Conjunctive Regular Path Queries under Injective Semantics

Diego Figueira
diego.figueira@cnrs.fr
Univ. Bordeaux, CNRS,
Bordeaux INP, LaBRI, UMR 5800
F-33400, Talence, France

Miguel Romero
miguel.romero.o@uai.cl
Faculty of Engineering and Science,
Universidad Adolfo Ibáñez
Chile

ABSTRACT

We introduce injective semantics for Conjunctive Regular Path Queries (CRPQs), and study their fundamental properties. We identify two such semantics: atom-injective and query-injective semantics, both defined in terms of injective homomorphisms. These semantics are natural generalizations of the well-studied class of RPQs under simple-path semantics to the class of CRPQs. We study their evaluation and containment problems, providing useful characterizations for them, and we pinpoint the complexities of these problems. Perhaps surprisingly, we show that containment for CRPQs becomes undecidable for atom-injective semantics, and PSPACE-complete for query-injective semantics, in contrast to the known ExpSpace-completeness result for the standard semantics. The techniques used differ significantly from the ones known for the standard semantics, and new tools tailored to injective semantics are needed. We complete the picture of complexity by investigating, for each semantics, the containment problem for the main subclasses of CRPQs, namely Conjunctive Queries and CRPQs with finite languages.

CCS CONCEPTS

• **Information systems → Query languages for non-relational engines**; • **Theory of computation** → *Database query processing and optimization (theory)*.

KEYWORDS

graph databases, regular path queries (RPQ), containment, evaluation, simple paths, injective homomorphisms

1 INTRODUCTION

Graph databases are important for many applications nowadays [2, 29]. In a nutshell, graph databases are abstracted as edge-labeled directed graphs, where nodes represent entities and labeled edges represent relations between these entities. A fundamental way to query graph databases is by finding *patterns* on the interrelation between entities. In this respect, a central querying mechanism for modern graph query languages is that of *regular path queries* (RPQs). RPQs provide a simple form of recursion tailored to discovering entities linked by paths with certain properties. These are queries of the form $x \xrightarrow{L} y$, where L is a regular expression on the alphabet of database edge labels. Such a query returns all pairs of nodes (u, v) in the database such that there is a (directed) path from u to v whose label matches L.

The closure under conjunction and existential quantification of RPQ yields what is known as *Conjunctive RPQs* (CRPQs). Indeed, CRPQs can be understood as the generalization of conjunctive queries with this simple form of recursion. CRPQs are part of SPARQL [20], the W3C standard for querying RDF data (a widespread format for graph databases). Some examples of RDF databases are well-known knowledge bases such as DBpedia and Wikidata. For example, (C)RPQs are popular for querying Wikidata [7, 23]. More generally, CRPQs are basic building blocks for querying graph-structured databases [2, 4]. They are part of G-core [1] and Cypher [17], the latter being the query language of Neo4j, which is currently one of the most popular commercial graph databases. They are also part of the ongoing standardization effort GQL for graph query languages [13, 19].

Alternative semantics. The semantics for the evaluation of an RPQ $x \xrightarrow{L} y$ as above often assumes that any, arbitrary, (directed) path from u to v is allowed as long as it satisfies the regular property L. However, there are alternative semantics in which one may restrict the path to have no repeated vertices (*a.k.a. simple path*), or no repeated edges (*a.k.a. trail*). In this way, only a *finite number* of paths need to be considered. As a matter of fact, these alternative semantics have received considerable attention both in practice and from the database theory community. Indeed, they are part of Neo4j's Cypher query language and are included in GQL as possible ways to evaluate RPQs. Furthermore, these alternative semantics have been extensively studied in the literature from the late 80s onwards [3, 12, 22, 24–26]. However, rather surprisingly, this body of research has focused mainly on RPQs, leaving the case of CRPQs under alternative semantics essentially unexplored.

Contribution. We introduce *injective* semantics for CRPQs, which generalize the simple-path semantics for RPQs, and we investigate the fundamental properties of CRPQs under these semantics. Concretely, we identify two possible natural semantics:

(1) *Atom-injective.* The first semantics is to require that each CRPQ *atom* is interpreted with the simple-path semantics of RPQs, that is, atoms of the form $x \xrightarrow{L} y$ must be mapped to *simple paths* and atoms of the form $x \xrightarrow{L} x$ must be mapped to *simple cycles*. In particular, there is no requirement that paths from different atoms be disjoint. Under this semantics, a Boolean CRPQ like $Q = \exists x, y, z \, (x \xrightarrow{(a+b)^+} y \wedge x \xrightarrow{(b+c)^+} z)$ holds true in the graph database consisting of a directed path of b's by mapping x to the first node of the path and y and z to the last node.

(2) *Query-injective.* The alternative is to consider a more restrictive semantics in which paths corresponding to different atoms must be mapped to different nodes, and hence there cannot be repeated nodes neither in paths nor between paths. This semantics

generalizes both RPQ under simple path semantics and Conjunctive Queries under injective semantics. In this case, the query Q above may only be true if the database contains two simple paths starting in the same node with the corresponding language which are disjoint (except for the origin).

We call the former *atom-injective semantics* since the "no repeated nodes" condition is required separately for each atom, and the latter *query-injective semantics* since injectivity is required for the query as a whole. The three semantics (standard, atom-injective, query-injective) form a hierarchy, where query-injective is the most restrictive and standard semantics is the least.

We first consider the *evaluation problem* for CRPQs, that is, checking whether a tuple $\bar{v}$ belongs to the results of a query Q over a particular graph database G. For standard semantics, this problem is NP-complete in *combined complexity*, that is, when both query and database are part of the input, and NL-complete in *data complexity*, that is, when the query is considered to be fixed. Under both injective semantics, the evaluation problem remains NP-complete in combined complexity, and becomes NP-complete in data complexity. This follows straight from the fact that RPQ evaluation under simple-path semantics is NP-complete, even for very simple RPQs [4, 26].

We therefore turn our attention to the *containment problem*, which is the main focus of this paper. This problem asks whether every result of a query Q_1 is also returned by a query Q_2, independently of the underlying database. Checking containment is one of the most basic static analysis tasks, and it can be a means for query optimization. The containment problem for CRPQs is known to be ExpSpace-complete [9, 16] under standard semantics, and the study of containment has been extended to queries with restricted shapes [14] or restricted regular expressions [15].

Rather surprisingly, as we show, the hierarchy of the three semantics (standard, atom-injective, query-injective) is not reflected in the complexity of the containment problem: the most restrictive semantics (query-injective) is PSpace-complete, the least restrictive one (standard) is ExpSpace-complete, and the middle one (atom-injective) is undecidable. These results for atom- and query-injective semantics are the main technical contributions of this paper. We complete the picture with a thorough study on the containment problem for the two main subclasses of CRPQs: Conjunctive Queries, and CRPQs with no Kleene star operator. We provide complexity completeness results for all possible combinations (*cf.* Figure 1). Our main results require the development of novel techniques, which yield insights on the subtle difficulties for handling static analysis under these semantics.

Organization. After a preliminary section §2, we define and characterize the evaluation and containment problems in §3 and §4, respectively. We study the containment problem for arbitrary CRPQs in §5, and for subclasses of CRPQs in §6. We conclude with §7. Omitted or sketched proofs can be found in the Appendix.

2 PRELIMINARIES

We assume familiarity with regular languages, regular expressions and non-deterministic finite automata (NFA). We often blur the distinction between a regular expression and the language it defines; similarly for NFAs.

Graph databases and paths. A *graph database* over a finite alphabet $\mathbb{A}$ is a finite edge-labeled graph $G = (V, E)$ over $\mathbb{A}$, where V is a finite set of vertices and $E \subseteq V \times \mathbb{A} \times V$ is the set of labeled edges (or simply *edges*). We write $u \xrightarrow{a} v$ to denote an edge $(u, a, v) \in E$. A *path* from u to v in a graph database $G = (V, E)$ over alphabet $\mathbb{A}$ is a (possibly empty) sequence $\pi = v_0 \xrightarrow{a_1} v_1, v_1 \xrightarrow{a_2} v_2, \ldots, v_{k-1} \xrightarrow{a_k} v_k$ of edges of G, where $k \geq 0$, $u = v_0$ and $v = v_k$. An *internal node* of such a path is any node v_i with $0 < i < k$. The *label* of π is the word $a_1 \ldots a_k \in \mathbb{A}^*$. When $k = 0$ the label of π is the empty word ε. We say that π is a *simple path* if all the nodes v_i are pairwise distinct, and a *simple cycle* if $v_0 = v_k$ and all the nodes v_i (for $i < k$) are pairwise distinct.

Conjunctive queries and homomorphisms. In the setting of graph databases, a *conjunctive query* (CQ) Q over a finite alphabet $\mathbb{A}$ is an expression $Q(x_1, \ldots, x_n) = A_1 \wedge \cdots \wedge A_m$, for $m \geq 0$, where $(x_1, \ldots, x_n)$ is a tuple of variables, and each A_i is an *atom* of the form $x \xrightarrow{a} y$, for variables x and y, and $a \in \mathbb{A}$. We denote by $vars(Q)$ the set of variables appearing in Q. We often write $Q(\bar{x})$ instead of Q to emphasize the tuple $\bar{x} = (x_1, \ldots, x_n)$ of *free variables* of Q. We assume that the free variables x_i are not necessarily distinct. The variables of Q which are not in $\{x_1, \ldots, x_n\}$ are (implicitly) existentially quantified. As usual, if $\bar{x}$ is empty, we say that the CQ Q is *Boolean*. Note that every CQ can be seen as a graph database (each atom is an edge), hence, by slightly abusing notation, we sometimes use graph database terminology for CQs.

A *homomorphism* h from a CQ $Q(\bar{x})$ to a graph database $G = (V, E)$ is a mapping from $vars(Q)$ to V such that $h(x) \xrightarrow{a} h(y)$ belongs to E for each atom $x \xrightarrow{a} y$ of Q. We say that h is *injective* if additionally we have $h(x) \neq h(y)$ for all pairs of distinct variables x and y. We write $Q \to G$ if there is a homomorphism from Q to G and $h : Q \to G$ if h is such a homomorphism. Similarly, for a tuple $\bar{v}$, we write $Q \to (G, \bar{v})$ if there is a homomorphism h from Q to G such that $h(\bar{x}) = \bar{v}$ and $h : Q \to (G, \bar{v})$ to make such h explicit. We use similar notation for injective homomorphisms, replacing $\to$ by $\xrightarrow{inj}$. Homomorphisms between CQs are essentially defined as before with the difference that free variables are mapped to free variables. That is, given two CQs $Q_1(\bar{x}_1)$, $Q_2(\bar{x}_2)$, we have $h : Q_1 \to Q_2$ if $h : Q_1 \to (G, \bar{x}_2)$, and $h : Q_1 \xrightarrow{inj} Q_2$ if $h : Q_1 \xrightarrow{inj} (G, \bar{x}_2)$, where G is the graph database denoted by Q_2.

We also work with CQs with *equality atoms*, which are queries of the form $Q(\bar{x}) = P \wedge I$, where P is a CQ (without equality atoms) and I is a conjunction of equality atoms of the form $x = y$ (the variables x and y may not belong to $vars(P)$). Again, we denote by $vars(Q)$ the set of variables appearing in Q. We define the binary relation $=_Q$ over $vars(Q)$ to be the reflexive-symmetric-transitive closure of the binary relation $\{(x, y) : x = y$ is an equality atom in $Q\}$. In other words, we have $x =_Q y$ if the equality $x = y$ is forced by the equality atoms of Q. Note that every CQ with equality atoms $Q(\bar{x}) = P \wedge I$ is equivalent to a CQ without equality atoms $Q^{\equiv}$, which is obtained from Q by collapsing each equivalence class of the relation $=_Q$ into a single variable. This transformation gives us a *canonical* renaming, which we always denote by Φ, from $vars(Q)$ to $vars(Q^{\equiv})$, defined by $\Phi(x) = C$, where C is the equivalence class containing x. In particular, the tuple of free variables of $Q^{\equiv}$ is $\Phi(\bar{x})$.

	CQ/CQ	CQ/CRPQ	CRPQ/CQ	CQ/CRPQ$^{\text{fin}}$	CRPQ$^{\text{fin}}$/CQ
standard	NP-c [10]	NP-c (†)	Π_2^p-c (‡)	NP-c (†)	Π_2^p-c (§,‡)
query-injective	NP-c (F.2)	NP-c (F.2)	Π_2^p-c (6.1,F.7)	NP-c (F.2)	Π_2^p-c (6.1,F.7)
atom-injective	NP-c (F.4)	Π_2^p-c (6.2,F.10)	Π_2^p-c (F.6,F.7)	Π_2^p-c (6.2,F.10)	Π_2^p-c (F.6,F.7)

	CRPQ/CRPQ$^{\text{fin}}$	CRPQ$^{\text{fin}}$/CRPQ	CRPQ$^{\text{fin}}$/CRPQ$^{\text{fin}}$	CRPQ/CRPQ
standard	PSPACE-c (F.8,F.9)	Π_2^p-c (§,F.10)	Π_2^p-c (§,F.10)	EXPSPACE-c ($,¢)
query-injective	PSPACE-c (F.8,5.1)	Π_2^p-c (6.1,F.10)	Π_2^p-c (6.1,F.10)	PSPACE-c (F.8,5.1)
atom-injective	undec. (5.2)	Π_2^p-c (6.2,F.10)	Π_2^p-c (6.2,F.10)	undec. (5.2)

†: [15, Thm 4.2] §: [15, Thm 4.3] ‡: [15, Thm 4.4] $: [9, Thm. 6] ¢: [16, Thm. 4.8]

Figure 1: Complexity of the containment problem under standard, query-injective, and atom-injective semantics. Numbers in brackets reference proposition/theorem numbers (some of them in Appendix F).

Conjunctive regular path queries. A *conjunctive regular path query* (CRPQ) Q over a finite alphabet $\mathbb{A}$ is an expression $Q(x_1, \ldots, x_n) = A_1 \wedge \cdots \wedge A_m$, for $m \geq 0$, where each A_i is an atom of the form $x \xrightarrow{L} y$, for variables x and y, and a regular expression L over $\mathbb{A}$. As before, we denote by $vars(Q)$ the set of variables of Q and often write $Q(\bar{x})$ instead of Q where $\bar{x} = (x_1, \ldots, x_n)$ is the tuple of (not necessarily distinct) free variables of Q. If the tuple $\bar{x}$ is empty, we say that Q is *Boolean*. The class of CRPQs extends the class of CQs and the well-studied class of *regular path queries* (RPQs). Indeed, each CQ can be seen as a CRPQ where the regular expressions are single labels from $\mathbb{A}$. On the other hand, an RPQ corresponds to a CRPQ of the form $Q(x, y) = x \xrightarrow{L} y$.

In this paper we shall consider three basic classes: the class CQ of all Conjunctive Queries, the class CRPQ of all CRPQs, and the class CRPQ$^{\text{fin}}$ of CRPQs using regular expressions with no Kleene-star (denoting finite languages). Observe that the latter corresponds to the subclass of CRPQ without recursion.

2.1 Standard, atom-injective, and query-injective semantics

We now define the *standard semantics* for CRPQs (*i.e.*, the usual semantics from the database theory literature) and we introduce the two new sorts of injective semantics.

For simplicity of exposition, we first give the semantics for CRPQ's without ε, and we then show how to expand the semantics to languages that include ε. Let Q be a CRPQ of the form $Q(\bar{z}) = x_1 \xrightarrow{L_1} y_1 \wedge \cdots \wedge x_n \xrightarrow{L_n} y_n$ and assume that no language L_i contains ε (the empty word). Given a graph database G, the *evaluation* of Q over $G = (V, E)$ under *standard semantics* (*st*-semantics for short), denoted by $Q(G)^{st}$, is the set of tuples $\bar{v}$ of nodes for which there is a mapping $\mu : vars(Q) \to V$ such that $\mu(\bar{z}) = \bar{v}$ and for each i there is a path π_i from $\mu(x_i)$ to $\mu(y_i)$ in G whose label is in L_i. The evaluation under *atom-injective semantics* (*a-inj*-semantics for short), denoted by $Q(G)^{a\text{-}inj}$, is defined similarly, but we further require that each π_i is a simple path (if $x_i \neq y_i$) or a simple cycle (if $x_i = y_i$). Finally, the evaluation under *query-injective semantics* (*q-inj*-semantics for short), denoted by $Q(G)^{q\text{-}inj}$, is similar to the atom-injective semantics (*i.e.*, each π_i must be simple), but we additionally require that μ is injective and that for every $i \neq j$ there are no internal nodes shared by π_i and π_j.

The semantics for a CRPQ Q with ε-words is defined as expected: the query Q is equivalent to a *union* of ε-free CRPQs and hence
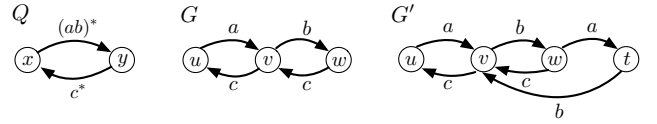

Figure 2: The CRPQ $Q(x, y)$ and graph databases G and G' from Example 2.1.

its evaluation is the union of the evaluation of these ε-free queries. More formally, for $\star \in \{st, a\text{-}inj, q\text{-}inj\}$, the semantics under $\star$-semantics of $Q(\bar{z}) = x \xrightarrow{L} y \wedge Q'$, where L contains ε and Q' is a CRPQ, is the union of the set of tuples given by the $\star$-evaluation of $Q(\bar{z}) = x \xrightarrow{L \setminus \{\varepsilon\}} y \wedge Q'$ and by the $\star$-evaluation of $Q(\bar{z}[x/y]) = Q'[x/y]$, where $X[x/y]$ is the result of replacing every occurrence of variable x with variable y in X.

Remark 2.1. *The three semantics form a hierarchy. In particular, for every CRPQ Q and every graph database G, we have $Q(G)^{q\text{-}inj} \subseteq Q(G)^{a\text{-}inj} \subseteq Q(G)^{st}$. The converse inclusions do not hold in general.*

Example 2.1. Consider the CRPQ $Q(x, y) = x \xrightarrow{(ab)^*} y \wedge y \xrightarrow{c^*} x$ and the graph database G from Figure 2. Observe that $(u, w) \in Q(G)^{a\text{-}inj}$ but $(u, w) \notin Q(G)^{q\text{-}inj}$. On the other hand, it is easy to check that $Q(G)^{st} = Q(G)^{a\text{-}inj}$. The graph database G' from Figure 2 provides a separation of the three semantics. Indeed, as before $Q(G')^{a\text{-}inj} \not\subseteq Q(G')^{q\text{-}inj}$, but additionally we have $(u, v) \in Q(G')^{st}$ and $(u, v) \notin Q(G')^{a\text{-}inj}$.

2.2 Characterizing evaluation

We state the semantics defined above in terms of restrictive notions of homomorphisms based on injectivity. This is based on the key notion of *expansion* of a CRPQ (*a.k.a.* canonical database), which will become useful for the technical developments of the next sections.

For any atom $x \xrightarrow{L} y$ of a CRPQ Q and $w \in L$, the *w-expansion* of $x \xrightarrow{L} y$ is the Boolean CQ with equality atoms of the form (i) $P = x \xrightarrow{a_1} z_1 \wedge z_1 \xrightarrow{a_2} z_2 \wedge \cdots \wedge z_{k-1} \xrightarrow{a_k} y$ if $w \neq \varepsilon$, such that the z_i are fresh new variables, or of the form (ii) $P = (y = z)$ if $w = \varepsilon$. We usually write $x \xrightarrow{w} y$ to denote such a w-expansion, with $w = a_1 \cdots a_k$. An *expansion* of $x \xrightarrow{L} y$ is a w-expansion for some $w \in L$. An *expansion profile* of the CRPQ Q is any function

φ mapping each atom of Q to an expansion thereof. An *expansion* of the CRPQ $Q(\bar{x}) = A_1 \wedge \cdots \wedge A_m$ is a CQ $E(\bar{y})$ for which there is an expansion profile φ of Q such that $E = \widetilde{E}^{\equiv}$, where $\widetilde{E}$ is the CQ with equality atoms defined by $\widetilde{E}(\bar{x}) = \varphi(A_1) \wedge \cdots \wedge \varphi(A_m)$. We denote by $\text{Exp}(Q)$ the set of all expansions of the CRPQ Q. Intuitively, an expansion of Q is obtained by expanding each atom of Q and then collapsing equivalent variables. For example, one possible expansion of the query $Q(x, y) = x \xrightarrow{(ab)^*} y \wedge y \xrightarrow{c^*} x$ from Figure 2 is $E_1(x, x) = x \xrightarrow{a} z \wedge z \xrightarrow{b} x$ through the expansion profile mapping $x \xrightarrow{(ab)^*} y$ to ab and $y \xrightarrow{c^*} x$ to ε, and another expansion $E_2(x, y) = x \xrightarrow{a} z \wedge z \xrightarrow{b} y \wedge y \xrightarrow{c} x$ mapping the atoms to ab and c respectively.

Standard and query-injective semantics. The standard and q-inj semantics can be rephrased as follows.

Proposition 2.2. *Let Q be a CRPQ and G be a graph database. Then $Q(G)^{st}$ [resp. $Q(G)^{q\text{-}inj}$] is the set of tuples $\bar{v}$ of nodes for which there is $E \in \text{Exp}(Q)$ such that $E \to (G, \bar{v})$ [resp. $E \xrightarrow{inj} (G, \bar{v})$].*

Atom-injective semantics. The atom-injective semantics corresponds to the less restrictive alternative that an arbitrary homomorphism can be allowed as long as it is injective when restricted to the expansions of each atom. Let $E(\bar{y})$ be an expansion of a CRPQ $Q(\bar{x})$, φ be an expansion profile producing E, and $\widetilde{E}(\bar{x}) = \varphi(A_1) \wedge \cdots \wedge \varphi(A_m)$ be the associated CQ with equality atoms (in particular $E = \widetilde{E}^{\equiv}$). Let $\Phi : \text{vars}(\widetilde{E}) \to \text{vars}(E)$ be the canonical renaming. We say that two variables $x, y \in \text{vars}(E)$ are φ-*atom-related* if there is some atom expansion $\varphi(A_i)$ containing some x', y' such that $\Phi(x') = x$ and $\Phi(y') = y$.

We now define a notion of injective homomorphism tailored to CRPQ expansions. We say that h is an *atom-injective homomorphism* from an expansion E of a CRPQ Q to a graph database G mapping free variables to $\bar{v}$ if $h : E \to (G, \bar{v})$ and there is an expansion profile φ producing E such that $h(x) \neq h(y)$ for every pair of distinct φ-atom-related variables x and y. We write $E \xrightarrow{a\text{-}inj} (G, \bar{v})$ if such an h exists. Atom-injective homomorphisms from E to CQs are defined in the obvious way.

Proposition 2.3. *Let Q be a CRPQ and G be a graph database. Then $Q(G)^{a\text{-}inj}$ is the set of tuples $\bar{v}$ of nodes for which there is $E \in \text{Exp}(Q)$ such that $E \xrightarrow{a\text{-}inj} (G, \bar{v})$.*

3 THE EVALUATION PROBLEM

The decision problem associated to evaluation is the *evaluation problem* for a class C of CRPQs and a semantics $\star \in \{st, a\text{-}inj, q\text{-}inj\}$.

Problem	Evaluation problem for C under $\star$-semantics
Given	A graph database G, a query $Q(\bar{x}) \in C$, and a tuple $\bar{v}$ of nodes.
Question	Is $\bar{v} \in Q(G)^{\star}$?

The evaluation problem is NP-complete under injective semantics, as it is the case under standard semantics.

Proposition 3.1. *The evaluation problem for CRPQ and CQ is NP-complete in combined complexity, for all semantics.*

Proof. The lower bound follows by an easy reduction from the injective-homomorphism testing problem, also known as the subgraph isomorphism problem, which is a well-known NP-complete problem. [11, 18]. Indeed, a Boolean CQ Q maps injectively to G iff $Q(G)^{q\text{-}inj} \neq \emptyset$ iff $Q^+(G^+)^{a\text{-}inj} \neq \emptyset$, where G^+ [resp. Q^+] is the result of adding, for a fresh symbol R, an R-edge between every pair of vertices [resp. an R-atom between every pair of variables].

The upper bound is a consequence of the polynomial-sized witness property. That is, if $Q \in \text{CRPQ}$, and $\bar{v} \in Q(G)^{\star}$, then there exists an expansion E of Q such that $E \xrightarrow{inj} (G, \bar{v})$ if $\star = q\text{-}inj$ and $E \xrightarrow{a\text{-}inj} (G, \bar{v})$ if $\star = a\text{-}inj$. In either case, E is linear in G and Q. One can then guess such an expansion and check the existence of the corresponding homomorphism. □

The data complexity for the alternative semantics (*i.e.*, when the query is considered to be of constant size) is also NP-complete, since evaluation of RPQs under simple path semantics is NP-complete, even for very simple regular expressions [26]:

Proposition 3.2. *The evaluation problem for CRPQ is NP-complete in data complexity, for atom-injective and query-injective semantics.*

The RPQs which can be evaluated efficiently in data complexity have been characterized via a trichotomy result: they can be either NP-complete, NL-complete, or in AC^0 [3, Theorem 2]. The generalization of this result to CRPQs under injective semantics seems highly non-trivial, and in particular it would necessitate a comprehensive understanding of the query equivalence problem, which is the focus of the next sections.

4 THE CONTAINMENT PROBLEM

A CRPQ Q_1 is *contained* in a CRPQ Q_2 under $\star$-semantics, denoted by $Q_1 \subseteq_{\star} Q_2$, if $Q_1(G)^{\star} \subseteq Q_2(G)^{\star}$ for every graph database G. We define the *containment problem*, which is parameterized by classes C_1 and C_2 of CRPQs as well as the semantics used (standard, query-injective, or atom-injective).

Problem	C_1/C_2 containment problem under $\star$-semantics
Given	CRPQs $Q_1 \in C_1$ and $Q_2 \in C_2$.
Question	Does $Q_1 \subseteq_{\star} Q_2$ hold?

Under standard semantics, all combinations among CQ, CRPQ and CRPQ$^{\text{fin}}$ have been studied and are decidable. In particular:

Theorem 4.1. *[9, 16] The CRPQ/CRPQ containment problem under standard semantics is ExpSpace-complete.*

We will dedicate the rest of the paper to study the situation for injective semantics. We show that one injective semantics becomes undecidable while the other becomes better behaved computationally under standard complexity theoretic assumptions (*cf.* the CRPQ/CRPQ column of Figure 1).

4.1 Characterizing containment

For the standard semantics, it is well-known that containment of CRPQs can be characterized in terms of expansions:

Proposition 4.2. *[9] Let Q_1 and Q_2 be CRPQs. Then $Q_1 \subseteq_{st} Q_2$ iff for every $E_1 \in \text{Exp}(Q_1)$ there is $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \to E_1$.*

A similar characterization holds for query-injective semantics:

PROPOSITION 4.3. *Let Q_1 and Q_2 be CRPQs. Then $Q_1 \subseteq_{q\text{-}inj} Q_2$ iff for every $E_1 \in \text{Exp}(Q_1)$ there is $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$.*

As it turns out, the previous characterization does not work for atom-injective semantics (replacing $\xrightarrow{inj}$ by $\xrightarrow{a\text{-}inj}$). In this case, the space of expansions of Q_1 is not enough and we need to check Q_2 against a *larger* space of expansions of Q_1 we define below.

An *atom-injective-expansion* (*a-inj*-expansion for short) of a CRPQ $Q(\bar{x})$ is a CQ $F(\bar{y})$ for which there is a CQ with equality atoms $\widetilde{F}(\bar{z}) = E(\bar{z}) \wedge J$ such that (i) $F = \widetilde{F}^{\equiv}$, (ii) $E(\bar{z})$ is an expansion of Q produced by some expansion profile φ, and (iii) J is a conjunction of equality atoms $x' = y'$, for variables $x', y' \in vars(E)$, such that for every pair of distinct φ-atom-related variables x, y in E, we have $x \neq_{\widetilde{F}} y$. We denote by $\text{Exp}^{a\text{-}inj}(Q)$ the set of all *a-inj*-expansions of Q. The intuition is that these types of expansions are obtained from an ordinary expansion of Q by identifying some pairs of variables which are not atom-related (the identifications are given by J). We use this for the following useful result:

LEMMA 4.4. *Let Q be a CRPQ, E' be a CQ, G be a graph database, and $\bar{v}$ be a tuple of nodes. The following are equivalent:*

(1) *There is $E \in \text{Exp}(Q)$ s.t. $E \xrightarrow{a\text{-}inj} (G, \bar{v})$ [resp. $E \xrightarrow{a\text{-}inj} E'$].*
(2) *There is $F \in \text{Exp}^{a\text{-}inj}(Q)$ s.t. $F \xrightarrow{inj} (G, \bar{v})$ [resp. $F \xrightarrow{inj} E'$].*

As a corollary of Lemma 4.4 we obtain an alternative definition of atom-injective semantics:

COROLLARY 4.5. *Let Q be a CRPQ, G be a graph database and $\bar{v}$ be a tuple of nodes. Then $\bar{v} \in Q(G)^{a\text{-}inj}$ if and only if there is $F \in \text{Exp}^{a\text{-}inj}(Q)$ such that $F \xrightarrow{inj} (G, \bar{v})$.*

We now give our characterization of atom-injective containment:

PROPOSITION 4.6. *For every pair Q_1, Q_2 of CRPQs, the following are equivalent:*

(1) $Q_1 \subseteq_{a\text{-}inj} Q_2$.
(2) *For every $F_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$ there is $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{a\text{-}inj} F_1$.*
(3) *For every $F_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$ there is $F_2 \in \text{Exp}^{a\text{-}inj}(Q_2)$ such that $F_2 \xrightarrow{inj} F_1$.*

From the characterizations above, for every pair of CRPQs Q_1, Q_2, we have that $Q_1 \subseteq_{q\text{-}inj} Q_2$ implies $Q_1 \subseteq_{st} Q_2$ and that $Q_1 \subseteq_{a\text{-}inj} Q_2$ implies $Q_1 \subseteq_{st} Q_2$ (while the converse implications do not hold in general). In contrast, and in spite of the hierarchy between the semantics, there is no such implication between query-injective and atom-injective containment, as the following example shows.

Example 4.7. Consider the Boolean CRPQs $Q_1 = x \xrightarrow{a} y \wedge y \xrightarrow{b} z$, $Q_2 = x \xrightarrow{ab} y$, $Q_1' = x \xrightarrow{a} y \wedge x \xrightarrow{b} y$ and $Q_2' = x \xrightarrow{a} y \wedge x' \xrightarrow{b} y'$. We have $Q_1' \subseteq_{a\text{-}inj} Q_2'$ (and $Q_1' \subseteq_{st} Q_2'$) but $Q_1' \not\subseteq_{q\text{-}inj} Q_2'$ as there cannot be an injective homomorphism from the unique expansion of Q_2' to the unique expansion of Q_1'. On the other hand, we have $Q_1 \subseteq_{q\text{-}inj} Q_2$ (and $Q_1 \subseteq_{st} Q_2$) but $Q_1 \not\subseteq_{a\text{-}inj} Q_2$. Indeed, we can take the *a-inj*-expansion F of Q_1 obtained from $x \xrightarrow{a} y \wedge y \xrightarrow{b} z$

by identifying x and z. Then, there cannot be an atom-injective homomorphism from the unique expansion of Q_2 to F.

In view of the characterizations above, in the sequel we will sometimes write *st-expansions* or *q-inj-expansions* to denote a (normal) expansion. For $\star \in \{st, q\text{-}inj, a\text{-}inj\}$, we say that $E_1(\bar{y})$ is a *counter-example* for $\star$-semantics if E_1 is a $\star$-expansion of Q_1 such that $\bar{y} \notin Q_2(E_1)^{\star}$ (recall that any CQ, in particular E_1, can be seen as a graph database). Note that the latter condition $\bar{y} \notin Q_2(E_1)^{\star}$ is equivalent to the non-existence of a (normal) expansion E_2 of Q_2 such that either (a) $E_2 \to E_1$ if $\star = st$; (b) $E_2 \xrightarrow{inj} E_1$ if $\star = q\text{-}inj$; or (c) $E_2 \xrightarrow{a\text{-}inj} E_1$ if $\star = a\text{-}inj$. Hence, $Q_1 \not\subseteq_\star Q_2$ if, and only if, there exists a counter-example for $\star$-semantics.

5 CONTAINMENT FOR UNRESTRICTED CRPQS

In this section we study the CRPQ/CRPQ containment problem under query-injective and atom-injective semantics. We show that the former is in PSPACE while the latter is undecidable. Both proofs are non-trivial and provide novel insights on how the semantics can be exploited for static analysis problems: In the first case by reducing the space needed from exponential to polynomial, and in the second case by enforcing counter-examples to witness solutions of the PCP problem, through an intricate encoding.

The CRPQ/CRPQ containment problem for standard semantics is in EXPSPACE [16] and all proofs [9, 14, 16] of which we are aware reduce the problem to test containment or universality on exponentially-sized NFA's, encoding the set of expansions or the set of (non-)counter-examples. The PSPACE bound for query-injective semantics is, however, quite different in nature, and uses some exponential number of polynomial-sized certificates to ensure that the containment holds.

THEOREM 5.1. *The CRPQ/CRPQ containment problem under query-injective semantics is in PSPACE.*

PROOF SKETCH(FULL PROOF IN APPENDIX C). Let Q_1, Q_2 be CRPQs; we want to test $Q_1 \subseteq_{q\text{-}inj} Q_2$. We only give a high-level description of the proof due to space constraints. We will work with polynomial-sized 'abstractions' of expansions of Q_1. These abstractions contain, for each atom A of Q_1, all the information on how the languages of Q_2 can be mapped into it. For example, it includes the information "*there is a partial run from state q to state q' of the NFA $\mathcal{A}_L$ of language L from Q_2 reading the expansion word of A*", or "*there is a partial run from the initial state of $\mathcal{A}_L$ to q reading some suffix of the expansion word of A*". Such an abstraction contains all the necessary information needed to retain from an expansion to check whether it is a counter-example. Indeed, any expansion having the same abstraction as a counter-example will be a counter-example.

In order to test whether an abstraction α abstracts a counter-example, we need to consider all possible ways of injectively mapping an expansion of Q_2 to an expansion of Q_1. We call this a *morphism type*, which contains the information of where each atom expansion of Q_2 is mapped to. For example, we can have the information that the path to which the expansion of atom A of Q_2 is mapped starts at some internal node of the expansion of atom $A_1 = x \xrightarrow{L_1} y$ of Q_1 then arrives to variable y with state q and
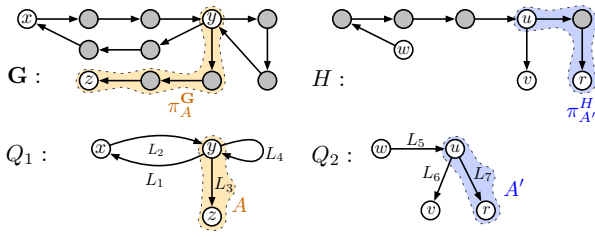
Figure 3: Example of definition of **G** and morphism type (H, h) from Q_1, Q_2. In this case, the injective morphism h from H to **G** maps each node of H to the node in the same position on **G** (*e.g.*, the lower-right node r of H maps to the lower-right node of **G**).

continues reading the full expansion of atom $A_2 = y \xrightarrow{L_2} z$ arriving to variable z with state q', and it ends its journey by reading a prefix of the expansion of atom $A_3 = z \xrightarrow{L_3} t$ arriving to a final state at some internal node. For each morphism type, we can check if it is *compatible* with an abstraction by checking, for example, that α indeed contains the information of having a partial run from q' to a final state reading a prefix of the expansion of A_3.

More concretely, consider the directed graph **G** consisting of replacing each atom $A = x \xrightarrow{L} y$ of Q_1 with a path $\pi_A^{\mathbf{G}}$ of length 3 (*i.e.*, adding two new internal vertices). A *morphism type* from Q_2 to Q_1 is a pair (H, h) such that $h : H \xrightarrow{inj} \mathbf{G}$ and H is a graph resulting from replacing each atom $A = x \xrightarrow{L} y$ of Q_2 with a path π_A^H from x to y. Figure 3 has an example of a morphism type. By injectivity, the size of H in any morphism type is linearly bounded on Q_1.

A morphism type (H, h) is *compatible* with an abstraction α if there is a mapping λ from the internal nodes of paths π_A^H to states of A, such that for every atom $A = x_1 \xrightarrow{L} x_2$ of Q_1, all the expected properties must hold. For example, if there is an atom A' of Q_2 and an infix π of the path $\pi_{A'}^H$ with $h(\pi) = \pi_A^{\mathbf{G}}$, then the abstraction α ensures that there is a run of $\mathcal{A}_{A'}$ from $\lambda(src(\pi))$ to $\lambda(tgt(\pi))$, where $\mathcal{A}_{A'}$ is the NFA of the language of A', and $src(\pi)$ and $tgt(\pi)$ are the first and last nodes of π, respectively. Or, as another example, one must also check that if there is an atom A' of Q_2 and a suffix π of $\pi_{A'}^H$ with $h(\pi)$ being a prefix of $\pi_A^{\mathbf{G}}$, then α ensures that there is a run from $\lambda(src(\pi))$ to some final state in $\mathcal{A}_{A'}$ on the prefix of the expansion of A. There are actually many other possible cases (17 in total), but each of these cases can be easily checked with the information compiled in an abstraction.

The key property of compatibility is that it captures whether an abstraction contains a Q_1-expansion which is a counter-example:

Claim 5.1. *The following are equivalent:*

(1) *There is a morphism type compatible with an abstraction α;*

(2) *for every expansion $E_1 \in \text{Exp}(Q_1)$ with abstraction α there exists some expansion $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$;*

(3) *there is an expansion $E_1 \in \text{Exp}(Q_1)$ with abstraction α and an expansion $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$.*
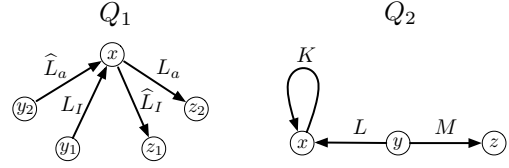

Figure 4: The general structure of Boolean CRPQs Q_1 and Q_2 from the reduction.

Finally, the PSpace algorithm guesses a mapping α from the atoms of Q_1 to subsets of P, checks that α is an abstraction of Q_1, and checks that there is no morphism type (H, h) which is compatible with α. Due to the Claim above, if the algorithm succeeds, then any expansion E_1 of Q_1 is a counter-example, and thus $Q_1 \not\subseteq_{q\text{-}inj} Q_2$; otherwise, for every expansion E_1 with abstraction α there is a compatible morphism type, which means that E_1 is not a counter-example and hence $Q_1 \subseteq_{q\text{-}inj} Q_2$. □

On the other hand, the CRPQ/CRPQ containment problem for atom-injective semantics becomes undecidable. Remarkably, the bound holds even when the right-hand side query has no infinite languages, and both queries are of very simple shape (*cf.* Figure 4).

THEOREM 5.2. *The* CRPQ/CRPQ *and* CRPQ/CRPQ$^{\text{fin}}$ *containment problems under atom-injective semantics are undecidable.*

PROOF SKETCH (FULL PROOF IN APPENDIX D). We reduce from the *Post Correspondence Problem* (PCP), a well-known undecidable problem. An instance of the PCP is a sequence of pairs $(u_1, v_1), \ldots, (u_\ell, v_\ell)$, where u_i and v_i are non-empty words over an alphabet Σ. The goal is to decide whether there is a *solution*, that is, a sequence $i_1, \ldots, i_k$ of indices from $\{1, \ldots, \ell\}$, with $k \geq 1$, such that the words $u_{i_1} \cdots u_{i_k}$ and $v_{i_1} \cdots v_{i_k}$ coincide.

We provide a high-level description of the reduction. The idea is to construct Boolean CRPQs Q_1 and Q_2 such that the PCP instance $(u_1, v_1), \ldots, (u_\ell, v_\ell)$ has a solution if and only if $Q_1 \not\subseteq_{a\text{-}inj} Q_2$. In particular, the PCP instance has a solution if and only if there exists a counterexample for *a-inj*-semantics, *i.e.*, an *a-inj*-expansion F of Q_1 such that there is no expansion $E \in \text{Exp}(Q_2)$ with $E \xrightarrow{a\text{-}inj} F$. The general structure of Q_1 is shown in Figure 4. We have a "middle" variable x, two "incoming" atoms and two "outgoing" atoms:

$$y_1 \xrightarrow{L_I} x \wedge y_2 \xrightarrow{\widehat{L_a}} x \wedge x \xrightarrow{\widehat{L_I}} z_1 \wedge x \xrightarrow{L_a} z_2$$

Words in the languages L_I and $\widehat{L_I}$ encode sequences of indices from $\{1, \ldots, \ell\}$, using special symbols from $\mathbb{I} := \{I_1, \ldots, I_\ell\}$ and $\widehat{\mathbb{I}} := \{\widehat{I_1}, \ldots, \widehat{I_\ell}\}$, respectively. On the other hand, words from L_a and $\widehat{L_a}$ encode sequences of words from $\{u_1, \ldots, u_\ell\}$ and $\{v_1, \ldots, v_\ell\}$, using symbols from the PCP alphabet Σ and $\widehat{\Sigma} := \{\widehat{a} : a \in \Sigma\}$, respectively. In the four languages, we have some extra symbols to make the reduction work. We stress that the finite alphabet used for the CRPQ Q_1 (and also for Q_2) depends on the PCP instance.

We are interested in a particular type of *a-inj*-expansions of Q_1 that we call *well-formed*. The idea is that well-formed *a-inj*-expansions correspond to solutions of the PCP instance. In particular, if there is a well-formed *a-inj*-expansion of Q_1 then there is a
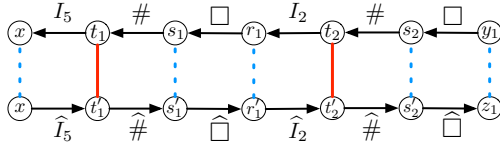
Figure 5: Example of the $I\text{-}\widehat{I}$-condition of well-formed expansions of Q_1. We show the expansions $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{\widehat{w_I}} z_1$ of the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L_I}} z_1$, respectively. The words $w_I = \square \# I_2 \square \# I_5$ and $\widehat{w_I} = \widehat{I_5} \widehat{\#} \widehat{\square} \widehat{I_2} \widehat{\#} \widehat{\square}$ encode the sequence of indices $5, 2$. Dotted blue lines indicate pairs of equal variables while red lines indicate distinct variables. We have some extra symbols $\#, \widehat{\#}, \square, \widehat{\square}$.

solution to the PCP instance and vice versa. We then construct Q_2 such that an *a-inj*-expansion of Q_1 is well-formed if and only if it is a counterexample for $Q_1 \subseteq_{a\text{-}inj} Q_2$.

Let F be an *a-inj*-expansion of Q_1 such that $F = \widetilde{F}^{\equiv}$ for $\widetilde{F} = E \wedge J$ (here $E \in \text{Exp}(Q_1)$ and J are the equality atoms). The *a-inj*-expansion F is *well-formed* if it satisfies four structural conditions we call $I\text{-}\widehat{I}$-, $I\text{-}a$-, $\widehat{a}\text{-}\widehat{I}$-, and $\widehat{a}\text{-}a$-condition. Intuitively, the $I\text{-}\widehat{I}$-condition requires that the words $w_I \in L_I$ and $\widehat{I} \in \widehat{L_I}$ chosen in the expansion E encode the *same* sequence of indices from $\{1, \ldots, \ell\}$. On the other hand, the $I\text{-}a$-condition ensures that the word $w_a \in L_a$, chosen in the expansion E, encodes a sequence of words from $\{u_1, \ldots, u_\ell\}$ *according* to the sequence encoded in $w_I \in L_I$. Similarly, the $\widehat{a}\text{-}\widehat{I}$-condition requires that the chosen word $\widehat{w_a} \in \widehat{L_a}$ encodes a sequence of words from $\{v_1, \ldots, v_\ell\}$ according to $\widehat{w_I} \in \widehat{L_I}$. Finally, the $\widehat{a}\text{-}a$-condition requires that the chosen words $w_a \in L_a$ and $\widehat{w_a} \in \widehat{L_a}$ "coincide" after removing the $\widehat{\cdot}$ superscripts from $\widehat{w_a}$ and focusing on the symbols from Σ. In other words, the $\widehat{a}\text{-}a$-condition ensures that the sequence of indices chosen by w_I (and $\widehat{w_I}$) is actually a solution to the PCP. In the four cases, we additionally need to require some conditions on the equality atoms J to make the reduction work (see Figure 5 for an example).

The key property of well-formedness is that it can be characterized in terms of the non-existence of a finite number of simple cycles and simple paths having certain labels. Let us illustrate this for the case of the $I\text{-}\widehat{I}$-condition and the expansion F of Figure 5. The forbidden labels for simple cycles are given by the finite language $K_{I\widehat{I}} = \mathbb{I}\widehat{\mathbb{I}}$. In the case of simple paths, these are given by $M_{I\widehat{I}} = \sum_{i \neq j} I_i \widehat{I_j} + \widehat{\mathbb{I}}\# + \widehat{\#}\mathbb{I} + \#\mathbb{I}\widehat{\mathbb{I}}\# + \square\widehat{\square}$. We have that an *a-inj*-expansion F of Q_1 satisfies the $I\text{-}\widehat{I}$-condition if and only if F does not contain simple cycles with labels in $K_{I\widehat{I}}$ nor simple paths with labels in $M_{I\widehat{I}}$.

To see the backward direction, note that t_1 and t_1' cannot be identified, otherwise we have a simple cycle from t_1 to itself with label in $\mathbb{I}\widehat{\mathbb{I}} \subseteq K_{I\widehat{I}}$. Now, the symbols I_5 and $\widehat{I_5}$ need to correspond to the same index from $\{1, \ldots, \ell\}$; otherwise we find a simple path from t_1 to t_1' with label in $\sum_{i \neq j} I_i \widehat{I_j} \subseteq M_{I\widehat{I}}$. To see the identification between s_1 and s_1', note first that t_1 and s_1' cannot be identified, as this would imply a simple path from t_1' to x with label in $\widehat{\#}\mathbb{I} \subseteq M_{I\widehat{I}}$. Analogously, we have that t_1' and s_1 cannot be identified. This implies that s_1 and s_1' are actually identified, otherwise we have a

simple path from s_1 to s_1' with label in $\#\mathbb{I}\widehat{\mathbb{I}}\# \subseteq M_{I\widehat{I}}$. Finally, r_1 and r_1' are identified, otherwise there would be a simple path from r_1 to r_1' with label in $\square\widehat{\square} \subseteq M_{I\widehat{I}}$. Note that we can repeat this argument from "left-to-right" starting from $r_i = r_i'$ instead of x, and obtain the $I\text{-}\widehat{I}$-condition. In order to ensure that the words w_I and $\widetilde{w_I}$ have the same length, we need to slightly modify the construction of $Q_1, K_{I\widehat{I}}$ and $M_{I\widehat{I}}$ (see Appendix D for details). The forward direction follows directly from the definition of the $I\text{-}\widehat{I}$-condition.

Since all the four conditions can be characterized via forbidden finite sets of simple cycles and paths, it is possible to write two CRPQs from CRPQ^{fin} of the form $Q_2^{\cup} = x \xrightarrow{K^{\cup}} x$ and $Q_2^{\rightarrow} = y \xrightarrow{M^{\rightarrow}} z$ such that for every *a-inj*-expansion F of Q_1, F is well-formed if and only if $Q_2^{\cup} \vee Q_2^{\rightarrow}(F)^{a\text{-}inj} = \emptyset$, where $Q_2^{\cup} \vee Q_2^{\rightarrow}$ is the *union* of both CRPQs. In particular, there is a solution to the PCP instance if and only if $Q_1 \not\subseteq_{a\text{-}inj} Q_2^{\cup} \vee Q_2^{\rightarrow}$. We finally show how to simulate the union $Q_2^{\cup} \vee Q_2^{\rightarrow}$ with a single query $Q_2 \in \text{CRPQ}^{\text{fin}}$ as in Figure 4. $\square$

6 CONTAINMENT FOR CRPQ SUBCLASSES

With the two previous results in place for the containment of unconstrained CRPQs, we now explore the C_1/C_2 containment problem under all the possible semantics, where C_1 and/or C_2 belong to simpler classes of queries, namely either Conjunctive Queries or CPRQs with no Kleene star (and hence with finite languages).

In many cases one can apply or adapt previously established techniques or reductions. There are, however, two noticeable exceptions: the lower bounds for CRPQ^{fin}/CQ under query-injective semantics and for CQ/CRPQ^{fin} under atom-injective semantics. We highlight only these two results. The remaining proofs can be found in Appendix F. In particular, as mentioned in Section 4, almost all the results for the standard semantics follow from previous work (in particular [9, 15, 16]).

THEOREM 6.1. *The* CRPQ^{fin}/CQ *containment problem under query-injective semantics is* Π_2^p-*hard.*

PROOF. We show that even when the languages of the left-hand query are unions of alphabet symbols, Π_2^p hardness still holds. We show a reduction from the following problem on graphs, which is known to be Σ_2^p-complete [28, Theorem 5], to non-containment. For a graph G let $V(G)$ denote its sets of vertices and let $G|_{V'}$ denote the subgraph induced by $V' \subseteq V(G)$.

PROBLEM	Generalized Two-Coloring Problem (GCP$_2$)
GIVEN	An undirected graph G, a number $n \in \mathbb{N}$ (in unary).
QUESTION	Is there a partition $V_1 \dot\cup V_2 = V(G)$ s.t. neither $G\|_{V_1}$ nor $G\|_{V_2}$ contains an n-vertex clique as subgraph?

We will produce two Boolean queries Q_1, Q_2 over the alphabet $\mathbb{A} = \{E, 1, 2, \#\}$ such that: (1) $Q_1 \not\subseteq_{q\text{-}inj} Q_2$ iff the GCP$_2$ instance is positive; and (2) Q_2 is a CQ, and every language of Q_1 is a set of one-letter words. Consider the input graph G, and the associated CQ Q_G on over the alphabet $\{E\}$, where for each edge $\{u, v\}$ in G we have atoms $u \xrightarrow{E} v \wedge v \xrightarrow{E} u$ in Q_G. Similarly, let K_n be the CQ associated to the n-vertex clique. For a CQ Q and $i \in \{1, 2\}$, let us
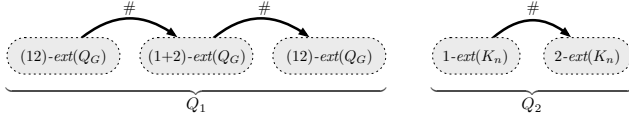
Figure 6: Definition of Q_1 and Q_2 in terms of G and n for the reduction of Theorem 6.1. The #-labeled thick arrows denote that there is an atom $x \xrightarrow{\#} y$ for each variable x from the source query to each variable y of the target query.

define $i\text{-}ext(Q)$ [resp. $(1+2)\text{-}ext(Q)$; $(12)\text{-}ext(Q)$] as the extension of Q by adding one atom $x \xrightarrow{i} x$ [resp. one atom $x \xrightarrow{1+2} x$; two atoms $x \xrightarrow{1} x \wedge x \xrightarrow{2} x$] for every variable $x \in \mathit{vars}(Q)$. We define Q_1, Q_2 as in Figure 6. On the one hand, if $Q_1 \nsubseteq_{q\text{-}inj} Q_2$ there must be some expansion E of Q_1 which is a counter-example. From E we can derive the partitioning $V_1 \mathbin{\dot\cup} V_2$ of $V(G)$ where V_i is the set of vertices labeled with an i-loop in the middle gadget of Q_1. Now observe that, for every i, K_n is not injectively mapped to $G|_{V_i}$, as otherwise we would have that $i\text{-}ext(K_n) \xrightarrow{inj} E$, implying $Q_2 \xrightarrow{inj} E$ and contradicting that E is a counter-example. This means that the GCP$_2$ instance is positive. On the other hand, if there is a partitioning $V_1 \mathbin{\dot\cup} V_2$ of $V(G)$ avoiding the n-clique as a subgraph, then the corresponding expansion E of Q_1 (by choosing to have an i-loop for each node $x \in V_i$) is such that Q_2 cannot be injectively mapped to E; in other words showing that E is a counter-example and thus $Q_1 \nsubseteq_{q\text{-}inj} Q_2$. □

THEOREM 6.2. *The* CQ/CRPQ$^{\mathrm{fin}}$ *containment problem under atom-injective semantics is* Π_2^p*-hard.*

PROOF SKETCH(FULL PROOF IN APPENDIX E). We show that even when all languages on the right-hand side are of the form $\{w\}$ with $|w| \le 2$ we have Π_2^p-hardness for containment. For this, we show how to adapt the proof of Π_2^p-hardness of [15, Theorem 4.3], which shows Π_2^p-hardness for CRPQ$^{\mathrm{fin}}$/CQ containment for the standard semantics. We reduce from $\forall\exists$-QBF (*i.e.*, Π_2-Quantified Boolean Formulas). Let $\Phi = \forall x_1, \ldots, x_n \, \exists y_1, \ldots, y_\ell \, \varphi(x_1, \ldots, x_n, y_1, \ldots, y_\ell)$ be an instance of $\forall\exists$-QBF such that φ is quantifier-free and in 3-CNF. We construct boolean queries Q_1 and Q_2 such that $Q_1 \subseteq_{a\text{-}inj} Q_2$ if, and only if, Φ is satisfiable.

The query Q_1 is defined in Figure 7, over the alphabet of labels $\{a, x_1, \ldots, x_n, y_1, \ldots, y_\ell, t, f, r\}$. We now explain how we define Q_2, over the same alphabet. Every clause of Φ is represented by a sub-query in Q_2, as depicted in Figure 7. All nodes with identical label ($y_{1,t}$ and $y_{1,f}$ in gadgets D, E) in the figures are the same node. Note that for every clause and every existentially quantified literal y_i therein we have one node named $y_{i,tf}$ in Q_2. The E-gadget is designed such that every represented literal can be homomorphically embedded, while exactly one literal has to be embedded in the D-gadget.

The intuitive idea is that the valuation of the x-variables is given by the $a\text{-}inj$-expansion E_1 of Q_1, whether the two nodes under x incident to t are equal or not: if they are equal this corresponds to a *false* valuation, otherwise a *true* valuation. On the other hand, the
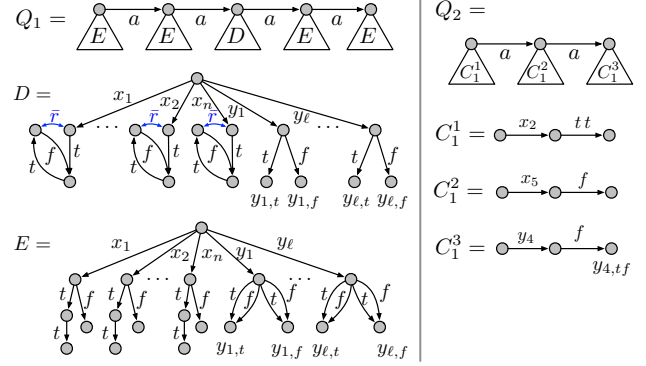

Figure 7: Left: Query Q_1 used in Theorem 6.2 and the gadgets D, and E used in its definition. The $\bar{r}$ blue edges depict the edges of the complement of r (*i.e.*, the edges which are *not* in relation r) for clarity. **Right:** Example of Q_2 for $\varphi = (x_2 \vee \neg x_5 \vee \neg y_4)$.

valuation of the y-variables is given by the homomorphism of an expansion of Q_2 into E_1 (i.e., whether the corresponding node is mapped to the node $y_{\_}t$ or $y_{\_}f$). The homomorphism of y-variables across several clauses has to be consistent, as all clauses share the same nodes $y_{\_}tf$, which uniquely get mapped either into $y_{\_}t$ or $y_{\_}f$. Hence, when the formula Φ is satisfiable, for any assignment to the variables $\{x_i\}$ (given by the choice of t/f edges in D), there is a mapping from $y_{\_}tf$ to one of $y_{\_}f$ or $y_{\_}t$. This gives $Q_1 \subseteq_{a\text{-}inj} Q_2$. Conversely, if an expansion of Q_2 can be mapped into K, then, for a choice of t/f edges in D, we have an embedding of each clause gadget of Q_2 in K. In particular, we can always map a literal in each clause of Q_2 to D, ensuring that φ is satisfied. As this is true for any expansion K obtained by any t/f assignment to $\{x_i\}$, we obtain that Φ is satisfiable. □

Other results. The remaining results are summarized on Figure 1 and the following theorem, whose proofs can be found in Appendix F.

THEOREM 6.3.

(1) *The* CQ/CRPQ *and* CQ/CQ *containment problems are* NP-*complete under query-injective semantics. (Proposition F.2 in Appendix F.)*

(2) *The* CQ/CQ *containment problem under atom-injective semantics is* NP-*complete. (Corollary F.4 in Appendix F.)*

(3) *The* CRPQ/CQ *and* CRPQ$^{\mathrm{fin}}$/CQ *containment problems are* Π_2^p*-hard, under standard and atom-injective semantics. (Proposition F.6 in Appendix F.)*

(4) *The* CRPQ/CQ *and* CRPQ$^{\mathrm{fin}}$/CQ *containment problems are in* Π_2^p*, under all semantics. (Proposition F.7 in Appendix F.)*

(5) *The* CRPQ/CRPQ$^{\mathrm{fin}}$ *containment problem is* PSPACE-*hard under all semantics. (Proposition F.8 in Appendix F.)*

(6) *The* CRPQ/CRPQ$^{\mathrm{fin}}$ *containment problem is in* PSPACE *under standard semantics. (Proposition F.9 in Appendix F.)*

(7) *The* CRPQ$^{\mathrm{fin}}$/CRPQ *containment problem is in* Π_2^p*, under all semantics. (Proposition F.10 in Appendix F.)*

7 DISCUSSION AND OUTLOOK

We have defined two possible injective semantics for CRPQs, providing two ways to extend the simple-path semantics of RPQs to the realm of CRPQs. On these semantics, we have shown that the containment problem differs drastically from the standard semantics, in some cases improving the complexity, and in some cases making the problem directly undecidable.

Both of these semantics are natural generalizations of simple-path semantics of RPQs. For instance, if we revert the role of edges and nodes, CRPQs under atom-injective semantics is present in the popular graph database Neo4j. While query-injective semantics is less common in practice, we still believe that this semantics, and in particular, looking for disjoint paths, may be useful for users and may provide an interesting feature for graph query languages. Further empirical investigation is needed to assess the practical usefulness of these two semantics.

While the fragments of the class of CRPQs we have studied are probably the three most fundamental subclasses, there are other more fine-grained fragments based on the form of regular expressions used in the CRPQs, which are practically very relevant [7, 8]. These fragments have been studied under standard semantics [15], and it would be interesting to understand how they behave under injective semantics. A different direction is to consider larger classes of queries, understanding how injective semantics are extended, and the impact on the bounds for containment – such as CRPQ with two-way navigation and union (UC2RPQ) [9], Extended CRPQ (ECRPQ) [6], or Regular Queries [27].

We have limited our investigation to (extensions of) the simple-path semantics. Another possibility is to consider *trail* semantics, which can be extended in a similar way to CRPQs, obtaining again two alternative semantics: query-edge-injective and atom-edge-injective, based on the notion of edge-injective homomorphisms. Many of our results can be extended to these semantics, and we suspect that complexities for query-edge-injective and query-injective coincide on all studied fragments, and likewise for atom-edge-injective and atom-injective. In particular, while neither the undecidability nor the PSpace upper-bound seem to go through when simply reversing the role of nodes and edges, we believe that both proofs can be adapted.

One possible research direction is on another fundamental static analysis problem for CRPQs, namely the *boundedness* problem, which checks whether a CRPQ is equivalent to a finite union of CQs. This problem is decidable for standard semantics [5].

REFERENCES

[1] Renzo Angles, Marcelo Arenas, Pablo Barceló, Peter A. Boncz, George H. L. Fletcher, Claudio Gutiérrez, Tobias Lindaaker, Marcus Paradies, Stefan Plantikow, Juan F. Sequeda, Oskar van Rest, and Hannes Voigt. 2018. G-CORE: A Core for Future Graph Query Languages. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1421–1432. https://doi.org/10.1145/3183713.3190654

[2] Renzo Angles, Marcelo Arenas, Pablo Barceló, Aidan Hogan, Juan Reutter, and Domagoj Vrgoč. 2017. Foundations of Modern Query Languages for Graph Databases. *ACM Comput. Surv.* 50, 5, Article 68 (sep 2017), 40 pages. https://doi.org/10.1145/3104031

[3] Guillaume Bagan, Angela Bonifati, and Benoît Groz. 2020. A trichotomy for regular simple path queries on graphs. *J. Comput. Syst. Sci.* 108 (2020), 29–48. https://doi.org/10.1016/j.jcss.2019.08.006

[4] Pablo Barceló. 2013. Querying graph databases. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, Richard Hull and Wenfei Fan (Eds.). ACM, 175–188. https://doi.org/10.1145/2463664.2465216

[5] Pablo Barceló, Diego Figueira, and Miguel Romero. 2019. Boundedness of Conjunctive Regular Path Queries. In *International Colloquium on Automata, Languages and Programming (ICALP) (Leibniz International Proceedings in Informatics (LIPIcs), Vol. 132)*. Leibniz-Zentrum für Informatik, 104:1–104:15. https://doi.org/10.4230/LIPIcs.ICALP.2019.104

[6] Pablo Barceló, Leonid Libkin, Anthony Widjaja Lin, and Peter T. Wood. 2012. Expressive Languages for Path Queries over Graph-Structured Data. *ACM Trans. Database Syst.* 37, 4 (2012), 31:1–31:46. https://doi.org/10.1145/2389241.2389250

[7] Angela Bonifati, Wim Martens, and Thomas Timm. 2019. Navigating the Maze of Wikidata Query Logs. In *World Wide Web Conference (WWW)*. 127–138.

[8] Angela Bonifati, Wim Martens, and Thomas Timm. 2020. An Analytical Study of Large SPARQL Query Logs. *VLDB Journal* (2020). To appear, https://doi.org/10.1007/s00778-019-00558-9.

[9] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. 2000. Containment of Conjunctive Regular Path Queries with Inverse. In *Principles of Knowledge Representation and Reasoning (KR)*. 176–185.

[10] Ashok K. Chandra and Philip M. Merlin. 1977. Optimal Implementation of Conjunctive Queries in Relational Data Bases. In *Proceedings of the 9th Annual ACM Symposium on Theory of Computing, May 4-6, 1977, Boulder, Colorado, USA*, John E. Hopcroft, Emily P. Friedman, and Michael A. Harrison (Eds.). ACM, 77–90. https://doi.org/10.1145/800105.803397

[11] Stephen A. Cook. 1971. The Complexity of Theorem-Proving Procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman (Eds.). ACM, 151–158. https://doi.org/10.1145/800157.805047

[12] Isabel F. Cruz, Alberto O. Mendelzon, and Peter T. Wood. 1987. A Graphical Query Language Supporting Recursion. In *Proceedings of the Association for Computing Machinery Special Interest Group on Management of Data 1987 Annual Conference, San Francisco, CA, USA, May 27-29, 1987*, Umeshwar Dayal and Irving L. Traiger (Eds.). ACM Press, 323–330. https://doi.org/10.1145/38713.38749

[13] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Wim Martens, Jan Michels, Filip Murlak, Stefan Plantikow, Petra Selmer, Oskar van Rest, Hannes Voigt, Domagoj Vrgoč, Mingxi Wu, and Fred Zemke. 2022. Graph Pattern Matching in GQL and SQL/PGQ. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22)*. 2246–2258. https://doi.org/10.1145/3514221.3526057

[14] Diego Figueira. 2020. Containment of UC2RPQ: the hard and easy cases. In *International Conference on Database Theory (ICDT) (Leibniz International Proceedings in Informatics (LIPIcs))*. Leibniz-Zentrum für Informatik.

[15] Diego Figueira, Adwait Godbole, S. Krishna, Wim Martens, Matthias Niewerth, and Tina Trautner. 2020. Containment of Simple Conjunctive Regular Path Queries. In *Principles of Knowledge Representation and Reasoning (KR)*. https://hal.archives-ouvertes.fr/hal-02505244

[16] Daniela Florescu, Alon Levy, and Dan Suciu. 1998. Query Containment for Conjunctive Queries with Regular Expressions. In *ACM Symposium on Principles of Database Systems (PODS)*. ACM Press, 139–148. https://doi.org/10.1145/275487.275503

[17] Nadime Francis, Alastair Green, Paolo Guagliardo, Leonid Libkin, Tobias Lindaaker, Victor Marsault, Stefan Plantikow, Mats Rydberg, Petra Selmer, and Andrés Taylor. 2018. Cypher: An Evolving Query Language for Property Graphs. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, Gautam Das, Christopher M. Jermaine, and Philip A. Bernstein (Eds.). ACM, 1433–1445. https://doi.org/10.1145/3183713.3190657

[18] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[19] GQL 2021. GQL standard website. https://www.gqlstandards.org/.

[20] Steve Harris and Andy Seaborne. 2013. SPARQL 1.1 Query Language. http://www.w3.org/TR/sparql11-query.

[21] Dexter Kozen. 1977. Lower Bounds for Natural Proof Systems. In *Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 254–266. https://doi.org/10.1109/SFCS.1977.16

[22] Katja Losemann and Wim Martens. 2013. The complexity of regular expressions and property paths in SPARQL. *ACM Trans. Database Syst.* 38, 4 (2013), 24:1–24:39. https://doi.org/10.1145/2494529

[23] Stanislav Malyshev, Markus Krötzsch, Larry González, Julius Gonsior, and Adrian Bielefeldt. 2018. Getting the Most Out of Wikidata: Semantic Technology Usage in Wikipedia's Knowledge Graph. In *International Semantic Web Conference (ISWC)*. 376–394.

[24] Wim Martens and Tina Popp. 2022. To appear. The Complexity of Regular Trail and Simple Path Queries on Undirected Graphs. In *ACM Symposium on Principles of Database Systems (PODS)*. ACM.

[25] Wim Martens and Tina Trautner. 2019. Dichotomies for Evaluating Simple Regular Path Queries. *ACM Trans. Database Syst.* 44, 4 (2019), 16:1–16:46. https://doi.org/10.1145/3331446

[26] Alberto O. Mendelzon and Peter T. Wood. 1995. Finding Regular Simple Paths in Graph Databases. *SIAM J. Comput.* 24, 6 (1995), 1235–1258. https://doi.org/10.1137/S009753979122370X

[27] Juan L. Reutter, Miguel Romero, and Moshe Y. Vardi. 2017. Regular Queries on Graph Databases. *Theory Comput. Syst.* 61, 1 (2017), 31–83. https://doi.org/10.1007/s00224-016-9676-2

[28] Vladislav Rutenburg. 1986. Complexity of Generalized Graph Coloring. In *Mathematical Foundations of Computer Science 1986, Bratislava, Czechoslovakia, August 25-29, 1996, Proceedings (Lecture Notes in Computer Science, Vol. 233),* Jozef Gruska, Branislav Rovan, and Juraj Wiedermann (Eds.). Springer, 573–581. https://doi.org/10.1007/BFb0016284

[29] Sherif Sakr, Angela Bonifati, Hannes Voigt, Alexandru Iosup, Khaled Ammar, Renzo Angles, Walid Aref, Marcelo Arenas, Maciej Besta, Peter A. Boncz, Khuzaima Daudjee, Emanuele Della Valle, Stefania Dumbrava, Olaf Hartig, Bernhard Haslhofer, Tim Hegeman, Jan Hidders, Katja Hose, Adriana Iamnitchi, Vasiliki Kalavri, Hugo Kapp, Wim Martens, M. Tamer Özsu, Eric Peukert, Stefan Plantikow, Mohamed Ragab, Matei R. Ripeanu, Semih Salihoglu, Christian Schulz, Petra Selmer, Juan F. Sequeda, Joshua Shinavier, Gábor Szárnyas, Riccardo Tommasini, Antonino Tumeo, Alexandru Uta, Ana Lucia Varbanescu, Hsiang-Yun Wu, Nikolay Yakovets, Da Yan, and Eiko Yoneki. 2021. The Future is Big Graphs: A Community View on Graph Processing Systems. *Commun. ACM* 64, 9 (aug 2021), 62–71. https://doi.org/10.1145/3434642

A APPENDIX TO SECTION 2

Proof of Proposition 2.2. The result is well-known for the case of standard semantics so we focus on query-injective semantics. Suppose $G = (V, E)$ and Q is of the form $Q(\bar{x}) = A_1 \wedge \cdots \wedge A_m$. Recall $Q(\bar{x})$ is equivalent to a union $\boldsymbol{Q}_{\varepsilon-free}$ of ε-free CRPQs. Assume first that $\bar{v} \in Q(G)^{q\text{-}inj}$. Then $\bar{v} \in Q'(G)^{q\text{-}inj}$ for some $Q' \in \boldsymbol{Q}_{\varepsilon-free}$ of the form $Q'(\bar{z}) = A_1' \wedge \cdots \wedge A_k'$. Without loss of generality, assume $A_{k+1}, \ldots, A_m$ are precisely the atoms of Q collapsed in the construction of Q' (that is, we take the ε-word on those atoms). In particular, the language of the atom A_i' is the language of A_i minus ε, for $i \in \{1, \ldots, k\}$. There exists then an injective mapping μ from $vars(Q')$ to V (satisfying $\mu(\bar{z}) = \bar{v}$), and for each atom $A_i' = x_i \xrightarrow{L_i'} y_i$, a simple path π_i from $\mu(x_i)$ to $\mu(y_i)$ such that distinct paths do not share internal nodes. We can take the expansion $E \in \text{Exp}(Q)$ produced by the expansion profile of Q that maps A_i to the word ε, if $i \in \{k+1, \ldots, m\}$, and maps A_i to the label of π_i, for $i \in \{1, \ldots, k\}$. We can define a homomorphism $h : E \to (G, \bar{v})$ by mapping the non-internal variables of E according to μ and each atom expansion to the corresponding simple path in G. As the paths do not share internal nodes this is an injective homomorphism, and hence $E \xrightarrow{inj} (G, \bar{v})$.

For the other direction, suppose that $h : E \xrightarrow{inj} (G, \bar{v})$ for some expansion $E \in \text{Exp}(Q)$. Let φ be the expansion profile generating E. We construct an ε-free CRPQ $Q' \in \boldsymbol{Q}_{\varepsilon-free}$ as follows: if φ assigns the word ε to A_i, then collapse A_i, otherwise, remove ε from its language. Suppose that Q' is of the form $Q'(\bar{z}) = A_1' \wedge \cdots \wedge A_k'$. We have that $\bar{v} \in Q'(G)^{q\text{-}inj}$. Indeed, define the mapping $\mu : vars(Q') \to V$ as the restriction of h to the non-internal variables of E, and define the path π_i to be the image via h of the expansion of the atom A_i' in E. Since h is injective, the π_i's are simple paths and do not share internal nodes. Moreover, the mapping μ is injective. Hence $\bar{v} \in Q'(G)^{q\text{-}inj}$, which implies that $\bar{v} \in Q(G)^{q\text{-}inj}$ as required. □

Proof of Proposition 2.3. The proof is analogous to the proof of Proposition 2.2, replacing injective homomorphisms by atom-injective homomorphisms. □

B APPENDIX TO SECTION 4

Proof of Proposition 4.3. The proof is identical to the proof of Proposition 4.2 replacing homomorphisms by injective homomorphisms. For the sake of completeness, we give the proof below.

Assume $Q_1 \subseteq_{q\text{-}inj} Q_2$ and take $E_1(\bar{y}) \in \text{Exp}(Q_1)$. Recall that we can see the query $E_1(\bar{y})$ as a graph database (of the same name) where each atom is interpreted as an edge. We have $E_1 \xrightarrow{inj} (E_1, \bar{y})$ and hence $\bar{y} \in Q_1(E_1)^{q\text{-}inj}$. By hypothesis, $\bar{y} \in Q_2(E_1)^{q\text{-}inj}$, that is, there is $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} (E_1, \bar{y})$, i.e, $E_2 \xrightarrow{inj} E_1$. For the other direction, assume $\bar{v} \in Q_1(G)^{q\text{-}inj}$ for some graph database G and tuple $\bar{v}$ of nodes. There is an expansion $E_1 \in \text{Exp}(Q_1)$ such that $E_1 \xrightarrow{inj} (G, \bar{v})$. By hypothesis, there exists $E_2 \in \text{Exp}(Q_2)$ with $E_2 \xrightarrow{inj} E_1$. By composition, we obtain $E_2 \xrightarrow{inj} (G, \bar{v})$, hence $\bar{v} \in Q_2(G)^{q\text{-}inj}$. □

Proof of Lemma 4.4. We start with (1)$\Rightarrow$(2). Let h be a witness for $E \xrightarrow{a\text{-}inj} (G, \bar{v})$. Define the query $\widetilde{F} = E \wedge J$ where J is the conjunction of all equality atoms $x = y$ with $x, y \in vars(E)$ and $h(x) = h(y)$. Since h is atom-injective, we have $F := \widetilde{F}^{\equiv} \in \text{Exp}^{a\text{-}inj}(Q)$. Moreover, there is $g : F \xrightarrow{inj} (G, \bar{v})$ as required. Indeed, let $\Phi : vars(E) \to vars(F)$ be the canonical renaming. For each $x \in vars(F)$, we set $g(x) = h(x')$, where x' is any variable in E with $\Phi(x') = x$. By construction, g is an injective homomorphism. Conversely, suppose $F = \widetilde{F}^{\equiv}$ for $\widetilde{F} = E \wedge J$ with expansion $E \in \text{Exp}(Q)$ and let g be a witness for $F \xrightarrow{inj} (G, \bar{v})$. Let $\Phi : vars(E) \to vars(F)$ be the canonical renaming. Observe that Φ is actually a homomorphism from E to F. By definition of $a\text{-}inj$-expansions, Φ is an atom-injective homomorphism from E to F. By composing Φ with g we obtain $E \xrightarrow{a\text{-}inj} (G, \bar{v})$. The case of E' instead of G is analogous. □

Proof of Proposition 4.6. By Lemma 4.4 it suffices to consider the equivalence between items (1) and (2). Suppose first $Q_1 \subseteq_{a\text{-}inj} Q_2$ and take $F_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$. We can see $F_1(\bar{y})$ as a graph database (of the same name) where each atom is interpreted as an edge. We have $F_1 \xrightarrow{inj} (F_1, \bar{y})$ and, by Corollary 4.5, we obtain $\bar{y} \in Q_1(F_1)^{a\text{-}inj}$. By hypothesis, $\bar{y} \in Q_2(F_1)^{a\text{-}inj}$, that is, there is $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{a\text{-}inj} (F_1, \bar{y})$, i.e, $E_2 \xrightarrow{a\text{-}inj} F_1$. For the other direction, assume $\bar{v} \in Q_1(G)^{a\text{-}inj}$ for some graph database G and tuple $\bar{v}$ of nodes. By Corollary 4.5, there is $F_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$ and $g : F_1 \xrightarrow{inj} (G, \bar{v})$. By hypothesis, there is $E_2 \in \text{Exp}(Q_2)$ and $f : E_2 \xrightarrow{a\text{-}inj} F_1$. By composing f with g we obtain that $E_2 \xrightarrow{a\text{-}inj} (G, \bar{v})$. We conclude that $\bar{v} \in Q_2(G)^{a\text{-}inj}$. □

C FULL PROOF OF THEOREM 5.1

Let $Q_1(\bar{x}_1), Q_2(\bar{x}_2)$ be CRPQs; we want to test $Q_1 \subseteq_{q\text{-}inj} Q_2$.

We often blur the distinction between a CRPQ and an edge-labeled graph, whose edges are regular expressions. Hence, the *degree* [resp. *in-degree*; *out-degree*] of a variable is the number of atoms containing it [resp. as a second variable; as a first variable].

High-level idea. We first give a high-level description of the proof. We will work with polynomial-sized 'abstractions' of expansions of Q_1. These abstractions contain, for each atom A of Q_1, all the information on how the languages of Q_2 can be mapped into it. For example, it includes the information "there is a partial run from state q to state q' of the NFA $\mathcal{A}_L$ of language L from Q_2 reading the expansion word of A", or "there is a partial run from the initial state of $\mathcal{A}_L$ to q reading some suffix of the expansion word of A". Such an abstraction contains all the necessary information needed to retain from an expansion to check if it is a counter-example. Indeed, any expansion with the same abstraction as a counter-example will be a counter-example.

In order to test whether an abstraction α abstracts a counter-example, we need to consider all possible ways of injectively mapping an expansion of Q_2 to an expansion of Q_1. We call this *morphism type*, which contains the information of where each atom expansion of Q_2 is mapped. For example, we can have the information that the path to which the expansion of atom A of Q_2 is mapped starts at some internal node of the expansion of atom $A_1 = x \xrightarrow{L_1} y$ of Q_1 then arrives to variable y with state q and continues reading

the full expansion of atom $A_2 = y \xrightarrow{L_2} z$ arriving to variable z with state q', and it ends its journey by reading a prefix of the expansion of atom $A_3 = z \xrightarrow{L_3} t$ arriving to a final state at some internal node. For each morphism type, we can check if it is *compatible* with an abstraction by checking, for example, that α indeed contains the information of having a partial run from q' to a final state reading a prefix of the expansion of A_3.

The important property is that an abstraction α is compatible with a morphism type τ iff for every expansion E_1 of Q_1 with abstraction α there is an expansion E_2 of Q_2 with morphism type τ such that $E_2 \xrightarrow{inj} E_1$. Hence, the PSPACE algorithm simply guesses α and checks that α, τ are not compatible, for every possible morphism type τ. We now give some more details for these ideas.

Remark C.1. *Any CRPQ is q-inj-equivalent to one in which there is no variable y incident to two atoms, with in-degree and out-degree equal to one. This is because $x \xrightarrow{L} y \wedge y \xrightarrow{L'} x'$ is equivalent (under q-inj or standard semantics) to $x \xrightarrow{L \cdot L'} x'$ (assuming $y \notin \{x, x'\}$).*

Due to Remark C.1, we can assume that the mapping from the expansion of Q_2 to the expansion of Q_1 is such that no two variables can be mapped to two internal nodes of an atom expansion.

Remark C.2. *For every CRPQ Q one can produce an equivalent union Q' of CRPQs such that (i) no language of Q' contains ε and (ii) there are no two distinct atoms $x \xrightarrow{L} y$ and $x \xrightarrow{L'} y$ in Q' with some single-letter word $a \in \mathbb{A}$ in $L \cap L'$. Further, Q' is an exponential union of polynomial-sized CRPQs, and testing whether a CRPQ is in the union is in PSPACE.*

Terminology. By *path* (of an expansion or directed graph) we mean a directed path, that is, a sequence of edges of the form $\pi = (v_0, v_1)(v_1, v_2) \cdots (v_{n-1}, v_n)$. An *internal node* of a path is any node excluding the initial and final ones (i.e., v_0 and v_n). For a path π, and a morphism h, we denote by $h(\pi)$ the path obtained by replacing each vertex v with $h(v)$. For a (directed) path π, we denote by $src(\pi)$ *[resp. $tgt(\pi)$]* the first [resp. last] vertex. A *subpath* of a path π as above, is a path of the form $(v_i, v_{i+1}) \cdots (v_j, v_{j+1})$ where $0 \le i \le j < n$ (in particular of length at least 1). An *infix [resp. prefix, suffix]* of a path of π is a subpath which does not contain $src(\pi)$ or $tgt(\pi)$ [resp. contains $tgt(\pi)$ and excludes $src(\pi)$, contains $src(\pi)$ and excludes $tgt(\pi)$]. We often blur the distinction between regular languages, regular expressions, NFA, and CRPQ atoms containing a regular language. For instance, we may write "q is a final state of atom $x \xrightarrow{L} y$", meaning that it is a final state of the NFA representing L.

Restriction of queries. To simplify the proof, we will assume that Q_1, Q_2 have the following properties:

- there is no ε in any of the languages;
- there are no two atoms $x \xrightarrow{L} y$ and $x \xrightarrow{L'} y$ with some single-letter word $a \in \mathbb{A}$ in $L \cap L'$;
- the queries are connected.

We will later show how to lift these assumptions.

Remark C.3. *As a consequence of Remark C.1, for the CRPQ/CRPQ containment problem of $Q_1 \subseteq_{q\text{-}inj} Q_2$ under q-inj semantics, and assuming the properties above, we can restrict our attention to injective*

homomorphisms $E_1 \xrightarrow{inj} E_1$ *(where $E_i \in \text{Exp}(Q_i)$) such that if two distinct variables $x, y \in vars(Q_1)$ are mapped to distinct internal nodes of an atom expansion, then they must both be of degree 1. That is, in view of the characterization of Proposition 4.3, $Q_1 \subseteq_{q\text{-}inj} Q_2$ iff for every $E_1 \in \text{Exp}(Q_1)$ there is $E_2 \in \text{Exp}(Q_2)$ such that $h : E_2 \xrightarrow{inj} E_1$, where h has the property above.*

Without any loss of generality, let us assume that all the NFA of the languages of Q_2 have pairwise disjoint sets of states, and that they are complete and co-complete (i.e., for every letter and state there is an incoming and an outgoing transition with that letter). Let us consider $\mathcal{A}_{Q_2}$ as an automaton having as transitions the (disjoint) union of all the transitions for the automata of Q_2. In this context we will denote by *initial state* [resp. *final state*] a state which is initial [resp. final] in the automaton from which it comes.

An *abstraction of an expansion E_1* of Q_1 is a mapping α from the atoms of Q_1 to subsets of P, where

$$P = \{\langle q\text{-}q'\rangle : q, q' \text{ states of } \mathcal{A}_{Q_2}\} \cup \{\langle q\dashv q'\rangle : q, q' \text{ states of } \mathcal{A}_{Q_2}\} \cup$$
$$\{\langle q\dashv\cdots\dashv q'\rangle : q, q' \text{ states of } \mathcal{A}_{Q_2}\} \cup \{\langle \cdots q\text{-}q'\cdots\rangle : q, q' \text{ states of } \mathcal{A}_{Q_2}\}$$

such that for every expansion $x \xrightarrow{w} y$ of an atom A we have:

- $\langle q\text{-}q'\rangle \in \alpha(A)$ if there is a partial run of $\mathcal{A}_{Q_2}$ from q to q' reading w;
- $\langle q\dashv q'\rangle \in \alpha(A)$ if for some $w = u \cdot v$ with $u, v \neq \varepsilon$ there is a partial run of $\mathcal{A}_{Q_2}$ from q to a final state reading u, and a partial run from an initial state to q' reading v;
- $\langle q\dashv\cdots\dashv q'\rangle \in \alpha(A)$ if for some $w = u \cdot s \cdot v$ with $s, u, v \neq \varepsilon$ there is a partial run of $\mathcal{A}_{Q_2}$ from q to a final state reading u, and a partial run of $\mathcal{A}_{Q_2}$ from an initial state to q' reading v;
- $\langle \cdots q\text{-}q'\cdots\rangle \in \alpha(A)$ if for some $w = u \cdot s \cdot v$ with $s, u, v \neq \varepsilon$ there is a partial run of $\mathcal{A}_{Q_2}$ from q to q' reading s.

Observe that the size of any abstraction is polynomial in Q_1, Q_2.

The set of *abstractions of Q_1*, is the set of abstractions of all its expansions.

Claim C.1. *Testing whether a mapping is an abstraction of Q_1 is in PSPACE.*

Indeed, a standard pumping argument shows that if α is an abstraction of Q_1, it has a witnessing expansion of at most exponential size. Using this bound, an on-the-fly PSPACE algorithm can guess the expansion for each atom and check that each atom A has abstraction $\alpha(A)$. This is done by guessing one letter at a time while keeping track of all possible partial runs it contains. The procedure also keeps a poly-sized counter to keep track of the size of the expansion being produced, and rejects the computation whenever the size exceeds the exponential bound.

Consider the directed graph $\mathbf{G}$ consisting of replacing each atom $A = x \xrightarrow{L} y$ of Q_1 with a path $\pi_A^{\mathbf{G}}$ of length 3 (i.e., adding two new internal vertices). A *morphism type* from $Q_2(\bar{x}_2)$ to $Q_1(\bar{x}_1)$ is a pair (H, h) such that $h : H \xrightarrow{inj} \mathbf{G}$, and H is a graph resulting from replacing each atom $A = x \xrightarrow{L} y$ of Q_2 with a (non-empty) path π_A^H from x to y. Further, we also ask that free variables are mapped accordingly, that is, $h(\bar{x}_2) = \bar{x}_1$. Figure 8 contains an example of a morphism type (H, h). It follows that, by injectivity, the size of H in any morphism type is linearly bounded on Q_1.
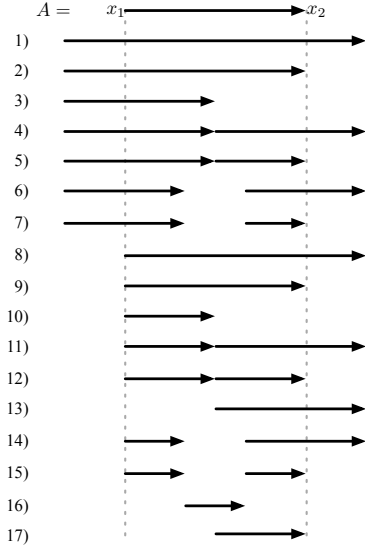
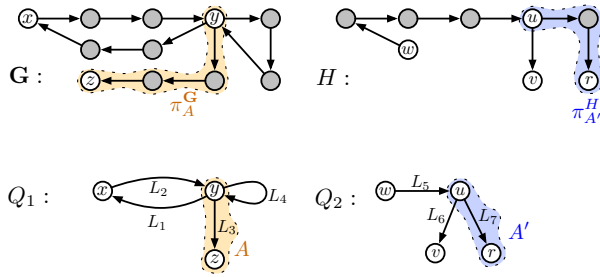**Figure 9: All the possible cases for compatibility. Observe that there are no other cases due to Remark C.3.**

Figure 8: Example of definition of G and morphism type (H, h) from Q_1, Q_2. In this case, the injective morphism h from H to G maps each node of H to the node in the same position on G (e.g., the lower-right node r of H maps to the lower-right node of G).

Claim C.2. *Testing whether a pair (H, h) is a morphism type is in* PSPACE.

A morphism type (H, h) is *compatible* with an abstraction α if there is a mapping λ from the internal nodes of paths π_A^H to states of A, such that for every atom $A = x_1 \xrightarrow{L} x_2$ of Q_1,

- if there is an atom A' of Q_2 and an infix subpath π of $\pi_{A'}^H$ with $h(\pi) = \pi_A^G$, then $\langle \lambda(src(\pi))\text{-}\lambda(tgt(\pi)) \rangle \in \alpha(A)$ (corresponding to case 1 in Figure 9);
- if there is an atom A' of Q_2 and a suffix subpath π of $\pi_{A'}^H$ with $h(\pi) = \pi_A^G$, then $\langle \lambda(src(\pi))\text{-}q_F \rangle \in \alpha(A)$ where q_F is a final state of A' (case 2 in Fig. 9);
- if there is an atom A' of Q_2 and a suffix subpath π of $\pi_{A'}^H$ with $h(\pi)$ being a prefix of π_A^G, then $\langle \lambda(src(\pi))\dashv q \rangle \in \alpha(A)$ for some q (case 3 in Fig. 9);

- if there are atoms A_1', A_2' of Q_2, a suffix subpath π_1 of $\pi_{A_1'}^H$ and a prefix subpath π_2 of $\pi_{A_2'}^H$ with $tgt(\pi_1) = src(\pi_2)$ and $h(\pi_1\pi_2) = \pi_A^G$, then $\langle src(\pi_1)\dashv tgt(\pi_2) \rangle \in \alpha(A)$ (case 4 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2 and a suffix subpath π of $\pi_{A_1'}^H$ with $tgt(\pi) = src(\pi_{A_2'}^H)$ and $h(\pi\pi_{A_2'}^H) = \pi_A^G$, then $\langle \lambda(src(\pi))\dashv q_F \rangle \in \alpha(A)$ where q_F is a final state of A_2' (case 5 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2, a suffix subpath π_1 of $\pi_{A_1'}^H$, and a prefix subpath π_2 of $\pi_{A_2'}^H$ with $tgt(\pi_1) \neq src(\pi_2)$, $h(\pi_1)$ is a prefix of π_A^G and $h(\pi_2)$ is a suffix of π_A^G, then $\langle \lambda(src(\pi_1))\dashv\cdots\dashv\lambda(tgt(\pi_2)) \rangle \in \alpha(A)$ (case 6 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2, and a suffix subpath π of $\pi_{A_1'}^H$ where $h(\pi)$ is a prefix of π_A^G, $h(\pi_{A_2'}^H)$ is a suffix of π_A^G and $tgt(\pi) \neq src(\pi_{A_2'}^H)$, then $\langle \lambda(src(\pi))\dashv\cdots\dashv q_F \rangle \in \alpha(A)$ for some final state q_F of A_2' (case 7 in Fig. 9);
- if there is an atom A' of Q_2 and a prefix subpath π of $\pi_{A'}^H$ with $h(\pi) = \pi_A^G$, then $\langle q_0\text{-}\lambda(tgt(\pi)) \rangle \in \alpha(A)$ where q_0 is an initial state of A' (case 8 in Fig. 9);
- if there is an atom A' of Q_2 with $h(\pi_{A'}^H) = \pi_A^G$, then $\langle q_0\text{-}q_F \rangle \in \alpha(A)$ where q_0/q_F is an initial/final state of A' (case 9 in Fig. 9);
- if there is an atom A' of Q_2 with $h(\pi_{A'}^H)$ a prefix of π_A^G, then $\langle q_0\dashv q \rangle \in \alpha(A)$ where q_0 is an initial state of A' and q is any state. (case 10 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2 and a prefix subpath π_2 of $\pi_{A_2'}^H$ with $tgt(\pi_{A_1'}^H) = src(\pi_2), h(\pi_{A_1'}^H \pi_2) = \pi_A^G$, then $\langle q_0\dashv\lambda(tgt(\pi_2)) \rangle \in \alpha(A)$ where q_0 is an initial state of A_1' (case 11 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2 with $tgt(\pi_{A_1'}^H) = src(\pi_{A_2'}^H)$, $h(\pi_{A_1'}^H \pi_{A_2'}^H) = \pi_A^G$, then $\langle q_0\dashv q_F \rangle \in \alpha(A)$ where q_0 is an initial state of A_1' and q_F a final state of A_2' (case 12 in Fig. 9);
- if there is an atom A' of Q_2 and a prefix π of $\pi_{A'}^H$ with $h(\pi)$ a suffix of π_A^G, then and $\langle q\dashv\cdots\dashv\lambda(tgt(\pi)) \rangle \in \alpha(A)$ for some q (case 13 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2 and a prefix subpath π_2 of $\pi_{A_2'}^H$ with $tgt(\pi_{A_1'}^H) \neq src(\pi_2)$, $h(\pi_{A_1'}^H)$ is a prefix of π_A^G and $h(\pi_2)$ is a suffix of π_A^G, then $\langle q_0\dashv\cdots\dashv\lambda(tgt(\pi_2)) \rangle \in \alpha(A)$ where q_0 is an initial state of A_1' (case 14 in Fig. 9);
- if there are atoms A_1', A_2' of Q_2 with $tgt(\pi_{A_1'}^H) \neq src(\pi_{A_2'}^H)$, $h(\pi_{A_1'}^H)$ is a prefix of π_A^G and $h(\pi_{A_2'}^H)$ is a suffix of π_A^G, then $\langle q_0\dashv\cdots\dashv q_F \rangle \in \alpha(A)$ where q_0 is an initial state of A_1' and q_F a final state of A_2' (case 15 in Fig. 9);
- if there is an atom A' of Q_2 with $h(\pi_{A'}^H)$ being an infix of π_A^G, then $\langle \cdots q_0\text{-}q_F\cdots \rangle \in \alpha(A)$ for q_0 and q_F initial and final states of A' (case 16 in Fig. 9);
- if there is an atom A' of Q_2 where $h(\pi_{A'}^H)$ is a suffix of π_A^G, then $\langle q\dashv q_F \rangle \in \alpha(A)$ for some state q and some final state q_F of A' (case 17 in Fig. 9).
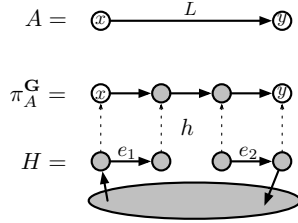
Figure 10: Example for proof of Claim C.4

The following statement is a direct consequence of H, h, λ being polynomially bounded and each of the conditions above being polynomial-time testable.

Claim C.3. *Testing whether a morphism type is compatible with an abstraction is in* PSPACE.

The key property of compatible abstractions, is that they allow to capture whether an expansion of Q_1 with a given abstraction is a counter-example for the containment problem $Q_1 \subseteq_{q\text{-}inj} Q_2$.

Claim C.4. *The following are equivalent:*

(1) *There is a morphism type compatible with an abstraction α;*
(2) *for every expansion $E_1 \in \text{Exp}(Q_1)$ with abstraction α there exists some expansion $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$;*
(3) *there is an expansion $E_1 \in \text{Exp}(Q_1)$ with abstraction α and an expansion $E_2 \in \text{Exp}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$.*

PROOF. $\boxed{1 \Rightarrow 2)}$ Assume $E_1 \in \text{Exp}(Q_1)$ has abstraction α, and (H, h) is a compatible morphism type through the mapping λ. For every atom A of Q_1, we replace every path π of H such that $h(\pi) = \pi_A^{\text{G}}$ with the expansion of A in E_1. The remaining edges of H are all part of paths which map *partially* to some π_A^{G}, these are replaced with paths according to the witnessing words for the elements of the form $\langle q\text{-}\!\cdots\!\text{-}q'\rangle$ and $\langle\cdots q\text{-}q'\cdots\rangle$ in each $\alpha(A)$.

For example, in the case depicted in Figure 10, we know that the expansion $w \in L$ of A is of the form $w = u \cdot s \cdot v$ such that there is a partial run of $\mathcal{A}_{Q_2}$ from q to some final state reading u, and a partial run of $\mathcal{A}_{Q_2}$ from an initial state to q' reading v. Hence, we replace edge e_1 with a path reading u, and edge e_2 with a path reading v. It follows that by the definition of abstraction the resulting CQ (*i.e.*, the query represented by the resulting edge-labeled graph) is an expansion of Q_2 that maps to E_1 through an injective homomorphism.

$\boxed{2 \Rightarrow 3)}$ This is trivial since an abstraction of Q_1 is the abstraction of an expansion thereof.

$\boxed{3 \Rightarrow 1)}$ Take any $E_1 \in \text{Exp}(Q_1)$ with abstraction α and $E_2 \in \text{Exp}(Q_2)$ such that $g : E_2 \xrightarrow{inj} E_1$. We now build a graph H from E_2 as follows. In the sequel, whenever we say that we replace a path π with a path of length n, we mean that we (1) remove all internal nodes of π, and all edges incident to these and (2) we add $n - 1$ fresh nodes and n edges in such a way that there is a path of length n from $src(\pi)$ to $tgt(\pi)$. For every path π in E_1 corresponding to the expansion of atom A :

- If there is a prefix [resp. suffix] of π which has no g-preimage, we replace the path $g^{-1}(\pi)$ with just one edge, and we send

the variable to the first internal node of π_A^{G}. For the remaining cases let us assume that every node of π has a g-preimage.

- If $g^{-1}(\pi)$ is a path with no Q_2-variables as internal nodes, then we replace it with an unlabeled path π' of length 3. We define h to map the first [resp. second] internal node of π' to the first [resp. second] internal node of π_A^{G}.
- If $g^{-1}(\pi)$ is a path that contains one variable $x \in vars(Q_2)$ as internal node, we replace the first half path of $g^{-1}(\pi)$ until x with just an unlabeled edge, and the other half with a path π' of length 2. We set h to map the variable x to the first internal node of π_A^{G} and the internal node of π to the second internal node of π_A^{G}.
- If $g^{-1}(\pi)$ contains two variables and two disjoint paths, we replace each of them with an unlabeled edge. We send the variables (*i.e.*, the endpoints of the paths) correspondingly to the two internal vertices of π_A^{G}.

The resulting graph H and mapping h is a morphism type which is compatible with α. □

Finally, the PSPACE algorithm guesses a mapping α from the atoms of Q_1 to subsets of P, checks that α is an abstraction of Q_1 (in PSPACE due to Claim C.1), and checks that there is no morphism type (H, h) which is compatible with α (in PSPACE, due to Claims C.2 and C.3, and closure under complement of PSPACE). Due to Claim C.4, if the algorithm succeeds, then any expansion E_1 of Q_1 is a counter-example, and thus $Q_1 \not\subseteq_{q\text{-}inj} Q_2$; otherwise, for every expansion E_1 with abstraction α there is a compatible morphism type, which means that E_1 is not a counter-example and hence $Q_1 \subseteq_{q\text{-}inj} Q_2$.

First note that if Q_2 is not connected, we can adapt the PSPACE algorithm by testing that there are no morphism types for the connected components of Q_2 such that all of them are compatible with the guessed abstraction of Q_1. Further, observe that the procedure can be extended to an exponential union of polynomial-sized CRPQs: the PSPACE algorithm first chooses one CRPQ Q_1 from the left-hand side union, guesses an abstraction of Q_1 and checks that no CRPQ coming from the right-hand side union has a compatible morphism type. For this reason, combined with Remark C.2, the same argument extends to (unions of) arbitrary CRPQ's. □

D FULL PROOF OF THEOREM 5.2

We reduce from *Post Correspondence Problem* (PCP), a well-known undecidable problem. An instance of the PCP is a sequence of pairs $(u_1, v_1), \ldots, (u_\ell, v_\ell)$, where u_i and v_i are non-empty words over an alphabet Σ. The goal is to decide whether there is a *solution*, that is, a sequence $i_1, \ldots, i_k$ of indices from $\{1, \ldots, \ell\}$, with $k \geq 1$, such that the words $u_{i_1} \cdots u_{i_k}$ and $v_{i_1} \cdots v_{i_k}$ coincide.

For an alphabet $\mathbb{A}$, we denote by $\widehat{\mathbb{A}}$ the alphabet $\widehat{\mathbb{A}} = \{\widehat{a} : a \in \mathbb{A}\}$. Let $(u_1, v_1), \ldots, (u_\ell, v_\ell)$ be a PCP instance and let Σ be its underlying alphabet. Let $\mathbb{I}$ and $\mathbb{A}$ be the alphabets $\mathbb{I} = \{I_1, \ldots, I_\ell\}$ and $\mathbb{A} = \Sigma \cup \mathbb{I} \cup \{\#, \#_\infty, \square, \$, \$', \$_\infty, \blacksquare, \blacksquare'\}$. We construct Boolean CRPQs Q_1 and Q_2 over alphabet $\mathbb{A} \cup \widehat{\mathbb{A}}$ such that the PCP instance $(u_1, v_1), \ldots, (u_\ell, v_\ell)$ has a solution if and only if $Q_1 \not\subseteq_{a\text{-}inj} Q_2$. In particular, the PCP instance has a solution if and only if there exists a counterexample for *a-inj*-semantics, *i.e.*, an *a-inj*-expansion F of Q_1 such that there is no expansion $E \in \text{Exp}(Q_2)$ with $E \xrightarrow{a\text{-}inj} F$.
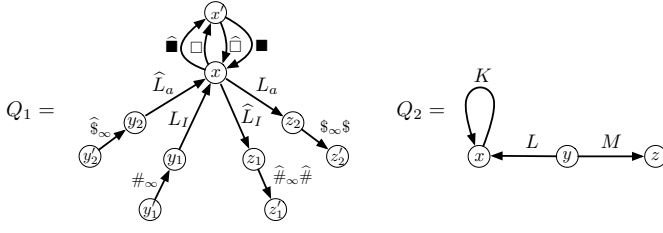
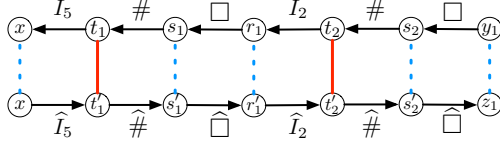**Figure 11: The Boolean CRPQs Q_1 and Q_2 from the reduction.**

Figure 12: Example of the $I\text{-}\widehat{I}$-condition of well-formed expansions of Q_1. We show the expansions $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{\widehat{w}_I} z_1$ of the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L}_I} z_1$, respectively. The words $w_I = \square \# I_2 \square \# I_5$ and $\widehat{w}_I = \widehat{I_5} \widehat{\#} \widehat{\square} \widehat{I_2} \widehat{\#} \widehat{\square}$ encode the sequence of indices $5, 2$. Dotted blue lines indicate pairs of equal variables while red lines indicate distinct variables. We have some extra symbols $\#, \widehat{\#}, \square, \widehat{\square}$.

The symbols in $\mathbb{A}$ are associated with the words u_i, while the symbols in $\widehat{\mathbb{A}}$ with the words v_i. For each $u_i = a_1 \cdots a_k$, we define the word $U_i = a_1 \$ \blacksquare a_2 \$ \blacksquare \cdots a_k \$' \blacksquare'$. Similarly, for each $v_i = a_1 \cdots a_k$ we define $V_i = \widehat{\blacksquare}' \widehat{\$}' \widehat{a_k} \widehat{\blacksquare} \widehat{\$} \widehat{a_{k-1}} \cdots \widehat{\blacksquare} \widehat{\$} \widehat{a_1}$. The CRPQ Q_1 is defined as follows (see Figure 11 for an illustration):

$$Q_1 = y_1 \xrightarrow{L_I} x \wedge y_2 \xrightarrow{\widehat{L}_a} x \wedge x \xrightarrow{\widehat{L}_I} z_1 \wedge x \xrightarrow{L_a} z_2$$

$$\wedge \; x \xrightarrow{\square} x' \wedge x \xrightarrow{\widehat{\blacksquare}} x' \wedge x' \xrightarrow{\widehat{\square}} x \wedge x' \xrightarrow{\blacksquare} x$$

$$\wedge \; y_1' \xrightarrow{\#_\infty} y_1 \wedge y_2' \xrightarrow{\widehat{\$}_\infty} y_2 \wedge z_1 \xrightarrow{\widehat{\#_\infty \#}} z_1' \wedge z_2 \xrightarrow{\$_\infty \$} z_2'$$

where:

$$L_I = (\square \# \mathbb{I})^+ \qquad \widehat{L}_I = (\widehat{\mathbb{I}} \widehat{\#} \widehat{\square})^+$$

$$L_a = (U_1 + \cdots + U_\ell)^+ \qquad \widehat{L}_a = (V_1 + \cdots + V_\ell)^+$$

Intuitively, a word from L_I corresponds to a choice of indices from $\{1, \ldots, \ell\}$, similarly for $\widehat{L}_I$. On the other hand, a word from L_a [resp. $\widehat{L}_a$] corresponds to a choice of words from $\{u_1, \ldots, u_\ell\}$ [resp. $\{v_1, \ldots, v_\ell\}$].

We are interested in a particular type of *a-inj*-expansions of Q_1 that we call *well-formed* and define below. The idea is that well-formed *a-inj*-expansions correspond to solutions of the PCP instance. In particular, if there is a well-formed *a-inj*-expansion of Q_1 then there is a solution to the PCP instance and vice versa. We then show how to construct Q_2 such that an *a-inj*-expansion of Q_1 is well-formed if and only if it is a counterexample for $Q_1 \subseteq_{a\text{-}inj} Q_2$.

Let F be an *a-inj*-expansion of Q_1 such that $F = \widetilde{F}^\equiv$ for $\widetilde{F} = E \wedge J$ (here $E \in \text{Exp}(Q_1)$ and J are the equality atoms). We say that F is *well-formed* if it satisfies the following four conditions:

(1) *$I\text{-}\widehat{I}$-condition :* This condition applies to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L}_I} z_1$ of Q_1. Let $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{\widehat{w}_I} z_1$ be the expansions associated to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L}_I} z_1$ in the expansion E. The condition requires the words w_I and $\widehat{w}_I$ to be of the form $w_I = \square \# I_{i_k} \cdots \square \# I_{i_1}$ and $\widehat{w}_I = \widehat{I_{i_1}} \widehat{\#} \widehat{\square} \cdots \widehat{I_{i_k}} \widehat{\#} \widehat{\square}$, for a sequence of indices $i_1, \ldots, i_k \in \{1, \ldots, \ell\}$. In other words, the expansions of the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L}_I} z_1$ correspond to the same sequence on indices. The $I\text{-}\widehat{I}$-condition also requires a particular behavior on the equality atoms J (see Figure 12). Suppose the expansion $y_1 \xrightarrow{w_I} x$ is of the form:

$$y_1 \xrightarrow{\square} s_k \xrightarrow{\#} t_k \xrightarrow{I_{i_k}} r_{k-1} \xrightarrow{\square} s_{k-1} \xrightarrow{\#} t_{k-1} \xrightarrow{I_{i_{k-1}}} r_{k-2} \cdots$$
$$\cdots r_1 \xrightarrow{\square} s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x$$

and $x \xrightarrow{\widehat{w}_I} z_1$ is of the form:

$$x \xrightarrow{\widehat{I_{i_1}}} t_1' \xrightarrow{\widehat{\#}} s_1' \xrightarrow{\widehat{\square}} r_1' \xrightarrow{\widehat{I_{i_2}}} t_2' \xrightarrow{\widehat{\#}} s_2' \xrightarrow{\widehat{\square}} r_2' \cdots$$
$$\cdots r_{k-1}' \xrightarrow{\widehat{I_{i_k}}} t_k' \xrightarrow{\widehat{\#}} s_k' \xrightarrow{\widehat{\square}} z_1$$

Then the relation $=_{\widetilde{F}}$ produced by the equality atoms J satisfies:

(a) $t_1 \neq_{\widetilde{F}} t_1', \cdots, t_k \neq_{\widetilde{F}} t_k'$
(b) $s_1 =_{\widetilde{F}} s_1', \cdots, s_k =_{\widetilde{F}} s_k'$
(c) $r_1 =_{\widetilde{F}} r_1', \cdots, r_k =_{\widetilde{F}} r_k'$
where $r_k := y_1$ and $r_k' := z_1$.

(2) *$I\text{-}a$-condition :* This condition applies to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{L_a} z_2$ of Q_1. Let $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{w_a} z_2$ be the expansions associated to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{L_a} z_2$ in the expansion E. The condition requires the words w_I and w_a to be of the form $w_I = \square \# I_{i_k} \cdots \square \# I_{i_1}$ and $w_a = U_{i_1} \cdots U_{i_k}$, for a sequence of indices $i_1, \ldots, i_k \in \{1, \ldots, \ell\}$. Intuitively, the word w_a chooses words from $\{u_1, \ldots, u_\ell\}$ according to the sequence $i_1, \ldots, i_k$. The $I\text{-}a$-condition also requires a particular behavior on the equality atoms J. Suppose the expansion $y_1 \xrightarrow{w_I} x$ is of the form:

$$y_1 \xrightarrow{\square} s_k \xrightarrow{\#} t_k \xrightarrow{I_{i_k}} r_{k-1} \xrightarrow{\square} s_{k-1} \xrightarrow{\#} t_{k-1} \xrightarrow{I_{i_{k-1}}} r_{k-2} \cdots$$
$$\cdots r_1 \xrightarrow{\square} s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x$$

and the expansion $x \xrightarrow{w_a} z_2$ is of the form:

$$x \xrightarrow{\widetilde{U_{i_1}}} s_1' \xrightarrow{\blacksquare'} r_1' \xrightarrow{\widetilde{U_{i_2}}} s_2' \xrightarrow{\blacksquare'} r_2' \cdots r_{k-1}' \xrightarrow{\widetilde{U_{i_k}}} s_k' \xrightarrow{\blacksquare'} z_2$$

where $\widetilde{U_i}$ is the word obtained from U_i by removing the last symbol $\blacksquare'$. Then the relation $=_{\widetilde{F}}$ produced by the equality atoms J satisfies:

(a) $t_j \neq_{\widetilde{F}} t$, for every internal variable t of the expansion
$$r_{j-1}' \xrightarrow{\widetilde{U_{i_j}}} s_j' \text{ (here } r_0' := x)$$
(b) $s_1 =_{\widetilde{F}} s_1', \cdots, s_k =_{\widetilde{F}} s_k'$

(c) $r_1 =_{\widetilde{F}} r'_1, \cdots, r_k =_{\widetilde{F}} r'_k$
where $r_k := y_1$ and $r'_k := z_2$.

(3) $\widehat{a}\text{-}\widehat{I}\text{-condition}$: This is analogous to the $I\text{-}a$-condition and applies to the atoms $y_2 \xrightarrow{\widehat{L_a}} x$ and $x \xrightarrow{\widehat{L_I}} z_1$ of Q_1. Let $y_2 \xrightarrow{\widehat{w_a}} x$ and $x \xrightarrow{\widehat{w_I}} z_1$ be the expansions associated to the atoms $y_2 \xrightarrow{\widehat{L_a}} x$ and $x \xrightarrow{\widehat{L_I}} z_1$ in the expansion E. The condition requires the words $\widehat{w}_a$ and $\widehat{w}_I$ to be of the form $\widehat{w}_I = \widehat{I}_{i_1} \widehat{\#}\widehat{\square} \cdots \widehat{I}_{i_k} \widehat{\#}\widehat{\square}$ and $\widehat{w}_a = V_{i_k} \cdots V_{i_1}$, for a sequence of indices $i_1, \ldots, i_k \in \{1, \ldots, \ell\}$. That is, the word $\widehat{w}_a$ chooses words from $\{v_1, \ldots, v_\ell\}$ according to the sequence $i_1, \ldots, i_k$. We also require some conditions on the equality atoms J. Suppose the expansion $y_2 \xrightarrow{\widehat{w_a}} x$ is of the form:

$$y_2 \xrightarrow{\widehat{\blacksquare}'} s_k \xrightarrow{\widetilde{V}_{i_k}} r_{k-1} \xrightarrow{\widehat{\blacksquare}'} s_{k-1} \xrightarrow{\widetilde{V}_{i_{k-1}}} r_{k-2} \cdots r_1 \xrightarrow{\widehat{\blacksquare}'} s_1 \xrightarrow{\widetilde{V}_{i_1}} x$$

where $\widetilde{V}_i$ is the word obtained from V_i by removing the first symbol $\widehat{\blacksquare}'$. Suppose also the expansion $x \xrightarrow{\widehat{w_I}} z_1$ is of the form:

$$x \xrightarrow{\widehat{I}_{i_1}} t'_1 \xrightarrow{\widehat{\#}} s'_1 \xrightarrow{\widehat{\square}} r'_1 \xrightarrow{\widehat{I}_{i_2}} t'_2 \xrightarrow{\widehat{\#}} s'_2 \xrightarrow{\widehat{\square}} r'_2 \cdots$$
$$\cdots r'_{k-1} \xrightarrow{\widehat{I}_{i_k}} t'_k \xrightarrow{\widehat{\#}} s'_k \xrightarrow{\widehat{\square}} z_1$$

Then the relation $=_{\widetilde{F}}$ produced by the equality atoms J satisfies:

(a) $t'_j \neq_{\widetilde{F}} t$, for every internal variable t of the expansion
$$s_j \xrightarrow{\widetilde{V}_{i_j}} r_{j-1} \text{ (here } r_0 := x)$$
(b) $s_1 =_{\widetilde{F}} s'_1, \cdots, s_k =_{\widetilde{F}} s'_k$
(c) $r_1 =_{\widetilde{F}} r'_1, \cdots, r_k =_{\widetilde{F}} r'_k$
where $r_k := y_2$ and $r'_k := z_1$.

(4) $\widehat{a}\text{-}a\text{-condition}$: This condition applies to the atoms $y_2 \xrightarrow{\widehat{L_a}} x$ and $x \xrightarrow{L_a} z_2$ of Q_1. Let $y_2 \xrightarrow{\widehat{w_a}} x$ and $x \xrightarrow{w_a} z_2$ be the expansions associated to the atoms $y_2 \xrightarrow{\widehat{L_a}} x$ and $x \xrightarrow{L_a} z_2$ in the expansion E. The condition requires the words $\widehat{w}_a$ and w_a to be of the form $\widehat{w}_a = \clubsuit\clubsuit\widehat{a}_n \cdots \clubsuit\clubsuit\widehat{a}_1$ and $w_a = a_1\clubsuit\clubsuit \cdots a_n\clubsuit\clubsuit$, for a word $a_1 \cdots a_n \in \Sigma^*$ (recall Σ is the alphabet of the PCP instance). Here, $\clubsuit$ is a placeholder representing some symbol. Intuitively, the word $\widehat{w}_a$ and w_a represent the same word from Σ^*. The $\widehat{a}\text{-}a$-condition also requires some conditions on the equality atoms J. Assume the expansion $y_2 \xrightarrow{\widehat{w_a}} x$ is of the form:

$$y_2 \xrightarrow{\clubsuit} s_n \xrightarrow{\clubsuit} t_n \xrightarrow{\widehat{a}_n} r_{n-1} \xrightarrow{\clubsuit} s_{n-1} \xrightarrow{\clubsuit} t_{n-1} \xrightarrow{\widehat{a}_{n-1}} r_{n-2} \cdots$$
$$\cdots r_1 \xrightarrow{\clubsuit} s_1 \xrightarrow{\clubsuit} t_1 \xrightarrow{\widehat{a}_1} x$$

and the expansion $x \xrightarrow{w_a} z_2$ is of the form:

$$x \xrightarrow{a_1} t'_1 \xrightarrow{\clubsuit} s'_1 \xrightarrow{\clubsuit} r'_1 \xrightarrow{a_2} t'_2 \xrightarrow{\clubsuit} s'_2 \xrightarrow{\clubsuit} r'_2 \cdots$$
$$\cdots r'_{n-1} \xrightarrow{a_n} t'_n \xrightarrow{\clubsuit} s'_n \xrightarrow{\clubsuit} z_2$$

Then the relation $=_{\widetilde{F}}$ produced by the equality atoms J satisfies:
(a) $t_1 \neq_{\widetilde{F}} t'_1, \cdots, t_n \neq_{\widetilde{F}} t'_n$
(b) $s_1 =_{\widetilde{F}} s'_1, \cdots, s_n =_{\widetilde{F}} s'_n$
(c) $r_1 =_{\widetilde{F}} r'_1, \cdots, r_n =_{\widetilde{F}} r'_n$
where $r_n := y_2$ and $r'_n := z_2$.

The key property of well-formedness is that it can be characterized in terms of the non-existence of a finite number of simple cycles and simple paths having certain labels. In order to do this, we need to define some finite languages. Recall $\widetilde{U}_i$ is obtained from U_i by removing the last symbol $\blacksquare'$ and $\widetilde{V}_i$ is obtained from V_i by removing the first symbol $\widehat{\blacksquare}'$. We define N to be the maximum length of the words U_i. We denote by $e^{i,j}$, for $i \leq j$, the regular expression $(e^i + e^{i+1} + \cdots + e^j)$. We have:

$$K_{I\widehat{I}} = \mathbb{I}\widehat{\mathbb{I}} + \#_\infty \widehat{\mathbb{I}} + \mathbb{I}\widehat{\#}_\infty$$

$$M_{I\widehat{I}} = \sum_{i \neq j} I_i\widehat{I}_j + \widehat{\mathbb{I}}\# + \widehat{\#}\mathbb{I} + \#\mathbb{I}\widehat{\mathbb{I}}\widehat{\#} + \square\widehat{\square} + \#_\infty \widehat{\mathbb{I}} + \mathbb{I}\widehat{\#}_\infty$$

$$K_{Ia} = \mathbb{I}\Sigma + \#_\infty \Sigma + \mathbb{I}\$_\infty$$

$$M_{Ia} = (\Sigma + \$ + \$' + \blacksquare)\,\mathbb{I} + (\Sigma + \$ + \blacksquare)^{1,N}\,\# + \sum_i \sum_{j \neq i} I_i\widetilde{U}_j +$$
$$+ \#\mathbb{I}\,(\widetilde{U}_1 + \cdots + \widetilde{U}_\ell) + \square\,\blacksquare' + \#_\infty \Sigma + \mathbb{I}\$_\infty$$

$$K_{\widehat{a}\widehat{I}} = \widehat{\Sigma}\widehat{\mathbb{I}} + \widehat{\$}_\infty \widehat{\mathbb{I}} + \widehat{\Sigma}\widehat{\#}_\infty$$

$$M_{\widehat{a}\widehat{I}} = \widehat{\mathbb{I}}\,(\widehat{\Sigma} + \widehat{\$} + \widehat{\$}' + \widehat{\blacksquare}) + \widehat{\#}\widehat{\Sigma} + \widehat{\mathbb{I}}\widehat{\#}\,(\widehat{\Sigma} + \widehat{\$} + \widehat{\blacksquare}) + \sum_i \sum_{j \neq i} \widetilde{V}_j\widehat{I}_i +$$
$$+ (\widetilde{V}_1 + \cdots + \widetilde{V}_\ell)\,\widehat{\mathbb{I}}\widehat{\#} + \widehat{\blacksquare}'\widehat{\square} + \widehat{\$}_\infty \widehat{\mathbb{I}} + \widehat{\Sigma}\widehat{\#}_\infty$$

$$K_{\widehat{a}a} = \widehat{\Sigma}\Sigma + \widehat{\$}_\infty \Sigma + \widehat{\Sigma}\$_\infty$$

$$M_{\widehat{a}a} = \sum_{a \neq b} \widehat{a}b + \Sigma\,(\widehat{\$} + \widehat{\$}') + (\$ + \$')\,\widehat{\Sigma} + (\widehat{\$} + \widehat{\$}')\,\widehat{\Sigma}\Sigma\,(\$ + \$')+$$
$$+ (\widehat{\blacksquare} + \widehat{\blacksquare}')(\blacksquare + \blacksquare') + \widehat{\$}_\infty \Sigma + \widehat{\Sigma}\$_\infty$$

Claim D.1. Let F be an a-inj-expansion of Q_1. Then:
(1) F satisfies the $I\text{-}\widehat{I}$-condition iff F does not contain a simple cycle with label in $K_{I\widehat{I}}$ nor a simple path with label in $M_{I\widehat{I}}$.
(2) F satisfies the $I\text{-}a$-condition iff F does not contain a simple cycle with label in K_{Ia} nor a simple path with label in M_{Ia}.
(3) F satisfies the $\widehat{a}\text{-}\widehat{I}$-condition iff F does not contain a simple cycle with label in $K_{\widehat{a}\widehat{I}}$ nor a simple path with label in $M_{\widehat{a}\widehat{I}}$.
(4) F satisfies the $\widehat{a}\text{-}a$-condition iff F does not contain a simple cycle with label in $K_{\widehat{a}a}$ nor a simple path with label in $M_{\widehat{a}a}$.

Proof. Suppose $F = \widetilde{F}^{\equiv}$ for $\widetilde{F} = E \wedge J$, where $E \in \text{Exp}(Q_1)$ and J are equality atoms. We start with item (1). Let $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{\widehat{w_I}} z_1$ be the expansions associated to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{\widehat{L_I}} z_1$ in the expansion E. Suppose that $w_I = \square\# I_{i_k} \cdots \square\# I_{i_1}$ and $\widehat{w}_I = \widehat{I}_{j_1} \widehat{\#}\widehat{\square} \cdots \widehat{I}_{j_p} \widehat{\#}\widehat{\square}$, for indices $i_1, \ldots, i_k, j_1, \ldots, j_p \in \{1, \ldots, \ell\}$. Assume also that $y_1 \xrightarrow{w_I} x$ is of the form:

$$y_1 \xrightarrow{\square} s_k \xrightarrow{\#} t_k \xrightarrow{I_{i_k}} r_{k-1} \xrightarrow{\square} s_{k-1} \xrightarrow{\#} t_{k-1} \xrightarrow{I_{i_{k-1}}} r_{k-2} \cdots$$
$$\cdots r_1 \xrightarrow{\square} s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x$$

and $x \xrightarrow{\widehat{w_I}} z_1$ is of the form:

$$x \xrightarrow{\widehat{I_{j_1}}} t'_1 \xrightarrow{\widehat{\#}} s'_1 \xrightarrow{\widehat{\square}} r'_1 \xrightarrow{\widehat{I_{j_2}}} t'_2 \xrightarrow{\widehat{\#}} s'_2 \xrightarrow{\widehat{\square}} r'_2 \cdots$$
$$\cdots r'_{p-1} \xrightarrow{\widehat{I_{j_p}}} t'_p \xrightarrow{\widehat{\#}} s'_p \xrightarrow{\widehat{\square}} z_1$$

We consider first the backward direction of item (1). We have $t_1 \neq_{\widetilde{F}} t'_1$, otherwise there would be a simple cycle $t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\widehat{I_{j_1}}} t'_1$ with label in $\mathbb{I}\widehat{\mathbb{I}} \subseteq K_{\widehat{I\widehat{I}}}$. This implies that $i_1 = j_1$. If this is not the case, we would have a simple path $t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\widehat{I_{j_1}}} t'_1$ with a label in $\sum_{i \neq j} I_i \widehat{I_j} \subseteq M_{\widehat{I\widehat{I}}}$. We claim that $s_1 =_{\widetilde{F}} s'_1$. Note first that $t_1 \neq_{\widetilde{F}} s'_1$ and $t'_1 \neq_{\widetilde{F}} s_1$. Indeed, if $t_1 =_{\widetilde{F}} s'_1$, then there would be a simple path $t'_1 \xrightarrow{\widehat{\#}} s'_1 \xrightarrow{I_{i_1}} x$ with label in $\widehat{\#}\mathbb{I} \subseteq M_{\widehat{I\widehat{I}}}$. If $t'_1 =_{\widetilde{F}} s_1$, then there would be a simple path $x \xrightarrow{\widehat{I_{j_1}}} t'_1 \xrightarrow{\#} t_1$ with label in $\widehat{\mathbb{I}}\# \subseteq M_{\widehat{I\widehat{I}}}$. It follows that $s_1 =_{\widetilde{F}} s'_1$ as otherwise the path $s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\widehat{I_{j_1}}} t'_1 \xrightarrow{\widehat{\#}} s'_1$ would be a simple path with a label in $\#\mathbb{I}\widehat{\mathbb{I}}\widehat{\#} \subseteq M_{\widehat{I\widehat{I}}}$. Finally, we have that $r_1 =_{\widetilde{F}} r'_1$. If this is not true, then $r_1 \xrightarrow{\square} s_1 \xrightarrow{\widehat{\square}} r'_1$ would be a simple path with a label in $\square\widehat{\square} \subseteq M_{\widehat{I\widehat{I}}}$.

We can iterate this argument, replacing in each step the "middle" variable x by the corresponding new "middle" variable r_i (see Figure 12). We obtain the following ($\alpha = \min\{k, p\}$):

- $i_1 = j_1, i_2 = j_2, \ldots, i_\alpha = j_\alpha$
- $t_1 \neq_{\widetilde{F}} t'_1, \cdots, t_\alpha \neq_{\widetilde{F}} t'_\alpha$
- $s_1 =_{\widetilde{F}} s'_1, \cdots, s_\alpha =_{\widetilde{F}} s'_\alpha$
- $r_1 =_{\widetilde{F}} r'_1, \cdots, r_\alpha =_{\widetilde{F}} r'_\alpha$

where $r_k := y_1$ and $r'_p := z_1$. Note that to conclude the I-$\widehat{I}$-condition, it suffices to show that $\alpha = k = p$. Towards a contradiction, suppose first that $k < p$. We know that $y_1 =_{\widetilde{F}} r'_k$, and $r'_k \neq z_1$. In particular, we have at least the following atoms:

$$y'_1 \xrightarrow{\widehat{\#_\infty}} y_1 \qquad r'_k \xrightarrow{\widehat{I_j}} t'_{k+1}$$

We have two cases. If $y'_1 =_{\widetilde{F}} t'_{k+1}$ then we have a simple cycle $y'_1 \xrightarrow{\widehat{\#_\infty}} y_1 \xrightarrow{\widehat{I_j}} t'_{k+1}$ with a label in $\widehat{\#_\infty}\widehat{\mathbb{I}} \subseteq K_{\widehat{I\widehat{I}}}$. On the other hand, if $y'_1 \neq_{\widetilde{F}} t'_{k+1}$ then we have a simple path $y'_1 \xrightarrow{\widehat{\#_\infty}} y_1 \xrightarrow{\widehat{I_j}} t'_{k+1}$ with a label in $\widehat{\#_\infty}\widehat{\mathbb{I}} \subseteq M_{\widehat{I\widehat{I}}}$. In either case, we obtain a contradiction. Suppose now that $k > p$. We know that $r_p =_{\widetilde{F}} z_1$, and $r_p \neq y_1$. We have at least the following atoms:

$$t_{p+1} \xrightarrow{I_i} r_p \qquad z_1 \xrightarrow{\widehat{\#_\infty}} z''_1$$

Again we have two cases. If $t_{p+1} =_{\widetilde{F}} z''_1$ then we have a simple cycle $t_{p+1} \xrightarrow{I_i} r_p \xrightarrow{\widehat{\#_\infty}} z''_1$ with a label in $\mathbb{I}\widehat{\#_\infty} \subseteq K_{\widehat{I\widehat{I}}}$. On the other hand, if $t_{p+1} \neq_{\widetilde{F}} z''_1$ then we have a simple path $t_{p+1} \xrightarrow{I_i} r_p \xrightarrow{\widehat{\#_\infty}} z''_1$ with a label in $\mathbb{I}\widehat{\#_\infty} \subseteq M_{\widehat{I\widehat{I}}}$. We obtain a contradiction in either case. We conclude that $k = p$ and hence the I-$\widehat{I}$-condition holds.

The forward direction of item (1) follows directly from the definition of the I-$\widehat{I}$-condition and inspection of the languages $K_{\widehat{I\widehat{I}}}$ and $M_{\widehat{I\widehat{I}}}$.

Now we turn to item (2). Let $y_1 \xrightarrow{w_I} x$ and $x \xrightarrow{w_a} z_2$ be the expansions associated to the atoms $y_1 \xrightarrow{L_I} x$ and $x \xrightarrow{L_a} z_2$ in the expansion E. Suppose that $w_I = \square \# I_{i_k} \cdots \square \# I_{i_1}$ and $w_a = U_{j_1} \cdots U_{j_p}$ for indices $i_1, \ldots, i_k, j_1, \ldots, j_p \in \{1, \ldots, \ell\}$. Assume that the expansion $y_1 \xrightarrow{w_I} x$ is of the form:

$$y_1 \xrightarrow{\square} s_k \xrightarrow{\#} t_k \xrightarrow{I_{i_k}} r_{k-1} \xrightarrow{\square} s_{k-1} \xrightarrow{\#} t_{k-1} \xrightarrow{I_{i_{k-1}}} r_{k-2} \cdots$$
$$\cdots r_1 \xrightarrow{\square} s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x$$

and the expansion $x \xrightarrow{w_a} z_2$ is of the form:

$$x \xrightarrow{\widetilde{U}_{j_1}} s'_1 \xrightarrow{\blacksquare'} r'_1 \xrightarrow{\widetilde{U}_{j_2}} s'_2 \xrightarrow{\blacksquare'} r'_2 \cdots r'_{p-1} \xrightarrow{\widetilde{U}_{j_p}} s'_p \xrightarrow{\blacksquare'} z_2$$

where $\widetilde{U}_j$ is the word obtained from U_j by removing the last symbol $\blacksquare'$.

We consider first the backward direction of item (2). Suppose the expansion $x \xrightarrow{\widetilde{U}_{j_1}} s'_1$ has the form:

$$x \xrightarrow{\clubsuit} o_1 \xrightarrow{\clubsuit} o_2 \cdots o_m \xrightarrow{\clubsuit} s'_1$$

where $\clubsuit$ is a placeholder representing some symbol. We claim that $t_1 \neq_{\widetilde{F}} t$, for all $t \in \{o_1, \ldots, o_m, o_{m+1}\}$, where $o_{m+1} := s'_1$. Note first that $t_1 \neq_{\widetilde{F}} o_1$, otherwise there would be a simple cycle $t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\clubsuit} o_1$, where $\clubsuit \in \Sigma$. In particular, the label would belong to $\mathbb{I}\Sigma \subseteq K_{Ia}$; a contradiction. Now we argue by induction. Suppose $t_1 \neq_{\widetilde{F}} o_h$, for some $h \in \{1, \ldots, m\}$. By contradiction, assume $t_1 =_{\widetilde{F}} o_{h+1}$. We have a simple path $o_h \xrightarrow{\clubsuit} o_{h+1} \xrightarrow{I_{i_1}} x$, where $\clubsuit \in \Sigma \cup \{\$, \blacksquare, \$'\}$ (note that $\clubsuit$ cannot be $\blacksquare'$). Then the label belongs to $(\Sigma + \$ + \$' + \blacksquare)\mathbb{I} \subseteq M_{Ia}$; a contradiction.

We now claim that $s_1 \neq_{\widetilde{F}} t$, for all $t \in \{o_1, \ldots, o_m\}$. By contradiction, suppose that $s_1 =_{\widetilde{F}} t$, for some $t \in \{o_1, \ldots, o_m\}$. Then there is a simple path $x \xrightarrow{U} t \xrightarrow{\#} t_1$, where $U \in (\Sigma + \$ + \blacksquare)^{1,N}$ (note how we use the fact that $t_1 \neq_{\widetilde{F}} t$, for all $t \in \{o_1, \ldots, o_m\}$; otherwise the path would not be necessarily simple). The label of the simple path belongs to $(\Sigma + \$ + \blacksquare)^{1,N}\# \subseteq M_{Ia}$; a contradiction.

We have that $i_1 = j_1$. If this is not the case, then we would have the simple path $t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\widetilde{U}_{j_1}} s'_1$ with a label in $\sum_i \sum_{j \neq i} I_i \widetilde{U}_j \subseteq M_{Ia}$. Moreover, we have $s_1 =_{\widetilde{F}} s'_1$, otherwise we would have the simple path $s_1 \xrightarrow{\#} t_1 \xrightarrow{I_{i_1}} x \xrightarrow{\widetilde{U}_{j_1}} s'_1$ with a label in $\#\mathbb{I}(\widetilde{U}_1 + \cdots + \widetilde{U}_\ell) \subseteq M_{Ia}$. Finally, we have $r_1 =_{\widetilde{F}} r'_1$. If this is not the case, then we have a simple path $r_1 \xrightarrow{\square} s_1 \xrightarrow{\blacksquare'} r'_1$ with a label in $\square\blacksquare' \subseteq M_{Ia}$.

As in the case of item (1), we can iterate this argument, replacing in each step the "middle" variable x by the corresponding new "middle" variable r_i. We obtain the following ($\alpha = \min\{k, p\}$):

- $i_1 = j_1, i_2 = j_2, \ldots, i_\alpha = j_\alpha$
- For every $j \in \{1, \ldots, \alpha\}$, we have $t_j \neq_{\widetilde{F}} t$, for every internal variable t of the expansion $r'_{j-1} \xrightarrow{\widetilde{U}_{i_j}} s'_j$ (here $r'_0 := x$)
- $s_1 =_{\widetilde{F}} s'_1, \cdots, s_\alpha =_{\widetilde{F}} s'_\alpha$
- $r_1 =_{\widetilde{F}} r'_1, \cdots, r_\alpha =_{\widetilde{F}} r'_\alpha$

where $r_k := y_1$ and $r'_p := z_2$. By using the same arguments as in the case of item (1) we obtain that $k = p$, and hence the I-a-condition holds.

The forward direction of item (2) follows directly from the definition of the I-a-condition and inspection of the languages K_{Ia} and M_{Ia}.

The cases of item (3) and (4) are analogous to cases (1) and (2). □

Let $Q_2^{\cup}$ and $Q_2^{\rightarrow}$ be the following Boolean CRPQs in CRPQ$^{\text{fin}}$:

$$Q_2^{\cup} = x \xrightarrow{K^{\cup}} x \qquad Q_2^{\rightarrow} = y \xrightarrow{M^{\rightarrow}} z$$

where $K^{\cup} := K_{I\widehat{I}} + K_{Ia} + K_{\widehat{a}\widehat{I}} + K_{\widehat{a}a}$ and $M^{\rightarrow} := M_{I\widehat{I}} + M_{Ia} + M_{\widehat{a}\widehat{I}} + M_{\widehat{a}a}$.

From Claim D.1, we obtain the reduction for the case when the right-hand side query is the *union* of the CRPQs $Q_2^{\cup}$ and $Q_2^{\rightarrow}$, which we denote by $Q_2^{\cup} \vee Q_2^{\rightarrow}$.

Claim D.2. *Let F be an a-inj-expansion of Q_1. Then F is well-formed if and only if $Q_2^{\cup} \vee Q_2^{\rightarrow}(F)^{a\text{-}inj} = \emptyset$. Moreover, there is a solution to the PCP instance if and only if $Q_1 \not\subseteq_{a\text{-}inj} Q_2^{\cup} \vee Q_2^{\rightarrow}$.*

From Claim D.2, we obtain the undecidability of containment under atom-injective semantics of a CRPQ in a union of two CRPQs from CRPQ$^{\text{fin}}$. We conclude our proof explaining how to simulate the union $Q_2^{\cup} \vee Q_2^{\rightarrow}$ with a single query $Q_2 \in$ CRPQ$^{\text{fin}}$ as in Figure 11.

We define the following languages:

$$K_{dummy} = (\square + \widehat{\blacksquare} + \widehat{\blacksquare}')(\widehat{\square} + \blacksquare + \blacksquare')$$

$$M_{dummy} = \widehat{\#} + \$ + \$'$$

$$L = \varepsilon + \mathbb{I} + \#\mathbb{I} + \widehat{\#}\widehat{\mathbb{I}} + \square\#\mathbb{I} + \#_\infty + (\Sigma + \$ + \$' + \blacksquare)\,\mathbb{I} +$$
$$\widehat{\Sigma} + \widehat{\#}\widehat{\Sigma} + (\widetilde{V}_1 + \cdots + \widetilde{V}_\ell) + \blacksquare'\,(\widetilde{V}_1 + \cdots + \widetilde{V}_\ell) + \widehat{\$}_\infty +$$
$$+ (\$ + \$')\,\widehat{\Sigma} + (\widehat{\$} + \widehat{\$}')\,\widehat{\Sigma} + (\widehat{\blacksquare} + \widehat{\blacksquare}')(\widehat{\$} + \widehat{\$}')\,\widehat{\Sigma}$$

Let Q_2 be the CRPQ defined as:

$$Q_2 = x \xrightarrow{K} x \wedge y \xrightarrow{L} x \wedge y \xrightarrow{M} z$$

where $K := K^{\cup} + K_{dummy}$ and $M := M^{\rightarrow} + M_{dummy}$. We conclude with the following claim:

Claim D.3. *Let F be an a-inj-expansion of Q_1. Then $Q_2^{\cup} \vee Q_2^{\rightarrow}(F)^{a\text{-}inj} \neq \emptyset$ if and only if $Q_2(F)^{a\text{-}inj} \neq \emptyset$.*

PROOF. For the forward direction, suppose $Q_2^{\cup} \vee Q_2^{\rightarrow}(F)^{a\text{-}inj} \neq \emptyset$. Assume first that $Q_2^{\cup}(F)^{a\text{-}inj} \neq \emptyset$. We consider two cases for the label w of the simple cycle mapping to F and provide an expansion of Q_2 that maps to F:

- $w \in K_{I\widehat{I}} \cup K_{\widehat{a}\widehat{I}}$: take expansion $x \xrightarrow{w} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{\widehat{\#}} z$.
- $w \in K_{Ia} \cup K_{\widehat{a}a}$: take either expansion $x \xrightarrow{w} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{\$} z$ or $x \xrightarrow{w} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{\$'} z$.

Note above that $\varepsilon \in L$ and $\widehat{\#}, \$, \$' \in M_{dummy} \subseteq M$.

Suppose now that $Q_2^{\rightarrow}(F)^{a\text{-}inj} \neq \emptyset$. Again, We consider all the possible cases for the label w of the simple path mapping to F and

provide an expansion of Q_2 that maps to F. We start with the case $w \in M_{I\widehat{I}}$:

- $w = I_p \widehat{I_q} \in \sum_{i \neq j} I_i \widehat{I_j}$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{I_p} x \wedge y \xrightarrow{w} z$.
- $w \in \widehat{\mathbb{I}}\#$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{\#}I_p \in \widehat{\#}\widehat{\mathbb{I}}$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{\widehat{\#}I_p} x \wedge y \xrightarrow{w} z$.
- $w = \#I_p\widehat{I_q}\# \in \#\mathbb{I}\widehat{\mathbb{I}}\#$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{\#I_p} x \wedge y \xrightarrow{w} z$.
- $w \in \square\widehat{\square}$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{\square\#I_i} x \wedge y \xrightarrow{w} z$, for a suitable $I_i \in \mathbb{I}$.
- $w \in \#_\infty \widehat{\mathbb{I}}$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{\#_\infty} x \wedge y \xrightarrow{w} z$.
- $w = I_p\widehat{\#}_\infty \in \mathbb{I}\widehat{\#}_\infty$: take expansion $x \xrightarrow{\square\widehat{\square}} x \wedge y \xrightarrow{I_p} x \wedge y \xrightarrow{w} z$.

Note that $\square\widehat{\square} \in K_{dummy} \subseteq K$ and $I_p, \varepsilon, \widehat{\#}I_p, \#I_p, \square\#I_i, \#_\infty \in L$. For the case $w \in M_{Ia}$ we have the following:

- $w = \clubsuit I_p \in (\Sigma + \$ + \$' + \blacksquare)\,\mathbb{I}$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{\clubsuit I_p} x \wedge y \xrightarrow{w} z$.
- $w \in (\Sigma + \$ + \blacksquare)^{1,N}\#$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{w} z$.
- $w = I_p\widetilde{U}_q \in \sum_i \sum_{j \neq i} I_i\widetilde{U}_j$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{I_p} x \wedge y \xrightarrow{w} z$.
- $w = \#I_p\widetilde{U}_q \in \#\mathbb{I}\,(\widetilde{U}_1 + \cdots + \widetilde{U}_\ell)$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{\#I_p} x \wedge y \xrightarrow{w} z$.
- $w \in \square\blacksquare'$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{\square\#I_i} x \wedge y \xrightarrow{w} z$, for a suitable $I_i \in \mathbb{I}$.
- $w \in \#_\infty \Sigma$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{\#_\infty} x \wedge y \xrightarrow{w} z$.
- $w = I_p\$_\infty \in \mathbb{I}\$_\infty$: take expansion $x \xrightarrow{\square\blacksquare'} x \wedge y \xrightarrow{I_p} x \wedge y \xrightarrow{w} z$.

Observe that $\square\blacksquare' \in K_{dummy} \subseteq K$ and $\clubsuit I_p \in L$, for $\clubsuit \in \Sigma + \$ + \$' + \blacksquare$, and $\varepsilon, I_p, \#I_p, \square\#I_i, \#_\infty \in L$. For the case $w \in M_{\widehat{a}\widehat{I}}$ we have:

- $w \in \widehat{\mathbb{I}}(\widehat{\Sigma} + \widehat{\$} + \widehat{\$}' + \widehat{\blacksquare})$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{\#}\widehat{a} \in \widehat{\#}\widehat{\Sigma}$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widehat{\#}\widehat{a}} x \wedge y \xrightarrow{w} z$.
- $w \in \widehat{\mathbb{I}}\#(\widehat{\Sigma} + \widehat{\$} + \widehat{\blacksquare})$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{w} z$.
- $w = \widetilde{V}_p\widehat{I}_q \in \sum_i \sum_{j \neq i} \widetilde{V}_j\widehat{I}_i$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widetilde{V}_p} x \wedge y \xrightarrow{w} z$.
- $w = \widetilde{V}_p\widehat{I}_q\widehat{\#} \in (\widetilde{V}_1 + \cdots + \widetilde{V}_\ell)\widehat{\mathbb{I}}\widehat{\#}$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widetilde{V}_p} x \wedge y \xrightarrow{w} z$.
- $w \in \widehat{\blacksquare}'\widehat{\square}$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widehat{\blacksquare}'\widetilde{V}_i} x \wedge y \xrightarrow{w} z$, for a suitable $\widetilde{V}_i \in (\widetilde{V}_1 + \cdots + \widetilde{V}_\ell)$.
- $w \in \widehat{\$}_\infty \widehat{\mathbb{I}}$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widehat{\$}_\infty} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{a}\widehat{\#}_\infty \in \widehat{\Sigma}\widehat{\#}_\infty$: take expansion $x \xrightarrow{\widehat{\blacksquare}'\widehat{\square}} x \wedge y \xrightarrow{\widehat{a}} x \wedge y \xrightarrow{w} z$.

Note that $\widehat{\blacksquare}'\widehat{\square} \in K_{dummy} \subseteq K$ and $\varepsilon, \widehat{\#a}, \widetilde{V_p}, \widehat{\blacksquare}'\widetilde{V_i}, \widehat{\$}_\infty, \widehat{a} \in L$. For the case $w \in M_{\widehat{a}a}$ we have ($\clubsuit \in \{\widehat{\blacksquare}, \widehat{\blacksquare}'\}$ and $\spadesuit \in \{\blacksquare, \blacksquare'\}$ are suitable symbols in each case):

- $w = \widehat{p}q \in \sum_{a \neq b} \widehat{ab}$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\widehat{p}} x \wedge y \xrightarrow{w} z$.
- $w \in \Sigma\,(\widehat{\$} + \widehat{\$}')$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\varepsilon} x \wedge y \xrightarrow{w} z$.
- $w = \diamond\widehat{p} \in (\$+\$')\,\widehat{\Sigma}$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\diamond\widehat{p}} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{\diamond p}q\diamond \in (\widehat{\$} + \widehat{\$}')\,\widehat{\Sigma}\,\Sigma\,(\$ + \$')$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\widehat{\diamond p}} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{\bigstar}\bigstar \in (\widehat{\blacksquare} + \widehat{\blacksquare}')(\blacksquare + \blacksquare')$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\widehat{\bigstar}\widehat{\diamond a}} x \wedge y \xrightarrow{w} z$, for suitable $\widehat{\diamond} \in (\widehat{\$} + \widehat{\$}')$ and $\widehat{a} \in \widehat{\Sigma}$.
- $w \in \widehat{\$}_\infty\,\Sigma$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\widehat{\$}_\infty} x \wedge y \xrightarrow{w} z$.
- $w = \widehat{p}\$_\infty \in \widehat{\Sigma}\,\$_\infty$: take expansion $x \xrightarrow{\clubsuit\spadesuit} x \wedge y \xrightarrow{\widehat{p}} x \wedge y \xrightarrow{w} z$.

Note that $\clubsuit\spadesuit \in K_{dummy} \subseteq K$, and $\widehat{p}, \varepsilon, \widehat{\$}_\infty \in L$, and $\diamond\widehat{p}, \widehat{\diamond p}, \widehat{\bigstar}\widehat{\diamond a} \in L$, for $\diamond \in (\$ + \$')$, $\widehat{\diamond} \in (\widehat{\$} + \widehat{\$}')$, and $\widehat{\bigstar} \in (\widehat{\blacksquare} + \widehat{\blacksquare}')$.

For the backward direction, suppose that $Q_2(F)^{a\text{-}inj} \neq \emptyset$. If suffices to show that the expansion of Q_2 mapping to F cannot use simultaneously words in K_{dummy} and M_{dummy}. It is possible to check that any mapping of an expansion $x \xrightarrow{w} x \wedge x \xleftarrow{w'} y \wedge y \xrightarrow{w''} z$, where $w \in K_{dummy}$, $w' \in L$ and $w'' \in M_{dummy}$, maps y to a variable $\bullet$ such that the labels of all outgoing edges of $\bullet$ belongs to the set

$$\{\square, \mathbb{I}, \#, \widehat{\blacksquare}, \widehat{\blacksquare}', \widehat{\Sigma}, \widehat{\$}, \widehat{\$}', \widehat{\square}, \widehat{\mathbb{I}}, \blacksquare, \blacksquare', \Sigma, \#_\infty, \widehat{\#}_\infty, \$_\infty, \widehat{\$}_\infty\}$$

However, this set of symbols is disjoint from M_{dummy} and hence z cannot be mapped to any variable. □

E FULL PROOF OF THEOREM 6.2

We show that even when all languages on the right-hand side are of the form $\{w\}$ with $|w| \leq 2$ we have Π_2^p-hardness for containment. For this, we show how to adapt the proof of Π_2^p-hardness of [15, Theorem 4.3], which shows Π_2^p-hardness for $\mathrm{CRPQ}^{fin}/\mathrm{CQ}$ containment for the standard semantics.[1]

We use a reduction from $\forall\exists$-QBF. The main idea is to use sets $\{t, f\}$ in Q_1 to encode true or false.

More precisely, let

$$\Phi \quad = \quad \forall x_1, \ldots, x_n\, \exists y_1, \ldots, y_\ell\, \varphi(x_1, \ldots, x_n, y_1, \ldots, y_\ell)$$

be an instance of $\forall\exists$-QBF such that φ is quantifier free and in 3-CNF. We construct boolean queries Q_1 and Q_2 such that $Q_1 \subseteq_{a\text{-}inj} Q_2$ if and only if Φ is satisfiable.

The **query** Q_1 is sketched in Figure 13 and built as follows: The basis is an a-path of length 4. We add 4 gadgets E to the outer nodes of the path and one gadget D at the innermost. The choice of 4 E gadgets surrounding the D gadget will be made clear once we discuss Q_2. Basically, the E-gadgets will accept everything while the D-gadget will ensure that the chosen literal evaluates to true. The

---

[1]Actually, it shows hardness for the fragment where the left-hand side can only have regular expressions of the form $a_1 + \cdots + a_n$. In some sense, we simulate disjunction with the choice of an atom-injective expansion for a CQ.

gadgets are also depicted in Figure 13. The gadgets are constructed as follows.

The **gadget** D is constructed such that the root node has one outgoing edge for each variable in Φ, that is, $n+\ell$ many. Each edge is labeled differently, that is, $x_1, \ldots, x_n, y_1, \ldots, y_\ell$. After each x_i-edge we add a t-edges which leads to a different node. From each of these nodes we have cycle of length 2 reading $t\,f$. For each $i \in \{1, \ldots, \ell\}$ we do the following. We add a t-edge to a node we name $y_{i,t}$ after the y_i-edge and an edge labeled f that leads to a node we name $y_{i,f}$. We named these nodes because we need those nodes also in the E-gadgets. Nodes with the same names across gadgets are actually the same node.

Each **gadget** E is constructed similar to the D gadget. The root node has one outgoing edge for each variable in Φ, that is $n+\ell$ many. Each edge is labeled differently, that is $x_1, \ldots, x_n, y_1, \ldots, y_\ell$. After each x_i-edge we add path of length 2 reading $t\,t$- and an f-edge. Each of those edges leads to a different node. After each y_i-edge we add a t-edge and an f-edge to both $y_{i,t}$ and to $y_{i,f}$.

We now explain the construction of Q_2. An example is given in Figure 13. For each clause i, query Q_2 has a small DAG, which might share nodes $(y_{k,tf})$ with the DAGs constructed for the other clauses. For clause i, we construct C_i^1, with an a-edge to the gadget C_i^2, and from there again a a-edge to the gadget C_i^3.

The gadget C_i^j represents the jth literal in the ith clause. Since the QBF is in 3-CNF, we have $j \in \{1, 2, 3\}$. If the literal is the positive variable x_k, C_i^j is a path labeled $x_k\,t\,t$. If it is the negative variable $\neg x_k$, C_i^j is a path labeled $x_k f$. If the literal is the positive variable y_k, C_i^j is a path labeled $y_k t$ and it ends in a node we call $y_{k,tf}$ and, if it is the negative variable $\neg y_k$, C_i^j is a path labeled $y_k f$ and it ends in $y_{k,tf}$, too.

This completes the construction. We will now give some intuition. The gadget D controls via the $\{t, f\}$ (simple) paths, which variables x_i are set to true and which to false. We will consider it *false* whenever there is a $(x_i f)$-path, meaning that the two nodes non-related via r are equal in the $a\text{-}inj$-expansion of Q_1. Otherwise, x_i is set to true. Observe that whenever x_i is false, it is not possible to map in an injective way any path $v \xrightarrow{x_i} v' \xrightarrow{t\,t} v''$ coming from a clause encoded from Q_2. And vice-versa, whenever x_i is true there is no way to map a $(x_i f)$-path. Hence, depending on this, we can either map C_i^j into it or not. The E gadgets are constructed such that every C_i^j can be mapped into it. The choice of the $a\text{-}inj$-expansion of Q_2 determines which path should be mapped into D and, therefore, which literal should be verified. The structure of Q_1 where two E gadgets each surround the D gadget aids in embedding the clauses C_i^1, C_i^2, C_i^3 for each i in $a\text{-}inj$-expansion E_1. If the ith clause is $(x_2 \vee \neg y_1 \vee \neg x_3)$, we have the assignment of f to x_2, t to x_3, then we can embed C_i^1, C_i^3 in the second and third E's, and $\neg y_1$ can be embedded in y_{1f} in D. Embedding $\neg y_1$ in y_{1f} fixes the assignment f to y_1 across all gadgets E, D, and all clauses in Q_2. Likewise, for a clause $(x_1 \vee \neg x_4 \vee y_5)$ in Φ, and an assignment f to x_1, t to x_4 in the canonical model G, we can embed $x_1, \neg x_4$ in the first and second E's and y_5 to the node y_{5t}.

We will now show correctness, that is: $Q_1 \subseteq_{a\text{-}inj} Q_2$ if and only if Φ is satisfiable. Let $Q_1 \subseteq_{a\text{-}inj} Q_2$. Then there exists an injective
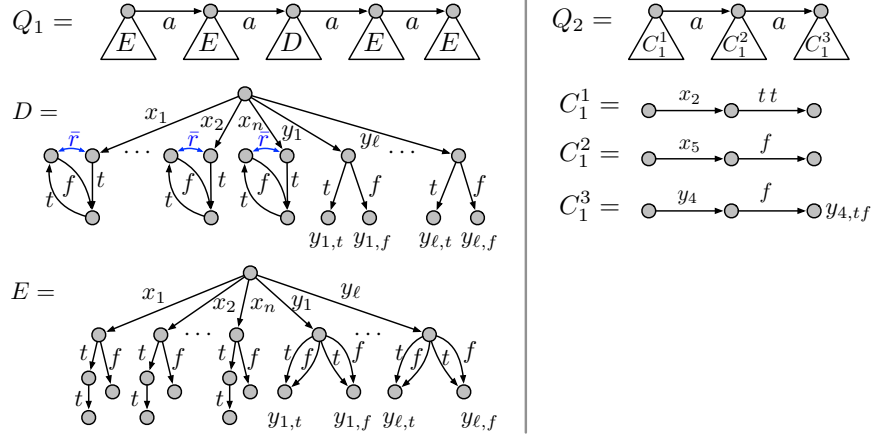
Figure 13: Query Q_1 used in Theorem 6.2 and the gadgets D, and E used in its definition. For the r edge relation (in blue) we depict the edges of its complement (*i.e.*, the edges which are *not* in relation r). Example of Q_2 for the formula $\varphi = (x_2 \vee \neg x_5 \vee \neg y_4)$.

homomorphism from Q_2 to each *a-inj*-expansion of Q_1. The *a-inj*-expansions of Q_1 look exactly like Q_1 except that each some pairs of vertices non-related via r may have been identified together.

Let B be an arbitrary *a-inj*-expansion of Q_1 and D_B the gadget D in B. We define $\theta_B(x_i) = 1$ if there are two distinct vertices non-related via r accessible via x_i, and $\theta_B(x_i) = 0$ otherwise. Let h be an injective homomorphism mapping an *a-inj*-expansion of Q_2 to B. We furthermore define $\theta_B(y_i) = 1$ if h maps $y_{i,tf}$ to $y_{i,t}$ and $\theta_B(y_i) = 0$ otherwise, *i.e.*, if $y_{i,tf}$ is mapped to $y_{i,f}$. We now show that θ_B is well-defined and satisfies φ. It is obvious that each C_i^j will be mapped either to the gadget D_B or to E and that for each $i \in \{1, \ldots, m\}$ exactly one C_i^j is mapped to D_B. If C_i^j corresponds to x_k, i.e., it is a path labeled $x_k t$, then it can only be mapped into D_B if $\theta_B(x_k) = 1$. Analogously, if C_i^j corresponds to $\neg x_k$, it is a path labeled $x_k f$, and can therefore only be mapped into D_B if $\theta_B(x_k) = 0$. If C_i^j corresponds to y_k or $\neg y_k$, it can always be mapped into D_B, but since $y_{k,tf}$ can only be mapped either to $y_{k,t}$ or $y_{k,f}$, we can either map positive y_k into D_B or negative ones, but not both. Therefore, the definition of $\theta_B(y_k)$ is unambiguous, and it indeed satisfies φ.

Since B is arbitrary, we obtain a choice $y_1, \ldots, y_\ell$ for all possible truth-assignments to $x_1, \ldots, x_n$ this way. Therefore, Φ is satisfiable.

For the only if direction let Φ be satisfiable. Then we find for each truth-assignment to $x_1, \ldots, x_n$ an assignment to $y_1, \ldots, y_\ell$ such that $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_\ell)$ is true. Let θ be a function that, given the x_i, returns an assignment for all y_i such that the formula evaluates to true. We will show how to map Q_2 into an arbitrary *a-inj*-expansion B of Q_1.

Let B and θ be given. Let D_B be again the gadget D in B. We use θ to obtain truth-values for $y_1, \ldots, y_\ell$ as follows. Since this assignment is satisfiable, there is a literal that evaluates to *true* in each clause. We map this literal to D_B and the others in this clause to gadgets E. If this literal is x_i, then we can map to the $(x_i\, t\, t)$-path in D_B. If it is $\neg x_i$, then we can map to the $(x_i\, f)$-path in D_B. If the literal is y_i, we can map the $(y_i\, t)$-path ending in $y_{i,tf}$ to D_B. This also implies that each $y_{i,tf}$ in Q_2 is mapped to $y_{i,t}$, which is no problem since each path mapped to E can choose freely between $y_{i,t}$

and $y_{i,f}$ and, since θ is a function, we only have either $\theta(y_i) = 1$ or $\theta(y_i) = 0$. Analogously, if the literal is $\neg y_i$, we can map the $(y_i\, f)$-path ending in $y_{i,tf}$ to D_B, which implies that each $y_{i,tf}$ in Q_2 is mapped to $y_{i,f}$. □

Observe that under standard semantics, the right-hand query Q_2 in the reduction of Theorem 6.2 above would be in fact equivalent to a CQ, but under *a-inj* semantics this is not the case.

F OTHER RESULTS

The following theorem summarizes all other complexity results which complete the picture of complexity results of Figure 1, whose proofs can be found below.

THEOREM F.1 (RESTATEMENT OF THEOREM 6.3).

(1) The CQ/CRPQ and CQ/CQ *containment problems are* NP-*complete under query-injective semantics. (Proposition F.2)*

(2) The CQ/CQ *containment problem under atom-injective semantics is* NP-*complete. (Corollary F.4)*

(3) The CRPQ/CQ and CRPQ^{fin}/CQ *containment problems are* Π_2^p-*hard, under standard and atom-injective semantics. (Proposition F.6)*

(4) The CRPQ/CQ and CRPQ^{fin}/CQ *containment problems are in* Π_2^p, *under all semantics. (Proposition F.7)*

(5) The $\text{CRPQ}/\text{CRPQ}^{\text{fin}}$ *containment problem is* PSPACE-*hard under all semantics. (Proposition F.8)*

(6) The $\text{CRPQ}/\text{CRPQ}^{\text{fin}}$ *containment problem is in* PSPACE *under standard semantics. (Proposition F.9)*

(7) The $\text{CRPQ}^{\text{fin}}/\text{CRPQ}$ *containment problem is in* Π_2^p, *under all semantics. (Proposition F.10)*

PROPOSITION F.2. *The* CQ/CRPQ *and* CQ/CQ *containment problems are* NP-*complete under query-injective semantics.*

PROOF. For the upper bound, let $Q_1(\bar{x})$ be a CQ and $Q_2(\bar{y})$ a CRPQ. Remember that for any $\star \in \{a\text{-}inj, q\text{-}inj\}$, we have that $Q_1 \subseteq_\star Q_2$ iff $\bar{x} \in Q_2(Q_1)^\star$, where Q_1 is seen as a graph database. By Proposition 3.1 we then have NP-membership.

The lower bound follows by a direct reduction from the respective evaluation problem for CQ together with Proposition 3.1. □

Let us call a homomorphism $h : A \to B$ *contracting* if for some $x \xrightarrow{a} y$ in A such that $x \neq y$, we have $h(x) = h(y)$. Observe that the composition of two non-contracting homomorphism is non-contracting.

Lemma F.3. *For any two CQ Q_1, Q_2, the following are equivalent:*
(1) there is a non-contracting homomorphism $Q_2 \to Q_1$,
(2) $Q_1 \subseteq_{a\text{-}inj} Q_2$.

Proof. From top to bottom, let $h : Q_2 \to Q_1$ be a non-contracting homomorphism, that is, such that for every atom $x \xrightarrow{a} y$ of Q_2 we have $h(x) \neq h(y)$. Let $E_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$. Observe that there exists $g : Q_1 \to E_1$ which is not contracting, and hence their composition $g(h) : Q_2 \to E_1$ is non-contracting either. Let U be the conjunction of all atoms $x = y$ such that $g(h(x)) = g(h(y))$, and observe that $E_2 = (Q_2 \wedge U)^{\equiv} \in \text{Exp}^{a\text{-}inj}(Q_2)$. Further, we have $g(h) : E_2 \xrightarrow{inj} E_1$. Summing up, for every $E_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$ there is $E_2 \in \text{Exp}^{a\text{-}inj}(Q_2)$ such that $E_2 \xrightarrow{inj} E_1$, which by the characterization of Proposition 4.6 proves that $Q_1 \subseteq_{a\text{-}inj} Q_2$.

From bottom to top, observe that $Q_1 \in \text{Exp}^{a\text{-}inj}(Q_1)$, and hence by Proposition 4.6 we have that there is some $E_2 \in \text{Exp}^{a\text{-}inj}(Q_2)$ such that $h : E_2 \xrightarrow{inj} Q_1$, which in particular means that h is non-contracting. On the other hand, as argued before, there must be a homomorphism $g : Q_2 \to E_2$ which is non-contracting. Since the composition of non-contracting homomorphisms yields a non-contracting homomorphism, we obtain that $h(g) : Q_2 \to Q_1$ is non-contracting. □

Corollary F.4. *The CQ/CQ containment problem under atom-injective semantics is NP-complete.*

Proof. The upper bound follows from Lemma F.3 above. For the lower bound, it is easy to see that the standard reduction from 3-colorability for CQ under standard semantics [10] still applies in this setting. □

Corollary F.5. *The CQ/CRPQ(A) containment problem under simple-path semantics is NP-complete.*[2]

Proposition F.6. *The CRPQ/CQ and CRPQ^{fin}/CQ containment problems are Π_2^p-hard, under standard and atom-injective semantics.*

Proof. The lower bound for standard semantics follows from [15, Theorem 4.3]. Further, it is easy to see that the Π_2^p-hardness proof of [15, Theorem 4.3] goes through for atom-injective semantics. This is because, in the reduction, all atoms of queries contain languages of words of length 1, and they have no self-loops. Indeed, as a consequence of Lemma F.3, under such restrictive conditions the *a-inj* and standard containment problems coincide. □

Proposition F.7. *The CRPQ/CQ and CRPQ^{fin}/CQ containment problems are in Π_2^p, under all semantics.*

Proof. Let $\star \in \{st, q\text{-}inj, a\text{-}inj\}$. Given Q_1, Q_2, let N be the number of atoms of Q_2. Consider the set S of all expansions of Q_1 where every atom expansion $x \xrightarrow{w} y$ of size greater than $2N$ is replaced with $x \xrightarrow{u\#v} y$, where # is a fresh symbol, and u [resp. v] is the N-prefix [resp. N-suffix] of w. Observe that every element of S is of polynomial size, and that we can check in polynomial time whether any given polysized CQ is in S. Consider the following Σ_2^p algorithm for non-containment. We check that there exists some connected component $\hat{Q}_2$ of Q_2 and element $E_1^{\#} \in S$ such that the following two conditions hold:

(i) $E_1^{\#} \not\subseteq_{\star} \hat{Q}_2$ (which is in co-NP due to Proposition F.2 for *q-inj* and [15, Theorem 4.2] for standard);

(ii) for each atom $x \xrightarrow{u\#v} y$ of $E_1^{\#}$ associated to an atom $x \xrightarrow{L} y$ of Q_1, there is no w such that (a) $uwv \in L$ and (b) $E_1^w \subseteq_{\star} \hat{Q}_2$, where $E_1^w() = x \xrightarrow{uwv} y$.

Since $\hat{Q}_2$ is connected it has to be mapped through a homomorphism [resp. injective homomorphism, *a-inj* homomorphism] either to the N-neighbourhood of a variable of Q_1, or entirely inside an atom expansion. The two items above ensure that none of these cases can occur, and hence that there exists a counter-example for the containment $Q_1 \subseteq_{\star} Q_2$. Observe that in item (ii), if a $\star$-expansion of $\hat{Q}_2$ maps into a directed path, it means that $\hat{Q}_2$ is $\star$-equivalent to a directed path, and hence that w can be taken of polynomial size. This, in turn, means that (ii) can be done in co-NP. □

Proposition F.8. *The CRPQ/CRPQ^{fin} containment problem is PSpace-hard under all semantics.*

Proof. First observe that for Boolean queries Q_1, Q_2 of the form $Q_i() = x \xrightarrow{L_i} y$, we have $Q_1 \subseteq_{st} Q_2$ iff $Q_1 \subseteq_{a\text{-}inj} Q_2$ iff $Q_1 \subseteq_{q\text{-}inj} Q_2$. In [15, Theorem 4.5] it was shown that the containment problem (under standard semantics) for this kind of queries is PSpace-hard, even when L_2 is a star-free expression and the alphabet is of fixed size. □

Proposition F.9. *The CRPQ/CRPQ^{fin} containment problem is in PSpace under standard semantics.*

Proof. Given Q_1, Q_2, let N be the maximum number of atoms of an expansion of Q_2. Consider the set S of all expansions of Q_1, where every expansion $x \xrightarrow{w} y$ of an atom $x \xrightarrow{L} y$ thereof such that $|w| > 2N$ is replaced with $u \cdot \# \cdot v$, where u [resp. v] is the N-prefix [resp. N-suffix] of w, and # is a fresh symbol. The PSpace algorithm then guesses an element $E_1^{\#}$ of S and checks whether there exists some expansion E_1 of Q_1 from which $E_1^{\#}$ could be obtained such that no expansion E_2 of Q_2 can be homomorphically mapped to E_1. For this, we check that there exists some connected component $\hat{Q}_2$ of Q_2 such that the following two conditions hold:

(i) $E_1^{\#} \not\subseteq \hat{Q}_2$ (which is in co-NP [15, Theorem 4.2]);

(ii) for each path of the form $x \xrightarrow{u\#v} y$ in $E_1^{\#}$ associated with the expansion of an atom $x \xrightarrow{L} y$ of Q_1, there is no w such that: (a) $uwv \in L$ and (b) $E_1^w \subseteq \hat{Q}_2$, where $E_1^w() = x \xrightarrow{uwv} y$.

Since $\hat{Q}_2$ is connected it needs to be mapped either to the N-neighbourhood of a variable of Q_1 (ruled out by item i), or entirely inside an atom expansion (ruled out by item ii). On the other hand, $E_1 \not\subseteq Q_2$ iff $E_1 \not\subseteq \hat{Q}_2$ for some component $\hat{Q}_2$. These two items hence ensure that none of these cases can occur, and that there exists a counter-example for the containment $Q_1 \subseteq Q_2$. Observe that item (ii) can be seen as an instance of the intersection emptiness problem for regular languages, that is, the problem of whether $\bigcap_{i \in I} L_i = \emptyset$ for a given set $\{L_i\}_{i \in I}$ of regular expressions, which is a PSpace-complete problem [21]. □

PROPOSITION F.10. *The* CRPQ$^{\text{fin}}$/CRPQ *containment problem is in* Π_2^p, *under all semantics.*

PROOF. Let $\star \in \{st, a\text{-}inj, q\text{-}inj\}$ and let $Q_1(\bar{x}), Q_2(\bar{x})$ be an instance. One can test non-containment by guessing a $\star$-expansion $E_1(\bar{x}) \in \text{Exp}^\star(Q_1(\bar{x}))$ (of linear size) and test that $\bar{x} \notin Q_2(E_1)$ under $\star$ semantics, which is in co-NP by Proposition 3.1. □