Containment of Regular Path Queries Under Path Constraints

Sylvain Salvati
Université de Lille, INRIA, CRIStAL/CNRS UMR 9189, France
Sophie Tison
Université de Lille, INRIA, CRIStAL/CNRS UMR 9189, France

Abstract

Data integrity is ensured by expressing constraints it should satisfy. One can also view constraints as data properties and take advantage of them for several tasks such as reasoning about data or accelerating query processing. In the context of graph databases, simple constraints can be expressed by means of path constraints while simple queries are modeled as regular path queries (RPQs). In this paper, we investigate the containment of RPQs under path constraints. We focus on word constraints that can be viewed as tuple-generating dependencies (TGDs) of the form

$$\forall x_1, x_2, \exists \bar{y}, a_1(x_1, y_1) \wedge \ldots \wedge a_i(y_{i-1}, y_i) \wedge \ldots \wedge a_n(y_{n-1}, x_2) \longrightarrow \exists \bar{z}, b_1(x_1, z_1) \wedge \ldots \wedge b_i(z_{i-1}, z_i) \wedge \ldots \wedge b_m(z_{m-1}, x_2) \ .$$

Such a constraint means that whenever two nodes in a graph are connected by a path labeled $a_1 \ldots a_n$, there is also a path labeled $b_1 \ldots b_m$ that connects them. Rewrite systems offer an abstract view of these TGDs: the rewrite rule $a_1 \ldots a_n \to b_1 \ldots b_m$ represents the previous constraint. A set of constraints $\mathcal{C}$ is then represented by a rewrite system R and, when dealing with possibly infinite databases, a path query p is contained in a path query q under the constraints $\mathcal{C}$ iff p rewrites to q with R. Contrary to what has been claimed in the literature we show that, when restricting to finite databases only, there are cases where a path query p is contained in a path query q under the constraints $\mathcal{C}$ while p does not rewrite to q with R. More generally, we study the finite controllability of the containment of RPQs under word constraints, that is when this containment problem on unrestricted databases does coincide with the finite case. We give an exact characterisation of the cases where this equivalence holds. We then deduce the undecidability of the containment problem in the finite case even when RPQs are restricted to word queries. We prove several properties related to finite controllability, and in particular that it is undecidable. We also exhibit some classes of word constraints that ensure the finite controllability and the decidability of the containment problem.

2012 ACM Subject Classification Theory of computation → Database constraints theory; Theory of computation → Rewrite systems; Theory of computation → Regular languages; Theory of computation → Grammars and context-free languages; Theory of computation → Database theory

Keywords and phrases Graph databases, rational path queries, query containment, TGDs, word constraints, rewrite systems, finite controllability, decision problems

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.17

Funding This work was financially supported by the ANR project CQFD (ANR-18-CE23-0003).

Acknowledgements We thank anonymous reviewers and Diego Figueira for their useful comments and Pierre Bourhis and Lily Gallois for fruitful discussions.

1 Introduction

The problem. In this paper, we investigate the containment of regular path queries (RPQs) under path constraints. We model graph databases as finite edge-labeled graphs. We call *ω-graph database* (or *ω-database*) graph databases where we remove the finiteness constraint. Queries we consider here are RPQs that test whether two nodes of the graph are connected by a path whose label belongs to a given regular language. Query containment and query equivalence are important properties when dealing with data: they play a central role in query optimizations, and also in reasoning about data. Query containment for RPQs without constraints is simply the problem of regular languages containment. In practice, query containment is also often used when dealing with particular databases for which we have knowledge about the actual data. We focus here on knowledge expressed by *word constraints* of the form $a_1 \dots a_n \sqsubseteq b_1 \dots b_m$. Such a constraint means that whenever two nodes in a graph are connected by a path labeled $a_1 \dots a_n$, there is also a path labeled $b_1 \dots b_m$ that connects them. From the logical point of view, this constraint can be seen as the following tuple-generating dependency (TGD):

$$\forall x_1, x_2, \exists \bar{y}, a_1(x_1, y_1) \wedge \dots \wedge a_i(y_{i-1}, y_i) \wedge \dots \wedge a_n(y_{n-1}, x_2) \longrightarrow \\ \exists \bar{z}, b_1(x_1, z_1) \wedge \dots \wedge b_i(z_{i-1}, z_i) \wedge \dots \wedge b_m(z_{m-1}, x_2) \ .$$

When $b_1 \dots b_m$ is the empty word, the constraint $a_1 \dots a_n \sqsubseteq \varepsilon$ is actually an Equality-Generating Dependency (EGD) which can be written:

$$\forall x_1, x_2, \exists \bar{y}, a_1(x_1, y_1) \wedge \dots \wedge a_i(y_{i-1}, y_i) \wedge a_n(y_{n-1}, x_2) \longrightarrow x_1 = x_2 \ .$$

In the rest of the paper we are going to be concerned only by word constraints $p \sqsubseteq q$ where neither p nor q is the empty string. EGDs pose problems slightly different from TGDs and part of the results of the paper does not apply when we remove this hypothesis.

Given a set of word constraints $\mathcal{C}$ and two RPQs P and Q, we may wonder whether for every ω-database that satisfies $\mathcal{C}$ the answer set of the query P is included in that of Q. In this case we write $P \sqsubseteq^{\omega}_{\mathcal{C}} Q$. If we restrict our attention to (finite) databases, we then write $P \sqsubseteq^{f}_{\mathcal{C}} Q$. In the context of databases without any constraints, the query containment problem boils down to language inclusion and the relation of query containment coincides in the finite case and in the infinite case. Such properties are called *finitely controllable.*

Word constraints are able to define precisely complex notions and then ensure that they are well used in databases. A simple example consists in defining what it means to be of the same generation in a genealogical tree, i.e. connecting two persons that are at the same distance of a common ancestor. We assume that we are given the edge labels `child` (that connects a person x to a person y when x is the child of y), `parent` (that connects a person x to a person y when x is the parent of y) and `sg` (for *same generation*), the following constraints give a definition of the relation `sg`:

$\texttt{child}\,\texttt{parent} \subseteq \texttt{sg}$
$\texttt{child}\,\texttt{sg}\,\texttt{parent} \subseteq \texttt{sg}$

Word constraints are basic and more refined properties would require more logical connective, e.g. modalities, joins on paths etc. However, the undecidability results of the paper for this basic class of constraints apply to more involved and more expressive classes.

Rewrite systems and word constraints. Suppose that there is a path labeled $p_1 p p_2$ in an ω-database that satisfies the constraint $p \sqsubseteq q$. We then know that there is a path $p_1 q p_2$ that connects the two vertices in the ω-database. If we further know that $p_1 q p_2 \sqsubseteq q'$, we can deduce that there exists a path q' between the two vertices. We can then apply the same kind of reasoning any number of times. This deduction mechanism is similar to a well known model of computation: *rewrite systems* or *semi-Thue systems*. Rewrite systems offer an abstract view of these particular TGDs: the rewrite rule $a_1 \dots a_n \to b_1 \dots b_m$ can be associated with the constraint $a_1 \cdots a_n \sqsubseteq b_1 \cdots b_m$. A (finite) set of constraints $\mathcal{C}$ is then represented by a (finite) rewrite system R.

Given $\mathcal{C}$ a finite set of word constraints, consider the containment problem $u \sqsubseteq^{\omega}_{\mathcal{C}} v$ where u and v are words. It is easy to see that if u rewrites to v , we have $u \sqsubseteq^{\omega}_{\mathcal{C}} v$, as we have seen that $\sqsubseteq^{\omega}_{\mathcal{C}}$ is transitive and closed under context (i.e. $u \sqsubseteq^{\omega}_{\mathcal{C}} v$ implies $w_1uw_2 \sqsubseteq^{\omega}_{\mathcal{C}} w_1vw_2$). With a classic construction, we can build an infinite model D of $\mathcal{C}$ such that $\mathrm{D} \models u \sqsubseteq v$ iff u rewrites to v (see Theorem 2). So, for word constraints, $\sqsubseteq^{\omega}_{\mathcal{C}}$ coincides with the associated rewrite relation and so, it is undecidable as the word problem (deciding whether two words are in the rewrite relation) for rewrite systems is in general undecidable.

Word constraints in databases and rewrite systems have already been connected in different frameworks, e.g. for *rooted* path constraints in *rooted* databases [1, 8] and for constraints in document stores [5]. The framework we consider here is the same as in [14] that emphasizes this strong connection between the query containment problem in the presence of word constraints and the word problem in rewrite systems. However, the connection is slightly more subtle than expected and we investigate it.

Whereas $\sqsubseteq^{\omega}_{\mathcal{C}}$ and $\xrightarrow{*}_{\mathcal{C}}$ coincide, $\sqsubseteq^{f}_{\mathcal{C}}$ and $\xrightarrow{*}_{\mathcal{C}}$ do not coincide in general - contrary to what is stated in [14, Theorem 2]. Indeed, on the one hand, the set of descendants of u by $\xrightarrow{*}_{\mathcal{C}}$ is recursively enumerable. On the other hand, the set $\{v \mid u \sqsubseteq^{f}_{\mathcal{C}} v\}$ is co-recursively enumerable [8]: one can enumerate the databases until one finds D so that $\mathrm{D} \models \mathcal{C}$ and there is a path labeled u between two nodes and no path labeled v between these two nodes. So, if they coincide, they are recursive: as soon as the set of descendants of u by $\xrightarrow{*}_{\mathcal{C}}$ is not recursive, it cannot be the case that the two sets coincide. As a consequence, *there must be cases where $u \sqsubseteq^{f}_{\mathcal{C}} v$ while it is not true that $u \xrightarrow{*}_{\mathcal{C}} v$.* We exhibit concrete examples that illustrate this phenomenon in the paper.

Query containment is not finitely controllable. By the preceding remark, in the setting of word constraints, for RPQs , $\sqsubseteq^{\omega}_{C}$ and $\sqsubseteq^{f}_{C}$ do not coincide: *query containment is not finitely controllable.* This result is central in this paper.

We study the finite controllability of the containment of RPQs under word constraints. We give an exact characterization of $P \sqsubseteq^{f}_{C} Q$ relying on $\xrightarrow{*}_{C}$. More precisely, $P \sqsubseteq^{f}_{C} Q$ holds iff every regular language closed under R that intersects with P intersects with Q. We then deduce from this characterization the undecidability of the containment problem in the finite case even when RPQs are restricted to word queries. This characterization also allows us to better understand when this containment problem on unrestricted databases does coincide with the finite case. We investigate several aspects of the finite controllabillty, and, in particular, we prove its undecidability. We also exhibit some classes of word constraints that ensure the finite controllability and the decidability of the containment problem.

Related work. As we already pointed out, this paper is strongly related to [14]. The setting we consider is the same. We correct some false claims -mainly the finite controllability of the query containment problem- announced in the paper and give new proofs of correct results whose proofs were relying on the finite controllability of the query containment problem. If the proofs are new ones, some ideas in our proofs were already present in [14].

The strong link between rewriting and path constraints has been investigated in [1, 3] in the rooted case: graphs are rooted, and queries are always evaluated starting from the root. In this setting, this amounts to use rewrite system with the *prefix rewriting strategy*, i.e. if $u \to v$ is a rule, only rewritings of the form "*up* rewrites to *vp*" are allowed. Prefix rewriting preserves regularity[7], and given a word u, it is easy to build a finite database with two nodes n_1, n_2, such that there is a path v between n_1 and n_2 iff v is a descendant of u by this prefix rewriting. This ensures the finite controllability of query containment and the decidability of regular path queries containment in the rooted case. This construction cannot be extended in general to the non-rooted case: indeed, the language of paths between two nodes in a finite model is a regular language, so cannot coincide with the set of descendants of u, if this set is non regular. Preservation of regularity is a key property, even if we prove that this is sufficient but not necessary to guarantee finite controllability in our setting. Links between rewriting and path constraints have also been used in the context of ontology-mediated query answering [5] and consistent query answering [4].

Undecidability of path constraint implication has already been proved in different contexts. In particular, undecidability of (resp. finite) implication has been proved r.e. (resp. co-r.e.) complete in the context of rooted graphs for a constraint language allowing constraints of the form $\forall x, (\alpha(r, x) \implies \forall y(\beta(x, y) \implies \gamma(x, y)))$ where r is a root of the graph, α, β, γ are paths [8]. In [9], similar constraints are expressed in Description Logic (DL) and finite implication is proved undecidable both in the rooted and in the global semantics. The global semantics is actually more general than our setting. The undecidability result concerning the global semantics of [9, Theorem 15] is strong enough to prove the undecidability of the query containment problem. For the sake of completeness, we give an original and direct proof of this result in Section 5.

Finite controllability for containment of conjunctive queries under inclusion and functional dependencies was introduced in [16]. The notion of finite controllability was later studied in several papers. In particular, finite controllability of containment for conjunctive queries under arbitrary inclusion dependencies and under keys and foreign keys has been proved in [18, 19] and finite controllability of UCQs was later showed for several classes of constraints, e.g. [12, 13, 2]. Consistent query answering for CRPQs under conjunction regular-path constraints have been studied in [4]. Finite Controllability for Ontology-Mediated Query Answering of C2RPQs has been studied in [10] where a complete classification of fragments of C2RPQs w.r.t. finite controllability under different classes of constraints, is provided according to the class of the underlying graph structure underlying the query. The results we obtain here for finite controllability are disjoint from these results, as we restrict to word constraints and as we focus to RPQs containment. Let us note that the classes of word constraints ensuring finite controllability that we exhibit don't fall, as far as we know, in any of the classes identified as ensuring finite controllability of CQs in the literature.

The problems we consider. Here follow the definitions of the main decision problems at the heart of this paper:

QC	**Query Containment**
Input	A set of word constraints $\mathcal{C}$, two RPQs P, Q
Question	$P \sqsubseteq^f_{\mathcal{C}} Q$?
QC^ω	$\omega-$**Query Containment**
Input	A set of word constraints $\mathcal{C}$, two RPQs P, Q
Question	$P \sqsubseteq^\omega_{\mathcal{C}} Q$?
UFC	**Uniform Finite Controllability**
Input	A set of word constraints $\mathcal{C}$
Question	For any RPQs P, Q , $P \sqsubseteq^f_{\mathcal{C}} Q$ iff $P \sqsubseteq^\omega_{\mathcal{C}} Q$?
FC	**Finite Controllability**
Input	A set of word constraints $\mathcal{C}$, two RPQs P, Q
Question	$P \sqsubseteq^f_{\mathcal{C}} Q$ iff $P \sqsubseteq^\omega_{\mathcal{C}} Q$?

Unfortunately, we will see that these problems are undecidable in general. So, we also consider subclasses of word constraints. Given *Problem* in $\{QC, QC^\omega, UFC, FC\}$, *Problem*$(\mathcal{C})$ will denote *Problem* restricted to the class $\mathcal{C}$ of word constraints. These

Table 1 Decidability of query containment under word constraints (U: undecidable, D: decidable).

Decision Problem \ Word Constraints $\mathcal{C}$	Unrestricted	Recursive	$\mathcal{C}$ preserves effectively regularity	$\mathcal{C}^{-1}$ preserves effectively regularity	Class of Proposition 26
$wwQC^\omega$	U	D	D	D	U
QC^ω	U	U	U	D	U
$wwQC$	U	?	D	D	D
QC	U	U	U	D	D

Table 2 Finite controllability of query containment under word constraints (U: undecidable).

Decision Problem \ Word Constraints $\mathcal{C}$	Unrestricted	$\mathcal{C}$ preserves effectively regularity	$\mathcal{C}^{-1}$ preserves effectively regularity	Class of Proposition 26	Example of Theorem 13
$wwUFC$	U	Yes	Yes	No	Yes
$wuUFC$	U	Yes	Yes	No	No

problems can also be restricted to the cases when P or Q is a word query. We denote by $\textsc{xy}Problem(\mathcal{C})$ with $\textsc{x},\textsc{y} \in \{\textsc{u},\textsc{w}\}$, the problem $Problem(\mathcal{C})$ where P (resp. Q) is restricted to words if $\textsc{x} = \textsc{w}$ (resp. $\textsc{y} = \textsc{w}$), unrestricted if $\textsc{x} = \textsc{u}$ (resp. $\textsc{y} = \textsc{u}$). When both $\textsc{x}$ and $\textsc{y}$ are $\textsc{u}$, they can be omitted. If $\mathcal{C}$ corresponds to the whole class of word constraints, it can be omitted. E.g. $\textsc{uw}QC$ denotes the problem of query containment where P is an RPQ and Q is a word; let $\mathcal{CF}$ the class of word constraints associated with context-free grammars, $\textsc{ww}QC(\mathcal{CF})$ denotes the problem of query containment where P and Q are words and R belongs to $\mathcal{CF}$.

In the sequel, when we mention *query containment* we mean QC, i.e. we refer to the *finite case*.

Summary of the results. Table 1 and Table 2 summarize most of the results presented in the paper.

2 Preliminaries

Graph databases. We model graph databases as edge-labeled graphs. For this we fix a finite alphabet of *labels* Σ, a *graph database* (or *database*) D is a pair (V, E) where V is a finite set of *objects* and $E \subseteq V \times \Sigma \times V$ is a finite set of *directed labeled edges*. We call *ω-graph database* (or *ω-database*) graph databases where we remove the finiteness constraint. When there is an edge (x, a, y) in an ω-database we say that there is an *edge labeled a between x and y* or *from x to y*. As it is customary, we allow ourselves to write $a(x, y)$ for the edge (x, a, y). Finally, we abuse notation and write $x \in \mathrm{D}$ or say x is in D to mention that x is an object of D. Similarly, for edges, we write $a(x, y) \in \mathrm{D}$ or say $a(x, y)$ belongs to D.

Path queries and database contraints. The *set of paths of labels* (or simply *paths*) over the alphabet Σ is Σ^* the set of (possibly empty) words or sequences built on Σ. We write ε for the empty path and Σ^+ for $\Sigma^* \setminus \{\varepsilon\}$. We inductively define the fact that two objects x and y of a ω-database D are connected by a path labeled u (we note this property $u(x, y) \in \mathrm{D}$) as follows:

- if $u = \varepsilon$, then $u(x, y) \in \mathrm{D}$ iff $x = y$,
- if $u = va$, then $u(x, y) \in \mathrm{D}$ iff there is z so that $v(x, z) \in \mathrm{D}$ and $a(z, y) \in \mathrm{D}$.

A path labeled u is considered as a *query* whose answer on an ω-database D is

$$\mathrm{ans}(u, \mathrm{D}) = \{(x, y) \mid u(x, y) \in \mathrm{D}\} \ .$$

Adopting the logical point of view, a path query $u = a_1 \cdots a_n$ can be seen as a particular kind of conjunctive query:

$$\exists x_1. \dots \exists x_{n-1}. \, a_1(x, x_1) \wedge a_2(x_1, x_2) \wedge \cdots \wedge a_n(x_{n-1}, y) \ .$$

In actual graph database systems, queries do not restrict to path labels but rather to *path labels languages*. Subsets of Σ^* are called *languages*. A language Q can also be seen as a query. The answer to that query on the ω-database D is:

$$\mathrm{ans}(Q, \mathrm{D}) = \{(x, y) \mid \exists u \in Q. \, u(x, y) \in \mathrm{D}\} \ .$$

In other words, such a query collects all the ordered pairs of nodes that are connected by a path labeled in Q. When Q is a regular language, the query induced by Q is called *regular path query* (RPQ).

Given two languages P and Q included in Σ^* and an ω-database D, whenever $\mathrm{ans}(P, \mathrm{D}) \subseteq \mathrm{ans}(Q, \mathrm{D})$ we say that D satisfies the *constraint* $P \sqsubseteq Q$ which we denote with $\mathrm{D} \models P \sqsubseteq Q$. An ω-database D satisfies a set constraints $\mathcal{C}$ when for every $P \sqsubseteq Q \in \mathcal{C}$, $\mathrm{D} \models P \sqsubseteq Q$; in that case we write $\mathrm{D} \models \mathcal{C}$. When for every ω-database D, $\mathrm{D} \models \mathcal{C}$ implies $\mathrm{D} \models P \sqsubseteq Q$, we write $P \sqsubseteq^{\omega}_{\mathcal{C}} Q$. When for every (finite) database D, $\mathrm{D} \models \mathcal{C}$ implies $\mathrm{D} \models P \sqsubseteq Q$, we write $P \sqsubseteq^{f}_{\mathcal{C}} Q$. Clearly, $P \sqsubseteq^{\omega}_{\mathcal{C}} Q$ implies $P \sqsubseteq^{f}_{\mathcal{C}} Q$. In this paper, we focus on finite sets of *word constraints*, i.e. constraints of the form $\{p\} \sqsubseteq \{q\}$ where $p \neq \varepsilon$ and $q \neq \varepsilon$, that we also write $p \sqsubseteq q$. We also focus on properties $P \sqsubseteq^{f}_{\mathcal{C}} Q$ and $P \sqsubseteq^{\omega}_{\mathcal{C}} Q$ when P and Q are regular languages that do not contain ε and sometimes more specifically when P or Q are singleton sets, i.e. represent words.

Rewrite systems. A rewrite system R on an alphabet Σ is a finite set of rules of the form $u \to v$ with u, v in Σ^+. The *one-step rewrite relation of* R, noted $\to_R$, is defined as follows: $p \to_R q$ when $p = u_1 u u_2$, $q = u_1 v u_2$ and R contains the rule $u \to v$. We note $\xrightarrow{*}_R$ the reflexive transitive closure of $\to_R$. We write $D_R(u)$ for the set of *descendants* of u, $\{v \mid u \xrightarrow{*}_R v\}$. For a language L, $D_R(L)$ is $\bigcup_{u \in L} D_R(u)$. We similarly define the set $A_R(u)$ of *ancestors* of u, $\{v \mid v \xrightarrow{*}_R u\}$ and $A_R(L) = \bigcup_{u \in L} A_R(u)$. A language L is *closed under* R when $D_R(L) = L$. We let R^{-1} be the rewrite system obtained by reversing each rule of R ($A_R(L) = D_{R^{-1}}(L)$).

We restrict rewrite systems to rules with non-empty words as we only wish to consider word constraints that are representable by means of TGDs. Notice that this restriction does not diminish the *computational power* of rewrite systems.

The *word problem* for rewrite systems is the question, given two words u and v, whether $u \xrightarrow{*}_R v$. This question is known to be undecidable, even with rules with non-empty words. We denote the set of left-hand (right-hand) sides of the rules in R by $lhs(R)$ ($rhs(R)$). We will use the following "modularity" property whose proof can be found in Appendix A:

▶ **Lemma 1.** *Let $R = R_1 \cup R_2$ a rewrite system such that the letters occurring in $lhs(R_1)$ do not occur in $rhs(R_2)$, then,* $\xrightarrow{*}_R = \xrightarrow{*}_{R_1} \circ \xrightarrow{*}_{R_2}$

Grammars. We will also use particular types of rewrite systems: grammars. A *type-0 grammar* G is a tuple (N, Σ, S, R) where N is a finite set of *non-terminals*, Σ is a finite set of *terminals*, S is an element of N, *the axiom of* G, and R is a finite set of rewrite rules on the alphabet $N \cup \Sigma$. The language defined by G is $\mathcal{L}(G) = \{w \in \Sigma^* \mid S \xrightarrow{*}_R w\}$. Notice that the rules of grammars being rewrite rules, they use non-empty words. It is well known that every recursively enumerable language can be defined by means of type-0 grammars. If L is a recursive language in Σ^+, L and its complement in Σ^+ can be generated by a type-0 grammar. So, there exists a rewrite system R (resp. R') over $\Sigma \cup N$, for some alphabet N (resp. N') such that there exists s (resp. s') in N (resp. N') s.t., for every u in Σ^+, $u \xrightarrow{*}_R s$ (resp. $u \xrightarrow{*}_{R'} s'$) iff u belongs (resp. does not belong) to L. A type-0 grammar $G = (N, \Sigma, S, R)$ is a *context-free grammar* when each its rule is of the form $A \to w$ where A is in N.

3 Query Containment and Rewriting

3.1 From word constraints to rewrite rules

As already explained, we can view word constraints and rewrite rules as similar objects. The rewrite rule $u \to v$ is naturally mapped to the constraint $u \sqsubseteq v$ and vice-versa. So given a set of contraints $\mathcal{C}$ we may consider it as a rewrite system and simply write $u \xrightarrow{*}_{\mathcal{C}} v$ to mean that the rewrite system naturally associated with $\mathcal{C}$ rewrites the word u to v. Similarly, given a rewrite system R, we may write $\mathrm{D} \models R$, $u \sqsubseteq^{\omega}_R v$ or $u \sqsubseteq^{f}_R v$ to denote the fact that in the set of constraints $\mathcal{C}_R$ that is naturally associated with R, we have $\mathrm{D} \models \mathcal{C}_R$, $u \sqsubseteq^{\omega}_{\mathcal{C}_R} v$ or $u \sqsubseteq^{f}_{\mathcal{C}_R} v$. *In the sequel, we will often conflate the set of word constraints and its naturally associated rewrite system.*

3.2 Comparing $\sqsubseteq^{\omega}_R$, $\sqsubseteq^{f}_R$ *and* $\xrightarrow{*}_R$

For a set of word constraints R, a natural question is then how the $\xrightarrow{*}_R$, $\sqsubseteq^{\omega}_R$ and $\sqsubseteq^{f}_R$ are related. We are first going to see that $\xrightarrow{*}_R$ and $\sqsubseteq^{\omega}_R$ coincide, using a construction inspired from [1]:

▶ **Theorem 2.** *Given a set of word constraints R, we have*

$$u \sqsubseteq^{\omega}_R v \textit{ iff } u \xrightarrow{*}_R v \ .$$

Proof. The right to left part of the equivalence does not present any difficulty. For the other direction, let $\mathrm{D} = (V, E)$ the ω-database defined by $V = \Sigma^*$ and $E = \{(v, a, u) \mid u \xrightarrow{*}_R va\}$. An easy induction shows that there is a path labeled w between the vertices v and u iff $u \xrightarrow{*}_R vw$. So if there is a path labeled w between the vertices v and u and $w \xrightarrow{*}_R t$, then $u \xrightarrow{*}_R vw \xrightarrow{*}_R vt$ and so there is a path labeled t between v and u. Thus if $w \xrightarrow{*}_R t$, then $\mathrm{D} \models w \sqsubseteq t$. This shows that $\mathrm{D} \models R$. Now, if $u \sqsubseteq^{\omega}_R v$, as there is a path labeled by u from the vertex ε to the vertex u, we get that there is also a path labeled by v from ε to u, and then, by what precedes that $u \xrightarrow{*}_R v$. ◀

This theorem tells us also the conditions under which query inclusion holds.

▶ **Corollary 3.** *Given a set of word constraints R and two queries Q_1 and Q_2 on that alphabet, the following are equivalent:*
- $Q_1 \sqsubseteq^{\omega}_{R} Q_2$,
- *for every p in Q_1, $D_R(p) \cap Q_2 \neq \emptyset$.*
- $Q_1 \subseteq A_R(Q_2)$.

Proof. Consider the ω-database used in the proof of Theorem 2. If $Q_1 \sqsubseteq^{\omega}_{R} Q_2$, as $\mathrm{D} \models R$, for every p in Q_1, there is a path labeled by some q in Q_2 from the vertex ϵ to the vertex p, so by what precedes, $D_R(p) \cap Q_2 \neq \emptyset$. The other implications do not present any difficulty. ◀

An important remark is that our characterization fails if we authorize rewrite rules with empty right hand sides, i.e. of the form $p \to \varepsilon$. For example, consider the rewrite system R that contains only one rule $a \to \varepsilon$. Clearly we do not have $a \xrightarrow{*}_R aa$, however, for every ω-database, if there is a path labeled a between two nodes x and y, as $a \to_R \varepsilon$, we must have $x = y$, so for every k there is path labeled a^k from x to y: in that case $a \sqsubseteq^{\omega}_{R} aa$ while it is not the case that $a \xrightarrow{*}_R aa$.

We have seen in the introduction that contrary to what is stated in of [14, Theorem 2], it is not true that $\sqsubseteq^{f}_{R}$ and $\xrightarrow{*}_R$ coincide. To summarize, we get:

▶ **Theorem 4.**
$u \xrightarrow{*}_R v$ iff $u \sqsubseteq^{\omega}_{R} v$.
$Q_1 \sqsubseteq^{\omega}_{R} Q_2$ iff $Q_1 \subseteq A_R(Q_2)$.
If $Q_1 \sqsubseteq^{\omega}_{R} Q_2$, then $Q_1 \sqsubseteq^{f}_{R} Q_2$.
In general, $u \sqsubseteq^{f}_{R} v$ does not imply that $u \sqsubseteq^{\omega}_{R} v$.

3.3 Characterizing $\sqsubseteq^{f}_{R}$: from query containment to non-separability

We have seen that $u \sqsubseteq^{f}_{R} v$ and $u \xrightarrow{*}_R v$ do not coincide. However, we will give a precise characterization of $Q_1 \sqsubseteq^{f}_{R} Q_2$ that uses closure under R:

▶ **Theorem 5.** *The following propositions are equivalent:*
- $Q_1 \sqsubseteq^{f}_{R} Q_2$,
- *every regular language closed under R that intersects with Q_1 intersects with Q_2.*

Proof. Suppose that it is not the case that $Q_1 \sqsubseteq^{f}_{R} Q_2$. Then there exists D, a model of R with two vertices x and y so that:
- there is a path of Q_1 labeled q from x to y,
- there is no path labeled by a word of Q_2 from x to y.

Seeing D as an automaton with initial state x and final state y, it must be the case that:
- it defines a regular language that is closed under R,
- it intersects with Q_1,
- it does not intersect with Q_2.

So there is a regular language closed under R that intersects with Q_1 and does not intersect with Q_2.

We now suppose that there is a regular language K closed under R that intersects with Q_1 (i.e. $K \cap Q_1 \neq \emptyset$) and does not intersect with Q_2 (i.e. $K \cap Q_2 = \emptyset$). We will build a database D so that $\mathrm{D} \models R$ and D does not satisfy $Q_1 \sqsubseteq Q_2$.

Let $K_1, \ldots, K_n$ be the finite set of *left residuals* of K. The left residual of K by a word q, noted $q^{-1}K$, is the language $q^{-1}K = \{p \mid qp \in K\}$. A language is a left residual of K when it is of the form $q^{-1}K$ for some q. It is well-known that a language is regular iff the set of its left residuals is finite. We start by making the following remark about the K_i's:

▶ **Lemma 6.** *For every i in $[n]$, K_i is closed under R.*

Proof. Take i in $[n]$, p in K_i and p' such that $p \xrightarrow{*}_R p'$. There is q so that $K_i = q^{-1}K$ and qp is in K. Since K is closed under R, qp' is also in K implying that p' is in $q^{-1}K = K_i$. ◀

We define the database D as follows:
- the set of vertices is $\{K_1, \ldots, K_n\}$,
- there is an edge labeled a between K_i and K_j iff $K_j \subseteq a^{-1}K_i$.

▶ **Lemma 7.** *There is a path in D labeled by $p \in \Sigma^+$ between K_i and K_j iff $p^{-1}K_i \supseteq K_j$.*

Proof. We proceed by induction on p.

When $p = a$, the conclusion directly follows from the definition.

Let $p = p'a$ with $p' \in \Sigma^+$. We first suppose that there is a path labeled p from K_i to K_j. So, there is K_k so that there is a path labeled p' from K_i to K_k and there is an arc labeled a between K_k and K_j. From the induction hypothesis, we have that $p'^{-1}K_i \supseteq K_k$ and by definition $a^{-1}K_k \supseteq K_j$, thus $p^{-1}K_i = a^{-1}p'^{-1}K_i \supseteq a^{-1}K_k \supseteq K_j$. Suppose now that $p^{-1}K_i \supseteq K_j$, we let $K_k = p'^{-1}K_i$. By induction, there is a path labeled p' between K_i and K_k, and moreover $a^{-1}K_k = p^{-1}K_i \supseteq K_j$ so that there is an edge labeled a between K_k and K_j. Therefore there is a path labeled p between K_i and K_j. ◀

▶ **Lemma 8.** *If there is a path labeled p between K_i and K_j, for any constraint $p \sqsubseteq q$ in R, there is a path labeled q between K_i and K_j.*

Proof. If there is a path labeled p between K_i and K_j, then $p^{-1}K_i \supseteq K_j$ from Lemma 7. So, if t belongs to K_j, pt belongs to K_i. As $p \sqsubseteq q$ in R, we have $p \rightarrow_R q$ and therefore $pt \rightarrow_R qt$. Now, from Lemma 6, qt also belongs to K_i and thus t is in $q^{-1}K_i$. Consequently $q^{-1}K_i \supseteq K_j$ and Lemma 7 implies that there is a path labeled q between K_i and K_j. ◀

The previous lemma shows that $\mathrm{D} \models R$. Now let $x = K$ and $y = q^{-1}K$ with $q \in K \cap Q_1$ (recall that $K \cap Q_1 \neq \emptyset$).

We now show that the set of words that label paths between x and y intersects with Q_1 and is included in K and so does not intersect with Q_2. It intersects with Q_1 as from Lemma 7, there is a path labeled by q between K and $q^{-1}K$. When there is a path labeled by p between K and $q^{-1}K$, we have that $p^{-1}K \supseteq q^{-1}K$ (Lemma 7). Now since $q \in K$, we have that $\varepsilon \in q^{-1}K$ and therefore ε is also an element of $p^{-1}K$ so p belongs to K and does not belong to Q_2 by hypothesis.

In a nutshell, we have $\mathrm{D} \models R$, there is a path between x and y labeled by a word of Q_1 (the word q) and no path labeled by a word in Q_2. This finally shows that it is not the case that $Q_1 \sqsubseteq^f_R Q_2$. ◀

So, we get as corollaries:

▶ **Corollary 9.** *The following propositions are equivalent:*
- $p \sqsubseteq^f_R Q_2$,
- *every regular language closed under R that contains p intersects with Q_2.*

▶ **Corollary 10.** *The following propositions are equivalent:*
- $p_1 \sqsubseteq^f_R p_2$,
- *every regular language closed under R that contains p_1 contains p_2.*

4 About (non) finite controllability

4.1 Non finitely controllable systems

We now show how to construct examples where we have $u \sqsubseteq_R^f v$ and we do not have $u \sqsubseteq_R^\omega v$. We first illustrate the idea of the construction by taking $R = \{c \to acb\}$. The set of descendants of c by R is $\{a^n cb^n \mid n \in \mathbb{N}\}$. Every finite database D such that $\mathrm{D} \models R$ and that contains a path labeled c between two nodes has necessarily (by a usual pumping argument) a path labeled $a^n cb^m$ with $n > m$ between these two nodes. Now, let $T = \{acb \to c, ac \to c, ac \to 0\}$. It is easy to see that any word $a^n cb^m \xrightarrow{*}_T 0$ iff $n > m$. Thus, in every finite database D such that $\mathrm{D} \models R \cup T$, if there is a path labeled by c between two vertices, then there is also a path labeled by 0: $c \sqsubseteq_{R\cup T}^f 0$. However, one can check that c does not rewrite to 0 with $R \cup T$. The idea underlying this construction can be used for any rewrite system R and any word u such that $D_R(u)$ is recursive but not regular.

▶ **Proposition 11.** *Let any set of word constraints R and any word u such that $D_R(u)$ is recursive but not regular. Let $R' = R \cup T \cup B$ defined as above: then $u \sqsubseteq_{R'}^f 0$ while it is not the case that $u \sqsubseteq_{R'}^\omega 0$.*

Proof. We assume that the symbols used by R are taken from the finite set Σ. We now take a finite set $\overline{\Sigma}$ with the same number of elements as Σ and a bijection between Σ and $\overline{\Sigma}$. We define $B = \{a \to \overline{a} \mid a \in \Sigma\}$. Given a word w in $(\Sigma \cup \Gamma)^*$ with $\Gamma \cap \Sigma = \emptyset$, we write $\overline{w}$, for the word obtained by replacing all occurrences of a in Σ by $\overline{a}$ and leaving other letters unchanged. As we suppose that $D_R(u)$ is recursive, we can assume the existence of a rewrite system T based on an alphabet Γ disjoint from Σ and that contains $\overline{\Sigma}$ and $\{0\}$ so that: $\overline{v} \to_T 0$ iff v is not in $D_R(u)$. Furthermore, we can suppose that 0 does not occur in $lhs(T)$. We take $R' = R \cup T \cup B$.

▶ **Lemma 12.** *For every v in Σ^+, $v \xrightarrow{*}_{R'} 0$ iff there is w in Σ^+ so that $v \xrightarrow{*}_R w$ and $w \notin D_R(u)$.*

Proof. The *if* part of the statement is a simple consequence of the definitions. For the *only if* part we prove a slightly stronger property. Given a word v we prove that there is w so that:

- $v \xrightarrow{*}_R w$ and
- $\overline{w} \xrightarrow{*}_T 0$.

Indeed, by Lemma 1, using the fact that the alphabet of $rhs(T)$ is disjoint from the alphabet of $lhs(R \cup B)$, that the alphabet of $rhs(B)$ is disjoint from the alphabet of the $lhs(R)$, we get that there exists w in Σ^+,w' in $(\Sigma \cup \overline{\Sigma})^*+$ such that $v \xrightarrow{*}_R w \xrightarrow{*}_B w' \xrightarrow{*}_T 0$. As 0 is only produced by T and as the alphabet of $lhs(T)$ is disjoint from Σ,we have $w' = \overline{w}$. ◀

Now, let K be a language containing u that is regular and closed under R'. Then $K \cap \Sigma^*$ is regular and contains $D_R(u)$: as $D_R(u)$ is not regular $K \cap \Sigma^*$ contains a word v in Σ^* that is not in $D_R(u)$: then $v \xrightarrow{*}_B \overline{v} \xrightarrow{*}_T 0$: as K is closed under R', K contains 0 and by Theorem 5, $u \sqsubseteq_{R'}^f 0$ while it is not the case that $u \xrightarrow{*}_{R'} 0$. ◀

4.2 Finite controllability: word queries vs RPQs

A consequence of [14, Theorem 2] that is also false, is that $Q_1 \sqsubseteq_R^f Q_2$ iff for every q_1 in Q_1 there is q_2 in Q_2 so that $q_1 \sqsubseteq_R^f q_2$ [14, Lemma 3]. We construct here an example of a rewrite system for which this property does not hold. We take the following rewrite system R:

$$a \to aab \qquad aab \to a$$
$$b \to abb \qquad abb \to b$$
$$ab \to ba \qquad ba \to ab$$

Let p, q be two words. If $p = \epsilon$ or $q = \epsilon$, it is easy to check that $p \xrightarrow{*}_R q$ iff $p = q$ iff $p \sqsubseteq^f_R q$. Let us now suppose that both are non empty. First, it is easy to check that $p \xrightarrow{*}_R q$ iff $|q|_a - |q|_b = |p|_a - |p|_b$ [1]. Second, $p \sqsubseteq^f_R q$ iff $p \xrightarrow{*}_R q$. Indeed, let $N_q = |q|_a - |q|_b$, $N_p = |p|_a - |p|_b$. If we do not have $p \xrightarrow{*}_R q$, then $N_q \neq N_p$. Let $K = \{w \mid |w|_a - |w|_b \equiv N_p \bmod (N_q+1)(N_p+1)\}$. The language K is regular and closed under R, contains p and does not contain q. So, using Theorem 5, we do not have $p \sqsubseteq^f_R q$.

Now, let $p = ab$ and let Q be the regular language b^+. By what precedes, there is no q in Q so that $p \sqsubseteq^f_R q$. Now, let K a regular language that contains p and that is closed under R. Then K contains $\{a^n b^n \mid n > 0\}$. Using the pumping lemma for regular languages we deduce that for some $m > 0$ we must have that $a^n b^{n+m}$ is in K. As K is closed under R, it contains b^m.

So, by Theorem 5, *we do have $p \sqsubseteq^f_R Q$, while there is no word q of Q so that $p \sqsubseteq^f_R q$.*

Furthermore, by Corollary 3, $P \sqsubseteq^\omega_R Q$ iff for every q_1 in P there is q_2 in Q_2 so that $q_1 \sqsubseteq^\omega_R q_2$. So, R provides an example of system such that containment of word queries is finitely controllable whereas the containment for regular path queries is not:

▶ **Theorem 13.** *There are sets of word constraints for which the containment of word queries is finitely controllable, while the containment of regular path queries is not.*

5 Query containment under word constraints is undecidable

In [14], the proof of undecidability of query containment under word constraints is a mere corollary of the assertion that $u \sqsubseteq^f_R v$ and $u \xrightarrow{*}_R v$ coincide. As we have seen earlier, this assertion is false. However, query containment under word constraints is actually undecidable. In this section, we give a proof of that fact. Notice that this result can also be derived from [9, Theorem 15].

▶ **Theorem 14.** *WWQC is undecidable.*

Proof. The undecidability result is obtained by reduction from the problem of the separability of context-free languages by some regular language. Formally this decision problem is stated as follows:

Input	two context free grammars G_1 and G_2
Question	Is there a regular language R so that $L(G_1) \subseteq R$ and $L(G_2) \cap R = \emptyset$?

This problem is known to be undecidable [15].

Take $G_1 = (N_1, \Sigma, S_1, R_1)$ and $G_2 = (N_2, \Sigma, S_2, R_2)$ two context free grammars on the alphabet Σ. We assume w.l.o.g. that

- $N_1 \cap N_2 = \emptyset$,
- they do not have rules of the form $A \to \varepsilon$,
- they are reduced (every non-terminal is reachable from the start symbol and defines a non-empty language).

We let R be the rewrite system on the alphabet $\Gamma = \Sigma \uplus N_1 \uplus N_2$ containing the rules of R_1 and R_2^{-1}.

[1] $|u|_x$ denotes the number of occurrences of the letter x in the word u.

▶ **Lemma 15.** *Given u and v in Γ^*, we have that $u \xrightarrow{*}_R v$ iff there is w so that $u \xrightarrow{*}_{R_1} w$ and $w \xrightarrow{*}_{R_2^{-1}} v$. Furthermore, if u does not contain any non-terminal of G_2 and v does not contain any non-terminal of G_1, then w is in Σ^+.*

Proof. As we have supposed that $N_1 \cap N_2 = \emptyset$, it must be the case that left hand side of a rule in R_1 does not share any symbol with the right hand side of a rule in R_2^{-1}. So, we get the first part of the lemma by Lemma 1. Finally, as R_1 (resp. R_2^{-1}) cannot generate (resp. eliminate) nonterminals of G_2 (resp. G_1), w does not contain any non-terminal symbol of G_2 (resp. G_1). ◀

We are going to show that it is not the case that $S_1 \sqsubseteq_R^f S_2$ iff there is a regular language that separates $L(G_1)$ and $L(G_2)$.

Suppose that it is not the case that $S_1 \sqsubseteq_R^f S_2$: by what precedes, there exists a regular language L closed under R that contains S_1 and does not contain S_2. As it is closed under R, it is closed under R_1: as it contains S_1, it contains $L(G_1)$. Similarly, it is closed under R_2^{-1} and as it does not contain $S2$, so it does not contain any word of $L(G_2)$: L is a regular language that separates $L(G_1)$ and $L(G_2)$.

Suppose now that L is a regular language that separates the languages of G_1 and of G_2, e.g. that contains $L(G_1)$ and does not intersect with $L(G_2)$. We can suppose $L \subseteq \Sigma^*$. Let us note Σ_1 the alphabet Σ enriched with the non terminals of G_1. By Theorem 5, we only have to prove that there is a regular language closed under R that contains S_1 and does not contain S_2.

Let us first notice that R_1 (resp. R_2^{-1}) preserves regularity by ascendants (resp. descendants).

Let $K_1 = \Sigma_1^* / A_{R_1}(\Sigma^*/L)$. K_1 is regular. Let us prove that K_1 is closed under R_1; let u in K_1, $u \xrightarrow{*}_{R_1} v$: then v belongs to Σ_1^*; if v does not belong to K_1, v belongs to $A_{R_1}(\Sigma^*/L)$ and, as $u \xrightarrow{*}_{R_1} v$, u belongs to $A_{R_1}(\Sigma^*/L)$, which contradicts the fact that u belongs to K_1. By construction, $K_1 \cap \Sigma^* \subseteq L$. As $D_{R_1}(S_1) \cap \Sigma^* = L(G_1)$, $D_{R_1}(S_1) \cap \Sigma^* \subseteq L$: S_1 belongs to K_1 and as K_1 is closed under R_1, $K_1 \cap \Sigma^* \supseteq L$. So, $K_1 \cap \Sigma^* = L$.

Now, let $K = D_{R_2^{-1}}(K_1)$; $S1$ belongs to K, as $S1$ belongs to K_1. By Lemma 15, K is closed by R, as K_1 is closed by R_1. Furthermore, by the same lemma, $K \cap \Sigma_2^* = D_{R_2^{-1}}(K_1 \cap \Sigma^*)$, i.e. $K \cap \Sigma_2^* = D_{R_2^{-1}}(L)$. As $L \cap L(G_2)$ is empty, K does not contain S_2.

So, K is a regular language closed under R that contains S_1 and does not contain S_2 : by Theorem 5, it is not the case that $S_1 \sqsubseteq_R^f S_2$. So $L(G_1)$ and $L(G_2)$ are separable by a regular language iff we do not have $S_1 \sqsubseteq_R^f S_2$.

So, given R, u, v, it is undecidable whether $u \sqsubseteq_R^f v$. ◀

6 How to ensure decidability and finite controllability of QC?

As we have proven that generally, $u \sqsubseteq_R^f v$ does not necessarily imply that $u \xrightarrow{*}_R v$, we naturally wonder which classes of rewrite systems ensure the equivalence between $u \sqsubseteq_R^f v$ and $u \xrightarrow{*}_R v$. We will see later that this equivalence is in general undecidable for arbitrary rewrite systems. This entails, that we must content ourselves with finding particular restrictions which have this property without ever having a complete and effective characterization. More generally, we can also try to identify classes of rewrite systems which ensure the equivalence between $Q_1 \sqsubseteq_R^f Q_2$ and $Q_1 \sqsubseteq_R^\omega Q_2$ for any RPQs Q_1 and Q_2.

By Subsection 3.3, we get easily (Proofs in Appendix B) the three following lemmas:

▶ **Lemma 16.** *For any word query p and RPQ Q, if $D_R(p)$ is regular, then $p \sqsubseteq_R^f Q$ iff $p \sqsubseteq_R^\omega Q$.*

▶ **Lemma 17.** *There is a set of word constraints R and RPQs P and Q so that $D_R(P)$ is regular, $P \sqsubseteq_R^f Q$, but we do not have $P \sqsubseteq_R^\omega Q$.*

▶ **Lemma 18.** *For any RPQs Q_1, Q_2, if $A_R(Q_2)$ is regular, then*
$Q_1 \sqsubseteq_R^f Q_2$ iff $Q_1 \sqsubseteq_R^\omega Q_2$.

So, an obvious property ensuring finite controllability of a rewrite system R is the *preservation of regularity* by R or its inverse R^{-1}. A rewrite system R (resp. R^{-1}) *preserves regularity* when for every regular language Q, $D_R(Q)$ (resp. $D_{R^{-1}}(Q) = A_R(Q)$) is a regular language. Lemma 17 tells us that it is not enough that the rewrite system preserves the regularity of a particular regular language.

▶ **Corollary 19.** *Let R a rewrite system. If R (resp. R^{-1}) preserves regularity, query containment is finitely controllable.*

Proof. If R^{-1} preserves regularity, we get the result by Lemma 18. If R preserves regularity, let us suppose that $Q_1 \sqsubseteq_R^f Q_2$. It implies that for every p in Q_1, $p \sqsubseteq_R^f Q_2$, and then by Lemma 16 $p \sqsubseteq_R^\omega Q_2$: so, $Q_1 \sqsubseteq_R^\omega Q_2$. ◀

We say that R *effectively preserves regularity* when it preserves regularity and when given a regular language Q (effectively presented by a regular expression or a finite state automaton) it is possible to compute (a representation of) $D_R(Q)$. We write RewRec to denote this class of rewrite systems and $\textsc{RewRec}^{-1}$ the class rewrite systems R whose inverse R^{-1} effectively preserves regularity.

▶ **Corollary 20.**
- *The problem* wuQC(RewRec) *is decidable.*
- *The problem* uu$QC(\textsc{RewRec}^{-1})$ *is decidable.*

Deciding whether a rewrite system preserves regularity is an undecidable property [17]. However, several classes of string rewrite systems that effectively preserve regularity have been identified, e.g. monadic systems [6] and match-bounded systems [11]. Monadic systems are systems whose rules have a letter as right-hand side – so the corresponding TGDs can be viewed as a Datalog program. A context-free grammar can be viewed as the inverse of a monadic rewrite system and so query containment is finitely controllable and decidable for the corresponding class of word constraints.

The first part of Corollary 20 cannot be generalized to uuQC(RewRec) which happens to be undecidable. We prove a slightly stronger result in Theorem 21 using the problem of universality of context-free languages which is well known to be undecidable. It can be stated as follows:

Input	A context free grammar G
Question	$\mathcal{L}(G) = \Sigma^+$?

The reduction is as follows: take a context free grammar $G = (N, \Sigma, S, \Delta)$ (w.l.o.g. we suppose that ϵ does not belong to G and does not occur in r.h.s. of Δ). The rewrite system $R = \Delta^{-1}$ is monadic and thus effectively preserves regularity [6]. Thus, Lemma 18 and Theorem 2 tell us that $\Sigma^+ \sqsubseteq_{R^{-1}}^f S$ is equivalent to the universality problem for G and is thus undecidable. This gives us an alternative proof of [14, Theorem 4]. Both proofs rely on similar constructions but the proof in [14] relies on a false theorem, namely [14, Theorem 3].

▶ **Theorem 21.** *The problem* uwQC(RewRec) *is undecidable.*

Our characterization leaves room for finding rewrite systems that do not preserve regularity but for which query containment is finitely controllable. The following proposition (Proof in Appendix C) states a modularity property: if two sets of word constraints are alphabet-disjoint and if both ensure finite controllability of query containment, their union does also ensure finite controllability.

▶ **Proposition 22.** *Let R_1 and R_2 two rewrite systems on disjoint alphabets Σ_1 and Σ_2 that ensure finite controllability of regular query containment. Then, $R = R_1 \cup R_2$ also ensures finite controllability of regular query containment.*

As the preservation of regularity of rewrite systems is not in general closed under union, this proposition allows us to construct rewrite systems R that ensure finite controllability of regular queries containment while neither R nor R^{-1} preserves regularity. E.g., let $R_1 = \{c \to acb\}$ and $R_2 = \{dfe \to f\}$. R_1^{-1} and R_2 are monadic and so preserve regularity and then R_1 and R_2 ensure both finite controllability. By the preceding proposition, $R = \{c \to acb, dfe \to f\}$ ensures finite controllability. However, neither R nor R^{-1} preserve regularity, as $D_R(c) = \{a^n cb^n \mid n \in \mathcal{N}\}$ and $A_R(f) = \{d^n fe^n \mid n \in \mathcal{N}\}$.

7 Finite controllability is undecidable

We are now looking at the decidability of *finite controllability* (wwFC) and of *uniform finite controllability* (wwUFC) defined in Section 1.

As $u \sqsubseteq_R^\omega v$ implies $u \sqsubseteq_R^f v$, we focus on the undecidability of $u \sqsubseteq_R^\omega v$ under the hypothesis that $u \sqsubseteq_R^f v$. We use a reduction from the following undecidable problem (Proof in Appendix D):

▶ **Lemma 23.** *The following problem is undecidable:*

Input L_1, L_2 *two recursive sets that are not separable by a regular set.*
Question *Is* $L_1 \cap L_2$ *empty?*

So, let L_1, L_2 be two recursive sets on the alphabet Σ^+ that are not separable by a regular set. As previously, we take a finite set $\overline{\Sigma}$ in bijection with Σ. We define $B = \{a \to \overline{a} \mid a \in \Sigma\}$. Given an alphabet Γ disjoint from Σ and a word w in $(\Sigma \uplus \Gamma)^*$, we write $\overline{w}$, for the word obtained by replacing all occurrences of a in Σ by $\overline{a}$ and leaving other letters unchanged.

As L_1, L_2 are recursive, there exist two rewrite systems :

- R_1 on an alphabet Σ_1 containing Σ and a symbol s_1 such that for any u in Σ^*, $s_1 \xrightarrow{*}_{R_1} u$ iff $u \in L_1$.
- R_2 on an alphabet Σ_2 containing $\overline{\Sigma}$ and a symbol s_2 such that for any u in Σ^*, $\overline{u} \xrightarrow{*}_{R_2} s_2$ iff $u \in L_2$.

We can suppose that $\Sigma_1 \cap \Sigma_2 = \emptyset$. Let $\Delta = \Sigma_1 \cup \Sigma_2 \cup \{\sharp_l, \sharp_r, g\}$ where $\sharp_l$, $\sharp_r$ and g are fresh symbols. We define:

$R_\sharp = \{x \to \sharp_l s_1 \sharp_r \mid x \in \Delta \setminus \{\sharp_l, \sharp_r\}\}$
$R_g = \{\sharp_l s_2 \sharp_r \to g\} \cup \{g \to xg, g \to x, xg \to g, gx \to g \mid x \in \Delta\}$
$R_{L_1,L_2} = R_1 \cup R_2 \cup R_\sharp \cup B \cup R_g$.

In the sequel, we denote R_{L_1,L_2} by R. Then, we get (Proof in Appendix E):

▶ **Lemma 24.**

1. *If* $L_1 \cap L_2 \neq \emptyset$, $u \xrightarrow{*}_R v$ *for any* (u, v) *in* $\Delta^+ \setminus \{\sharp_l, \sharp_r\}^* \times \Delta^+$.
2. *If* $L_1 \cap L_2 = \emptyset$, *we do not have* $\sharp_l s_1 \sharp_r \xrightarrow{*}_R \sharp_l s_2 \sharp_r$.
3. $\sharp_l s_1 \sharp_r \sqsubseteq_R^f \sharp_r s_2 \sharp_r$.
4. *If* $u \in \{\sharp_l, \sharp_r\}^*$, *then* $u \sqsubseteq_R^f v$ *(resp.* $u \xrightarrow{*}_R v$*) iff* $u = v$.
5. $u \sqsubseteq_R^f v$ *iff* $u \in \Delta^+ \setminus \{\sharp_l, \sharp_r\}^*$ *or* $u = v$.

A consequence of Lemma 24 is that deciding equivalence of $\sharp_l s_1 \sharp_r \sqsubseteq^f_R \sharp_r s_2 \sharp_r$ and $\sharp_l s_1 \sharp_r \xrightarrow{*}_R \sharp_r s_2 \sharp_r$ amounts to decide non emptiness of $L_1 \cap L_2$. More generally, for every $u, v \in \Delta^+$, $u \sqsubseteq^f_R v$ is equivalent to $u \xrightarrow{*}_R v$ only when $L_1 \cap L_2 \neq \emptyset$. Therefore we get the undecidability of finite controllability and uniform finite controllability of a rewrite system:

▶ **Theorem 25.** *WWFC and WWUFC are undecidable.*

Furthermore, let C be the class of systems R_{L_1,L_2} for L_1, L_2 recursive sets on the alphabet Σ that are not separable by a regular set. On the one hand, for any rewrite system R in C, $u \sqsubseteq^f_R v$ iff $u \in \Delta^+/\{\sharp_l, \sharp_r\}^*$ or $u = v$. So, for any rewrite system R in C, $\sqsubseteq^f_R$ is decidable. On the other hand, $s_1 \xrightarrow{*}_R s_2$ is equivalent to $L_1 \cap L_2 = \emptyset$ so is undecidable in C. So, we get:

▶ **Proposition 26.** *There exists a class of rewrite systems for which the WWQC is decidable whereas WWQC$^\omega$ is undecidable.*

This proposition contradicts [14, Corallary 2] that is a consequence of [14, Theorem 2].

8 Conclusion

Starting with an error in the proof of [14], we have studied the containment of RPQs under word constraints. Contrary to what was claimed in [14], we have showed that this property is not finitely controllable in general. We also have given counter-examples to properties that were corollaries of this false claim and alternate proofs to those that were correct.

For this, we have studied the relation between word constraints and rewrite systems. We have given a precise characterization of $P \sqsubseteq^f_R Q$ in terms of separability and closure under rewriting by R. This characterization has played a key role in identifying the properties of query containment in this setting and in giving a correct proof of the undecidability of the containment problem.

The stage being set we have studied further properties of finite controllability in this setting. In particular, we have showed that it is undecidable and we have exhibited some classes of constraints that ensure the finite controllability and the decidability of query containment. This study allowed us to show that the finite controllability of the containment of word queries and that of RPQs do not coincide. More specifically we give examples of constraints for which the containment of word queries is finitely controllable, whereas it is not the case for general RPQs. Interestingly we have also showed that when $p \sqsubseteq^f_R Q$, we do not necessarily have $p \sqsubseteq^f_R q$ for some word q in Q, i.e. the "witness" of containment depends on the model.

We observe that for obtaining finite controllability in this setting, it suffices to consider constraints for which the underlying rewrite system preserves regularity by inverse rewriting. We also observe that those for which the underlying rewrite system preserves regularity have nice properties. Such rewrite systems have been widely studied. We show that other rewrite systems can also have interesting properties with respect to that containment problem (as done in Proposition 22).

Finally many of the results of the paper could be extended to RPQ constraints of the form $P \subseteq u$, where P is an RPQ , u a word. In particular, we think that the characterization of $u \sqsubseteq^f_R v$ in terms of separability could likely be extended. An interesting consequence would then be that decidability results about finite controllability and query containment would then hold for some classes of RPQs .

References

1 Serge Abiteboul and Victor Vianu. Regular path queries with constraints. *Journal of Computer and System Sciences*, 58(3):428–452, 1999. doi:10.1006/jcss.1999.1627.

2 Giovanni Amendola, Nicola Leone, and Marco Manna. Finite controllability of conjunctive query answering with existential rules: Two steps forward. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 5189–5193. ijcai.org, 2018. doi:10.24963/ijcai.2018/719.

3 Yves André, Anne-Cécile Caron, Denis Debarbieux, Yves Roos, and Sophie Tison. Path constraints in semistructured data. *Theoretical Computer Science*, 385(1-3):11–33, oct 2007. doi:10.1016/j.tcs.2007.05.010.

4 Pablo Barceló and Gaëlle Fontaine. On the data complexity of consistent query answering over graph databases. *Journal of Computer and System Sciences*, 88:164–194, 2017. doi:10.1016/j.jcss.2017.03.015.

5 Meghyn Bienvenu, Pierre Bourhis, Marie-Laure Mugnier, Sophie Tison, and Federico Ulliana. Ontology-mediated query answering for key-value stores. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 844–851. ijcai.org, 2017. doi:10.24963/ijcai.2017/117.

6 Ronald V. Book and Friedrich Otto. *String-Rewriting Systems*. Texts and Monographs in Computer Science. Springer, 1993. doi:10.1007/978-1-4613-9771-7.

7 R. Büchi. *Regular canonical systems*. Arch. Math. Logik Grundlag., 1964.

8 Peter Buneman, Wenfei Fan, and Scott Weinstein. Path constraints in semistructured databases. *Journal of Computer and System Sciences*, 61(2):146–193, 2000. doi:10.1006/jcss.2000.1710.

9 Diego Calvanese, Magdalena Ortiz, and Mantas Simkus. Verification of evolving graph-structured data under expressive path constraints. In Wim Martens and Thomas Zeume, editors, *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, volume 48 of *LIPIcs*, pages 15:1–15:19. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICDT.2016.15.

10 Diego Figueira, Santiago Figueira, and Edwin Pin Baque. Finite Controllability for Ontology-Mediated Query Answering of CRPQ. In *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR'20), Rhodes, Greece, sep 2020. doi:10.24963/kr.2020/39.

11 Alfons Geser, Dieter Hofbauer, and Johannes Waldmann. Match-bounded string rewriting systems. *Appl. Algebra Eng. Commun. Comput.*, 15(3-4):149–171, 2004. doi:10.1007/s00200-004-0162-8.

12 Tomasz Gogacz and Jerzy Marcinkowski. Converging to the chase – A tool for finite controllability. *Journal of Computer and System Sciences*, 83(1):180–206, 2017. doi:10.1016/j.jcss.2016.08.001.

13 Georg Gottlob, Marco Manna, and Andreas Pieris. Finite model reasoning in hybrid classes of existential rules. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1831–1837. International Joint Conferences on Artificial Intelligence Organization, jul 2018. doi:10.24963/ijcai.2018/253.

14 Gösta Grahne and Alex Thomo. Query containment and rewriting using views for regular path queries under constraints. In *Proceedings of the Twenty-Second ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 9-12, 2003, San Diego, CA, USA*, pages 111–122, 2003. doi:10.1145/773153.773165.

15 Harry B. Hunt III. On the decidability of grammar problems. *J. ACM*, 29(2):429–447, 1982. doi:10.1145/322307.322317.

16 D.S. Johnson and A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *Journal of Computer and System Sciences*, 28(1):167–189, 1984. doi:10.1016/0022-0000(84)90081-3.

17 Friedrich Otto. Some undecidability results concerning the property of preserving regularity. *Theor. Comput. Sci.*, 207(1):43–72, 1998. doi:10.1016/S0304-3975(98)00055-3.

18 Riccardo Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In Stijn Vansummeren, editor, *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 356–365. ACM, 2006. doi:10.1145/1142351.1142404.

19 Riccardo Rosati. On the finite controllability of conjunctive query answering in databases under open-world assumption. *J. Comput. Syst. Sci.*, 77(3):572–594, 2011. doi:10.1016/j.jcss.2010.04.011.

A Proof of Lemma 1

Proof. First let us note that if $u \to_{R_2} w \to_{R_1} v$, there exists w' such that $u \to_{R_1} w' \to_{R_2} v$. Indeed, as $u \to_{R_2} w$, $u = u_1 l_2 u_2$, $w = u_1 r_2 u_2$ for some rule (l_2, r_2) in R_2. As $w \to_{R_1} v$, $w = v_1 l_1 v_2$ for some rule (l_1, r_1) in R_2 and $v = v_1 r_1 v_2$. By hypothesis the letters of l_1 do not occur in r_2: so either $v_1 l_1$ is a prefix of u_1, or $l_1 v_2$ is a suffix of u_2. W.l.o.g., let us suppose that $v_1 l_1$ is a prefix of u_1: $w = v_1 l_1 w_1 r_2 u_2$, $v = v_1 r_1 w_1 r_2 u_2$ and $u = v_1 l_1 w_1 l_2 u_2$: $u \to_{R_1} v_1 r_1 w_1 l_2 u_2 \to_{R_2} v_1 r_1 w_1 r_2 u_2 = v$.

Now, let $u \xrightarrow{*}_R v$: there exists a derivation $u \to_R u_1 \to_R \ldots u_n = v$. At step i either a rule of R_1 or a rule of R_2 is applied. An inversion in the derivation is a couple (i, j) with $i < j$ such that the rule applied at step i is in R_2 whereas the rule applied at step j is in R_1. We will prove that the derivation can be "sorted" and that $u \xrightarrow{*}_{R_1} \circ \xrightarrow{*}_{R_2} v$ by induction of the number of inversions in the derivation. If the number of inversions is 0, by definition, $u \xrightarrow{*}_{R_1} \circ \xrightarrow{*}_{R_2} v$. Otherwise, there is an inversion: this implies that there exists a step i such that the rule applied at step i is in R_2 whereas the rule applied at step $i+1$ is in R_1. By what precedes, we can permute these two steps and we get a derivation $u \xrightarrow{*}_R v$ whose number of inversions is strictly smaller: so, by induction, $u \xrightarrow{*}_{R_1} \circ \xrightarrow{*}_{R_2} v$. ◀

B Proof of Lemmas 16, 17, 18

Proof of Lemma 16. By Theorem 4 we just need to prove that if $p \sqsubseteq^f_R Q$ then $p \sqsubseteq^\omega_R Q$. $D_R(p)$ is a regular language closed by R and contains p. Then, by Theorem 5, if $p \sqsubseteq^f_R Q$, $D_R(p)$ intersects with Q and so, there exists q in Q such that $p \xrightarrow{*}_R q$; then, by Theorem 2, $p \sqsubseteq^\omega_R q$ and so, $p \sqsubseteq^\omega_R Q$. ◀

Proof of Lemma 17. We take the rewrite system R of Section 4.2. We let $P = \Sigma^+$ and $Q = a^+ + b^+$. We define B as the set $\{u \mid u \neq \varepsilon \wedge |u|_a = |u|_b\}$. For every $u \notin B$, we either have that $|u|_a > |u|_b$ or $|u|_a < |u|_b$. In the first case, $u \xrightarrow{*}_R a^k$ for $k = |u|_a - |u|_b$ and, in the second case, $u \xrightarrow{*}_R b^k$ for $k = |u|_b - |u|_a$. So for every $u \notin B$, we have that $u \sqsubseteq^\omega_R a^+ + b^+$ and thus $u \sqsubseteq^f_R a^+ + b^+$. In contrast, we have that $D_R(B) = B$ and $B \cap a^+ + b^+ = \emptyset$. Therefore for every $u \in B$, we do not have $u \sqsubseteq^\omega_R a^+ + b^+$. Thus, we do not have that $\Sigma^+ \sqsubseteq^\omega_R b^+ + a^+$.

However, for every $u \in B$, let $n = |u|_a$, as $u \xrightarrow{*}_R a^n b^n$ we have that $u \sqsubseteq^f_R a^n b^n$. We have seen Section 4.2 that $a^n b^n \sqsubseteq^f_R b^m$ for some $m > 0$. This shows that $\Sigma^+ \sqsubseteq^f_R a^+ + b^+$. ◀

Proof of Lemma 18. By Theorem 4 we just need to prove that if $Q_1 \sqsubseteq^f_R Q_2$ then $Q_1 \sqsubseteq^\omega_R Q_2$. $K = \Sigma^*/A_R(Q_2)$ is a regular language closed under R that does not intersect with Q_2. From Theorem 5, if $Q_1 \sqsubseteq^f_R Q_2$, K does not intersect with Q_1; then $Q_1 \subseteq A_R(Q_2)$ and by Theorem 4 $Q_1 \sqsubseteq^\omega_R Q_2$. ◀

C Proof of Proposition 22

Proof. We only need to prove that $q \sqsubseteq_R^f Q$ implies $q \sqsubseteq_R^\omega Q$. Take a path q and a regular set of paths Q such that there is no p in Q with $q \xrightarrow{*}_R p$. We will prove the existence of a regular language K closed under R containing q and not intersecting with Q.

Let $q_1 p_1 \ldots q_n p_n$ be the unique decomposition of q such that $q_1 \in \Sigma_1^*$, $p_n \in \Sigma_2^*$, $q_2, \ldots, q_n \in \Sigma_1^+$ and $p_1, \ldots, p_{n-1} \in \Sigma_2^+$.

Let $P = \Sigma_1^*(\Sigma_2^+\Sigma_1^+)^{n-1}\Sigma_2^*$. If $Q \cap P$ is empty, we can choose P for K: it is easy to check that it satisfies all the conditions. Otherwise, $Q \cap P$ is a non empty regular language included in P.

We use the following property: given three regular sets N, A, B, if $N \subseteq AB$, then $N = \bigcup_{i \in I} A_i B_i$ where the A_i (resp. B_i) are all regular, all included in A (resp. B) and I is finite. Indeed it is easy to check that $N = \bigcup_{u \in A}([u] \cap A).(u^{-1}N \cap B)$, where $[u] = \{v \mid u^{-1}N = v^{-1}N\}$. The number of distincts $[u]$ and $u^{-1}N$ is finite as N is regular, so we get the required finite decomposition.

Iterating the aforementioned property, we get that $Q \cap P$ can be decomposed in a finite union of products of regular languages $\bigcup_{i \in I} A_{1,i}B_{1,i} \ldots A_{n,i}B_{n,i}$ for some finite I, where $A_{1,i} \subseteq \Sigma_1^*, B_{n,i} \subseteq \Sigma_2^*$, $A_{j,i} \subseteq \Sigma_1^+$ for $j \neq 1$, $B_{j,i} \subseteq \Sigma_2^+$, for $j \neq n$.

As, by hypothesis, there is no p in Q so that $q \xrightarrow{*}_R p$, $Q \cap P \cap D_R(q) = \emptyset$. Thus, for every i in I, $A_{1,i}B_{1,i} \ldots A_{n,i}B_{n,i} \cap D_R(q)$ is empty and so there exists j such that $A_{j,i} \cap D_{R_1}(q_j) = \emptyset$ or $B_{j,i} \cap D_{R_2}(p_j) = \emptyset$. Since R_1 (resp. R_2) is finitely controllable, there is a regular language K_1 (resp. K_2) closed under R_1 (resp. R_2) containing $D_{R_1}(q_j)$ (resp. $D_{R_2}(p_j)$) and not intersecting with $A_{j,i}$ (resp.$B_{j,i}$).

Define L_i as:

- $L_i = \Sigma_1^*(\Sigma_2^+\Sigma_1^+)^{j-2}\Sigma_2^+(K_1 \cap \Sigma_1^+)(\Sigma_2^+\Sigma_1^+)^{n-j}\Sigma_2^*$
  in case $A_{j,i} \cap D_{R_1}(q_j) = \emptyset$,
- $L_i = \Sigma_1^*(\Sigma_2^+\Sigma_1^+)^{j-1}(K_2 \cap \Sigma_2^+)(\Sigma_1^+\Sigma_2^+)^{n-j-1}\Sigma_1^+\Sigma_2^*$
  otherwise (in this case $B_{j,i} \cap D_{R_2}(p_j) = \emptyset$).

It is easy to check that in both cases L_i is regular, closed under R, contains q, does not intersect with $A_{1,i}B_{1,i} \ldots A_{n,i}B_{n,i}$.

Define K as $\bigcap_{i \in I} L_i$: K is regular, closed under R, contains q, does not intersect with Q. By Theorem 5, we obtain that it is not the case that $q \sqsubseteq_R^f Q$. ◀

D Proof of Lemma 23

Proof. Our proof relies on undecidability of emptiness of the intersection of recursive sets of numbers. We represent sets of numbers as one letter languages. For $k \neq 0$, we let $\mathbf{p}_k$ to be the k^{th} prime number and, given a letter a and a set of numbers $\mathcal{N}$, we write $\mathcal{P}_a(\mathcal{N})$ for the language $\{a^{\mathbf{p}_n} \mid n \in \mathcal{N}\}$. We write $\mathbf{P}_a$ for the language $\{a^p \mid p \text{ prime number}\}$ and $\mathbf{P}_a^c$ for $a^* - \mathbf{P}_a$.

First, let us notice that if L is regular and L is included in $\mathbf{P}_a$, then L is finite. Indeed, by a pumping argument, we get that if L is an infinite regular language included in Σ^*, there exists $n \geq 0, m > 0$ such that a^{n+pm} belongs to L for any integer p. But, then $a^{n+(n+2m+2)m}$ belongs to L. As $n + (n + 2m + 2) * m = (n + 2m) * (m + 1)$ is not prime, L is not included in $\mathbf{P}_a$.

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ two infinite arbitrary recursive sets of numbers. We let $\mathcal{L}_1 = \mathbf{P}_a^c \cup \mathcal{P}_a(\mathcal{N}_1)$, $\mathcal{L}_2 = \mathcal{P}_a(\mathcal{N}_2)$. We have that $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ iff $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$. If R is a regular language containing $\mathcal{L}_1$, a^*/R is included in P_a, and then, from what precedes, is finite; so, R contains all, but finitely, many elements of a^* and as $\mathcal{L}_2$ is infinite, we must have $\mathcal{L}_2 \cap R \neq \emptyset$. In other words, $\mathcal{L}_1$ and $\mathcal{L}_2$ cannot be separated by a regular set. However deciding $\mathcal{L}_1 \cap \mathcal{L}_2 = \emptyset$ is equivalent to deciding $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$. ◀

E Proof of Lemma 24

Proof. We prove below the different items, the second one being the most technical.

1. If $L_1 \cap L_2$ is non empty, there exists w such that $s_1 \xrightarrow{*}_{R_1} w \xrightarrow{*}_B \overline{w} \xrightarrow{*}_{R_2} s_2$, so $s_1 \xrightarrow{*}_R s_2$. Let (u,v) in $\Delta^+/\{\sharp_l,\sharp_r\}^* \times \Delta^+$: u can be decomposed in $u_1 x u_2$ with $x \notin \{\sharp_l,\sharp_r\}$. Then, $u \to_{R_\sharp} u_1\sharp_l s_1 \sharp_r u_2 \xrightarrow{*}_R u_1 \sharp_l s_2 \sharp_r u_2 \to_{R_g} u_1 \sharp_l g \sharp_r u_2 \xrightarrow{*}_{R_g} g$. As g can generate any word of Δ^+, we have $u \xrightarrow{*}_R v$.
2. Suppose $\sharp_l s_1 \sharp_r \xrightarrow{*}_R \sharp_l s_2 \sharp_r$. We first prove that $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2 \cup R_\sharp} m_1 \sharp_l s_2 \sharp_r m_2$. If no g occurs in the derivation, the derivation is in $R_1 \cup B \cup R_2 \cup R_\sharp$ and satisfies the requirements. Otherwise, let us consider i_g, the first step of the derivation where g occurs. As g is only produced by $\sharp_l s_2 \sharp_r \to g$ or by a rule containing g in its l.h.s., the derivation truncated at its $i_g - 1$ first steps is of the form $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2 \cup R_\sharp} m_1 \sharp_l s_2 \sharp_r m_2$ for some m_1, m_2. Now, let us prove that by induction of the length of the derivation the following property: Let u in $(\Sigma_1 \cup \Sigma_2)^*$; If $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2 \cup R_\sharp} m_1 \sharp_l u \sharp_r m_2$ for some m_1, m_2, there exists a derivation $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2} \sharp_l u \sharp_r$. If the derivation is of length 1, either $\sharp_l s_1 \sharp_r \to_{R_1} \sharp_l u \sharp_r$ or $\sharp_l s_1 \sharp_r \to_{R_\sharp} \sharp_l \sharp_l s_1 \sharp_r \sharp_r$ and $u = s_1$. In both cases, $\sharp_l s_1 \sharp_r \xrightarrow{\leq 1}_{R_1} \sharp_l u \sharp_r$. Let us now suppose that the property is true for derivations of length n and let a derivation of length $n+1$: If $\sharp_l u \sharp_r$ is not concerned by the last rewriting step, we have $\sharp_l s_1 \sharp_r \xrightarrow{n}_R m_1' \sharp_l u \sharp_r m_2'$ for some m_1', m_2' and we have the property by induction. If $\sharp_l u \sharp_r$ is concerned by the last step, as u is in $(\Sigma_1 \cup \Sigma_2)^*$, either the last step uses a rule $x \to_{R_\sharp} \sharp_l s_1 \sharp_r$ and $u = s_1$, so the property is trivial or the derivation is of the form $\sharp_l s_1 \sharp_r \xrightarrow{n}_{R_1 \cup B \cup R_2 \cup R_\sharp} m_1 \sharp_l u' \sharp_l m_2 \to_{R_1 \cup B \cup R_2} m_1 \sharp_l u \sharp_r m_2$. Then by induction, $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2} \sharp_l u' \sharp_r \to_{R_1 \cup B \cup R_2} \sharp_l u \sharp_r$. Applying the property to $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2 \cup R_\sharp} m_1 \sharp_l s_2 \sharp_r m_2$, we get that there is a derivation $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1 \cup B \cup R_2} \sharp_l s_2 \sharp_r$. By adapting the reasonment already used in Lemma 12, there exists a derivation $\sharp_l s_1 \sharp_r \xrightarrow{*}_{R_1} \sharp_l m \sharp_r \xrightarrow{*}_B \sharp_l \overline{m} \sharp_r \xrightarrow{*}_{R_2} \sharp_l s_2 \sharp_r$. But then m belongs to $L_1 \cap L_2$ that would not be empty.
3. Every regular language closed under R containing $\sharp_g s_1 \sharp_d$ contains $\sharp_g L_1 \sharp_d$ and then intersects with $\sharp_g L2 \sharp_d$ as L_1 and $L2$ are not regularly separable. So, by Corollary 10, $\sharp_g s_1 \sharp_d \not\sqsubseteq^f_R \sharp_g s_2 \sharp_d$.
4. If $u \in \{\sharp_g, \sharp_d\}^*$, then $D_R(u) = \{u\}$ is a regular language closed under R. By Corollary 10, $u \sqsubseteq^f_R v$ iff $u = v$ and, therefore, iff $u \xrightarrow{*}_R v$.
5. Let $u \in \Delta^+/\{\sharp_l,\sharp_r\}^*$: we have $u = u_1 x u_2$ with $x \in \Delta$. Therefore, $u \to_{R_\sharp} u_1 \sharp_l s_1 \sharp_r u_2$, so $u \sqsubseteq^f_R u_1 \sharp_l s_1 \sharp_r u_2$, As $\sharp_l s_1 \sharp_r \sqsubseteq^f_R \sharp_l s_2 \sharp_r$, $u \sqsubseteq^f_R u_1 \sharp_l s_2 \sharp_r u_2 \sqsubseteq^f_R u_1 g u_2 \sqsubseteq^f_R v$ for any v. In case $u \in \{\sharp_l,\sharp_r\}^*$, we have already seen that that $u \sqsubseteq^f_R v$ iff $u = v$. In a nutshell, $u \sqsubseteq^f_R v$ iff $u \in \Delta^+/\{\sharp_l,\sharp_r\}^*$ or $u = v$. ◀