Regular path queries on graphs with data: A rigid approach

Zhilin Wu
Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
wuzl@ios.ac.cn

ABSTRACT

Regular path queries (RPQ) is a classical navigational query formalism for graph databases to specify constraints on labeled paths. Recently, RPQs have been extended by Libkin and Vrgoč to incorporate data value comparisons among different nodes on paths, called regular path queries with data (RDPQ). It has been shown that the evaluation problem of RDPQs is PSPACE-complete and NLOGSPACE-complete in data complexity. On the other hand, the containment problem of RDPQs is in general undecidable. In this paper, we propose a novel approach to extend regular path queries with data value comparisons, called rigid regular path queries with data (RRDPQ). The main ingredient of this approach is an automata model called nondeterministic rigid register automata (NRRA), in which the data value comparisons are rigid, in the sense that if the data value in the current position $x$ is compared to a data value in some other position $y$, then by only using the labels (but not data values), the position $y$ can be uniquely determined from $x$. We show that NRRA is robust in the sense that nondeterministic, deterministic and two-way variant of NRRA, as well as an extension of regular expressions, are all of the same expressivity. We then argue that the expressive power of RRDPQs are robust by demonstrating that for every graph database, there is a localized transformation of the graph database so that every RDPQ in the original graph database can be turned into an equivalent RRDPQ over the transformed one. Finally, we investigate the computational properties of RRDPQs and conjunctive RRDPQs (CRRDPQ). In particular, we show that the containment of CRRDPQs (and RRDPQs) can be decided in 2EXPSPACE.

1. INTRODUCTION

Graph data management is a classical research field in database community and has achieved a recent resurgence, with the momentum from new application domains, such as online social networks, bioinformatics, and semantic web. Various query languages have been proposed for graph databases (see [2, 21, 3] for surveys). Among them, regular path queries (RPQ) are basic query formalisms to specify path constraints in graph databases. Graph databases are usually modelled as edge-labeled graphs. A RPQ looks for a pair of nodes connected by a path whose sequence of labels belongs to a regular language ([9]). For the convenience of specifications, RPQs can be extended in a natural way, called RPQs with inverse word symbols (2RPQ), to allow traversing edges in both directions. Since the availability of inverse symbols much eases the specifications, we will focus on path queries with inverse symbols through this paper.

Extensions of RPQs to specify the relationships among multiple paths have been investigated intensively, e.g. conjunctive RPQs (CRPQ) which specify the existence of several paths on the whole ([12, 8, 11]), nested regular expressions where multiple RPQs are organized into a tree structure ([19, 6]), extended CRPQs where regular or rational relations over paths are allowed ([5, 4]).

RPQs have also been extended in another way, called regular path queries with data (RDPQ), to incorporate data value comparisons between two nodes in a path ([18]). RDPQs are interpreted over data graphs, which extend graph databases by assigning a data value to every node. A RDPQ looks for a pair of nodes connected by a path whose sequence of data values and labels is accepted by a nondeterministic register automata (NRA). NRA is an extension of finite state automata, where a fixed number of registers are used to store the data values. Similarly to 2RPQs and CRPQs, RDPQs with inverse (2RDPQ), conjunctive RDPQs (CRDPQ), or conjunctive 2RDPQs (C2RDPQ), can also be defined.

Evaluation and containment are two basic problems for database query languages. These two problems have been investigated extensively for RPQs and 2RPQs, CRPQs and C2RPQs (see [3] for a survey). For RD

PQs, the evaluation problem is PSPACE-complete, and NLOGSPACE-complete in data complexity. On the other hand, the containment problem of RD

PQs is undecidable, as a result of the undecidability of the inclusion problem of NRAs ([18]). Since the containment and equivalence problem are essential for the optimization of queries, this undecidability result of RD

PQs seems to undermine the validity of RD

PQs as a fundamental formalism of path queries that combines the labelling
and data constraints.

Our goal in this paper is to propose an alternative extension of 2RPQs with data value comparisons, called rigid regular data path queries with inverse (2RRDPQ), which, we believe, achieves a good balance between the expressive power and the computational properties (decidability and complexity).

2RRDPQs are based on an automaton model also proposed in this paper, called nondeterministic rigid register automata (NRRA), where the data value comparisons are “rigid” in the sense that if the data value in the current position \( x \) is compared to a data value in some other position \( y \), then by only using the labels (but not data values), the position \( y \) can be uniquely determined from \( x \). With the rigidity constraint, we are able to show that NRRA enjoy nice properties as finite state automata, that is, NRRA can be determined, they are closed under all Boolean operations, the two-way variant of NRRA is expressively equivalent to (one-way) NRRA, and they are expressively equivalent with a natural extension of regular expressions. In addition, while the expressive power of NRRA and NRAs are incomparable, we demonstrate that the expressive power of NRRA can be captured by an extension of NRAs with nondeterministic guessing.

To justify the expressibility of 2RRDPQs, we show that although the expressive power of 2RRDPQs and 2RPQs are incomparable, every 2RPQ can in fact be turned into a 2RRDPQ if a localized transformation is applied to graph databases. By “localized transformation”, we mean that the transformation is obtained by adding for each node \( v \) a new node \( n_v \) which is only connected to \( v \) and the global topology of the original graph is preserved (see Section 4).

We then investigate the computational properties of 2RRDPQs. We show that 2RRDPQs can be evaluated over data graphs with the same (data and combined) complexity as RDPQs. In addition, we consider conjunctive 2RRDPQs (C2RRDPQ) and show that the containment problem of C2RRDPQs can be decided in 2EXPSPACE. From this, we deduce that the containment problem of 2RRDPQs can be decided in 2EXPSPACE as well. The 2EXPSPACE result is proved by a nontrivial extension of the proof for the EXPSPACE-completeness result of C2RPQs in [8], and is the most technical part of this paper.

Related work. The idea of rigidity is inspired by event clock automata from the verification community ([1]), where for every event \( a \), a clock \( x_a \) is used to record the time that has been elapsed from the last occurrence of \( a \), and a clock \( y_a \) is used to predict the time that will elapse until the next occurrence of \( a \). Although in spirit similarly to event clock automata, NRRA are defined to allow much more complicated data value comparisons. For instance, in NRRA, the current data value can be compared to the data value in the position corresponding to the last occurrence of the word symbol \( a \) before the next occurrence of the symbol \( b \). This capability of data value comparisons is essential for the proof of the 2EXPSPACE result of the containment problem of C2RRDPQs (see Section 5). NRAs were introduced in [13]. A restriction of NRAs, window memory automata, has been proposed in [7], where only local data value comparisons are allowed. NRAs strictly extend window memory automata, since non-local data value comparisons are allowed. Various query formalisms have been proposed for data graphs to combine the topology and data constraints, e.g. XPath with data comparisons [15], TriAL for RDF [16]. Although the containment problem of data path queries is in general undecidable, it has been examined in detail for various fragments with positive data value comparisons ([15]).

Organization of this paper. Definitions are given in the next section. NRRA and their variants are presented in Section 3. 2RRDPQs are investigated in Section 4. Section 5 deals with the C2RRDPQs.

2. DEFINITIONS

For a natural number \( k \) such that \( k > 0 \), let \([k]\) denote \( \{1, \ldots, k\} \) and \([-k]\) denote \( \{-k, \ldots, -1\} \).

Fix a finite alphabet \( \Sigma \) and an infinite set of data values \( \mathbb{D} \). Let \( \Sigma^* = \Sigma \cup \{ \alpha^- | \alpha \in \Sigma \} \). For \( \alpha \in \Sigma^* \), we use \( \alpha^- \) to denote the inverse of \( \alpha \). In particular, \( \alpha^- \alpha = \alpha \) for \( \alpha \in \Sigma \).

A word \( w \) over the alphabet \( \Sigma \) is a finite sequence of elements from \( \Sigma \). For a word \( w \), \( |w| \) is used to denote the length of \( w \).

A data path \( \alpha \) over \( \Sigma \) is a sequence \( d_0a_1d_1 \ldots a_n d_n \), where \( d_0, \ldots, d_n \in \mathbb{D} \) and \( a_1, \ldots, a_n \in \Sigma \). The data path of the minimum length is a single data value \( d \). Given two data paths \( \alpha_1 = d_0a_1d_1 \ldots a_n d_n \) and \( \alpha_2 = d_0a_{n+1}d_{n+1} \ldots a_m d_m \), the concatenation of \( \alpha_1 \) and \( \alpha_2 \), denoted by \( \alpha_1 \cdot \alpha_2 \), is defined as the following data path, \( d_0a_1 \ldots a_n d_n a_{n+1} \ldots a_m d_m \). Note that \( \alpha_1 \cdot \alpha_2 \) is defined only if the last data value of \( \alpha_1 \) is the same as the first data value of \( \alpha_2 \). The definition naturally extends to the concatenation of multiple data paths.

A language over \( \Sigma \) is a set of words over \( \Sigma \) and a data language over \( \Sigma \) is a set of data paths over \( \Sigma \).

Let \( \Sigma \) and \( \Gamma \) be two finite alphabets. Then a letter projection \( \text{prj} \) from \( \Sigma \) to \( \Gamma \) is a surjective function from \( \Sigma \) to \( \Gamma \). The letter projections of words, data paths, languages and data languages can be defined in a natural way. For a letter projection \( \text{prj} : \Sigma \rightarrow \Gamma \) and \( \gamma \in \Gamma \), we use \( \text{prj}^{-1}(\gamma) \) to denote the set \( \{ \alpha \in \Sigma | \text{prj}(\alpha) = \gamma \} \). Note that \( \text{prj}^{-1}(\gamma) \subseteq \Sigma \) forms a partition of \( \Sigma \). In addition, for \( B \subseteq \Gamma \), let \( \text{prj}^{-1}(B) = \bigcup_{\gamma \in B} \text{prj}^{-1}(\gamma) \).

A graph database \( G \) is an edge-labeled graph \((V, E)\), where \( V \) is the set of nodes and \( E \subseteq V \times V \times \mathbb{V} \). For \( e = (v, a, v') \in E \), let \( \lambda(e) \) denote the label \( a \). A semipath \( \pi \)
in $G$ is a sequence $v_0a_1v_1 \ldots v_{n-1}a_nv_n$ such that for every $i: 1 \leq i \leq n$, either $(v_{i-1}, a_i, v_i) \in E$ or $(v_i, a_i, v_{i+1}) \in E$. A path $\pi$ in $G$ is a semipath $v_0a_1v_1 \ldots v_{n-1}a_nv_n$ such that for every $i: 1 \leq i \leq n$, $(v_{i-1}, a_i, v_i) \in E$. Let $\lambda(\pi)$ denote the sequence of labels on a semipath $\pi$, that is, $a_1 \ldots a_n$. A semipath $\pi$ is simple if no nodes are repeated on $\pi$.

A regular path query (RPQ) over $\Sigma$ is a tuple $\xi = (x, L, y)$, where $L$ is a regular language over the alphabet $\Sigma$. The regular language $L$ can be given by a finite state automaton or a regular expression. Given a graph database $G = (V, E)$, the evaluation result of $\xi$ over $G$, denoted by $\xi(G)$, consists of the set of pairs $(v, v')$ such that there is a path $\pi$ from $v$ to $v'$ such that $\lambda(\pi) \in L$.

A regular path query with inverse (2RPQ) over $\Sigma$ is a tuple $\xi = (x, L, y)$, where $L$ is a regular language over the alphabet $\Sigma^*$. The semantics of 2RPQs are defined similarly to RPQs, with paths replaced by semipaths.

The evaluation problem for a RPQ or 2RPQ is defined as follows: Given a RPQ or 2RPQ $\xi$, a graph database $\mathcal{G} = (V, E, \eta)$, a node pair $(v, v')$ in $\mathcal{G}$, decide whether $(v, v') \in \xi(\mathcal{G})$.

The containment problem of a RPQ or 2RPQ is defined as follows: Let $\xi_1, \xi_2$ be two RPQs or 2RPQs. Then $\xi_1$ is contained in $\xi_2$, denoted by $\xi_1 \subseteq \xi_2$, if for every graph database $\mathcal{G}$, $\xi_1(\mathcal{G}) \subseteq \xi_2(\mathcal{G})$.

A conjunctive regular path query (CRPQ) $\xi$ over $\Sigma$ is an expression of the form $\text{Ans}(z) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$, where for every $i$, $(y_{2i-1}, L_i, y_{2i})$ is a RPQ over $\Sigma$, and $z$ is a tuple of variables from $\{y_1, \ldots, y_{2l}\}$ (z are called the distinguished variables of $\xi$). Note that in the above definition, $y_i$ and $y_j$ ($i \neq j$) may be the same variable.

Given a graph database $\mathcal{G} = (V, E)$, a CRPQ $\xi := \text{Ans}(z) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$, and $\nu : \{y_1, \ldots, y_{2l}\} \rightarrow V$, we say $(\mathcal{G}, \nu) = \xi$, if $(\nu(y_{2i-1}), \nu(y_{2i}))$ belongs to the evaluation result of $(y_{2i-1}, L_i, y_{2i})$ over $\mathcal{G}$ for every $i: 1 \leq i \leq l$. The evaluation result of $(y_{2i-1}, L_i, y_{2i})$ over $\mathcal{G}$ for every $i: 1 \leq i \leq l$. The evaluation result of $\xi$ over $\mathcal{G}$, denoted by $\xi(\mathcal{G})$, is the set of all tuples $\nu(z)$ for $\nu : \{y_1, \ldots, y_{2l}\} \rightarrow V$ such that $(\mathcal{G}, \nu) = \xi$. Similarly, 2CRPQs can be defined, with RPQs replaced by 2RPQs.

The evaluation and containment problem of CRPQs or 2CRPQs can be defined similarly to RPQs.

A data graph $\mathcal{G}$ is a tuple $(V, E, \eta)$, where $(V, E)$ is a graph database and $\eta: V \rightarrow \mathbb{D}$ assigns each node a data value. For a semipath $\pi = v_0a_1v_1 \ldots v_{n-1}a_nv_n$ in $(V, E)$, the data path corresponding to $\pi$, denoted by $\eta(\pi)$, is $\eta(v_0)a_1\eta(v_1)\ldots a_nv_n$. Let $k$ be a natural number. A $k$-register data constraint is defined by the following rules:

$$c ::= r_i \leftarrow r_j \mid r_i \leftarrow c \mid c \leftarrow c \mid c \circ c,$$

where $0 \leq i, j \leq k$, $i \neq j$, and $r_0$ is a special register reserved for the current data value. Let $\mathcal{C}_k$ denote the set of data constraints.

Let $c$ be a data constraint, $\theta \in (\mathbb{D} \cup \{\bot\})^k$ be the current state of registers (where $\theta(i) = \bot$ denotes the fact that no data value is stored into the register $r_i$), and $d$ be a data value, then the semantics of $c$ is defined over $(\theta, d)$ as follows: If $c = r_i \leftarrow r_j$, then $(\theta, d) \models c$ iff $\theta[0 \leftarrow d(i)] \neq \bot$ and $\theta[0 \leftarrow d(j)] \neq \bot$, and $(\theta[0 \leftarrow d(i)], \theta[0 \leftarrow d(j)])$, where $\theta[0 \leftarrow d]$ is the function extending $\theta$ by assigning $d$ to 0. The semantics of $c = r_i \leftarrow c$ can be defined similarly. In addition, the semantics of $c = c_1 \lor c_2$ and $c = c_1 \land c_2$ are defined in a natural way.

Let $k$ be a natural number. A nondeterministic k-register data path automaton (NRA, [18]) $A$ over $\Sigma^*$ is a tuple $(Q, \delta, I, F)$, where $Q = Q_w \cup Q_{\|=}$ such that $Q_w$ and $Q_{\|=}$ are two finite disjoint sets of word states and data states, $k$ is the number of registers, $I \subseteq Q_{\|=}$ is the set of initial states, $F \subseteq Q_{\|=}$ is the set of final states, $\delta = \delta_{w} \cup \delta_{d}$ such that $\delta_{w} \subseteq Q_w \times \Sigma^* \times Q_{\|=}$ is the word transition relation and $\delta_{d} \subseteq Q_{\|=} \times C_k \times Q_{\|=} \times 2^k$ is the data transition relation.

The intuition of the definition of NRAs is that since data paths alternate between data values and word symbols, when $A$ is in a data (resp. word) state, it is ready to read a data value (resp. a word symbol). Since data paths begin and end with data values, an initial state should be a state before reading a data value, so $I$ is defined as a subset of $Q_{\|=}$, dually, a final state should be a state after reading a data value, so $F$ is defined as a subset of $Q_w$.

Given a data path $\alpha = d_{0}a_{1}d_{1} \ldots a_{n}d_{n}$ and a NRA $A = (Q, k, \delta, I, F)$, a configuration of $A$ on $\alpha$ is a tuple $(q, j, \theta)$, where $q \in Q$, $j$ is the current position (where $j = 0$ means the first position) of the symbol that $A$ reads, and $\theta \in (\mathbb{D} \cup \{\bot\})^k$ is the current state of the registers. An initial configuration of $A$ over $\alpha$ is $(q(0, \theta_1))$, where $q \in I$, and $\theta_1(i) = \bot$ for every $i \in [k]$. Let $(q, j, \theta), (q', j + 1, \theta')$ be two configurations (where $0 \leq j \leq 2n$). Then $(q', j + 1, \theta')$ is said to be a successor of $(q, j, \theta)$, denoted by $(q, j, \theta) \xrightarrow{\alpha} (q', j + 1, \theta')$, if one of the following conditions holds:

- If the $j$-th symbol of $\alpha$ is a word symbol $a$, then $(q, a, q') \xrightarrow{\delta_w}, \theta' = \theta$.
- If the $j$-th symbol of $\alpha$ is a data value $d$, then there are $c \in \mathcal{C}_k, X \subseteq [k]$ such that $(q, c, q', X) \in \delta_d$, $(\theta, d) = c$, and $\theta'$ is obtained from $\theta$ by assigning $d$ to every $i \in X$.

A data path $\alpha = d_{0}a_{1}d_{1} \ldots a_{n}d_{n}$ is accepted by a NRA $A$ if there are $q \in I, q' \in F$ and a data assignment $\theta$ such that $(q, 0, \theta) \xrightarrow{\alpha} (q', 2n + 1, \theta')$, where $\xrightarrow{\alpha}$ is the reflexive and transitive closure of $\xrightarrow{\alpha}$. The set of data paths accepted by $A$ is denoted by $L(A)$. In addition, for every $\theta, \theta' \in (\mathbb{D} \cup \{\bot\})^k$, we use $\xrightarrow{\alpha}(A, \theta, \theta')$ to denote the set of data paths $\alpha = d_{0}a_{1}d_{1} \ldots a_{n}d_{n}$ such that there are $q \in I, q' \in F$ satisfying $(q, 0, \theta) \xrightarrow{\alpha}(q', 2n + 1, \theta')$.

Let $k$ be a natural number. Regular expressions with
k-memory (REM, [18]) over \( \Sigma^* \) are defined by the following rules:
\[
e := e | \emptyset | a | e \cdot e | e \cup e | e^* | \downarrow x e | e[c],
\]
where \( a \in \Sigma^* \), \( e \in \mathcal{E}_a \), and \( X \subseteq \{k\} \).

The semantics of REMs is defined by a relation \( \theta \models_{e,a} \theta' \), where \( e \) is a REM, \( \alpha \) is a data path, \( \theta, \theta' \in (\Sigma \cup \{1\})^{[k]} \). In the following, due to space constraints, we only present the semantics for the last two rules above, that is, \( e = \downarrow x e_1 \) and \( e = e_1[c] \), the semantics of the other rules are obvious and can be found in [18].

- If \( e = \downarrow x e_1 \), then \( \theta \models_{e,c} \theta' \) if \( \theta \models_{x,d} \theta_1 \models_{e_1,a} \theta' \), where \( d \) is the first data value of \( \alpha \), and \( \theta \models_{x,d} \) is obtained from \( \theta \) by assigning \( d \) to all the registers in \( X \).
- If \( e = e_1[c] \), then \( \theta \models_{e,c} \theta' \) if \( \theta \models_{e_1,a} \theta' \) and \( (\theta', d) = c \), where \( d \) is the last data value of \( \alpha \).

A data path \( \alpha \) is accepted by a REM \( e \) if there exists \( \theta \in (\Sigma \cup \{1\})^{[k]} \) such that \( \theta \models_{e,a} \theta \). The set of data paths accepted by a REM \( e \) is denoted by \( \mathcal{L}(e) \). For every \( \theta, \theta' \in (\Sigma \cup \{1\})^{[k]} \), we use \( \mathcal{L}(e, \theta, \theta') \) to denote the set of data paths \( \alpha \) such that \( \theta \models_{e,a} \theta' \).

**Theorem 1** ([10, 18, 17]). The following facts hold for NRAs and REMs.
- NRAs and REMs are expressively equivalent.
- The nonemptiness problem of NRAs and REMs is PSPACE-complete.
- The universality and equivalence problem of NRAs and REMs are undecidable.

A regular path query with data (RDPQ) \( \xi \) over \( \Sigma \) is a tuple \((x, L, y)\), where \( L \) is a language of data paths defined by a NFA or a REM over the alphabet \( \Sigma \). Given a data graph \( \mathcal{G} = (V, E, \eta) \), the evaluation result of \( \xi \) over \( \mathcal{G} \), denoted by \( \xi(\mathcal{G}) \), is the set of node pairs \((v, v')\) in \( \mathcal{G} \) such that there is a path \( \pi \) from \( v \) to \( v' \) such that \( \eta(\pi) \), the path data corresponding to \( \pi \), belongs to \( L \).

A regular path query with inverse data (2RDPQ) \( \xi \) over \( \Sigma \) is a tuple \((x, L, y)\), where \( L \) is a language of data paths defined by a NFA or a REM over \( \Sigma^* \). The semantics of 2RDPQ \( \xi \) over a data graph \( \mathcal{G} = (V, E, \eta) \) is defined similarly to that of RDPQ, with paths replaced by semipaths.

Similarly to 2RPs, regular path queries with inverse data (2RDPQ) can be defined. Moreover, CRDPQs and C2RDPQs can be defined in the same way as CRPs and 2RPs. The evaluation and containment problem of RDPQs, 2RDPQs, CRDPQs, C2RDPQs can also be defined similarly.

**Theorem 2** ([18]). The following results hold for RDPQs, 2RDPQs, CRDPQs and C2RDPQs.
- The evaluation problem of RDPQs and 2RDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.
- The containment problem of RDPQs, 2RDPQs, CRDPQs and C2RDPQs is undecidable.

3. RIGID REGISTER AUTOMATA AND ITS RELATIVES

In this section, we first define nondeterministic rigid register automata (NRRA). Then we show the robustness of this model by proving that NRRA can be determinized and their two-way as well as alternating variants are expressively equivalent to NRRA. We also show that there is a natural extension of regular expressions equivalent to NRRA.

3.1 Rigid data constraints

A position term \( t \) over the alphabet \( \Sigma^* \) is defined by the following rules,
\[
t := cur(t) | suc(t) | pred(t) | suc\_A(t) | pred\_A(t),
\]
where \( A \) is a nonempty subset of \( \Sigma^* \). Intuitively, the constant “cur” denotes the position of the current data value, “suc” and “pred” denote the position of the next and the previous data value, “suc\_A” denotes the position of the data value immediately after the next occurrence of a word symbol from \( A \), dually, “pred\_A” denotes the position of the data value immediately before the previous occurrence of a word symbol from \( A \).

Let \( T_p[\Sigma^*] \) denote the set of position terms over \( \Sigma^* \).

For brevity, position terms of the form \( suc\_A(t) \) or \( pred\_A(t) \) (where \( a \in \Sigma^* \) are written as \( suc\_A(t) \) or \( pred\_A(t) \). In addition, we use \( suc^i \) to denote the repetitions of \( suc \) for \( i \) times. Similarly, we use the abbreviations \( pred^i \), \( suc\_A^i \), and \( pred\_A^i \).

The set of subterms of \( t \in T_p[\Sigma^*] \), denoted by \( sub(t) \), are defined in a natural way, e.g. \( sub(suc\_A(t)) = \{ suc\_A(t) \} \cup sub(t) \). We use \( t' \preceq t \) to denote the fact that \( t' \in sub(t) \), and \( t' \prec t \) to denote the fact that \( t' \preceq t \) and \( t \neq t' \). Suppose \( t, t', t_1 \in T_p[\Sigma^*] \) and \( t' \preceq t \), let \( t[t' / t_1] \) denote the position term obtained from \( t \) by replacing \( t' \) with \( t_1 \).

The semantics of position terms are defined as follows: Give a data path \( \alpha = a_0a_1d_1 \ldots a_n \) and a position \( 2i \) (where \( 0 \leq i \leq n \), \( 2i \) is the position for the data value \( d_1 \), and the first position is indexed by \( 0 \)), the position represented by \( t \) over \( \alpha \) and \( 2i \), denoted by \( t_{\alpha}[2i] \), is defined as follows.
- \( suc_{\alpha}[2i] = 2i. \)
- If \( i < n \), then \( suc(suc_{\alpha})_{\alpha}[2i] = 2(i+1) \). Otherwise, \( suc(suc_{\alpha})_{\alpha}[2i] = 1 \).
- If \( i > 0 \), then \( pred(pred_{\alpha})_{\alpha}[2i] = 2(i - 1) \). Otherwise, \( pred(pred_{\alpha})_{\alpha}[2i] = 1 \).
• If \((t_1)_\alpha[2i] \neq 1\), then
  \((\text{succ}(t_1))_\alpha[2i] = (\text{succ}(\text{cur}))_\alpha[(t_1)_\alpha[2i]]\).
  Otherwise, \((\text{succ}(t_1))_\alpha[2i] = 1\).

• If \((t_1)_\alpha[2i] \neq 1\), then
  \((\text{pred}(t_1))_\alpha[2i] = (\text{pred}(\text{cur}))_\alpha[(t_1)_\alpha[2i]]\).
  Otherwise, \((\text{pred}(t_1))_\alpha[2i] = 1\).

• If there exists \(j : i < j \leq n\) such that \(a_j \in A\) and \(j\) is the minimum number satisfying this condition, that is, for every \(j' : i < j' < j\), we have \(a_{j'} \notin A\), then \((\text{succ}(\text{cur}))_\alpha[2i] = 2j\). Otherwise, \((\text{succ}(\text{cur}))_\alpha[2i] = 1\).

• If there exists \(j : j \leq i\) such that \(a_j \in A\) and \(i\) is the maximum number satisfying this condition, that is, for every \(j' : j < j' < i\), we have \(a_{j'} \notin A\), then \((\text{pred}(\text{cur}))_\alpha[2i] = 2(j - 1)\). Otherwise, \((\text{pred}(\text{cur}))_\alpha[2i] = 1\).

• If \((t_1)_\alpha[2i] \neq 1\), then
  \((\text{succ}(t_1))_\alpha[2i] = (\text{succ}(\text{cur}))_\alpha[(t_1)_\alpha[2i]]\).
  Otherwise, \((\text{succ}(t_1))_\alpha[2i] = 1\).

• If \((t_1)_\alpha[2i] \neq 1\), then
  \((\text{pred}(t_1))_\alpha[2i] = (\text{pred}(\text{cur}))_\alpha[(t_1)_\alpha[2i]]\).
  Otherwise, \((\text{pred}(t_1))_\alpha[2i] = 1\).

Example 1. Suppose
\[
\alpha = \begin{bmatrix} a & d_0 & a & d_1 & b & d_2 & a & d_3 & a & d_4 & b & d_5 \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{bmatrix}
\]
where the second arrow is the sequence of positions. Let \(t_1 = \text{succ}_a(\text{succ}(\text{cur}))\) and \(t_2 = \text{pred}_b(\text{pred}(\text{cur}))\). Let us consider \((t_1)_\alpha[0]\) and \((t_2)_\alpha[10]\). At first, \((t_1)_\alpha[0] = (\text{succ}_a(\text{cur}))_\alpha[(\text{succ}(\text{cur}))_\alpha[0]]\). Since \((\text{succ}(\text{cur}))_\alpha[0] = 2\), the first occurrence of a is the position \(2\) is in the position \(5\), we get \((t_1)_\alpha[0] = 6\). On the other hand, \((t_2)_\alpha[10] = (\text{pred}_b(\text{cur}))_\alpha[(\text{pred}(\text{cur}))_\alpha[10]]\). Because \((\text{pred}(\text{cur}))_\alpha[10] = 8\) and the last occurrence of \(b\) before the position \(8\) is in the position \(3\), we get \((t_2)_\alpha[10] = 2\).

A rigid data constraint \(c\) over \(\Sigma^*\) is defined by the following rules,
\[
c := t_1 \sim t_2 | t_1 \sim t_2 | c \lor c | c \land c,\quad \text{where} \ t_1, t_2 \in \mathcal{T}_\delta[\Sigma^*].
\]

We use \(C_{\text{rigd}}[\Sigma^*]\) to denote the set of rigid data constraints over \(\Sigma^*\).

The semantics of rigid data constraints can be defined inductively. In the following, we will define the semantics for the case \(c = t_1 \sim t_2\). The semantics of \(c = t_1 \sim t_2\) can be defined similarly. Moreover, the semantics of \(c = c_1 \lor c_2\) and \(c = c_1 \land c_2\) can be defined in a standard way. Let \(c \in C_{\text{rigd}}[\Sigma^*]\), \(\alpha = a_0d_1a_1 \ldots a_n\), and \(i : 0 \leq i \leq n\), then \((\alpha, 2i)\) is said to satisfy \(c = t_1 \sim t_2\), denoted by \((\alpha, 2i) \models c\), if \((t_1)_\alpha[2i] \neq 1\), \((t_2)_\alpha[2i] \neq 1\), and \(d_{(t_1)_\alpha[2i]} = d_{(t_2)_\alpha[2i]}\).

Given a rigid data constraint \(c\), we use \(\bar{c}\) to denote the negation of \(c\). More specifically, \(\bar{c}\) is obtained from \(c\) by swapping \(\sim\) for \(\lor\), and \(\lor\) for \(\land\). For instance, if \(c = \text{cur} \sim \text{succ}(\text{cur}) \lor \text{cur} \sim \text{pred}(\text{cur})\), then \(\bar{c} = \text{cur} \land \text{succ}(\text{cur}) \land \text{cur} \land \text{pred}(\text{cur})\).

**Proposition 1.** The satisfiability problem of rigid data constraints is NP-complete.

### 3.2 Non-deterministic and deterministic rigid register automata

A non-deterministic rigid register automaton (NRRA) \(A\) over the alphabet \(\Sigma^*\) is a tuple \((Q, \delta, I, F)\), where \(Q, F\) are as those in NRA, \(\delta = \delta_w \cup \delta_d\) such that \(\delta_w \subseteq Q_w \times \Sigma^* \times Q_d\) and \(\delta_d \subseteq Q_d \times C_{\text{rigd}}[\Sigma^*] \times Q_w\).

A run of \(A\) over a data path \(\alpha = a_0d_1a_1 \ldots a_n\) is a state sequence \(q_0a_0q_1a_1 \ldots q_{2n-1}a_nq_{2n}q_{2n+1}\) such that \(q_0 \in I\), for every \(1 \leq i \leq n\), \((q_{2i-1}, a_i, q_{2i+1}) \in \delta_d\) and \((\alpha, 2i) \models c_i\), and for every \(1 \leq i \leq n\), \((q_{2i-1}, a_i, q_{2i}) \in \delta_w\). A run \(\rho = q_0c_0q_1c_1 \ldots q_{2n-1}a_nq_{2n}q_{2n+1}\) is accepting if \(q_{2n+1} \in F\).

A deterministic rigid register automaton (DRRA) over \(\Sigma^*\) is a NRRA \(A = (Q, \delta, I, F)\) such that \(I\) is a singleton, and \(\delta\) satisfies that for every \(q \in Q_w, a \in \Sigma^*\), there is at most one \(q' \in Q_d\) such that \((q, a, q') \in \delta_w\), and for every \((q, c_1, q_1), (c_2, c_2) \in \delta_d\), if \(q_1 \neq q_2\), then \(c_1 \land c_2\) is unsatisfiable.

Let \(A\) be a NRRA. Then \(\mathcal{T}_A\) is used to denote the minimal set of position terms satisfying that for every \(t_1 \sim t_2\) or \(t_1 \not\sim t_2\) occurring in \(A\), we have \(t_1, t_2 \in \mathcal{T}_A\); moreover, if \(t \in \mathcal{T}_A\) and \(t' \not\sim t\), then \(t', [t \text{\lor cur}] \in \mathcal{T}_A\).

In addition, \(C_A\) is used to denote the set of rigid data constraints occurring in \(A\).

**Example 2.** Let \(\Sigma = \{a, b\}\). Let \(L\) denote the language of data paths satisfying that the sequence of word symbols on the data path belongs to ab* a, the first data value occurs in some other position, and the last data value does not occur elsewhere. Then \(L\) is defined by the NRRA \(A\) illustrated in Figure 1, where \(\text{succ}_a\) is an abbreviation of \(\text{succ}(\text{cur})\). \(Q_d = \{q_0, q_2, q_4, q_6\}\) and \(Q_w = \{q_1, q_3, q_5, q_7, q_9, q_{11}\}\).

Since NRRA are able to compare the current data value with the data values in the future, NRAs and NRRA are expressively incomparable.

**Proposition 2.** NRA and NRRA are expressively incomparable.

Let \(A = (Q, \delta, I, F)\) be a NRA over the alphabet \(\Sigma^*\) and \(prj\) a letter projection from \(\Sigma^*\) to \(\Gamma\). Then the letter projection of \(A\), denoted by \(prj(A)\), is obtained from \(A\) by replacing each transition \((q, a, q') \in \delta_w\) with \((q, prj(a), q')\), and each \(\text{succ}_a\) (resp. \(\text{pred}_d\)) occurring in \(\delta_d\), where \(A \subseteq \Sigma^*\), with \(\text{succ}_{prj(A)}\) (resp. \(\text{pred}_{prj(A)}\)).

Note that \(prj(A)\) may not define \(prj(\mathcal{Z}(A))\), as witnessed by the following result.
Figure 1: An example for NRRA

Proposition 3. The class of languages definable by NRRA is not closed under letter projections.

In the following, we will introduce a constraint for a pair \( (A, prj) \), where \( A \) is a NRA and \( prj \) is a letter projection, so that \( prj(A) \) does define \( prj(\mathcal{L}(A)) \).

Let \( A = (Q, \delta, I, F) \) be a NRA over \( \Sigma^* \), \( prj \) be a letter projection from \( \Sigma^* \) to \( \Gamma \). Then \( A \) is said to be position-invariant under \( prj \) if for every \( suc(A) \) occurring in \( \delta \), where \( \Sigma \subseteq \Sigma^* \), there is \( B \subseteq \Gamma \) such that \( A = prj^{-1}(B) \). It is easy to observe that the position-invariance guarantees that for every \( t \in T_A \), every data path \( a \) and every position \( 2i \) of \( a \), it holds \( t_\alpha[2i] = (prj(t))_{prj(\alpha)}[2i] \), where \( prj(t) \) is obtained from \( t \) by replacing each occurrence of \( suc(A) \) (resp. \( pred(A) \)) in \( t \) with \( suc(prj(A)) \) (resp. \( pred(prj(A)) \)). From this, we deduce that if \( A \) is position invariant under \( prj \), then \( prj \) does not affect the interpretations of the rigid data constraints in \( A \). So we have the following result.

Proposition 4. Suppose \( A \) is a NRA over \( \Sigma^* \) and \( prj \) is a letter projection from \( \Sigma^* \) to \( \Gamma \). If \( A \) is position-invariant under \( prj \), then \( \mathcal{L}(prj(A)) = prj(\mathcal{L}(A)) \).

For a NRA, in every position, only a bounded number of data values occurring before this position are stored into the registers for the future references. On the other hand, in the first sight, in a NRA, it is only required that a bounded number of positions are referenced to by a data transition in a single position, but it is not required that only a bounded number of positions are referenced to by \( all \) the data transitions after a position. In the following, we show that this is indeed the case. By utilizing this property, we then show that NRRA can be simulated by an extension of NRAs with nondeterministic guessing.\(^1\)

Let \( A = (Q, \delta, I, F) \) be a NRA over the alphabet \( \Sigma^* \), \( \alpha = a_0a_1a_2 \ldots a_n \) be a data path over \( \Sigma^* \), \( \rho = q_0q_1q_2 \ldots q_{2n-1}^\alpha a_0q_2nq_{2n+1}^\alpha \) be a run of \( A \) over \( \alpha \), and \( i : 0 \leq i \leq n \). Define the set of future positions of the position \( 2i \) of \( \alpha \) with respect to \( \rho \), denoted by \( Pos^\alpha_{\rho}[\alpha, 2i] \), as

\[
\{ t_\alpha[2j] | j \leq i, t \text{ occurs in } c_j, t_\alpha[2j] \neq \bot, t_\alpha[2j] > 2i \}.
\]

Similarly, define the set of past positions of the position \( 2i \) of \( \alpha \) with respect to \( \rho \), denoted by \( Pos^\alpha_{\rho}[\alpha, 2i] \), as

\[
\{ t_\alpha[2j] | i \leq j, t \text{ occurs in } c_j, t_\alpha[2j] \neq \bot, t_\alpha[2j] < 2i \}.
\]

Lemma 1. Let \( A = (Q, \delta, I, F) \) be a NRA over the alphabet \( \Sigma^* \), \( \alpha = a_0a_1a_2 \ldots a_n \) be a data path, then for every run \( \rho \) of \( A \) over \( \alpha \) and every \( i : 0 \leq i \leq n \),

\[
Pos^\rho_{\rho}[\alpha, 2i] \cup Pos^\rho_{\rho}[\alpha, 2i] \subseteq \{ t_\alpha[2j] | t \in T_A \}.
\]

Intuitively, Lemma 1 says that for every run \( \rho \) over a data path \( \alpha \) and every position \( 2i \) of \( \alpha \), only a bounded number of positions before (resp. after) the position \( 2i \) are referred to by \( \rho \) after (resp. before) reaching the position \( 2i \).

A nondeterministic register data path automaton with guessing (NRAG) \( A \) over \( \Sigma^* \) is a tuple \( (Q, k, I, F) \), where \( Q, k, I, F \) are as those in the definition of NRA, \( \delta \subseteq \delta_w \cup \delta_d \subseteq \delta_w \subseteq Q_w \times \Sigma^* \times Q_d \) and \( \delta_d \subseteq Q_d \times C_k \times Q_w \times \Sigma^* \times C_k \).

The intuition of a transition \( (q, c, q', X, Y, c') \in \delta \) is that if the current state is \( q \), the data values stored in the registers together with the current data value \( d \) satisfies \( c \), then the state is changed to \( q' \), \( d \) is stored into every register in \( X \). Meanwhile, for each register in \( Y \), a data value is nondeterministically guessed. In addition, the guessed data values should satisfy the constraint \( c' \).

The semantics of NRAGs are defined similarly as those of NRAs, that is, a successor relation of configurations \( (q, j, \theta) \mapsto (q', j + 1, \theta) \) can be defined, with the following adjustment for data transitions.

If the \( j \)-th symbol is a data value \( d \), then there exist \( c, c' \) such that \( (q, c, q', X, Y, c') \in \delta_d \), and \( \theta' \) is obtained from \( \theta \) as follows,

- for each \( i \in X \), \( d \) is assigned to \( i \) (thus \( \theta'(i) = d \)),
- for each \( i \in Y \), a data value \( d' \) is guessed (thus \( \theta'(i) = d' \)), so that the guessed data values satisfy the following condition: The function \( \theta_g \) extending \( \theta \) by assigning \( d' \) to \( k + 1 \) for each \( i \in Y \) satisfies that \( (\theta_g, d) = c' \),
- for each \( i \notin X \cup Y \), \( \theta'(i) = \theta(i) \).

Note that data values are not allowed to be copied explicitly among the registers in NRAG. But this can be achieved by guessing. For instance, if we want to copy a data value from \( r_i \) to \( r_j \), then we can guess a data value for \( r_j \) and add the constraint \( r_i \sim r_{k+j} \) for the guessing. Later on, when we mention copying a data value from a register to the other, we always mean the implicit copying by guessing.

Since the nonemptiness of NRAGs can be solved similarly to that of NRAs, we have the following result.
Proposition 5. The nonemptiness problem of NRAG is PSPACE-complete.

In the following, we will show that the expressive power of NRRAs can be captured by NRAGs.

Theorem 3. From a NRRRA $A = (Q, \delta, I, F)$, an equivalent NRRA $B = (Q', k, \delta', I', F')$ can be constructed such that $|Q'|$ is polynomial over $|Q|$ and exponential over $|T_A|$ and $k$ is polynomial over $|T_A|$.

We will present a proof sketch for Theorem 3 and illustrate the main ideas. These ideas are also used for the proof of Theorem 6 in Section 5.

Proof. Let $A = (Q, \delta, I, F)$ be a NRRRA. In the following, we will construct a NRRA $B$ to simulate $A$.

We first give an intuitive description of the construction. Let $\rho$ be a run of $A$ over a data path $\alpha$. Then in the position $2i$, $B$ simulates $\rho$ as follows: $B$ records the data values in the positions belonging to $\operatorname{Pos}[\rho][2i]$, guesses the data values in the positions belonging to $\operatorname{Pos}[\rho][2i]$, and records the order for the positions in $\operatorname{Pos}[\rho][2, 2i]$ and $\operatorname{Pos}[\rho][2, 2i]$.

We introduce some additional notations.

Let $\alpha = d_0a_1d_1...d_{n-1}a_n$ be a data path and $i : 0 \leq i \leq n$. The profile of the position $2i$ in $\alpha$, denoted by $\operatorname{prof}_{2i}(\alpha)$, is defined as a triple $(S, \chi, \sim)$, where

- $S = \{t \in T_A | t \operatorname{cur}[2i] = 1\}$,
- $\chi$ is a sequence
  
  $$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})$$
  $$(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$
  
  such that
  - for every $j : -m_1 \leq j \leq m_2$, $T_j \notin S$ and $T_j \neq \varnothing$,
  - the collection $\{T_{-m_1}, \ldots, T_0, \ldots, T_{m_2}\}$ forms a partition of $S$, and $\operatorname{cur} \in T_0$,
  - for every $t, t' \in S$, if $t \operatorname{cur} \in T_j$ and $t' \operatorname{cur} \in T_k$, then $j < k$ if $\operatorname{prof}[2i](t) \leq \operatorname{prof}[2i](t')$ (in particular, $j = k$ if $\operatorname{prof}[2i](t) = \operatorname{prof}[2i](t')$),
  - for every $j : -m_1 < j < m_2$, $b_j = a(t_0[2i]) / 2$ for some $t \in T_j$, and $b_{m_1} = a(t_0[2i]) / 2$ if $t_0[2i] > 0$ for some $t \in T_{m_1}$, otherwise $b_{m_1} = 1$,
  - for every $j : -m_2 < j < m_2$, $b'_j = a(t_0[2i]) / 2 + 1$ for some $t \in T_j$, and $b'_{m_2} = a(t_0[2i]) / 2 + 1$ if $t_0[2i] < 2n$ for some $t \in T_{m_2}$, otherwise $b'_{m_2} = 1$,
  - $s_{m_2} = 1$, and for every $j : -m_2 \leq j < m_2$, if $t_0[2i] = t_0[2i] + 1$ for some $t \in T_j$ and $t' \operatorname{cur} \in T_{j+1}$, then $s_{j+1} = 1$, otherwise $s_{j+1} = 0$.

- $\sim$ is an equivalence relation over $S$ defined as follows: Let $t, t' \in S$, then $t \sim t'$ if $d_{t_0[2i]} = d_{t'_0[2i]}$.

Let $\Sigma_{\operatorname{prof}}$ denote the set of all triples $(S, \chi, \sim)$ such that $S \subseteq T_A$.
then after the position $2i$ is visited (that is, the reading head is in $2i+1$), for each $j : -m_{i+2} \leq j \leq m_{i+2}$, $B$ stores in the register $r_{j \mid |T_A|}$ the data value corresponding to $T_{ij}$. In particular, $B$ stores the data value $d_i$ in $r_{|T_A|}$.

- Over each pair of positions $2i$ and $2(i+1)$ (where $0 \leq i < n$), $B$ checks that $\pi_i \xrightarrow{\text{alt alt}} \pi_{i+1}$. To do this, $B$ copies (by guessing) data values between registers and guesses some data values for a few registers.

  - At the same time, $B$ simulates the run of $A$ as follows.

    - If $A$ makes a transition $(q, a_i, q')$ over $a_i$, then $B$ checks that $b'_{i-1,0} = a_i$ and changes the state from $q$ to $q'$.

    - If $A$ makes a transition $(q, c, q')$ over $d_i$, then $B$ checks that $\pi_i$ satisfies $c$, verifies that $d_i$ is equal to the data value stored in $r_{j \mid |T_A|}$ for each $j : -m_{i+1} \leq j \leq m_{i+1}$ such that there is $t \in T_j$ satisfying $\text{car -dir} t$ (in particular, $d_i$ should be equal to the data value in $r_{|T_A|}$), and changes the state from $q$ to $q'$.

    - $B$ accepts if $A$ accepts and a final profile is reached.

From the above construction, we know that in its states, $B$ should record the states of $A$ and the guessed profiles. Therefore, the number of states of $B$ is polynomial over $|Q|$ and exponential over $|T_A|$. □

**Proposition 6.** The nonemptiness of NRRAs and DRRAs is PSPACE-complete.

By using a slight extension of the subset construction, we are able to show that NRRA can be determinized.

**Proposition 7.** For every NRRA $A$, there is an equivalent DRRA of exponential size.

**Corollary 1.** NRRAs are closed under all Boolean operations.

**Corollary 2.** The language inclusion problem of NRRAs is PSPACE-complete.

### 3.3 Two-way nondeterministic rigid register automata

In this subsection, we will show that two-way nondeterministic rigid register automata are of the same expressibility as NRRA.

A two-way nondeterministic rigid register automaton (2NRRA) $A$ over $\Sigma^*$ is a tuple $(Q, \tau, \delta, I, F)$, where $Q, I, F$ are as those in the definition of NRRAs, $\tau, \delta \subseteq \Sigma^*$ are respectively the left and right endmarkers, $\delta = \delta_w \cup \delta_d$ such that

- $\delta_w \subseteq Q \times (\Sigma^* \cup \{-\}, \bullet) \times Q \times \{1, -1\}$ (where $1, -1$ denote the direction of the head: “right” and “left”) satisfies that for every transition $(q, \tau, \gamma, \text{dir}) \in \delta_w$ (resp. $(q, \tau, \gamma', \text{dir}) \in \delta_w$), it holds that $\text{dir} = +1$ (resp. $\text{dir} = -1$),

- $\delta_d \subseteq Q \times \text{Crgd} \times Q \times \{1, -1\}$.

Let $\alpha = d_0 a_1 d_1 \ldots a_n d_n$ be a data path and $A$ be a 2NRRA. A run of $A$ over $\alpha$ is a sequence $(q_0, i_0) \theta_0 (q_1, i_1) \theta_1 \ldots \theta_{m-1} (q_m, i_m)$ such that $q_0 \in I$, $i_0 = 0$, $i_m = 2n + 2$,

- for every $j : 0 \leq j < m$, if the symbol of $\tau \alpha_i - \text{in}$ in the position $i_j$ is a word symbol $\alpha \in \Sigma^* \cup \{-\}$, then there is $\text{dir} \in \{1, -1\}$ such that $(q_j, \alpha, q_{j+1}, \text{dir}) \in \delta_w$, $\theta_j = \alpha$, and $i_j + 1 = i_j + \text{dir}$,

- for every $j : 0 \leq j \leq m$, if the symbol of $\tau \alpha_i - \text{in}$ in the position $i_j$ is a data value $\text{dir} \in \text{Crgd}$ and $\text{dir} \in \{1, -1\}$ such that $(q_j, c, q_{j+1}, \text{dir}) \in \delta_d$, $(\alpha, i_j - 1) \equiv c, \theta_j = c$, and $i_j + 1 = i_j + \text{dir}$.

A run is accepting if $q_m \in F$. Note that a run of a 2NRRA over $\alpha$ starts at the left endmarker (position 0) and stops at the right endmarker (position $2n + 2$).

**Proposition 8.** For every 2NRRA, there is an equivalent NRRA of exponential size.

### 3.4 Rigid regular expressions with memory

Rigid regular expressions with memory (RREM) is defined by the following rules,

$e ::= e \mid a \mid [c] \mid e \cdot e \mid e^+$, where $c \in \text{Crgd}[\Sigma^*]$.

Let $e$ be a RREM, $\alpha = d_0 a_1 d_1 \ldots a_n d_n$ be a data path, and $i, j : 0 \leq i \leq j \leq 2n$. The semantics of $e$ is defined by a relation $(\alpha, i) \rightarrow_e (\alpha, j)$ as follows.

- If $e = e$, then $(\alpha, i) \rightarrow_e (\alpha, j)$ if $i = j$ and the symbol of $\alpha$ at position $i$ is a data value (thus $i$ is even).

- If $e = a$, then $(\alpha, i) \rightarrow_e (\alpha, j)$ if $i = j + 2$, the symbol of $\alpha$ at position $i + 1$ is $a$.

- If $e = [c]$, then $(\alpha, i) \rightarrow_e (\alpha, j)$ if $i = j$, the symbol of $\alpha$ at position $i$ is a data value (thus $i$ is even), and $(\alpha, i) \equiv c$.

- The semantics for the rules $e_1 \cup e_2$, $e_1 \cdot e_2$ and $e_1^+$ are defined in a natural way and are omitted.

A data path $\alpha = d_0 a_1 d_1 \ldots a_n d_n$ is accepted by a RREM $e$ if $(\alpha, 0) \rightarrow_e (\alpha, 2n)$. Let $\mathcal{L}(e)$ denote the set of data paths accepted by a RREM $e$.

**Proposition 9.** NRRAs and RREM have the same expressive power.

- From a RREM $e$, a NRRA $A_e$ can be constructed in LOGSPACE such that $\mathcal{L}(e) = \mathcal{L}(A_e)$. 
• From a NRRA $A$, a RREM $e_A$ can be constructed in EXPTIME such that $L(A) = L(e_A)$.

**Corollary 3.** The nonemptiness problem of RREMs is PSPACE-complete.

### 4. RIGID REGULAR PATH QUERIES WITH DATA

A rigid regular path query with inverse and data (2RDPQ) $\xi$ over the alphabet $\Sigma$ is a tuple $(x, L, y)$ where $L$ is a language of data paths defined by a NRRA or a RREM over $\Sigma^\ast$.

Given a data graph $G = (V, E, \eta)$ and a RRDQP $\xi = (x, L, y)$, the evaluation result of $\xi$ over $G$, denoted by $\xi(G)$, is the set of all pairs $(v, v')$ such that there is a semipath $\pi$ from $v$ to $v'$ in $G$ such that $\eta(\pi) \in L$.

**Proposition 10.** The evaluation problem of 2RDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.

In the following we will show that every 2RDPQ can be turned into a 2RRDPQ, if data graphs are transformed in a natural way. Note that the transformation of data graphs presented in the following is localized in the sense that for each node, a new node is added and connected to the node by edges with special labels, and the relationships between the nodes in the original data graph are not changed.

Let $G = (V, E, \eta)$ be a data graph over the alphabet $\Sigma$, $k \geq 1$, and $\{A_i \mid 1 \leq i \leq k\} \cap \Sigma = \emptyset$. Then the data-to-node $k$-transformation $\xi$ of $G$, denoted by $\xi(G) = (V_{dn,k}, E_{dn,k}, \eta_{dn,k})$, is defined as follows.

- $V_{dn,k}$ is obtained from $V$ by adding a new node $n_v$ for each node $v \in V$,
- $E_{dn,k}$ is defined as the union of $E$ and the set of edges $(v, A_i, n_v)$ for every $v \in V$ and $i : 1 \leq i \leq k$,
- for each $v \in V$, $\eta_{dn,k}(v) = \eta(v)$ and $\eta_{dn,k}(n_v) = \eta(v)$.

The intuition of the above transformation is to copy the data value of each node $v$ to a new node connected to $v$ with $k$ edges. Note that the transformation does not change the edges between nodes in the original graph.

**Theorem 4.** Let $k \geq 1$, $\xi = (x, L, y)$ be a 2RDPQ over the alphabet $\Sigma$ such that $L$ is given by a NRA or REM containing at most $k$-registers. Then a 2RRDPQ $\xi = (x, L', y)$ over the alphabet $\Sigma^\ast \cup \{A_i, A_i' \mid 1 \leq i \leq k\}$ can be constructed in polynomial time such that for every data graph $G = (V, E, \eta)$, $\xi(G) = \xi'(G_{dn,k})$.

Note that in practice, the number $k$ in 2RDPQs are usually small, e.g. $k = 1, 2$, and can be assumed to be a constant. Then the above data-to-node transformation becomes query-independent.

### 5. CONJUNCTIVE RIGID REGULAR PATH QUERIES WITH DATA

Conjunctive 2RRDPQs (C2RRDPQ) can be defined similarly to C2RDPQs, with 2RDPQs replaced by 2RRDPQs.

**Proposition 11.** The evaluation of C2RRDPQs is PSPACE-complete, and NLOGSPACE-complete in data complexity.

**Theorem 5.** The containment of C2RRDPQs is in 2EXPSPACE and EXPSPACE hard.

The rest of this section is devoted to the proof of Theorem 5. The proof is a nontrivial extension of that of the EXPSPACE-completeness result for C2RDPQs in [8] and is the most technical part of this paper.

#### 5.1 Canonical data graph

Let $\xi := \text{Ans}(z) \leftarrow \bigwedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$ be a C2RRDPQ. $G = (V, E, \eta)$ be a data graph, and $\nu : \{y_1, \ldots, y_{2l}\} \rightarrow V$.

Then $G$ is said to be $\nu$-canonical for $\xi$ if

- $G$ consists of $l$ simple semipaths $\pi_1, \ldots, \pi_l$, one for each conjunct of $\xi$, such that only start and end nodes can be shared among different semipaths,
- for every $i : 1 \leq i \leq l$, $\pi_i$ is a semipath from $\nu(y_{2i-1})$ to $\nu(y_{2i})$ such that $\eta(\pi_i)$ belongs to $L_i$.

It is easy to see that if $G$ is $\nu$-canonical for $\xi$, then $\nu(z)$ belongs to $\xi(G)$.

In the rest of this section, we assume that $\xi_1, \xi_2$ are two C2RDPQs such that

- $\xi_1$ and $\xi_2$ have the same set of distinguished variables,
- the set of non-distinguished variables of $\xi_1$ and $\xi_2$ are disjoint.

More specifically, for $i = 1, 2$, let $\xi_i := \text{Ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{1 \leq j \leq l_i} (y_{2j-1, i}, L_{i,j}, y_{2j, i})$ such that $\{y_{1,1}, \ldots, y_{2l_1}\} \cap \{y_{2l_1 + 1, 1}, y_{2l_1 + 2, 1}, \ldots, y_{2l_2}\}$ is equal to $\{z_1, \ldots, z_n\}$.

Let $G = (V, E, \eta)$ be a $\nu$-canonical data graph for $\xi_1$.

Then a mapping $\mu : \{y_{2,1}, \ldots, y_{2l_2}\} \rightarrow V$ is said to be a $(\xi_1, G, \nu)$-mapping for $\xi_2$ if

- for every $j : 1 \leq j \leq n$, $\nu(z_j) = \mu(z_j)$,
- for every $j : 1 \leq j \leq l_2$, $\mu(y_{2j-1, 2}, \mu(y_{2j, 2})$ belongs to the evaluation result of $(y_{2j-1, 2}, L_{2,j}, y_{2j, 2})$ over $G$.

Note that the existence of a $(\xi_1, G, \nu)$-mapping for $\xi_2$ implies that $\nu(z) \in \xi_2(G)$.

The following result can be shown in the same way as a corresponding result for C2RDPQs (Theorem 2 in [8]).
Proposition 12. Let \( \xi_1, \xi_2 \) be two C2RRDPQs. Then 
\( \xi_1 \not\equiv \xi_2 \) iff there are a data graph \( \mathcal{G} \) and a mapping \( \nu \) from the variables in \( \xi_1 \) to the nodes in \( \mathcal{G} \) such that 
\[
\begin{align*}
\mathcal{G} & \text{ is } \nu\text{-canonical for } \xi_1, \\
\text{and there are no } (\xi_1, \mathcal{G}, \nu)\text{-mappings for } \xi_2.
\end{align*}
\]

5.2 Evaluating 2RRDPQs over canonical data graphs

Let \( \mathcal{G} = (V, E, \eta) \) be a \( \nu\text{-canonical data graph for } \xi_1 \) and \( \xi = (x, L, y) \) be a 2RRDPQ such that \( L \) is defined by a NRRA \( A = (Q, \delta, I, F) \) over \( \Sigma^* \). Then \( \mathcal{G} \) consists of \( L \)-simple semipaths \( \pi_1, \ldots, \pi_l \) such that for every \( j \leq l \leq l_1 \), \( \pi_j \) is a semipath from \( \nu(y_{1, j-1}) \) to \( \nu(y_{1, j}) \), and \( \eta(\pi_j) \in L_{1,j} \). Our goal is to evaluate \( \xi \) over \( \mathcal{G} \).

We use a similar idea to the evaluation of 2RRPQs over canonical graphs in [8]: the data graph \( \mathcal{G} \) is first encoded into a data path \( \alpha_{\mathcal{G}} \), then a 2NRRA \( A_{\mathcal{G}} \) is constructed from \( \xi_1 \) and \( \xi_1 \) so that \( \xi(\mathcal{G}) \) is nonempty iff \( \alpha_{\mathcal{G}} \) is accepted by \( A_{\mathcal{G}} \).

For every \( i \leq l \), let \( \text{ren}_i \), denote the renaming function that maps each \( a \in \Sigma^* \) to \( (a, i) \). For a data path \( \alpha \), let \( \text{ren}_i(a) \) denote the data path obtained from \( \alpha \) by replacing each \( a \in \Sigma^* \) with \( \text{ren}_i(a) \).

Let \( \Sigma_{\xi_1} = (\#) \cup \bigcup_{1 \leq j \leq l_1} (\Sigma^* \times \{j\} \cup \{\#\}) \). We represent \( \mathcal{G} \) as a data path \( \alpha_{\mathcal{G}} \) over the alphabet \( \Sigma_{\xi_1} \) as follows.

\[
\alpha_{\mathcal{G}} := d_1 \text{ren}_1(\eta(\pi_1)) d_2 d_3 \# d_4 d_5 \text{ren}_2(\eta(\pi_2)) d_6 d_7 \# \ldots d_{2l-1} \text{ren}_{l-1}(\eta(\pi_{l-1})) d_{2l} d_{2l+1},
\]

where \( d_1, d_2, \ldots, d_{2l+1} \) are data values from \( \mathbb{D} \) not occurring in \( \mathcal{G} \) such that \( d_i = d_i \) iff \( \nu(y_{1,i}) = \nu(y_{1,i+1}) \).

Intuitively, for each \( j \leq l \leq l_1 \), the \( j \)-th semipath \( \pi_j \) is represented by a data subpath \( \alpha_{\mathcal{G},\pi_j} \) in \( \alpha_{\mathcal{G}} \), where \( \alpha_{\mathcal{G},\pi_j} = d_j \text{ren}_j(\eta(\pi_j)) d_{j+1} \# d_{2j+1} \text{ren}_{j-1}(\eta(\pi_j)) d_{2j+2} \). The symbol \( \# \) is used to separate those data subpaths. It is easy to observe that for every pair \( (\pi_j, v) \) such that \( v \) is a node in \( \pi_j \), there is a unique position in \( \alpha_{\mathcal{G},\pi_j} \) corresponding to \( (\pi_j, v) \), denoted by \( p_{\alpha_{\mathcal{G},\pi_j}}(\pi_j, v) \). For instance, if \( v = \nu(y_{1, j-1}) = \nu(y_{1, j}) \), then \( p_{\alpha_{\mathcal{G},\pi_j}}(\pi_j, v) \) is the position immediately before the symbol \( \# \) and \( p_{\alpha_{\mathcal{G},\pi_j}}(\pi_j, v) \) is the position immediately after the symbol \( \# \) in \( \alpha_{\mathcal{G},\pi_j} \).

For the simplicity of presentations, we assume that for every \( j \leq l \), \( \pi_j \) contains at least two edges. All the proofs in the rest of this section can be easily adapted to deal with the situation that there is \( j \leq l \) such that \( \pi_j \) contains at most one edge.

Let \( \pi' = v_0 a_1 v_1 \ldots v_{l-1} a_l v_l \) be a semipath in \( \mathcal{G} \) (since \( \pi' \) is an arbitrary semipath in \( \mathcal{G} \), it may start or end in the middle of \( \pi_1, \ldots, \pi_l \)). Because \( \alpha_{\mathcal{G}} \) is an encoding of the data graph \( \mathcal{G} \) and \( \pi' \) is a semipath in \( \mathcal{G} \), there is also an encoding of \( \pi' \) in \( \alpha_{\mathcal{G}} \). We call this encoding as the \( \text{trace of } \pi' \) in \( \alpha_{\mathcal{G}} \), denoted by \( \text{trc}_{\alpha_{\mathcal{G}}}(\pi') \). A formal definition of \( \text{trc}_{\alpha_{\mathcal{G}}}(\pi') \) will be given later.
and a position $2i: 0 \leq i \leq \ell + 2r$ on $\eta(\text{ur}v(\pi'))$, define the adjusted position represented by $t$ over $\eta(\text{ur}v(\pi'))$ and $2i$, denoted by $t_{\text{adj}}(\text{ur}v(\pi'))[2i]$, similarly to the semantics of position terms, with the following adjustments for the rules $\text{suc}(t_1)$ and $\text{pred}(t_1)$. In the following, we only present the adjustments for $\text{suc}(t_1)$, and the adjustments for $\text{pred}(t_1)$ are symmetric. If $(t_1)_{\text{ur}v(\pi')}[2i] = 1$ or $(t_1)_{\text{ur}v(\pi')}[2i] = 2(\ell + 2r)$, then $(\text{suc}(t_1))_{\text{ur}v(\pi')}[2i] = 1$; otherwise,

- if $(t_1)_{\text{ur}v(\pi')}[2i]$ is not a position immediately before $\#$, then $(\text{suc}(t_1))_{\text{ur}v(\pi')}[2i] = (t_1)_{\text{ur}v(\pi')}[2i] + 2$;
- otherwise,

$$(\text{suc}(t_1))_{\text{ur}v(\pi')}[2i] = (t_1)_{\text{ur}v(\pi')}[2i] + 4.$$

LEMMA 2. Suppose $\pi' = \nu_0\nu_1\nu_2\ldots \nu_r$ is a semipath in $G$ such that $\text{ur}v(\pi') = \pi_0\#\ldots \pi'\#$ and $\text{tr}c(\pi') = p_0b_1p_1\ldots b_{t+2}r_{t+2}$ (where $b_1, \ldots, b_{t+2}$, $r_{t+2} \in \Sigma^{\pi' \cup \{\#\}}$). Then for every $i: 0 \leq i \leq \ell + 2r$, there exists a function $\text{pos}_i \in \{T(\Sigma_{\xi_{1}})\cup\{1\}\}^{T_B}$ such that for every $t \in T_A$, $\text{pos}_i(t) = 1$ iff $t_{\text{ur}v(\pi')}[2i] = 1$; moreover, if $\text{pos}_i(t) = 1$ and $t_{\text{ur}v(\pi')}[2i] = 2i'$, then $\text{pos}_i(t)=\text{pos}_i[p_i]=p_{i'}$.

PROOF. (Theorem 6)

Let $\pi'$ be a path in $G$, the $\pi'$-unraveling of $\pi$ be $\pi'_0\ldots \pi'_r$. In addition, for every $s: 0 \leq s \leq r$, all the edges on $\pi'_s = v_{i_s}a_{i_s+1}v_{i_s+1}\ldots v_{i_s+s}$ belong to $\pi_j$.

Our goal is to construct a 2NRRA $B$ that simulates the runs of $A'$ over $\eta(\text{ur}v(\pi'))$. Similarly to the construction of NRAGs from NRRA in the proof of Theorem 3, the 2NRRA $B$ goes through $\text{tr}c(\pi')$ in $\alpha_{\Theta}$ and guesses the profile of the current position of $\eta(\text{ur}v(\pi'))$, in order to simulate $A'$ over $\eta(\text{ur}v(\pi'))$. The difference is that instead of storing and guessing the data values, $B$ records and guesses position terms from $\text{T}_B[\Sigma_{\xi_1}]$ (interpreted over $\alpha_{\Theta}$) for position terms occurring in the profile of the current position in $\eta(\text{ur}v(\pi'))$. The most technical part of the construction is how to construct the consistency of the guessed position terms from $\text{T}_B[\Sigma_{\xi_1}]$ and how to update them during the simulation. Since the details of the consistency conditions and the updating of the guessed position terms are rather tedious, they are omitted due to the space limitation.

From the above description, we know that in its states, $B$ should record the states of $A'$, the guessed profiles, and the guessed position terms from $\text{T}_B[\Sigma_{\xi_1}]$. Because both the number of profiles and the number of possible guesses for the position terms from $\text{T}_B[\Sigma_{\xi_1}]$ are exponential over $|T_A|$, it follows that the number of states of $B$ is polynomial over $|Q|$ and exponential over $|T_A|$. □

5.3 Checking the non-containment

We will construct a NRRA $A'_1 = (Q', \delta', I', F')$ to check the non-containment of $\xi_1$ over $\xi_2$ as follows.

1. Construct a NRRA $A'_1$ which reads a data path $\alpha$ over the alphabet $\Sigma_{\xi_1}$ and verifies that $\alpha$ encodes a $\nu$-canonical data graph $G$ for $\xi_1$. In particular, for every 2RRDPQ $(y_{1,2,j-1}, L_{1,j}, y_{1,2,j})$, $A'_1$ checks that the $j$-th block of $\alpha$ encodes a data path over the alphabet $\Sigma^\pi$ belonging to $L_{1,j}$.

2. Construct a NRRA $A'_2$ verifying that there are no $(\xi_1, G, \nu)$-mappings for $\xi_2$ as follows.

(a) Construct a 2NRRA $B_1$ to verify a $(\xi_1, G, \nu)$-mapping for $\xi_2$ over $\alpha_{\Theta}$ annotated with subsets of $\{y_{1,1}, \ldots, y_{2,1}\}$. The intention is that the annotations encode an assignment of nodes in $G$ to the variables from $\{y_{1,1}, \ldots, y_{2,1}\}$. The alphabet of $B_2$ is $\Sigma_{\xi_1} = \Sigma_{\xi_1} \times 2^{\{y_{1,1}, \ldots, y_{2,1}\}}$. If the word symbol immediately before a position $2i$ of the annotated $\alpha_{\Theta}$ is $(a', Z)$, then this means that each variable in $Z$ is assigned to the node of $G$ represented by the position $2i$. Some consistency constraints for these annotations, e.g. the annotations in two distinct positions are disjoint, should be checked. To check the 2RRDPQs $(y_{1,2,j-1}, L_{2,j}, y_{2,2,j})$ of $\xi_2$ over the annotated $\alpha_{\Theta}$, the construction in the proof of Theorem 6 is used. Note that since all the rigid data constraints in the RRDPQs of $\xi_2$ are independent from the annotations, we are able to assume that for every $\text{suc}_B$ or $\text{pred}_B$ occurring in $B_1$, there is $A \subseteq \Sigma_{\xi_1}$ such that $B = A \times 2^{\{y_{1,1}, \ldots, y_{2,1}\}}$.

(b) Transform $B_1$ into an equivalent NRRA $B_2$ (cf. Proposition 8).

(c) Let $\text{pr}j : \Sigma_{\xi_{1}} \rightarrow \Sigma_{\xi_{2}}$ such that $\text{pr}j(a', Z) = a'$. Construct $B_3 = \text{pr}j(B_2)$. From the assumption above, we know that $B_1$ and $B_2$ are position-invariant under $\text{pr}j$. Then from Proposition 4, we deduce that $\text{L}(B_3) = \text{L}(\text{pr}j(B_2)) = \text{pr}j(\text{L}(B_2))$. So the NRRA $B_3$ guesses and verifies a $(\xi_1, G, \nu)$-mapping for $\xi_2$.

(d) Determinize and complement $B_3$ to get $A'_2$ (cf. Proposition 7).

3. $A'$ is the intersection of $A'_1$ and $A'_2$.

There is a final remark for the above construction: As pointed out in [8], letter projections are only meaningful
for one-way automata. This explains why we need go from the 2NRR A B1 to the NRR A B2 before applying the letter projection prj.

The complexity analysis.

The size of A2 is polynomial over the size of ξ. From Theorem 6, the size of B1 is exponential over the size of ξ. From Proposition 8, the size of B2 is exponential over the size of B1. The size of B3 is the same as the size of B2. From Proposition 7, the size of A2 is exponential over the size of B3. Therefore, the size of A2 is triple-exponential over the size of ξ.

To check the nonemptiness of A1 ∩ A3, we can guess “on the fly” an accepting run of A1 ∩ A3 in double exponential space. From Savitch’s theorem, we deduce that the containment of C2RRDPQs is in 2EXPSPACE.

On the other hand, the containment of C2RRDPQs is EXPSPACE-hard since this is already the case for C2RPQs ([8]).

6. CONCLUSION

In this paper, a novel approach to extend 2RPQs with data value comparisons, called rigid regular path queries with inverse and data (2RRDPQs), was proposed. 2RRDPQs rely on nondeterministic rigid register automata (NRRAs), also introduced in this paper. We demonstrated the robustness of NRRAs by showing that NRRAs can be determinized and the two-way NRRAs are expressively equivalent to NRRAs. We then argued that 2RRDPQs achieve a good balance between the expressibility and computational properties. On the one hand, we showed that every 2RDPQ can be turned into a 2RRDPQ if a localized transformation is applied to graph databases. On the other hand, we proved that 2RRDPQs enjoy nice computational properties, as witnessed by the decidability (as a matter of fact, 2EXPSPACE) of the containment problem of 2RRDPQs and conjunctive 2RRDPQs (C2RRDPQ). The proof for the 2EXPSPACE result of the containment problem of C2RRDPQs is the most technical part of this paper and can be seen as the main result of this paper.

There are several natural directions for future work. One direction is to investigate whether the evaluation and containment problem of acyclic C2RRDPQs have a lower complexity. Another direction is to investigate nested rigid regular expressions with memory.

7. REFERENCES


APPENDIX

A. PROOFS IN SECTION 3.1

Proposition 1 The satisfiability problem of rigid data constraints is NP-complete.

Proof. Lower bound: By an easy reduction from the satisfiability of Boolean formulas.
Upper bound: Let $c$ be a rigid data constraint over $\Sigma^*$. Let $T_c$ denote the minimal set of position terms satisfying the following conditions.

- for every $t_1 \sim t_2$ or $t_1 \succeq t_2$ occurring in $c$, $t_1, t_2 \in T_c$.
- for every $t \in T_c$ and $t' \in C_{rig}[\Sigma^*]$ such that $t' \preceq t$, we have $t', t'[\text{car}] \notin T_c$.

Similar to the construction of a NRAGs from NRRAs in the proof of Theorem 3, we can define concept of profiles with respect to $T_c$. More specifically, a profile is a triple $(S, \chi, \sim)$ such that $S \subseteq T_c$, $\sim$ is an equivalence relation on $S$, and

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2}) .$$

In addition, some consistency conditions can be defined such that $c$ is satisfiable iff there is a consistent profile $(S, \chi, \sim)$ with $c \in T_0$ in $\chi$.

Since the size of a profile is polynomial over that of $c$, a profile $(S, \chi, \sim)$ can be guessed and the consistency condition as well as $c \in T_0$ can be checked in polynomial time. Therefore, the satisfiability of rigid data constraints is in NP. □

B. PROOFS IN SECTION 3.2

Proposition 2 NRAs and NRRAs are expressively incomparable.

Proof. The data language “there are two distinct positions with the same data value” is definable in NRAs, but not in NRRAs.

On the other hand, the data language “the sequence of word symbols belongs to $ab^*a$ and the last data value does not occur elsewhere” is definable in NRRAs, but not in NRAs. □

Proposition 3 The class of languages definable by NRRAs are not closed under letter projections.

Proof. Let $\Sigma = \{ (a, 0), (a, 1) \}$, $\Gamma = \{ a \}$, and $\text{prj}$ be a letter projection from $\Sigma$ to $\Gamma$ such that $\text{prj}((a, 0)) = \text{prj}((a, 1)) = a$.

Let $L$ be the data language “there are exactly two distinct positions labeled by $(a, 1)$ and the data values before these two positions are the same”. Then $\text{prj}(L)$ is the data language “there are two distinct positions with the same data value”.

It is easy to see that $L$ can be defined by a NRRA $A$. On the other hand, $\text{prj}(L)$ is not definable by a NRRA.

We would like to remark that over the alphabet $\{ a \}$, the position terms $\text{suc}_A(t)$ (resp. $\text{pred}_A(t)$) in $\text{prj}(A)$ are equal to $\text{suc}(t)$ (resp. $\text{pred}(t)$). Therefore, $\text{prj}(A)$ does not define $\text{prj}(L)$. □

Proposition 4 Suppose $A$ is a NRRA over $\Sigma^*$ and $\text{prj}$ is a letter projection from $\Sigma^*$ to $\Gamma$. If $A$ is position-invariant under $\text{prj}$, then $\mathcal{L}(\text{prj}(A)) = \text{prj}(\mathcal{L}(A))$.

Proof. Suppose $A$ is a NRRA over $\Sigma^*$ and $\text{prj}$ is a letter projection from $\Sigma^*$ to $\Gamma$ such that $A$ is position-invariant under $\text{prj}$.

For every position term $t \in T_A$, define $\text{prj}(t)$ as the position term obtained from $t$ by replacing every occurrence of $\text{suc}_A$ with $\text{suc}_{\text{prj}(A)}$. In addition, for every $c \in C_A$, define $\text{prj}(c)$ as the rigid data constraint obtained from $c$ by replacing every position term $t$ with $\text{prj}(t)$.

We first prove the following claim.

Claim. Let $\alpha = d_0a_1d_1 \ldots a_nd_n$ be a data path, $c \in C_A$ and $i : 0 \leq i \leq n$. Then $(\alpha, 2i) = c$ iff $(\text{prj}(\alpha), 2i) = \text{prj}(c)$.

Proof. It is sufficient to prove that for every $t \in T_A$ and $i : 0 \leq i \leq n$, $t_\alpha[2i] = (\text{prj}(t))_{\text{prj}(\alpha)[2i]}$.

This result can be proved by an induction on the structure of the position terms. In the following, we take $t = \text{suc}_A(\text{car})$ as an example to illustrate the proof.

Suppose $(\text{suc}_A(\text{car}))_\alpha[2i] = 2j$ for some $j : i < j$. Then the first occurrence of word symbols from $A$ in $\alpha$ after the position $2i$ is in the position $2j$. It follows that the first occurrence of word symbols from $\text{prj}(A)$ in $\text{prj}(\alpha)$ after the
position $2i$ is in the position $2j$. Otherwise, there is $j' : i < j' < j$ such that a word symbol from $prj(A)$ occurs in the position $2j'$ of $prj(A)$. From the fact that $A$ is position-invariant under $prj$, we know that $A = prj^{-1}(prj(A))$. Thus, a word symbol from $A$ occurs in the position $2j' < 2j$ of $A$, a contradiction. Therefore, $(suc_{prj(A)}(cur))_{prj(A)[2i]} = 2j$.

Suppose $(suc_{prj(A)}(cur))_{prj(A)[2i]} = 2j$ for some $j : i < j$. Then the first occurrence of word symbols from $prj(A)$ in $prj(A)$ after the position $2i$ is in the position $2j$. It follows that the first occurrence of word symbols from $A$ in $A$ after the position $2i$ is in the position $2j$. Otherwise, there is $j' : i < j' < j$ such that a word symbol from $A$ occurs in the position $2j'$ of $A$. Thus, a word symbol from $prj(A)$ occurs in the position $2j' < 2j$ of $prj(A)$, a contradiction. Therefore, $(suc_{prj(A)}(cur))_{prj(A)[2i]} = 2j$.

Let

\[ \mathcal{L}(prj(A)) \subseteq prj(\mathcal{L}(A)) : \]

Suppose $\beta = d_0 \gamma_1 d_1 \ldots \gamma_n d_n \in \mathcal{L}(prj(A))$.

Then there is an accepting run of $prj(A)$ over $\beta$, say $\rho = q_0 c_0 \gamma_1 q_2 c_1 \ldots q_{2n-1} \gamma_n q_{2n} c_n q_{2n+1}$.

From the definition of $prj(A)$, we know that

- for every $i : 1 \leq i \leq n$, there is $a_i \in \Sigma^*$ such that $prj(a_i) = \gamma_i$ and $(q_{2i-1}, a_i, q_{2i}) \in \delta_w$,
- for every $i : 0 \leq i \leq n$, there is $c_i' \in C_A$ such that $(q_{2i}, c_i', q_{2i+1}) \in \delta_d$ and $c_i = prj(c_i')$.

Let $\alpha = d_0 a_1 d_0 a_2 d_0 a_3 \ldots a_n d_n$. Then $\beta = prj(\alpha)$.

From the claim, we know that for every $i : 0 \leq i \leq n$, $(\alpha, 2i) = c_i$ iff $(prj(\alpha), 2i) = c_i$.

Therefore, $q_0 c_0 q_1 a_1 q_2 a_2 \ldots q_{2n-1} a_n q_{2n} c_n q_{2n+1}$ is an accepting run of $A$ over $\alpha$. We conclude that $\alpha \in \mathcal{L}(A)$ and $\beta = prj(\alpha) \in prj(\mathcal{L}(A))$.

Proposition 5 The nonemptiness problem of NRAGs is PSPACE-complete.

Proof. The upper bound: Let $A = (Q, \delta, I, F)$ be a NRAG. Then similar to NRAs ([10, 18]), a NFA $B = (Q', \delta', I', F')$ can be constructed such that $\mathcal{L}(A)$ is nonempty iff $\mathcal{L}(B)$ is nonempty, and $|Q'|$ is polynomial over $|Q|$ and exponential over $k$. To decide the nonemptiness of $B$, an accepting run of $B$ can be guessed nondeterministically in polynomial space. From Savitch's theorem, we know that the nonemptiness of $A$ can be decided in PSPACE.

The lower bound: Follows from that of NRAs.

Lemma 1. Let $A = (Q, \delta, I, F)$ be a NRRA over the alphabet $\Sigma^*$ and $\alpha = d_0 a_1 d_1 \ldots a_n d_n$ be a data path. Then for every run $\rho$ of $A$ over $\alpha$ and every $i : 0 \leq i \leq n$, $Pos_{\rho}[\alpha, 2i] \cup Pos_{\rho}[\alpha, 2i] \subseteq \{ t_a[2i] | t \in T_A \}$.

Proof. Let $A = (Q, \delta, I, F)$ be a NRRA, $\alpha = d_0 a_1 d_1 \ldots a_n d_n$ be a data path, $\rho = q_0 c_0 q_1 a_1 q_2 \ldots q_{2n-1} a_n q_{2n} c_n q_{2n+1}$ be a run of $A$ over $\alpha$, and $i : 0 \leq i \leq n$.

Let $t_a[2j] \in Pos_{\rho}[\alpha, 2i]$ such that $j \leq i$ occurs in $c_j$, $t_a[2j] \neq 1$ and $t_a[2j] > 2i$. Then there is $t' \in T_A[\Sigma^*]$ such that $t' \leq t$, $t'_a[2j] \leq 2i$, and for every $t'' : t' < t'' \leq t$, $t''_a[2j] > 2i$. It follows that $suc(t') \leq t$ or $suc_A(t') \leq t$ for some $A \subseteq \Sigma^*$.

- If $suc(t') \leq t$, then $t'_a[2j] = 2i$, since $(suc(t'))_a[2j] = t'_a[2j] + 1 > 2i$ and $t'_a[2j] \leq 2i$. Thus, $t_a[2j] = \{(t'[\text{cur}])_a[t'_a[2j]] \}$. If $suc_A(t') \leq t$, then $t'_a[2j] \leq 2i$. From the above argument, it follows that $Pos_{\rho}[\alpha, 2i] \subseteq \{ t_a[2i] | t \in T_A[\Sigma^*] \}$. Similarly, we can show that $Pos_{\rho}[\alpha, 2i] \subseteq \{ t_a[2i] | t \in T_A \}$.
Proof. Let $A = (Q, \delta, I, F)$ be a NRRA. In the following, we will construct a NRAG $B$ to simulate $A$.

We first give an intuitive description of the construction. Let $p$ be a run of $A$ over a data path $\alpha = a_0a_1d_1 \ldots a_n d_n$. From Lemma 1, we know that for every $i : 0 \leq i \leq n$, $\text{Pos}^p_{\alpha}[\alpha, 2i] \cup \text{Pos}^p_{\alpha}[\alpha, 2i]$ contains only a bounded number of positions. It follows that only a bounded number of registers are needed to store them in the position $2i$. Therefore, $B$ can simulate $p$ as follows: In the position $2i$,

- $B$ records in its registers the data values in the positions belonging to $\text{Pos}^p_{\alpha}[\alpha, 2i]$.
- $B$ guesses in its registers the data values in the positions belonging to $\text{Pos}^p_{\alpha}[\alpha, 2i]$.
- $B$ records the order for the positions in $\text{Pos}^p_{\alpha}[\alpha, 2i]$ and $\text{Pos}^p_{\alpha}[\alpha, 2i]$.

We introduce some additional notations. Let $\alpha = a_0a_1d_1 \ldots a_n d_n$ be a data path and $i : 0 \leq i \leq n$. The profile of the position $2i$ in $\alpha$, denoted by $\text{prof}_\alpha(2i)$, is defined as a triple $(S, \chi, [~, ~])$, where

- $S = \{ t \in T_A | t_\alpha[2i] \neq \bot \}$
- $\chi$ is a sequence

$$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

where

- for every $j : -m_1 \leq j \leq m_2$, $T_j \subseteq S$ and $T_j \neq \emptyset$,
- the collection $T_{-m_1}, \ldots, T_0, \ldots, T_{m_2}$ forms a partition of $S$,
- for every $t, t' \in S$, if $t \in T_j$ and $t' \in T_{j'}$, then $j_1 \leq j_2$ iff $t_\alpha[2i] \leq t'_\alpha[2i]$ (in particular, $j_1 = j_2$ iff $t_\alpha[2i] = t'_\alpha[2i]$),
- $\text{cur} \in T_0$,
- for every $j : -m_1 < j < m_2$, $b_j = a(t_\alpha[2i])$ if $t \in T_j$ and $b_{-m_1} = a(t_\alpha[2i])$ if $t_\alpha[2i] > 0$ for some $t \in T_{-m_1}$, otherwise, $b_{-m_1} = 1$,
- for every $j : -m_1 \leq j < m_2$, $b'_j = a(t_\alpha[2i])$ if $t \in T_j$ and $b'_{m_2} = a(t_\alpha[2i])$ if $t_\alpha[2i] < 2n$ for some $t \in T_{m_2}$, otherwise, $b'_{m_2} = 1$,
- $s_{m_2} = 1$, and for every $j : -m_1 \leq j < m_2$, if $t'_\alpha[2i] = t_\alpha[2i] + 1$ for some $t \in T_j$ and $t' \in T_{j'+1}$, then $s_j = 1$, otherwise, $s_j = 0$.

$\sim$ is an equivalence relation over $S$ defined as follows: Let $t, t' \in S$, then $t \sim t'$ iff $d_\alpha[2i] = d_\alpha[2i]$.

Let $\Sigma_{\text{prof}}$ denote the set of all triples $(S, \chi, [~, ~])$ such that

- $S \subseteq T_A$,
- $\chi$ is a sequence

$$(b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

such that

- for every $j : -m_1 \leq j \leq m_2$, $T_j \neq \emptyset$,
- $\text{cur} \in T_0$,
- $T_{-m_1}, \ldots, T_{m_2}$ is a partition of $S$,
- $b_{-m_1} \in \Sigma^* \cup \{ \bot \}$, and for every $j : -m_1 < j \leq m_2$, $b_j \in \Sigma^*$,
- $b'_{m_2} \in \Sigma^* \cup \{ \bot \}$, and for every $j : -m_1 \leq j < m_2$, $b'_j \in \Sigma^*$,
- $s_{m_2} = 1$, and for every $j : -m_1 \leq j < m_2$, $s_j \in \{ 0, 1 \}$,

$\sim$ is an equivalence relation over $S$ such that for every $t, t' \in T_A$, if $t, t' \in T_j$ for some $j$, then $t \sim t'$.

Note that for $(S, \chi, [~, ~)] \in \Sigma_{\text{prof}}$, there may be no data paths $\alpha$ and a position in $\alpha$ such that the profile of the position in $\alpha$ is $(S, \chi, [~, ~])$. Nevertheless, we are able to define a consistency condition on the elements from $\Sigma_{\text{prof}}$ so that a consistent element from $\Sigma_{\text{prof}}$ indeed corresponds to the profile of a position in some data path. Moreover, for two consistent elements from $\Sigma_{\text{prof}}$, say $(S_1, \chi_1, [~, ~]), (S_2, \chi_2, [~, ~])$, and $a \in \Sigma^*$, we are able to define a syntactic successor relation $(S_1, \chi_1, [~, ~]) \rightarrow (S_2, \chi_2, [~, ~])$, which mimics the changes from $\text{prof}_\alpha(2i)$ to $\text{prof}_\alpha(2(i + 1))$ by reading a word symbol $a$ in the position $2i + 1$ of a data path.

For $A \subseteq \Sigma^*$, a sequence

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

...
Claim. Suppose $(S, \chi, \sim) \in \Sigma_{prof}$. Then $(S, \chi, \sim)$ is consistent iff there is a data path $\alpha$ and a position $2i$ in $\alpha$ such that $prof_i(2i) = (S, \chi, \sim)$.

Proof. The “if” direction is trivial.

The “only if” direction:
Suppose $(S, \chi, \sim)$ is consistent. Let

$$\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})$$

For each $T_j$, we assign a data value $d_j$, in a way that respects the equivalence relation $\sim$, that is, if $t \in T_j$, $t' \in T_{j'}$, and $t \sim t'$, then $d_j = d_{j'}$. In addition, let $d$ be a data value different from all these $d_j$'s.

For every $j : -m_1 \leq j \leq m_2$, we define a data path $\alpha_j$ as follows.

- For $j = -m_1$,
  - if $s_{-m_1} = 1$ and $b_{-m_1} \neq \bot$, then let $\alpha_{-m_1} = dB_{-m_1}d_{-m_1}$,
  - if $s_{-m_1} = 1$ and $b_{-m_1} = \bot$, then let $\alpha_{-m_1} = dB_{-m_1}$,
  - if $s_{-m_1} = 0$ and $b_{-m_1} \neq \bot$, then let $\alpha_{-m_1} = dB_{-m_1}b'_{-m_1}d$,
  - if $s_{-m_1} = 0$ and $b_{-m_1} = \bot$, then let $\alpha_{-m_1} = dB_{-m_1}b'_{-m_1}$.

- For $j < m_1$,
  - if $s_{j-1} = 1$ and $s_j = 1$, then let $\alpha_j = d_{j-1}b_jd_j$,
  - if $s_{j-1} = 1$ and $s_j = 0$, then let $\alpha_j = d_{j-1}b_jb'jd_j$,
  - if $s_{j-1} = 0$ and $s_j = 1$, then let $\alpha_j = db_jd_j$,
  - if $s_{j-1} = 0$ and $s_j = 0$, then let $\alpha_j = db_jd_j$.

- For $j = m_2$,
  - if $s_{m_2-1} = 1$ and $b'_{m_2} \neq \bot$, then let $\alpha_{m_2} = dB_{m_2}b_{m_2}b'_{m_2}d$,
  - if $s_{m_2-1} = 1$ and $b'_{m_2} = \bot$, then let $\alpha_{m_2} = dB_{m_2}$,
  - if $s_{m_2-1} = 0$ and $b'_{m_2} \neq \bot$, then let $\alpha_{m_2} = dB_{m_2}b_{m_2}d$,
  - if $s_{m_2-1} = 0$ and $b'_{m_2} = \bot$, then let $\alpha_{m_2} = dB_{m_2}$.
Consider the data path \( \alpha = \alpha_{-m_1} \cdot \alpha_{-m_1+1} \cdot \ldots \cdot \alpha_{-1} \cdot \alpha_0 \cdot \alpha_1 \cdot \ldots \cdot \alpha_{m_2} \).

For every \( j : -m_1 \leq j \leq m_2 \), let the position of \( \alpha \) corresponding to the data value \( d_j \) be \( 2i_j \).

From the construction of \( \alpha \) from \((S, \chi, \sim)\), by an induction on the structure of position terms, we can prove the following result.

For every \( t \in \mathcal{T}_A \) and every \( j : -m_1 \leq j \leq m_2 \), \( t \in T_j \) iff \( t_{00}[2i_0] = 2i_j \).

(\#)

Let us take \( t = \text{succ}(\alpha) \) as an example to illustrate the proof.

Suppose \( \text{succ}(\alpha) \in T_j \), then \( b_j \in A \) and for every \( j' : 0 < j' < j \), \( b_{j'} \notin A \), and for every \( j' : 0 \leq j' < j - 1 \), \( b_{j'} \notin A \); in addition, \( b_{j-1} \notin A \) implies \( s_{j-1} = 1 \). From this, we deduce that \( (\text{succ}(\alpha))_0[2i_0] = 2i_j \), since all the word symbols located after the position \( 2i_0 \) and before the position \( 2(i_j - 1) \) in \( \alpha \) do not belong to \( A \).

On the other hand, suppose \( (\text{succ}(\alpha))_n[2i_0] = 2i_j \), then the word symbol immediately before the position \( 2i_j \), that is, \( b_j \), belongs to \( A \), and all the word symbols located after the position \( 2i_0 \) and before the position \( 2(i_j - 1) \) in \( \alpha \) do not belong to \( A \). From the construction of \( \alpha \), it follows that \( b_j \in A \) and for every \( j' : 0 < j' < j \), \( b_{j'} \notin A \), and for every \( j' : 0 \leq j' < j - 1 \), \( b_{j'} \notin A \); in addition, \( b_{j-1} \notin A \) implies \( s_{j-1} = 1 \) and \( b_{j-1} = b_j \). From this, we conclude that \( \text{succ}(\alpha) \in T_j \).

From the result (\#), we conclude that \( \text{prof}_{\alpha}[2i_0] = (S, \chi, \sim) \). ∎

Let \( \text{Prof}_A \) denote the set of elements of \( \Sigma_{\text{prof}} \) that are consistent. Suppose \( (S, \chi, \sim) \in \text{Prof}_A \),

\[
\chi = (b, -m_1, T, -m_1, b', -m_1, \ldots, b, -1, T, -1, b', -1, s, -1)(b_0, T_0, b_0', s_0)(b_1, T_1, b_1', s_1) \ldots (b_{m_2}, T, b_m', s_{m_2})
\]

c and \( c \in C_A \). Then the satisfaction of \( c \) over \( (S, \chi, \sim) \), denoted by \( (S, \chi, \sim) \models c \), can be defined by interpreting \( c \) over \( (S, \chi, \sim) \) in a natural way. For instance, if \( c = t_1 \sim t_2 \), then \( (S, \chi, \sim) \models c \) if \( t_1, t_2 \in S \) and \( t_1 \sim t_2 \).

Suppose \( a \in \Sigma^*, (S_1, \chi_1, \sim_1) \in \text{Prof}_A \), for \( i = 1, 2, \ldots, m_2 \),

\[
\chi_i = (b_{i, -m_1, 1}, T_{i, -m_1, 1}, b'_{i, -m_1, 1}, s_{i, -m_1, 1}) \ldots (b_{i, -1, 1}, T_{i, -1, 1}, b'_{i, -1, 1}, s_{i, -1, 1})(b_{i, 0, 1}, T_{i, 0, 1}, b'_{i, 0, 1}, s_{i, 0})(b_{i, 1, 1}, T_{i, 1, 1}, b'_{i, 1, 1}, s_{i, 1}) \ldots (b_{i, m_2, 1}, T_{i, m_2, 1}, b'_{i, m_2, 1}, s_{i, m_2})
\]

In the following, we will define the concept that \( (S_2, \chi_2, \sim_2) \) is a successor of \( (S_1, \chi_1, \sim_1) \) with respect to \( a \), denoted by \( (S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2) \).

For every \( j : -m_1 \leq j \leq m_1 \), construct \( T'_{1, j} \subseteq \mathcal{T}_A \) from \( T_{1, j} \) as follows.

- For every \( t \in T_{1, j} \) such that \( \text{succ}(\alpha) \leq t \), let \( [\text{succ}(\alpha) \mid \text{cur}] \in T'_{1, j} \).
- For every \( t \in T_{1, j} \) and \( \alpha \subseteq \Sigma^* \) such that \( a \in A \) and \( \text{succ}(\alpha) \leq t \), let \( [\text{succ}(\alpha) \mid \text{cur}] \in T'_{1, j} \).
- For every \( t \in T_{1, j} \) such that \( [\text{cur} \mid \text{pred}(\alpha)] \in \mathcal{T}_A \), let \( [\text{cur} \mid \text{pred}(\alpha)] \in T'_{1, j} \).
- For every \( t \in T_{1, j} \) and \( \alpha \subseteq \Sigma^* \) such that \( a \in A \) and \( [\text{cur} \mid \text{pred}(\alpha)] \in \mathcal{T}_A \), let \( [\text{cur} \mid \text{pred}(\alpha)] \in T'_{1, j} \).
- For every \( t \in T_{1, j} \) such that \( a \notin A \), \( \text{succ}(\alpha) \leq t \) or \( \text{pred}(\alpha) \leq t \), let \( t \in T'_{1, j} \).
- If \( s_{1, 0} = 1 \), let \( c \in T'_{1, j} \).

Note that \( T'_{1, j} \)'s defined above may be empty for some \( j : -m_1 \leq j \leq m_2 \).

\( (S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2) \) if the following conditions hold.

- \( b'_{1, 0} = a \).
- Let \( j_1, j_2, \ldots, j_n : -m_1 \leq j_1 < j_2 < \ldots < j_n \leq 0 \) be the sequence of the non-positive indices such that for every \( r : 1 \leq r \leq \ell \), \( T'_{1, j_r} \neq \emptyset \) (and all the other \( T'_{1, j} \)'s for non-positive \( j \)'s are empty). Then the sequence

\[
(b_{1, j_1}, T_{1, j_1}, b'_{1, j_1}, s_{1, j_1}) \ldots (b_{1, j_n}, T_{1, j_n}, b'_{1, j_n}, s_{1, j_n})
\]

is equal to the sequence

\[
(b_{2, m_2 1}, T_{2, m_2 1}, b'_{2, m_2 1}, s_{2, m_2 1}) \ldots (b_{2, -m_2 1}, T_{2, -m_2 1}, b'_{2, -m_2 1}, s_{2, -m_2 1})
\]

is equal to the sequence

\[
(b_{1, j_1}, T_{1, j_1}, b'_{1, j_1}, s_{1, j_1}) \ldots (b_{1, j_n}, T_{1, j_n}, b'_{1, j_n}, s_{1, j_n})
\]

- There is a partial mapping \( f \) from \( \{1, \ldots, m_2 \} \) to \( \{0, \ldots, m_2 \} \) such that

\[
\begin{align*}
&\text{for every } j : 1 \leq j \leq m_2, f(j) \text{ is undefined iff } T'_{1, j} = \emptyset, \\
&\text{f is increasing, that is, if } j_1 < j_2 \text{ and } f(j_1), f(j_2) \text{ are defined, then } f(j_1) < f(j_2), \\
&\text{for every } j : 1 \leq j \leq m_2, \text{ if } f(j) \text{ is defined, then } T'_{1, j} \subseteq T_{2, f(j)}, b_{1, j} = b_{2, f(j)} \text{ and } b'_{1, j} = b'_{2, f(j)}, \\
&\text{if } f(1) \text{ is defined, then } f(1) = 0 \text{ iff } s_{1, 0} = 1, \\
&\text{for every } j : 1 \leq j < m_1, \text{ if } f(j), f(j + 1) \text{ are both defined, then } s_{1, j} = 1 \text{ implies } f(j + 1) = f(j) + 1 \text{ and } s_{2, f(j)} = 1,
\end{align*}
\]
Proposition 7

For every NRRA \( A \), there is an equivalent DRRA of exponential size.

Proof. Let \( A = (Q, \delta, I, F) \) be a NRRA. We construct a DRRA \( A' = (Q', \delta', q_0', F') \) as follows:

\begin{itemize}
  \item \( Q' = Q'_w \cup Q'_d \), where \( Q'_w = 2^{Q_w} \), \( Q'_d = 2^{Q_d} \),
  \item \( q_0' = I, F' = \{ S \in Q'_w | S \cap F \neq \emptyset \} \),
\end{itemize}
Proposition 8

C. PROOFS IN SECTION 3.3

The language inclusion problem for NRRAs is PSPACE-complete.

PROOF. The upper bound:

Let \( \mathcal{A} = (Q_1, \delta_1, I_1, F_1) \) and \( \mathcal{B} = (Q_2, \delta_2, I_2, F_2) \) be two NRRAs. To decide whether \( \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \), we use the following procedure.

1. Determine and complement \( \mathcal{B} \), let \( \mathcal{C} \) be the resulting DRRA.
2. Construct the product of \( \mathcal{A} \) and \( \mathcal{C} \), say \( \mathcal{C}' \), that defines \( \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{C}) \). Check whether \( \mathcal{L}(\mathcal{C}') \neq \emptyset \).

From the proof of Proposition 7, we know that the size of \( \mathcal{C} \) is exponential over \( |Q_2| \). Thus, the size of \( \mathcal{C}' \) is polynomial over \( |Q_1| \) and exponential over \( |Q_2| \). The set of position terms of \( \mathcal{C}' \) is the union of \( T_A \) and \( T_B \).

From the proof of Proposition 6, it follows that the nonemptiness of \( \mathcal{C}' \) can be reduced to that of a NFA of size polynomial over \( |Q_1| \), exponential over \( |Q_2| \), and exponential over \( |T_A| \) and \( |T_B| \).

From Savitch’s theorem, we conclude that \( \mathcal{L}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{B}) \) can be decided in PSPACE.

The lower bound:

The language inclusion of NFAs is already PSPACE-hard.

C. PROOFS IN SECTION 3.3

Proposition 8. For every 2NRA, there is an equivalent NRA of exponential size.

PROOF. The proof is an adaptation of Shepherdson’s method [20] to construct an equivalent NFA from a two-way NFA.

Let \( \mathcal{A} = (Q, \tau, \delta, I, F) \) be a 2NRA. We construct a NRA \( \mathcal{A}' = (Q', \delta', I', F') \) as follows.

1. \( Q' = Q_d \cup Q_w \), where
   - \( Q_d' \) is the set of all tuples \( (C, f) \in 2^{C \times (Q \cup \{1\})} \) such that \( \bigwedge_{c \in C} c \) is satisfiable, \( f(\cdot) \in Q_d \), and for every \( q \in Q_d, f(q) \in Q_w \cup \{1\} \),
   - \( Q_w' \) is the set of all tuples \( (a, f) \in (\Sigma^* \cup \{\tau\}) \times (Q \cup \{1\})^{Q_w \cup \{1\}} \) such that \( f(\cdot) \in Q_w \), and for every \( q \in Q_w, f(q) \in Q_w \cup \{1\} \).
2. \( I' \) is the set of \( (C, f) \in Q_d' \) satisfying the following conditions,
   - \( f(\cdot) = q \) such that \( (q', \tau, q, +1) \in \delta_w \) for some \( q' \in I \),
   - if \( f(q) = q' \), then there exist \( q_0, q_1, \ldots, q_k \in Q_d \) and \( p_1, \ldots, p_k \in Q_w \) such that
     * \( q_0 = q \),
     * for every \( i : 0 \leq i < k \), there is \( c_i \in C \) such that \( (q_i, c_i, p_i, +1) \in \delta_w \), and for every \( i : 1 \leq i \leq k, (p_i, \tau, q_i, +1) \in \delta_w \),
     * there is \( c \in C \) such that \( (q_k, c, q', +1) \in \delta_d \).
3. \( \delta' \) are defined as follows.
   - Let \( (a, f) \in Q_w \), \( (C, f) \in Q_d' \). Then \((a, f, a, (C, f')) \in \delta_w'\) if for every pair \( (q, q') \) such that \( q \in Q_d, q' \in Q_w \) and \( f'(q) = q' \), the following condition holds.
     There exist \( q_0, q_1, \ldots, q_k \in Q_d, p_1, \ldots, p_k \in Q_w \) such that \( q_0 = q \), and the following conditions hold,
     1. for every \( 0 \leq i < k \), there exists \( c \in C \) such that \( (q_i, c, p_{i+1}, -1) \in \delta_d \),
     2. for every \( 1 \leq i \leq k \), \( f(p_i) = q_i \),
For each $q$, $f(q') = q'$, and for every $q' \in Q_w$, $f(q')' = 1$.

F consists of all $(i, f) \in Q_w$ such that
- $f(\cdot) = q \in Q_w$,
- for every $q' \in Q_w$, $f(q') = 1$,
- there is $(C, f') \in Q'_d$ such that $((C, f') \wedge C, f) \in \delta_w'$, there are $p_1, \ldots, p_k \in Q_d$ and $q_0, \ldots, q_k \in Q_w$ satisfying that $q_0 = q$, for every $i : 0 \leq i < k$, $(q_i, a, p_{i+1}, -1) \in \delta_w$, and for every $i : 0 \leq i < k$, $f'(p_i) = q_i$, and $q_k \in F$.

Now we prove the correctness of the construction, that is, for every data path $\alpha = \delta_0 \delta_1 \cdots \delta_{m-1} \delta_m$, $\mathcal{A}$ accepts $\alpha$ iff $\mathcal{A}'$ accepts $\alpha$.

"Only if direction":

Suppose $\mathcal{A}$ accepts $\alpha$. Then there is an accepting run of $\mathcal{A}$ over $\alpha$, say $(q_0, i_0) \theta_0(q_1, i_1) \theta_1 \cdots \theta_{m-1}(q_m, i_m)$, such that
- $q_0 \in I$, $q_m \in F$,
- $i_0 = 0$, $i_m = 2n + 2$,
- for every $j : 0 \leq j \leq m$, if $i_j$ is even, then there is $\text{dir} \in \{+1, -1\}$ such that $(q_j, a_{i_j}/2, q_{j+1}, \text{dir}) \in \delta_w$ (where $a_0 = -1, a_{n+1} = 1$, $\theta_j = a_{i_j}/2$, and $i_{j+1} = i_j + \text{dir}$,
- for every $j : 0 \leq j \leq m$, if $i_j$ is odd, then there are $c \in C_{rgd}$ and $\text{dir} \in \{+1, -1\}$ such that $(q_j, c, q_{j+1}, \text{dir}) \in \delta_d$, \( (\alpha, i_j - 1) = c, \theta_j = c, \) and $i_{j+1} = i_j + \text{dir}$.

Without loss of generality, we assume that in the accepting run above, no states are repeated when the reading head moves to the same position, more precisely, the following condition holds.

For every $j_1, j_2 : 0 < j_1 < j_2 < m$ such that $i_{j_1} = i_{j_2}$, it holds that $q_{i_{j_1}} = q_{i_{j_2}}$. (*)

The above assumption is justified by the fact that if a state is repeated in the same position, then the subrun between the repetitions can be trimmed and the remaining part is still an accepting run.

For each $i : 1 \leq i \leq 2n + 2$, define $f_i$ as follows.

1. For every $q \in Q$, if there are $j_1, j_2 : 0 \leq j_1 < j_2 < m$ such that
   - $q_{j_1} = q$, $q_{j_2} = q'$,
   - $i_{j_1} = i$, $i_{j_2} = i + 1$, and for every $j' : j_1 < j' < j_2$, $i_{j'} \leq i$,
   - then $f_i(q) = q'$, otherwise $f_i(q) = 1$.

   Note that the assumption (*) guarantees that for every $i$, there is at most one pair $(j_1, j_2)$ satisfying the above condition. So $f_i$ is well-defined.

2. $f_i(q)$, where $q \in Q$ satisfies that there exists $j : 0 < j \leq m$ such that $i_j = i$, $q = q_i$, and for every $j' : 0 \leq j' < j$, $i_{j'} < i$.

For each $i : 0 \leq i \leq n$, define $C_i \subseteq C_\mathcal{A}$ as the set of $\theta_j$’s such that $i_j = 2i + 1$.

Then
\[
(C_0, f_1)(\bigwedge_{c \in C_0} c)(a_1, f_2)a_1(C_1, f_3) \cdots (a_n, f_{2n})a_n(C_n, f_{2n+1})(\bigwedge_{c \in C_n} c)(+, f_{2n+2})
\]
is an accepting run of $\mathcal{A}'$ over $\alpha$.

"If direction":

Suppose $\mathcal{A}'$ accepts $\alpha$. Then there exists $\alpha = \delta_0 \delta_1 \cdots \delta_{m-1} \delta_m$, $\mathcal{A}$ accepts $\alpha$, such that
- $\alpha \in F'$, then there exist $q_0, q_1, \ldots, q_k \in Q_w$ and $p_1, \ldots, p_k \in Q_d$ such that
  - $f_{2n+1}(\cdot) = q_0$,
  - for every $i : 0 \leq i < k$, $(q_i, a, p_{i+1}, -1) \in \delta_w$, and for every $i : 1 \leq i < k$, $f_{2n}(p_i) = q_i$,

20
\[ q_i \in F. \]

From the fact that \( f_{2n+1}(\cdot) = q_0 \neq \perp \), we deduce from the definition of \( \delta' \) in \( A' \) that \( f_i(\cdot) \neq \perp \) for every \( i : 0 \leq i \leq 2n \).

It follows that for every \( i : 1 \leq i \leq 2n + 1 \), \( f_i(\cdot) = f_{i-1}(f_{i-1}(\cdot)) \).

From the fact that \( f_0(\cdot) \neq \perp \), there is \( p_0 \in I \) such that \( (p_0,+,f_0(\cdot),+1) \in \delta_w \).

By induction on \( i : 0 \leq i \leq 2n \), we can show that if \( f_i(q) = q' \), then there is a subrun \( \rho^i_{q,q'} \) from \((q,i+1)\) to \((q',i+2)\) of \( A \) over \( \alpha \).

Consider the composition of the following subruns,

\[ (p_0,0) = (f_0(\cdot),1), \rho^1_{f_0(\cdot),f_0(\cdot)}, \rho^2_{f_0(\cdot),f_0(\cdot)}, \ldots, \rho^2_{f_0(\cdot),f_0(\cdot)}, (q_0,2n+2) = (p_1,2n+1), \rho^2_{p_1,q_1}, \]

\[ (q_1,2n+2) = (p_2,2n+1), \rho^2_{p_2,q_2}, \ldots, (q_{k-1},2n+2) = (p_k,2n+1), \rho^2_{p_k,q_k}. \]

Let \( \rho \) denote this composition. Then \( \rho \) is an accepting run of \( A \) over \( \alpha \). \( \square \)

**D. PROOFS IN SECTION 4**

**Proposition 10** The evaluation for 2RRDPQs is PS-PSPACE-complete, and NLLOGSPACE-complete in data complexity.

**Proof.** The upper bound:

We use the idea to prove the PS-PSPACE upper bound for 2RDPQs in [18].

Let \( \mathcal{G} = (V,E,\eta) \) be a data graph, \( \xi = (x,L,y) \) be a 2RDPQ, and \((v_1,v_2) \in V \times V \). Suppose \( L \) is given by a NRRRA \( A = (Q,\delta,I,F) \) over \( \Sigma^+ \).

Let \( D \) be the set of data values occurring in \( \mathcal{G} \). Then \( \mathcal{G} \) plus \((v_1,v_2)\) can be seen as a NFA \( A_{\mathcal{G},(v_1,v_2)} = (Q',\delta',I',F') \) with initial state \((v_1)\), final state \((v_2)\) over the alphabet \( \Sigma^+ \cup D \) as follows.

- \( Q' = \{ v_1, v \in V \} \)
- \( \delta' = \{(v_1,a,v_1'),(v_1,\alpha,v_1) \mid (v,a,v') \in E \} \cup \{(v_2,d,v_2) \mid v \in V, \eta(v) = d \} \)
- \( I' = \{ v_1 \}, F' = \{ v_2 \} \)

From the proof of Theorem 3, we know that from \( A \), an equivalent NRRRA \( B = (Q',k,\delta',I',F') \) can be constructed such that \( |Q'| \) is polynomial over \( |Q| \) and exponential over \( |T_A| \) and \( k \) is polynomial over \( |T_A| \).

When restricted to the data paths where all data values are from \( D \), the NRRRA \( B \) can be seen as a NFA \( B' \) over the alphabet \( \Sigma^+ \cup D \) with the state space \( Q' \times D^{|k|} \). It follows that the size of the state space of \( B' \) is exponential over the size of \( A \) and polynomial over the size of \( D \).

To decide whether \((v_1,v_2) \in \xi(\mathcal{G}) \), it is sufficient to check whether \( \mathcal{L}(A_{\mathcal{G},(v_1,v_2)}) \neq \emptyset \). Since an accepting run of \( A_{\mathcal{G},(v_1,v_2)} \cap B' \) can be guessed in polynomial space, from Savitch's theorem, we conclude that the evaluation problem of NRRAs is in PS-PSPACE.

If the size of \( A \) is bounded by a constant, then an accepting run of \( A_{\mathcal{G},(v_1,v_2)} \cap B' \) can be guessed in logarithmic space, it follows that the upper bound of the data complexity of the evaluation problem of NRRAs is NLLOGSPACE.

The PS-PSPACE lower bound is obtained by an easy reduction from the nonemptiness of NRRAs. The NLOGSPACE lower bound of data complexity is from that of RQPs. \( \square \)

**Theorem 4.** Let \( k \geq 1, \xi = (x,L,y) \) be a 2RDPQ over the alphabet \( \Sigma \) such that \( L \) is given by a NRA or REM containing at most \( k \)-registers. Then a 2RDPQ \( \xi' = (x,L',y) \) over the alphabet \( \Sigma^+ \cup \{ A_i, A_i^{-1} \mid 1 \leq i \leq k \} \) can be constructed in polynomial time such that for every data graph \( \mathcal{G} = (V,E,\eta) \), \( \xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn,k}) \).

**Proof.** Let \((x,L,y)\) be a 2RDPQ. We first consider the situation that \( L \) is given by a NRA \( A = (Q,k,\delta,I,F) \) over the alphabet \( \Sigma^+ \).

In the following, we will construct a NARRA \( A' = (Q',\delta',I',F') \) over \( \Sigma^+ \cup \{ A_i, A_i^{-1} \mid 1 \leq i \leq k \} \) so that \( \xi' = (x,\mathcal{L}(A'),y) \) satisfies that \( \xi(\mathcal{G}) = \xi'(\mathcal{G}_{dn,k}) \).

The intuition of \( A' \) is to simulate the run of \( A \), by using the following tricks.

Every time a data value is stored into the \( i \)-th register in \( A \), the sequence \( dA_i dA_i^{-1} d \) is read by \( A' \). Later on, we can refer to the data values stored in the \( i \)-th register by using the position terms \( pred_{A_i} \).

We formally define \( A' = (Q',\delta',I',F') \) as follows.

1. \( Q' = Q_w \cup Q_d \) such that
   - \( Q_w \) is the union of \( Q_w \) and \( \delta_d \times \{ A_i, A_i^{-1} \mid 1 \leq i \leq k \} \),
   - \( Q_d \) is the union of \( Q_d \) and \( \delta_d \times \{ \#_i \mid 1 \leq i \leq k \} \).
\[ \delta' = \delta_w' \cup \delta_d' \]

- For every transition \((q, c, q', X) \in \delta_d\), let \(e' \in C_{rgd}[\Sigma^*] \) be obtained from \(c\) by replacing every \(r_j\) \((1 \leq j \leq k)\) with \(pred_{A_j}\) and \(r_0\) with \(cur\). If \(X = \emptyset\), then \((q, e', q') \in \delta_d'\); otherwise, let \(X = \{r_1, \ldots, r_i\}\), then \(\delta'\) includes the following transitions,

\[
\begin{align*}
\gamma' &\rightarrow ((q, c, q', X), A_{i_1}) A_{i_1}^{true} \rightarrow ((q, c, q', X), A_{i_1}) \\
\gamma' &\rightarrow ((q, c, q', X), #_1) A_{i_2}^{true} \rightarrow ((q, c, q', X), A_{i_2}) \\
\gamma' &\rightarrow ((q, c, q', X), A_{i_2}) A_{i_2}^{true} \rightarrow ((q, c, q', X), A_{i_2}) \\
\gamma' &\rightarrow ((q, c, q', X), #_i) A_{i_i}^{true} \rightarrow ((q, c, q', X), A_{i_i}) \\
\gamma' &\rightarrow ((q, c, q', X), #_i) A_{i_i}^{true} \rightarrow ((q, c, q', X), A_{i_i}) \\
\gamma' &\rightarrow ((q, c, q', X), #_i) A_{i_i}^{true} \rightarrow ((q, c, q', X), A_{i_i}) \\
\gamma' &\rightarrow ((q, c, q', X), #_i) A_{i_i}^{true} \rightarrow ((q, c, q', X), A_{i_i}) \\
\end{align*}
\]

If \(L\) is given by a REM \(e\) over the alphabet \(\Sigma^*\), we construct a RREM \(e'\) such that \(\xi' = (x, L(e'), y)\) over the alphabet \(\Sigma^* \cup \{A_i, A_i^{-1} \mid 1 \leq i \leq k\}\) satisfies that \(\xi(\emptyset) = \xi'(\emptyset_{kn,k})\).

From a REM \(e\), we construct a RREM \(tr(e)\) by an induction on the structure of REMs. The nontrivial cases are \(e = \{X e_1\} e_1\) and \(e = e_1[e]\). For \(e = \{X e_1\} e_1\), suppose \(X = \{r_1, \ldots, r_i\}\), then \(tr(e) = A_{i_1} A_{i_1} \cdots A_{i_1} tr(e_1)\). For \(e = e_1[e]\), let \(e' \in C_{rgd}\) be obtained from \(c\) by replacing \(r_j\) with \(pred_{A_j}\) and \(r_0\) with \(cur\), then \(tr(e) = tr(e_1) \cdot [e']\). \(\square\)

\textbf{E. PROOFS IN SECTION 5}

\textbf{Proposition 11.} The evaluation of \(C2RDPQs\) is \(\text{PSPACE-complete, and NLOGSPACE-complete in data complexity.}\)

\textbf{Proof.} The PSPACE lower bound follows from that of \(2RDPQs\). The NLOGSPACE lower bound follows from that of \(PQs\).

The upper bound:

Let \(\xi := Anz(\emptyset) \leftarrow \wedge_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})\) be a \(C2RDPQ\), \(\mathcal{G} = (V, E, \eta)\) a data graph, and \(\pi\) a tuple of nodes of the same arity as \(\pi\). Suppose for every \(i: 1 \leq i \leq l\), \(L_i\) is given by an NRRA \(A_i = (Q_i, \delta_i, I_i, F_i)\) over the alphabet \(\Sigma^*\).

From the proof of Theorem 3, we know that for every \(i\) from each \(A_i\), an equivalent NRAG \(B_i = (Q'_i, k_i, \delta'_i, I'_i, F'_i)\) can be constructed such that \(|Q'_i|\) is polynomial over \(|Q_i|\) and exponential over \(|T_{A_i}|\), and \(k\) is polynomial over \(|T_{A_i}|\).

When restricted to the data paths where all data values are from \(D\), the NRAG \(B_i\) can be seen as an NFA \(B'_i\) over the alphabet \(\Sigma^* \cup D\) with the state space \(Q'_i \times D^{[k_i]}\). It follows that the size of the state space of \(B'_i\) is exponential over the size of \(A_i\) and polynomial over the size of \(D\).

To check whether \(e \in \xi(\emptyset)\), an assignment \(\nu\) of nodes in \(V\) to \(\{y_1, \ldots, y_{2l}\}\) is first guessed such that \(\nu(\pi) = \pi\).

Similarly to the proof of Proposition 10, for every pair \((v(y_{2i-1}), v(y_{2i}))\), the data graph \(\mathcal{G}\) together with \((\nu(y_{2i-1}), \nu(y_{2i}))\) can be seen as a NFA \(A_{\mathcal{G}, i}\) over the alphabet \(\Sigma^* \cup D\) with the initial state \((\nu(y_{2i-1}))\) and the final state \((\nu(y_{2i}))\).

Then for every \(i: 1 \leq i \leq l\), an accepting run of \(A_{\mathcal{G}, i} \cap B'_i\) can be guessed in polynomial space. To check whether \(\pi \in \xi(\emptyset)\), the accepting runs of the NFAs \(A_{\mathcal{G}, i} \cap B'_i\) can be guessed one by one. From Savitch’s theorem, we deduce that the nonemptiness of \(C2RDPQs\) is in PSPACE.

Similarly, if the size of \(\xi\) is bounded by a constant, then the assignment \(\nu\) and the accepting runs of \(A_{\mathcal{G}, i} \cap B'_i\) can be guessed in logarithmic space. Therefore, the evaluation problem of \(C2RDPQs\) is the NLOGSPACE data complexity. \(\square\)

\textbf{Lemma 2.} Suppose \(\pi' = v_0a_1v_1 \cdots a_la_l\) is a semipath in \(\mathcal{G}\) such that \(ur\pi'(\pi') = \pi_0' \# \cdots \# \pi_l'\) and \(tr\pi(\pi') = p_0b_1p_1 \cdots b_{2r}p_2r \) \((\text{where } b_1, \ldots, b_{2r} \in \Sigma^* \cup \{#\})\). Then for every \(i: 0 \leq i \leq l + 2r\), there exists a function \(pos_i \in \{T_{A_i}[\Sigma^*] \cup \{\}\}^{T}\) such that for every \(t \in T_A\), \(pos_i(t) = 1\) iff \(t_{ur\pi'(\pi')}[2i] = 1\); moreover, if \(pos_i(t) = 1\) and \(t_{ur\pi'(\pi')}[2i] = 1\), then \(pos_i(t)_{A_{\mathcal{G}, i}}[p_i] = p_i\).

\textbf{Proof.} Let \(\pi' = v_0a_1v_1 \cdots a_la_l\) be a semipath in \(\mathcal{G}\), \(ur\pi'(\pi') = \pi_0' \# \cdots \# \pi_l'\) such that for every \(s: 0 \leq s \leq r\), all the edges on \(\pi_s'\) belong to \(\pi_s\), for some \(j_s: 1 \leq j_s \leq l_1\), and \(tr\pi(\pi') = p_0b_1p_1 \cdots b_{2r}p_2r \) \((\text{where } b_1, \ldots, b_{2r} \in \Sigma^* \cup \{#\})\).

We prove the lemma by an induction on the structure of position terms.

Induction base: For every \(i: 0 \leq i \leq l + 2r\), \(pos_i(cur) = cur\).

Induction step:
Let us first consider the case \( t = suc(t_1) \).

Let \( i : 0 \leq i \leq \ell + 2r \).

If \( t^\text{adj}_{\text{urv}(\pi_s)}[2i] = 1 \), then let \( \text{pos}_i(t) = 1 \). Otherwise, \( t^\text{adj}_{\text{urv}(\pi_s)}[2i] = 2i' \) for some \( i' \). From \( t = suc(t_1) \), it follows that

\[
(t_1)^\text{adj}_{\text{urv}(\pi_s)}[2i] = 2i'' \text{ for some } i'' \text{ such that } 2i'' = 2i' + 2 \text{ or } 2i'' = 2i'' + 4.
\]

According to the induction hypothesis, there exists \( \text{pos}_i(t_1) \in T_p[\Sigma_{\xi_1}] \) such that (\( \text{pos}_i(t_1) \))\( _{\alpha_g} (p_i) = p_{i''} \).

- If \( 2i'' = 2i'' + 2 \), then let \( \text{pos}_i(t) = \text{pos}_i(suc(t_1)) = suc(\text{pos}_i(t_1)) \) if \( p_{i''} < p_{i'} \), otherwise, let \( \text{pos}_i(t) = \text{pos}_i(suc(t_1)) = pre\text{d}(\text{pos}_i(t_1)) \).
- If \( 2i'' = 2i'' + 4 \), then there are \( j_1, j_2 : 1 \leq j_1, j_2 \leq l_1 \) such that one of the following conditions holds,
  1. \( p_{i''} \) is the position immediately before \( \$_{2j_1} \), \( p_{i'} \) is the third position before \( \$_{2j_2} \) in \( \alpha_g \), and \( j_1 \neq j_2 \).
  2. \( p_{i''} \) is the position immediately before \( \$_{2j_2} \), \( p_{i'} \) is the second position after \( \$_{2j_2} \).
  3. \( p_{i''} \) is the position immediately after \( \$_{2j_1} \), \( p_{i'} \) is the position after \( \$_{2j_2} \).
  4. \( p_{i''} \) is the position immediately after \( \$_{2j_1} \), \( p_{i'} \) is the second position after \( \$_{2j_2} \).

We illustrate the argument by considering the second situation above. The arguments for the other three situations are similar.

\[
\begin{align*}
\text{if } j_1 > j_2, \text{ then let } & \text{pos}_i(suc(t_1)) = suc(suc_{z_{2j_1-1}}(\text{pos}_i(t_1))), \\
\text{if } j_1 < j_2, \text{ then let } & \text{pos}_i(suc(t_1)) = suc^2(\text{pred}_{z_{2j_2-1}}(\text{pos}_i(t_1))).
\end{align*}
\]

The case \( t = pre\text{d}(t_1) \) can be discussed similarly as \( t = suc(t_1) \).

Now consider the case \( t = suc_A(t_1) \).

Let \( i : 0 \leq i \leq \ell + 2r \).

If \( t^\text{adj}_{\text{urv}(\pi_s)}[2i] = 1 \), let \( \text{pos}_i(t) = 1 \). Otherwise, let \( t^\text{adj}_{\text{urv}(\pi_s)}[2i] = 2i' \). From \( t = suc_A(t_1) \), we know that \( (t_1)^\text{adj}_{\text{urv}(\pi_s)}[2i] = 2i'' \) for some \( i'' \) such that \( 2i'' < 2i' \).

From the induction hypothesis, \( (\text{pos}_i(t_1))_{\alpha_g} (p_i) = p_{i''} \).

If there are no \( \# \) symbols in the subpath of \( \text{urv}(\pi_s) \) from the position \( 2i'' \) to \( 2i' \), then the position \( 2i'' \) and \( 2i' \) both belong to \( \pi_s' \) for some \( s : 0 \leq s \leq r \). It follows that \( p_{i''} \) and \( p_{i'} \) are two positions between \( \$_{2j_1} \) and \( \$_{2j_2} \) in \( \alpha_g \).

Define \( \text{pos}_i(t) \) as follows.

- If \( p_{i''} < p_{i'} \), let \( \text{pos}_i(suc_A(t_1)) = pre\text{d}_{A_\pi}(p_i) \).
- If \( p_{i''} < p_{i'} \), let \( \text{pos}_i(suc_A(t_1)) = suc_{A_\pi}(p_i) \).

Otherwise (that is, there are \( \# \) symbols from \( 2i'' \) to \( 2i' \) ), let \( 2i'' \) be the position before the position \( 2i' \) on \( \text{urv}(\pi_s) \) such that \( 2i'' \) is a position immediately after \( \# \) and \( 2i'' \) is the last position before \( 2i' \) satisfying this property. Let \( s : 0 \leq s \leq r \) such that \( 2i'' \) and \( 2i' \) are two positions belonging to \( \pi_s' \). Then \( p_{2i''} \) is the position immediately after \( \$_{2j_2} \) or the position immediately before \( \$_{2j_1} \) in \( \alpha_g \). We illustrate the argument by considering the situation that \( p_{2i''} \) is the position immediately after \( \$_{2j_2} \). The discussion for the latter situation is similar. Define \( \text{pos}_i(t) \) as follows.

- If \( p_{i''} < p_{i'} \), let \( \text{pos}_i(suc_A(t_1)) = suc_{A_\pi}(p_i) \).
- If \( p_{i''} < p_{i'} \), let \( \text{pos}_i(suc_A(t_1)) = suc_{A_\pi}(suc_{z_{2j_2-1}}(p_i)) \).

The case \( t = pre\text{d}(t_1) \) can be discussed similarly to \( t = suc_A(t_1) \).

In summary, for every \( t \), \( op(t) \in T_p[\Sigma^+] \) such that \( \text{pos}_i(t) \neq 1 \) (where \( op = suc, pre\text{d}, suc_A, pre\text{d}_A \)), there is \( t_{op} \in T_p[\Sigma_{\xi_1}] \) such that \( \text{pos}_i(t) = t_{op}[\text{cur}(\text{pos}_i(t))] \).

\[ \square \]

**Theorem 6.** Let \( \mathcal{G} \) be a \( \nu \)-canonical data graph for \( \xi_1 \), \( \xi \) be a 2RRDPQ. Then a 2NRRA \( A_\xi \) can be constructed from \( \xi \) and \( \xi_1 \) such that \( (\xi, \mathcal{G}) \) is nonempty iff \( A_\xi \) accepts \( \nu \text{-}A \).
A locating profile $\text{loc}$ of $\mathcal{A}'$ over $\alpha_{\varphi}$, is defined as a pair $((S, \chi, \sim), \text{pos})$, where $(S, \chi, \sim) \in \text{Prof}_A$ (cf. proof of Theorem 3),

$$
\chi = \left( b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1} \right) \ldots \left( b_{-1}, T_{-1}, b'_{-1}, s_{-1} \right) \left( b_0, T_0, b'_0, s_0 \right) \left( b_1, T_1, b'_1, s_1 \right) \ldots \left( b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2} \right)
$$

and $\text{pos} : (T_p[\Sigma_{\xi_1}] \cup \{1\})^{T_A}$ such that

- $\text{pos}(t) = 1$ for every $t \in T_A \setminus S$,
- $\text{pos}(\text{cur}) = \text{cur}$,
- for every $t, t' \in S$ such that $t \neq t'$, we have $\text{pos}(t) \neq \text{pos}(t')$, for every $t_1, t_2 \in S$ such that there is $j : -m_1 \leq j \leq m_2$ satisfying that $t_1, t_2 \in T_j$, if $\text{op}(t_1), \text{op}(t_2) \in S$ for $\text{op} \in \{\text{succ}, \text{pred}, \text{succ}_A, \text{pred}_A\} \subseteq \Sigma^*$, then there is $\text{op}_j \in T_p[\Sigma_{\xi_1}]$ of the form as those in the proof of Lemma 2 (e.g. $\text{t}_{\text{succ}} = \text{succ}(\text{succ}_{\chi_{j+1}}(\text{cur}))$) such that $\text{pos}(\text{op}(t_j)) = \text{op}_j(\text{cur}^\text{\text{pos}}(t_j))$ for $j = 1, 2$.

Let $\Sigma_{\text{loc}}$ denote the set of locating profiles.

Similar to the construction of NRRGs from NRRAs, we define two successor relations between locating profiles. Let $((S_1, \chi_1, \sim_1), \text{pos}_1), ((S_2, \chi_2, \sim_2), \text{pos}_2) \in \Sigma_{\text{loc}}$, $a \in \Sigma^*$, $j : 1 \leq j \leq l_1$, $1 \leq k_1, k_2 \leq 2l_1$, and $\text{dir} \in \{+1, -1\}$.

In the following, we will define two relations $((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(\alpha_{\varphi}, \text{dir})} ((S_2, \chi_2, \sim_2), \text{pos}_2)$ and $((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a, \text{dir})} ((S_2, \chi_2, \sim_2), \text{pos}_2)$. The latter relation corresponds to the situation that the run of $\mathcal{A}'$ is jumping over # on urv$\mathcal{A}'(\pi)$, and the former relation corresponds to the situation that the run of $\mathcal{A}'$ is not.

At first, $((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a_{\text{dir}})} ((S_2, \chi_2, \sim_2), \text{pos}_2)$ if the following conditions hold.

- $((S_1, \chi_1, \sim_1), \text{pos}_1) \overset{a}{\rightarrow} ((S_2, \chi_2, \sim_2))$.
- If $\text{succ}(\text{cur}) \in S_1$, then $\text{pos}_1(\text{succ}(\text{cur})) = \text{succ}(\text{cur})$ if $\text{dir} = +1$, and $\text{pos}_1(\text{succ}(\text{cur})) = \text{pred}(\text{cur})$ otherwise.
- If $a \in A$ and $\text{succ}_A(\text{cur}) \in S_1$, then $\text{pos}_1(\text{succ}_A(\text{cur})) = \text{succ}_A(\text{cur})$ if $\text{dir} = +1$, and $\text{pos}_1(\text{succ}_A(\text{cur})) = \text{pred}_A(\text{cur})$ otherwise.
- For every $t \in S_1$ such that $\text{succ}(\text{cur}) \neq t$, if $\text{dir} = +1$, then $\text{pos}_2(t[\text{succ}(\text{cur})]) = \text{pos}_1(t)[\text{succ}(\text{cur})]$;
- For every $t \in S_2$ such that $\text{pred}(\text{cur}) \neq t$, $\text{pos}_2(t) = \text{pos}_1(t)[\text{pred}(\text{cur})]$. 
- For every $t \in S_1$ such that $\text{succ}(\text{cur}) \neq t$ and $a \in A$, $\text{pos}_2(t) = \text{pos}_1(t)$.
- For every $t \in S_1$ such that $\text{pred}(\text{cur}) \neq t$ and $a \notin A$, $\text{pos}_2(t) = \text{pos}_1(t)$.

In the following, we will define $((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a_{\text{dir}}, \text{dir})} ((S_2, \chi_2, \sim_2), \text{pos}_2)$ for $k_1, k_2$ satisfying that there are $k_1, k_2 : 1 \leq k_1, k_2 \leq l_1$ such that one of the following conditions hold.

1. $k_1 = 2k_1', k_2 = 2k_2'$, and $k_1' \neq k_2'$,
2. or $k_1 = 2k_1', k_2 = 2k_2' - 1$,
3. or $k_1 = 2k_1' - 1$, $k_2 = 2k_2'$,
4. $k_1 = 2k_1' - 1$, $k_2 = 2k_2' - 1$, and $k_1' \neq k_2'$.

We will illustrate the definition for the first case above, the other three cases can be discussed in the same way. Suppose $k_1 = 2k_1'$, $k_2 = 2k_2'$, and $k_1' \neq k_2'$ for some $k_1', k_2'$. Then $((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a_{\text{dir}}, \text{dir})} ((S_2, \chi_2, \sim_2), \text{pos}_2)$ if the following conditions hold.

- $((S_1, \chi_1, \sim_1), \text{pos}_1) \overset{a}{\rightarrow} ((S_2, \chi_2, \sim_2))$.
- If $\text{succ}(\text{cur}) \in S_1$, then $\text{pos}_1(\text{succ}(\text{cur})) = \text{pred}(\text{succ}_A(\text{cur}))$ if $k_1' < k_2'$, and $\text{pos}_1(\text{succ}(\text{cur})) = \text{pred}(\text{succ}_A(\text{cur}))$ otherwise.
• If \( a \in A \) and \( su\_A(car) \in S_1 \), then \( pos_1(su\_A(car)) = pred_{A_{\Sigma^k}}(suc_{S_{k_2}}(car)) \) if \( k_1 < k_2 \), and \( pos_1(su\_A(car)) = pred_{A_{\Sigma^k}}(pred_{S_{k_2}}(car)) \) otherwise.

• For every \( t \in S_1 \) such that \( su\_A(car) \leq t \), if \( k_1 < k_2 \), then
  \[
  pos_2(t[su\_A(car)] = pos_1(t)[pred^2(suc_{S_{k_2}}(car))],
  \]
  otherwise,
  \[
  pos_2(t[su\_A(car)] = pos_1(t)[pred(pred_{S_{k_2}}(car))].
  \]

• For every \( t \in S_1 \) such that \( su\_A(car) \leq t \) and \( a \in A \), if \( k_1 < k_2 \), then,
  \[
  pos_2(t[su\_A(car)] = pos_1(t)[pred_{A_{\Sigma^k}}(suc_{S_{k_2}}(car))],
  \]
  otherwise,
  \[
  pos_2(t[su\_A(car)] = pos_1(t)[pred_{A_{\Sigma^k}}(pred_{S_{k_2}}(car))].
  \]

• For every \( t \in S_2 \) such that \( pred_{A}(car) \leq t \), \( pos_2(t) = pos_1(t[\{pred_{A}(car)\}].

• For every \( t \in S_2 \) such that \( pred_{A}(car) \leq t \) and \( a \in A \), \( pos_2(t) = pos_1(t[\{pred_{A}(car)\}].

• For every \( t \in S_1 \) such that \( su\_A(car) \leq t \) and \( a \notin A \), \( pos_2(t) = pos_1(t).\)

We are ready to construct the 2NRA \( B \).

Suppose \( \pi' = \pi_0a_1v_1 \ldots a_sv_s \) is a semilpath in \( \alpha_\varphi \), \( urv_{\varphi}(\pi') = \pi_0' \# \ldots \# \pi_r' \), for every \( s : 0 \leq s \leq r \), all the edges on \( \pi_s' \) belong to \( \pi_j', (1 \leq j \leq l) \), and \( trc(\pi') = d_0d_1p_1 \ldots d_{t+2r}p_{t+2r} \) (where for every \( j : 1 \leq j \leq t + 2r, a_j' \in \Sigma^* \cup \{\#\} \)). Then \( B \) does the following.

• In each position \( p_i \) (\( 0 \leq i \leq t + 2r \)) of \( \alpha_\varphi \), \( B \) guesses a locating profile \( loc_i = ((S_i, \chi_i, \sim_i), pos_i) \in \Sigma_{loc} \) with
  \[
  \chi_i = (b_{i,-m_1}, T_{i,-m_1}, b_{i,-m_1}', s_{i,-m_1}) \ldots (b_{i,-1}, T_{i,-1}, b_{i,-1}', s_{i,-1}) \ldots (b_{i,0}, T_{i,0}, b_{i,0}', s_{i,0})(b_{i,1}, T_{i,1}, b_{i,1}', s_{i,1}) \ldots (b_{i,m_2}, T_{i,m_2}, b_{i,m_2}', s_{i,m_2}).
  \]

In addition,

– if \( i = 0 \), then \( loc_0 = (S_0, \chi_0, \sim_0) \) is an initial locating profile, that is, for every \( t \in T_A \) such that \( pred_{A}(car) \leq t \) or \( suc_{A}(car) \leq t \) for some \( A \in \Sigma^* \) if \( t \notin S_0 \),

– if \( i = t + 2r \), then \( loc_{t+2r} = (S_{t+2r}, \chi_{t+2r}, \sim_{t+2r}) \) is a final profile, that is, for every \( t \in T_A \) such that \( suc_{A}(car) \leq t \) or \( suc_{A}(car) \leq t \) for some \( A \in \Sigma^* \) if \( t \notin S_{t+2r} \).

• Over each pair of positions \( p_i \) and \( p_{i+1} \) (where \( 0 \leq i \leq t + 2r \)) of \( \alpha_\varphi \),

– if \( i_s + 2s \leq 2i < i_{s+1} + 2s \) for some \( s : 0 \leq s \leq r \), then \( B \) checks that \( loc_i = \{a_{i+1}, j_{s}, dir\} \rightarrow loc_{i+1} \), where \( dir = +1 \) if \( p_{i+1} = p_i + 2 \), and \( dir = 1 \) otherwise,

– if \( 2i = i_s + 2(s - 1) \) for some \( s : 1 \leq s \leq r \) (that is, \( 2i \) is the position immediately before \( \# \) in \( urv_{\varphi}(\pi') \)), then
  \( B \) jumps from \( p_i \) to \( p_{i+1} \), then to \( p_{i+2} \), and checks that \( loc_i = \{a_{i+1}, s_k, s_{k'}, \rightarrow \} \rightarrow loc_{i+1} \), where
  \[
  k = 2j_{s+1} \text{ if } p_{a_{s+1}}(\pi_{j_{s+1}}, v_{j_{s+1}}) \text{ is the position immediately before } S_{2j_{s+1} - 1}, \text{ and } k = 2j_{s+1} - 1 \text{ if } p_{a_{s+1}}(\pi_{j_{s+1}}, v_{j_{s+1}}) \text{ is the position immediately after } S_{2j_{s+1} - 1},
  \]

  \[
  k' = 2j_{s} \text{ if } p_{a_{s}}(\pi_{j_{s}}, v_{j_{s}}) \text{ is the position immediately before } S_{2j_{s}} - 1, \text{ and } k' = 2j_{s} - 1 \text{ if } p_{a_{s}}(\pi_{j_{s}}, v_{j_{s}}) \text{ is the position immediately after } S_{2j_{s} - 1}.\]

• At the same time, \( B \) simulates the run of \( \mathcal{A'} \) over \( \eta(u_{\varphi}(\pi')) \) as follows.

– If \( \mathcal{A'} \) makes a transition \( (q, a, c, q') \) over \( a_i \), then \( B \) checks that \( b_{i-1,0} = a_i \) and changes the state from \( q \) to \( q' \).

– If \( \mathcal{A'} \) makes a transition \( (q, c, q') \) in the position \( 2i \) of \( \eta(u_{\varphi}(\pi')) \), then \( B \) checks that \( (S_i, \chi_i, \sim_i) \) satisfies \( c \), verifies that the data value in the current position is equal to the data value in the position represented by \( pos_2(t) \) for each \( t \in S_i \) such that \( car \sim_i t \), and changes the state from \( q \) to \( q' \).

– \( B \) accepts if \( \mathcal{A'} \) accepts and a final profile is reached.

From the above construction, we know that in its states, \( B \) should record the states of \( \mathcal{A'} \) and the guessed locating profiles. Because both the number of profiles and the number of functions \( pos \) in locating profiles are exponential over \(|T_A|\), it follows that the number of states of \( B \) is polynomial over \(|Q|\) and exponential over \(|T_A|\). □