exact model of a set of path constraints \mathcal{C} satisfies the constraint $p \preceq q$ if and only if it is implied by \mathcal{C} . We characterize the sets \mathcal{C} of path inclusion constraints that have an exact model. After that we give an algorithm which computes this model. We finish by studying a particular case: word equality constraints.

1 Introduction

The development of the World Wide Web has led to the birth of semistructured data models with languages adapted to these models. Semistructured data are usually viewed as rooted edge-labeled directed graphs: indeed, we can model HTML pages as a graph (a page is a node, an hyper-link is an edge) or we can model XML documents as graphs. A presentation of this model and an overview of works done in this context can be found in [1].

Let us consider the datum figure 1 which represents a journal. This journal contains articles and each article is written by one or two authors. Sometimes we need to co-author two authors.

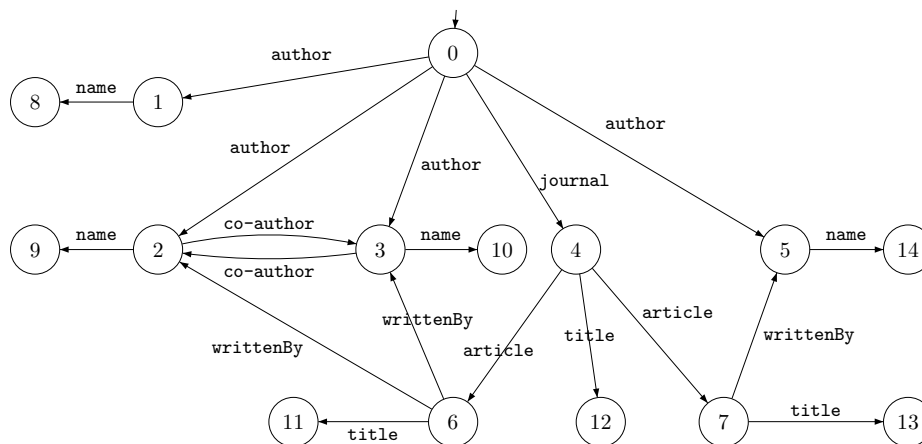


Fig. 1. Example of a semistructured datum

In this example, the graph is said to be non-deterministic as some nodes have several outgoing edges with the same label (for example, the root has several “author” edges).

Basic query mechanisms proposed for semistructured data and querying the web are based on path expressions (see for example Lorel [2], UnSQL [9]). In particular, a regular path

[★] This research was partially supported by Inria (MOSTRARE team).

expression or regular query is a regular expression on the alphabet of labels appearing in the data. The result of the regular query q is the set of nodes reached from the root by a path labeled by a word of q .

For example, `author`, `author.name`, `journal.article.title` are paths of the datum D (figure 1). The regular expression `author.co-author*` is a regular query whose result on D is $\{1, 2, 3, 5\}$.

Path expressions can also be used to express structural informations about the data. E.g., one could want to express that every coauthor of an author is also an author. This leads to path constraints which give restrictions on the data paths. Certain kinds of integrity constraints found in object-oriented databases and also common in semistructured databases can be expressed with path constraints. These constraints have been introduced by Abiteboul and Vianu in [3] and are useful in query optimization. Different classes of path constraints, (see for instance [10], [15] or [5]) have been studied. Here, we study path inclusion constraints. A path inclusion constraint is written $p \preceq q$ where p and q are regular path queries: a datum models (or satisfies) $p \preceq q$ if the set of nodes result of p is included in the set of nodes result of q . Continuing the example, since the result of `author.co-author+` is $\{2, 3\}$ and the result of `journal.article.writtenBy` is $\{2, 3, 5\}$ the path inclusion `author.co-author+ \preceq journal.article.writtenBy` is satisfied. If we denote by $p \equiv_C q$ the conjunction $p \preceq q \wedge q \preceq p$, the datum D satisfies `author.co-author \equiv_C author.co-author.co-author`. In this example, the constraint is of the form $u \equiv_C v$ where u and v are paths. We call *word equality* this kind of constraint. Similarly, the constraint $u \preceq v$ is called *word inclusion*. A set of path inclusions \mathcal{C} implies a path inclusion $p \preceq q$ denoted $\mathcal{C} \models p \preceq q$ if every datum model of \mathcal{C} is also a model of $p \preceq q$. A regular query p has the *boundedness property (strong boundedness property)* w.r.t a set \mathcal{C} of path inclusions if there exists a regular query f such that $\mathcal{C} \models p \preceq f$ ($\mathcal{C} \models p \equiv_C f$) and such that the language represented by f is finite.

In this paper, we are mainly interested in models of a set of constraints. Of course, any set of path inclusions has a model (e.g. the complete model reduced to the root which models any path inclusion); here we focus on the notion of exact model: a model of \mathcal{C} is said exact if it models only constraints satisfied conjointly by every model of \mathcal{C} . In other words, a datum D_C is an exact model of \mathcal{C} if D_C is a model of $p \preceq q$ if and only if \mathcal{C} implies $p \preceq q$.

According to the form of the constraints, existence of exact models is ensured or not, as we shall see. When existence of exact models is ensured, a natural question arises: is there a finite exact model? Having effectively a finite exact model of the set of constraints can be used for several purposes: of course, it provides an effective mean of checking whether a (regular) path inclusion is implied by the set of constraints. It ensures also that every query is strongly bounded. Lastly, as we can extract from a finite model, a finite set of word constraints which implies \mathcal{C} , it allows us to give a finite presentation of the primary set of constraints.

The first case we consider is the case of bounded path inclusion ($p \preceq u$ where p is a regular query and u is a non-empty word). In this case, we propose a decidable characterization of sets \mathcal{C} which have a finite exact model. Moreover, we give an effective way of computing such a model when it exists.

Secondly we consider only word equalities ($u \equiv_C v$ where u and v are words). In this case, we give a more efficient algorithm for deciding existence of a finite exact model and for constructing such a model.

The next section contains formal definitions related to path inclusions and to exact models. In section 3, we exhibit examples of set of constraints and the associate exact model. In section 4, we characterize the sets \mathcal{C} of bounded path constraints who have an exact model (4.1). Then we give an algorithm which computes the exact model from a set of bounded path inclusion (4.2). Then we particularize this algorithm to the case of word equality (4.3). We conclude this paper in section 5.

2 Preliminaries

In the sequel we use the following notions which were introduced in [3]. Let A be a fixed finite alphabet of labels.

Definition 1.

- A semistructured datum is a triple $D = \langle N, \text{root}, T \rangle$ where N is a set of nodes, $\text{root} \in N$ is called the root of datum D and $T \subseteq N \times A \times N$ is the set of transitions.
- If N is finite, the datum is said finite.
- If, for all n in N , for all a in A , there is at most one transition (n, a, n') in T , then the datum is said to be deterministic.
- If, for all n in N , for all a in A , there is at least one transition (n, a, n') in T , then the datum is said to be complete.
- If, for all n in N there are a finite number of transitions (n, a, n') in T , the datum is said to be finitely branching.

We are interested in the set of nodes which are reached by some paths in a datum. Then we can define the notions of query and result of query.

Definition 2.

- Given D a datum, $u \in A^*$, let $\text{acc}_D(u)$ be defined by :
 1. if $u = \epsilon$, then $\text{acc}_D(u) = \{\text{root}\}$
 2. if $u = u'a$ ($u' \in A^*$, $a \in A$) $\text{acc}_D(u) = \{n \in N \mid \exists n' \in \text{acc}_D(u'), (n', a, n) \in T\}$
- A regular query p is a regular expression over A . The result of a query p over a datum D is the set $\text{acc}_D(p) = \cup_{u \in L(p)} \text{acc}_D(u)$ where $L(p)$ denotes the regular language described by p .

Now, we formally define path inclusions, path equalities, and notions related to implication.

Definition 3.

- A path inclusion is an expression of the form $p \preceq q$ where p, q are regular queries.
- A path equality $p \equiv_C q$ represents the conjunction $(p \preceq q) \wedge (q \preceq p)$.
- A bounded path constraint is an expression of the form $p \preceq u$ where p is a regular query and u is a non-empty word.
- If u and v are paths, $u \preceq v$ is called word inclusion, and $u \equiv_C v$ is called word equality.
- A datum D satisfies (is a model of) a path inclusion $p \preceq q$, denoted $D \models p \preceq q$, if the set of nodes $\text{acc}_D(p)$ is included in $\text{acc}_D(q)$. D satisfies a set \mathcal{C} of path inclusions, denoted $D \models \mathcal{C}$, if D satisfies each path inclusion of \mathcal{C} .
- A set \mathcal{C} of path inclusions implies a path inclusion $p \preceq q$, denoted $\mathcal{C} \models p \preceq q$, if for each datum D such that $D \models \mathcal{C}$, $D \models p \preceq q$.
- A datum D_C is an exact model of a set of constraints \mathcal{C} if D_C is a model of $p \preceq q$ if and only if \mathcal{C} implies $p \preceq q$.
- Given a set \mathcal{C} of path inclusions, and two regular queries p, q , the implication problem for \mathcal{C}, p, q is to decide whether $\mathcal{C} \models p \preceq q$.

In [3], the authors prove that in the context of the implication problem we can restrict to finite models, since implication and finite implication of path inclusions are equivalent. It is stated in the following proposition.

Proposition 1. A set \mathcal{C} of path inclusions implies a path inclusion $p \preceq q$, denoted $\mathcal{C} \models p \preceq q$, if for each **finite** datum D such that $D \models \mathcal{C}$, $D \models p \preceq q$.

The implication of path inclusions is closed by right congruence, transitivity and reflexivity : for all p, q, r regular queries and \mathcal{C} set of path inclusions.

- $\mathcal{C} \models p \preceq q \Rightarrow \mathcal{C} \models pr \preceq qr$
- $\mathcal{C} \models p \preceq p$
- $(\mathcal{C} \models p \preceq q) \wedge (\mathcal{C} \models q \preceq r) \Rightarrow \mathcal{C} \models p \preceq r$

Related to implication, another interesting problem, from a query optimization point of view, is the decidability of the boundedness property.

Definition 4. A regular query p has the boundedness property (strong boundedness property) w.r.t a set \mathcal{C} of path inclusions if there exists a regular query f such that $\mathcal{C} \models p \preceq f$ ($\mathcal{C} \models p \equiv_C f$) and $L(f)$ is finite.

E.g., let \mathcal{C} be $\{a^2 \preceq a\}$ w.r.t. \mathcal{C} , the query a^* is bounded (by a), whereas the query ba^* is not.

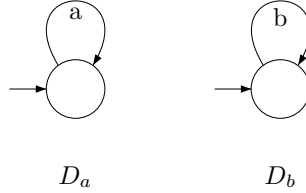
Definition 5. $|p|$ is the length of the regular query p , the size of $p \preceq q$, denoted by $|p \preceq q|$, is the sum $|p| + |q|$, and the size of the set of constraints \mathcal{C} , denoted by $|\mathcal{C}|$, is the sum of the sizes of its constraints.

3 Exact model

In this section, we will exhibit examples of sets of constraints corresponding to the four following situations :

1. There does not exist an exact model
2. There is an exact model but any exact model is not finitely branching
3. There is a finitely branching exact model but there is no exact finite model
4. There is an exact finite model

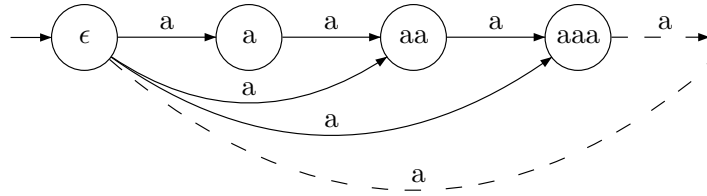
Example 1. This example of set of constraints which does not have a finite model uses the fact that we only consider rooted graph. This give to the empty word a particular property : for any datum $D = \langle N, r, T \rangle$, we know that $\text{acc}_D(\epsilon)$ is always the singleton $\{\epsilon\}$. It follows that if $D \models \epsilon \preceq v$ then for any prefix v' of v , $\text{acc}_D(v')$ is not empty. On the other hand, constraints $u \preceq \epsilon$ present us with a difficult problem: if $D \models u \preceq \epsilon$ then $\text{acc}_D(u)$ is empty else $\text{acc}_D(u)$ is equal to $\{\epsilon\}$: Let us consider the alphabet $A = \{a, b\}$ and the set of constraints $\mathcal{C} = \{a \preceq \epsilon, b \preceq \epsilon\}$. First let us consider D_a and D_b shown bellow. These two data satisfy \mathcal{C} but we will use them as a counterexample. Let D_c be an exact model of \mathcal{C} . As $\mathcal{C} \models a \preceq \epsilon$ and $\mathcal{C} \models b \preceq \epsilon$ they are only four values for the couple $(\text{acc}_{D_c}(a), \text{acc}_{D_c}(b))$:



$\text{acc}_{D_c}(a)$	$\text{acc}_{D_c}(b)$	consequence
\emptyset	\emptyset	$D_c \models a \equiv_{\mathcal{C}} b$ but $\mathcal{C} \not\models a \equiv_{\mathcal{C}} b$ (see D_a)
\emptyset	$\{\text{root}\}$	$D_c \models b \equiv_{\mathcal{C}} \epsilon$ but $\mathcal{C} \not\models b \equiv_{\mathcal{C}} \epsilon$ (see D_a)
$\{\text{root}\}$	\emptyset	$D_c \models a \equiv_{\mathcal{C}} \epsilon$ but $\mathcal{C} \not\models a \equiv_{\mathcal{C}} \epsilon$ (see D_b)
$\{\text{root}\}$	$\{\text{root}\}$	$D_c \models a \equiv_{\mathcal{C}} b \equiv_{\mathcal{C}} \epsilon$ but $\mathcal{C} \not\models a \equiv_{\mathcal{C}} b \equiv_{\mathcal{C}} \epsilon$ (see D_b)

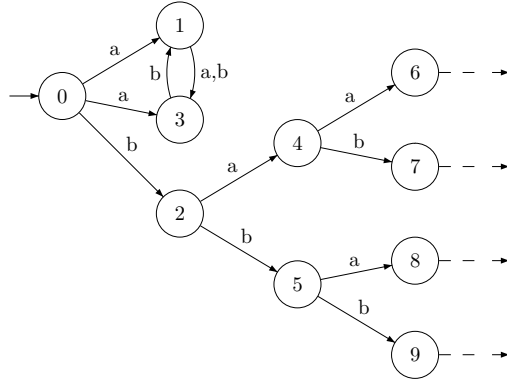
So it is impossible to find an exact model for \mathcal{C} .

Example 2. A finitely branching exact model is interesting since, in this case, $\text{acc}_{D_c}(u)$ is always finite. It follows from this remark that q has the strong boundedness property w.r.t. \mathcal{C} if and only if $\text{acc}_{D_c}(q)$ is finite. Unfortunately some sets of constraints, although having an exact model, do not have any finitely branching exact model : Let us consider the set $\mathcal{C} = \{a^2 \preceq a\}$ and D_c an exact model of \mathcal{C} . Let k and k' be two different integers such that $k' > k > 1$, then we must have $D_c \models a^{k'} \preceq a^k$ but we cannot have $D_c \models a^k \preceq a^{k'}$ since D_c is exact. It follows that $\text{acc}_{D_c}(a^k)$ is different from $\text{acc}_{D_c}(a^{k'})$ and D_c has an infinite number of nodes. Moreover, it is clear that, for any model of \mathcal{C} , every node except **root**, must be in $\text{acc}_D(a)$. Since D_c has an infinite number of nodes, it is not finitely branching and it contains the following subgraph :



Remark that for any exact model D_c $\text{acc}_{D_c}(a^*)$ is an infinite set nevertheless a^* is bounded by $a + \epsilon$. Moreover any query has the strong boundedness property w.r.t \mathcal{C} (an ϵ -free query q is equivalent to the shortest word which belongs to $L(q)$).

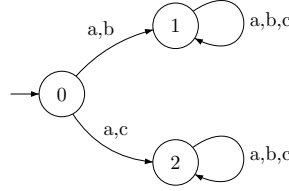
Example 3. Some sets of constraints have a finitely branching exact model but no exact finite model. It is the case of the set $\mathcal{C} = \emptyset$ whose exact model is the infinite prefix tree. The following example presents another set \mathcal{C} which has a finitely branching exact model but no exact finite model. Let us consider the set $\mathcal{C} = \{a \equiv_{\mathcal{C}} ab, aaba \equiv_{\mathcal{C}} aa, aabb \equiv_{\mathcal{C}} aa, aab \preceq a, aa \preceq a\}$. An exact model D_c is given by the following figure :



With an exact model the implication problem $\mathcal{C} \models p \preceq q$ is reduced to the problem of the inclusion between $\text{acc}_{D_{\mathcal{C}}}(p)$ and $\text{acc}_{D_{\mathcal{C}}}(q)$. As $\text{acc}_{D_{\mathcal{C}}}(abb) = \{1, 3\}$ and $\text{acc}_{D_{\mathcal{C}}}(aa) = \{3\}$, $D_{\mathcal{C}} \models aa \preceq abb$ and then $\mathcal{C} \models aa \preceq abb$. It is the same for the boundedness property : as $\text{acc}_{D_{\mathcal{C}}}(ab^*)$ is finite then ab^* has the boundedness property (bounded by aa) and then ab^* has the strong boundedness property ($\mathcal{C} \models ab^* \equiv_c a$). Whereas $\text{acc}_{D_{\mathcal{C}}}(b^*)$ is not finite and, since $D_{\mathcal{C}}$ is finitely branching, b^* has neither the boundedness property nor the strong boundedness property.

Example 4. Finally we show two examples of set of constraints which have an exact finite model.

If we consider a set \mathcal{C} which implies $\mathcal{C} \models u \equiv_c v$ for any word u and v , the complete model reduced to the root is an exact finite model. The next example will show another set of constraints which have an exact finite model. Let us consider the alphabet $\{a, b, c\}$ and $\mathcal{C} = \{a \preceq b+c, b \preceq a, c \preceq a\} \cup \{x \equiv_c xy \mid (x, y) \in A \times A\}$. The datum $D_{\mathcal{C}}$ below is an exact finite model of \mathcal{C}



Clearly, $D_{\mathcal{C}}$ is a model of \mathcal{C} . Moreover,

$$\mathcal{C} \models p \equiv_c \begin{array}{l} + \\ u \in L(p) \\ |u| \leq 1 \end{array}$$

Then it is easy to see that, if $D_{\mathcal{C}} \models p \preceq q$ for some regular expressions p and q then

$$\mathcal{C} \models \begin{array}{l} + \\ u \in L(p) \\ |u| \leq 1 \end{array} \preceq \begin{array}{l} + \\ u \in L(q) \\ |u| \leq 1 \end{array}$$

and $D_{\mathcal{C}}$ is exact.

4 Exact finite model

From now on, we will only consider the case of a finite set of inclusions of the form $p \preceq u$ where p is a regular path expression and u is a non-empty word: we call such path inclusions *bounded path inclusions*. This kind of constraints have been introduced in [7]. In [6], we have proved that any set of bounded path constraints has an exact model, but this model is always infinite and generally it is not a finitely branching datum. Nevertheless we can use the following lemma :

Lemma 1 ([6]). *Let \mathcal{C} be a set of bounded path inclusion, u be a word and q be a regular query. If $\mathcal{C} \models u \preceq p$ then there is a word u_p in $L(p)$ s.t. $\mathcal{C} \models u \preceq u_p$.*

Remark 1. This lemma does not hold in the case of general constraints : the exact model defined in example 4 satisfies $a^+ \preceq (b + c)$ but satisfies neither $a^n \preceq b$ nor $a^n \preceq c$ for all $n \geq 1$.

In order to decide if a set of constraint \mathcal{C} has an exact finite model, we give in 4.1 a characterization of such \mathcal{C} . And then, in 4.2, we give an algorithm which decides if \mathcal{C} has a finite exact model. This algorithm is able to compute a finite exact model. To do this we define and use a prefix rewriting system which simulates the path inclusion. We consider a particular case in 4.3 : in this section we consider word equality constraints (i.e. $\mathcal{C} = \{u_i \equiv_c v_i\}$ where u_i 's and v_i 's are word).

4.1 Characterization

In order to characterize the sets of constraints which have an exact finite model, we introduce the following equivalence relation :

Definition 6. *Let \mathcal{C} be a set of path inclusion. We will denote \equiv_c the relation on $A^* \times A^*$ defined by $u \equiv_c v$ if $\mathcal{C} \models u \equiv_c v$.*

Clearly \equiv_c is an equivalence relation. We will denote by $[u]_c$ the equivalence classes of the path u for the relation \equiv_c and we define the datum $D(\mathcal{C}) = \langle N, r, T \rangle$ by :

- $N = \{[u]_c \mid u \in A^*\}$
- $r = \{[\epsilon]_c\}$
- $T = \{([u]_c, x, [v]_c) \mid \mathcal{C} \models v \preceq ux\}$

The following lemma, which characterizes $\text{acc}_{D(\mathcal{C})}(u)$ for any word u , can easily be proven by induction on the length of u :

Lemma 2. $\forall u \in A^* \text{ acc}_{D(\mathcal{C})}(u) = \{[v]_c \mid \mathcal{C} \models v \preceq u\}$

We are now able to prove :

Proposition 2. *For any set \mathcal{C} of bounded path inclusion, $D(\mathcal{C})$ is an exact model.*

Proof. Let us first prove that $D(\mathcal{C})$ is a model of \mathcal{C} : let $(p \preceq u) \in \mathcal{C}$, v be a word in $L(p)$ and $[w]_c \in \text{acc}_{D(\mathcal{C})}(v)$. From lemma 2, it follows that $\mathcal{C} \models w \preceq v$. Since, by transitivity, we get $\mathcal{C} \models w \preceq u$ and using again lemma 2, we obtain $[w]_c \in \text{acc}_{D(\mathcal{C})}(u)$ and $D(\mathcal{C}) \models p \preceq u$. Let us prove now that $D(\mathcal{C})$ is exact. Let us suppose that $D(\mathcal{C}) \models p \preceq q$ for some regular expressions p and q . Let u be a word of $L(p)$. Since $[u]_c$ is in $\text{acc}_{D(\mathcal{C})}(u)$ and $D(\mathcal{C}) \models p \preceq q$, there exists a word v of $L(q)$ such that $[u]_c$ is $\text{acc}_{D(\mathcal{C})}(v)$. It follows from lemma 2 that $\mathcal{C} \models u \preceq v$ and then $\mathcal{C} \models p \preceq q$.

Clearly if \equiv_c is of finite index, then by construction $D(\mathcal{C})$ is finite, and conversely, it is easily seen that if there exists a finite exact model of \mathcal{C} then \equiv_c is of finite index. It follows :

Theorem 1. *Let \mathcal{C} be a set of bounded path inclusions. Then \mathcal{C} has an exact finite model if and only if \equiv_c is of finite index.*

Corollary 1. *A set of bounded path inclusion \mathcal{C} has an exact finite model if and only if $D(\mathcal{C})$ is finite.*

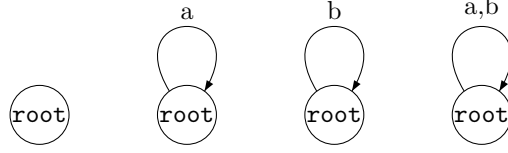
The following example shows that it is important to forbid empty word in right-hand side of constraints, else proposition 2 does not hold.

Example 5. Let us consider the following set of constraints $\mathcal{C} = \{a \preceq \epsilon, b \preceq \epsilon, aba \equiv ab, abb \equiv ab, ab \equiv ba\}$. Since, for any model D of \mathcal{C} , we have $\text{acc}_D(a) = \text{root}$ or $\text{acc}_D(a) = \emptyset$, we have $\mathcal{C} \models a \equiv aa$. For same reasons, we have $\mathcal{C} \models b \equiv bb$. Then it is easy to deduce that \equiv_c is of finite index : more precisely, the equivalence classes of \equiv_c are

- $[\epsilon]_c = \{\epsilon\}$

- $[a]_C = a^+$
- $[b]_C = b^+$
- $[ab]_C = (a+b)^*(ab+ba)(a+b)^*$

Nevertheless, for the same reasons than in example 1, C has no exact model, since it has only the four following models :



Let us consider now the converse problem : is every (finite) datum D an exact model of a (finite) set of bounded path constraints? The answer is *yes* :

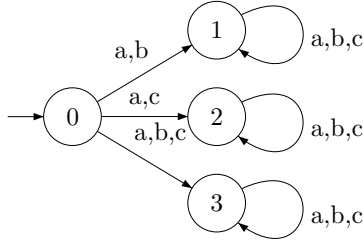
Proposition 3. *For every (finite) datum D , there exists a (finite) set of word constraints $C(D)$ such that $C(D) \models p \preceq q$ if and only if $D \models p \preceq q$*

Proof. Let $D = \langle N, r, T \rangle$ and $S(D) = \{\text{acc}_D(u) \mid u \in A^*\}$. For any set s of $S(D)$, $\text{lex}(s)$ will denote a word of $\{u \mid \text{acc}_D(u) = s\}$. For example we can take the shortest and the first word in alphabetical order. The only restriction is that $\text{lex}(\{r\})$ must be ϵ .

$C(D) = C_{\equiv} \cup C_{\preceq}$ where $C_{\equiv} = \{(\text{lex}(s), x, \text{lex}(s')) \mid s' = \{n' \mid \exists n \in s \wedge (n, x, n') \in T\} \text{ and } \text{lex}(s) = x \cdot \text{lex}(s')\}$ and $C_{\preceq} = \{\text{lex}(q_1) + \dots + \text{lex}(q_n) \preceq \text{lex}(q'_1) + \dots + \text{lex}(q'_k) \mid \bigcup_{1 \leq i \leq n} q_i \subseteq \bigcup_{1 \leq i \leq k} q'_i\}$. Clearly, D is a model of $C(D)$.

We have to prove that D is exact. As $C_{\equiv} \models u \equiv \text{lex}(\text{acc}_D(u))$, $C_{\equiv} \models q \equiv \text{lex}(\text{acc}_D(u))$, if $D \models p \preceq q$ then $C(D) \models p \equiv \text{lex}(\text{acc}_D(u))$ and $C(D) \models q \equiv \text{lex}(\text{acc}_D(u))$. Now, since $\bigcup_{u \in p} (\text{acc}_D(u)) \subseteq \bigcup_{u \in q} (\text{acc}_D(u))$ and $(\text{lex}(\text{acc}_D(u)) \preceq \text{lex}(\text{acc}_D(u))) \in C_{\preceq}$, it follows that $C(D) \models p \preceq q$.

Example 6. Let us consider the following datum with root 0 :



So $S(D) = \{\{0\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$ and $\text{lex}(\{0\}) = \epsilon$, $\text{lex}(\{1, 3\}) = b$, $\text{lex}(\{2, 3\}) = c$ and $\text{lex}(\{1, 2, 3\}) = a$.

$C_{\equiv} = \{a \equiv_c aa \equiv_c ab \equiv_c ac, b \equiv_c ba \equiv_c bb \equiv_c bc, c \equiv_c ca \equiv_c cb \equiv_c cc\}$. $C_{\preceq} = \{b \preceq a, c \preceq a, b+c \preceq a, a \preceq b+c\}$. Let us remark that $C(D) \models b+c \equiv_c a$ but neither $C \models b \equiv_c a$ nor $C \models c \equiv_c a$

Remark that, when D is a finite datum, the proof is effective since $C(D)$ can be built with an algorithm similar to the subset construction used in determinization of finite automata.

4.2 Algorithm

As proved in [14], any set of bounded path inclusions has an exact model. Moreover, in [14] we have simulated by a prefix rewriting system the path inclusions. In this section we will use this technique to check if $\{[u]_C \mid u \in A^*\}$ is finite and to construct, if it is possible, an exact model.

First, we associate with a set C of bounded path inclusions a prefix rewrite system such that u rewrites to v if and only if C models $u \preceq v$.

Definition 7. Let $\mathcal{C} = \{p_1 \preceq u_1, \dots, p_n \preceq u_n\}$ be a finite set of bounded path inclusions over an alphabet A . We consider the relation on paths defined by $u \xrightarrow{\mathcal{C}} v$ if and only if there exists i such that $u \in L(p_i)$ and $v = u_i$. By extension, we denote also $\xrightarrow{\mathcal{C}}$ its right congruent closure. Then $\xrightarrow{\mathcal{C}^*}$ denotes the reflexive, transitive closure of $\xrightarrow{\mathcal{C}}$.

This relation is a prefix rewriting relation as defined in [11] based on an infinite rewrite system. We have the following property:

Proposition 4. Let \mathcal{C} be a set of bounded path inclusions. For any paths u, v , $u \xrightarrow{\mathcal{C}^*} v$ if and only if $\mathcal{C} \models u \preceq v$.

In some sense it means that the word constraints we can deduce from a set of bounded path constraints \mathcal{C} can be deduced from \mathcal{C} using only right congruence, reflexivity, transitivity. Once more, this doesn't work when we have constraints of the form $u \preceq \epsilon$. Indeed in this case, in any model D , $\text{acc}_D(u) = \emptyset$ or $\{\text{root}_D\}$. So it easy to prove that $\mathcal{C} \models u \preceq \epsilon$ and $\mathcal{C} \models v \preceq w$ implies $\mathcal{C} \models uv \preceq uw$: this kind of “ ϵ left congruence” can't be simulated by a prefix rewrite system.

Thanks to property 4, we can use properties of prefix rewrite relations. Indeed, prefix rewrite relations are recognizable relations (in the sense of [12]). The idea behind recognizable relations is simply to encode a n -uple of words on the alphabet A by a word on the alphabet $A \cup \{\perp\} \times \dots \times A \cup \{\perp\}$, where \perp is a new symbol. This encoding is obtained by superposing the n words, aligning the words by the end. E.g. (ab, aaa, b) will be encoded in $[\perp, a, \perp][a, a, \perp], [b, a, b]$. Formally, we define the morphism π_j from $([A \cup \{\perp\}]^n)^*$ by $\pi_j[a_1, a_2, \dots, a_n] = a_j$, if it belongs to A , $\pi_j[a_1, a_2, \dots, a_n] = \epsilon$ if $a_j = \perp$. Now, let Cor be the recognizable language of the words $u_1 \dots u_n$ -on the alphabet $[A \cup \{\perp\}]^n$ - satisfying the two following properties:

- for each j , if $\pi_j(u_k)$ is in A then $\pi_j(u_l)$ is in A for any $l > k$
- there exists some j s.t. $\pi_j(u_1)$ is in A .

Then, the notion of recognizable relation is defined by:

Definition 8. A n -ary relation R on $A^* \times \dots \times A^*$ is recognizable if and only if there exists an automaton \mathcal{M} on the alphabet $A \cup \{\perp\} \times \dots \times A \cup \{\perp\}$ recognizing the language

$$L(\mathcal{M}) = \{u \in \text{Cor} \mid (\pi_1(u), \dots, \pi_n(u)) \in R\}$$

This definition corresponds to the definition of tree recognizable relations defined in [12], if we consider here a word as a tree whose root is associated with the end of the word. Extending slightly previous results[see for instance [12]], we get that prefix rewrite relations where left hand side of rules are recognizable and right hand side are words, are particular cases of recognizable relations and then:

Proposition 5 (see for instance [12]). If \mathcal{C} is a set of inclusion constraints, the prefix rewriting relation $\xrightarrow{\mathcal{C}^*}$ is a recognizable relation.

Rec , the set of recognizable relations, inherits good properties of recognizable languages (see [12] for more details):

Proposition 6.

- Rec is effectively closed under boolean operations (union, intersection, complementation) and under cylindrification and projection.¹
- Finiteness and emptiness are decidable on Rec .

Using this properties, we get the following corollary.

Corollary 2. We can decide whether a recognizable equivalence relation is of finite index.

¹ roughly speaking, cylindrification (projection) consists in adding (deleting) a component in a tuple.

Proof. Let r be a recognizable equivalence relation. We will construct a recognizable relation rep : $rep(u, v)$ means that v is the representative of $[u]_r$. To represent $[u]_r$, we choose the shortest (by the length and then by the alphabetical order) word of $[u]_r$, as *shorter* is obviously a recognizable relation (We could choose an other representative associated with an other recognizable total ordering). Then the relation rep defined by $rep(u, v) = r(u, v) \wedge \forall w r(u, w) \Rightarrow shorter(v, w)$ is a recognizable relation (proposition 6) and we can construct an automaton representing it. Using once more the proposition 6, the monadic relation $\exists u \mid rep(u, v)$ is recognizable and can be effectively associated with an automaton. As r is of finite index iff this relation is finite, and as finiteness is decidable in *Rec*, we can decide whether r is of finite index. Let us note, that in this case the relation is a finite union of Cartesian products of recognizable sets (even a finite union of sets of the form $L \times L$ where L is recognizable) and so is a recognizable subset of $A^* \times A^*$ [8,12].

Now, we get directly:

Proposition 7. *Let \mathcal{C} be a set of bounded path inclusion. The problem of the existence of a finite exact model \mathcal{C} is decidable.*

Proof. The relation $equiv$ defined by $equiv(u, v) = (u \xrightarrow[\mathcal{C}]{*} v) \wedge (v \xrightarrow[\mathcal{C}]{*} u)$ is a recognizable relation (proposition 6) and can be effectively associated with an automaton representing it. It follows from the theorem 1 that we can decide if \mathcal{C} has a finite exact model.

Computing $\{(u, v) \mid rep(u, v)\}$ by using classical automata constructions leads to an exponential algorithm (the complement of $equiv(u, v) \wedge shorter(u, v)$ has to be constructed). This provides also an algorithm to compute an exact model, as we are able to construct an automaton for the set of representatives of $equiv$ and an automaton for $\{(u, v) \mid \mathcal{C} \models u \leq v\}$.

4.3 Word equality constraints

In this section we only consider sets of word equality constraints (i.e. $\mathcal{C} = \{u_i \equiv_c v_i, 1 \leq i \leq n\}$ where u_i 's and v_i 's are words). When \mathcal{C} is a set of word equalities, the exact model defined in proposition 2 is $D(\mathcal{C}) = \langle N, r, T \rangle$ with $N = \{[u]_c \mid u \in A^*\}$, $r = [\epsilon]$ and $T = \{([u]_c, x, [ux]_c) \mid u \in A^*, x \in A\}$. We proved in [6] that $D(\mathcal{C})$ is the unique deterministic exact model of \mathcal{C} .

In order to check if $D(\mathcal{C})$ is finite, we shall define a finite graph which will coincide with $D(\mathcal{C})$ if $D(\mathcal{C})$ is finite. Let W be the set of all prefixes of $\{w \in A^* \mid \exists w' \in A^*, (w \equiv_c w') \in \mathcal{C}\}$. Let $D(\mathcal{C})_W = \langle N_W, r_W, T_W \rangle$ be the datum defined by $N_W = \{[w]_c \mid w \in W\}$, $r_W = [\epsilon]$ and $T_W = \{([u]_c, x, [ux]_c) \mid ux \in W, x \in A\}$. We have proved in [6] :

Proposition 8. *Let \mathcal{C} be a set of word equality constraints then $D(\mathcal{C})$ is finite iff $D(\mathcal{C})_W$ is complete.*

We will now give the complexity of the decision algorithm corresponding with proposition 8. In order to construct $D(\mathcal{C})_W$, we will use the union-find algorithm to build the equivalence classes which are the nodes of the graph $D(\mathcal{C})_W$. The union-find algorithm manages sets of disjoint elements (sets of classes) with three primitives: create a class, compute the union of two classes and find a representative of a class (in [13] the authors give the data structures and the algorithms). The following $find(u)$ will denote a representative of the class of u . We compute the equivalence classes by applying a procedure called **merge**, for all (u, v) in \mathcal{C} :

- If $(u \equiv_c v) \in \mathcal{C}$ then compute the union between the class of u and the class of v
- then compute recursively the union between the class of ux and the class of vx , for each label x .

Note that W contains $|\mathcal{C}|$ elements, where $|\mathcal{C}|$ is the sum of the sizes of the constraints in \mathcal{C} . This would lead to an $O(|A| \times |\mathcal{C}|^2)$ algorithm for **merge** procedure. In order to obtain a complexity in $O(|A|)$, we define an appropriate data structure : this structure is a graph $G = \langle N, r, T_G \rangle$ where $N = \{w \mid w \in W\}$, $r = \epsilon$ and T_G satisfies: $v = find(u)$ iff $((v, x, w) \in T_G$ and $find(w) = find(ux)$). Then, starting from graph G initialized as a graph $\langle N, r, T \rangle$ where $N = \{w \mid w \in W\}$, $r = \epsilon$ and $T_G = \{(w, x, wx) \mid x \in A^*, wx \in W\}$, we can apply the following **merge** procedure :

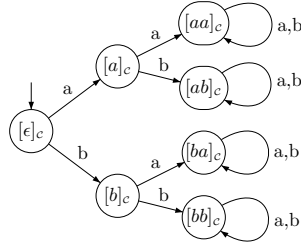
```

procedure merge(in u,v: Path; out S:SetOfClasses; out G:Graph)
% merge the class of u with the class of v .
  r_u = find(u) % compute the reprasantative of u
  r_v = find(v) % compute the reprasantative of v
  union(u,v,S)
  r_∪ = find(u) % r_∪ is the representative of the new class
  for each x ∈ A do
    let r_{ux} be the word s.t. (r_u, x, r_{ux}) ∈ T_G
    let r_{vx} be the word s.t. (r_v, x, r_{vx}) ∈ T_G
    let r_{∪x} be the word s.t. (r_∪, x, r_{∪x}) ∈ T_G
    if r_{ux} and r_{vx} exist and are different then merge(r_{ux}, r_{vx})
    else r_{ux} exists and r_{∪x} does not exist then add (r_∪, x, r_{ux}) to T_G
    else r_{vx} exists and r_{∪x} does not exist then add (r_∪, x, r_{vx}) to T_G
  end if
end

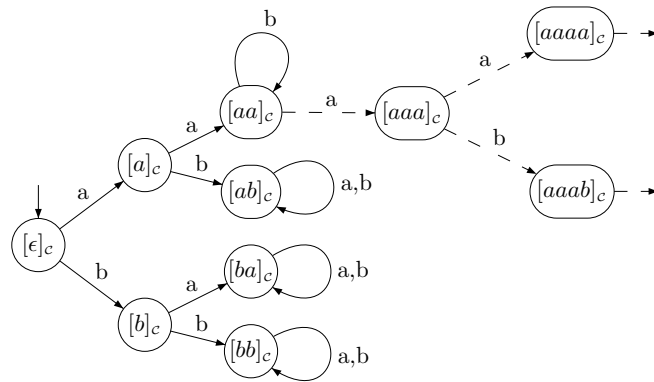
```

We proved that $[u]_c = [v]_c$ if and only if $find(u) = find(v)$ and then we represent $\{[w]_c \mid w \in W\}$ by $S = \{find(w) \mid w \in W\}$ which allows to build graph $D(\mathcal{C})_W$. In [4] and [16], it is proved that an algorithm using $n - 1$ unions and m finds ($m \geq n$) is an $O(n + m \cdot \alpha(n, m))$ algorithm². It follows that our algorithm which computes $D(\mathcal{C})_W$ is better than $O(|A| \times |\mathcal{C}| \times lg^*(|\mathcal{C}|))$.

Example 7. Let us consider the alphabet $A = \{a, b\}$ and the set of word equality constraints $\mathcal{C} = \{xyz \equiv_c xy \mid (x, y, z) \in A \times A \times A\}$. Then the set of all equivalence classes of relation \equiv_c is the set $\{[\epsilon]_c, [a]_c, [b]_c, [aa]_c, [ab]_c, [ba]_c, [bb]_c\}$. Since this set is finite, graphs $D(\mathcal{C})$ and $D(\mathcal{C})_W$ coincide and correspond with the following :



Now, if we consider $\mathcal{C}' = \mathcal{C} \setminus \{aaa \equiv_c aa\}$, then graph $D(\mathcal{C}')_W$, drawn with straight lines below, is not complete anymore, then the equivalence relation $\equiv_{\mathcal{C}'}$ is not of finite index and the only deterministic exact model of \mathcal{C}' is the infinite graph $D(\mathcal{C}')$.



² α is the inverse of the Ackermann function. The Ackermann function A is defined by $A(1, j) = 2^j \ j \geq 1$, $A(i, 1) = A(i - 1, 2)$ if $i \geq 2$, $A(i, j) = A(i - 1, A(i, j - 1))$ if $i, j \geq 2$

5 Conclusion

In this paper we have studied the problem of the existence of an exact model w.r.t. a set of bounded path inclusion constraints C . We have given an effective decidable characterization of sets of bounded path constraints which have a finite exact model. Finally we have studied the particular case of word equality constraints.

It would be interesting, now, to study the problem of the existence of an exact model for general sets of path constraints.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web*. Morgan Kaufmann Publishers, 2000.
2. S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The lorel query language for semistructured data. *Journal of Digital Libraries*, 1(1):68–88, 1997.
3. S. Abiteboul and V. Vianu. Regular path queries with constraints. In *Proc. of ACM Symposium on Principles of Database Systems*, 1997.
4. A. Aho, J. Hopcroft, and J. Ullman. *The design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.
5. N. Alechina, S. Demri, and M. de Rijke. A modal perspective on path constraints. *Journal of Logic and Computation*, to appear, 2004.
6. Y. Andre, A.C. Caron, D. Debarbieux, Y. Roos, and S. Tison. Extraction and implication of path constraints. In *Proceedings of the 29th Symposium on Mathematical Foundations of Computer Science (MFCS'04), Prague (Czech Republic) 2004*, Lecture Notes in Computer Science, Prague, 2004. Springer.
7. Y. Andr, F. Bossut, and A.C. Caron. On decidability of boundedness property for regular path queries. In *proceedings of DLT'99*, Aachen, Germany, 1999. Development in Language Theory, World Scientific Publishing Co.
8. J. Berstel. *Transductions and Context-Free Languages*. Teubner Studienbücher, Stuttgart, 1979.
9. P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization techniques for unstructured data. In *SIGMOD*, pages 505–516, Montreal, 1996.
10. P. Buneman, W. Fan, and S. Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2), 2000.
11. D. Caucal. On the regular structure of prefix rewritings. pages 87 – 102, Copenhagen, Denmark, May 1990.
12. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: <http://www.grappa.univ-lille3.fr/tata>, 1997.
13. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
14. D. Debarbieux, Y. Roos, S. Tison, Y. Andre, and A.C. Caron. Path rewriting in semistructured data. In *proceedings of words'03: 4th International Conference on Combinatorics on Words*, pages 358–369, Turk, Finland, 2003. TUCS General Publication.
15. G. Grahne and A. Thomo. Query containment and rewriting using views for regular path queries under constraints. In *proceedings of PODS'03*, pages 111–122. Symposium on Principles of Database Systems, ACM, 2003.
16. R. Tarjan. Efficiency of a good but non linear set union algorithm. In *Journal of the ACM*, volume 22, n2, pages 215 – 225, 1975.