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 are 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 RDPQs are reasonable 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 (3,4).

RPQs have also been extended in another way, called regular data path queries (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 RDPQs, the evaluation problem is PSPACE-complete, and NLOGSPACE-complete in data complexity. On the other hand, the containment problem of RDPQs 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 RDPQs seems
to undermine the validity of RDPQs 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 2RDPQs 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 NRAs enjoy nice properties as finite state automata, that is, NRAs can be determined, they are closed under all Boolean operations, the two-way variant of NRAs is expressively equivalent to (one-way) NRAs, and they are expressively equivalent with a natural extension of regular expressions. In addition, while the expressive power of NRAs and NRAs are incomparable, we demonstrate that the expressive power of NRAs 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 2RDPQs are incomparable, every 2RDPQ 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 \(2\textup{EXPSPACE}\). From this, we deduce that the containment problem of 2RRDPQs can be decided in \(2\textup{EXPSPACE}\) as well. The \(2\textup{EXPSPACE}\) result is proved by a nontrivial extension of the proof for the \(\textup{EXPSPACE}\)-completeness result of C2RDPQs 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, NRAs are defined to allow much more complicated data value comparisons. For instance, in NRAs, 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 \(2\textup{EXPSPACE}\) 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. NRAs 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 \(\Sigma^* = \Sigma \cup \{a^- \mid a \in \Sigma\}\). For \(a \in \Sigma^*\), we use \(a^-\) to denote the inverse of \(a\). In particular, \((a^-)^{−1} = a\) for \(a \in \Sigma\).

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 \(d_0d_1d_2\ldots d_n\), 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 \(\alpha_1 = d_0d_1\ldots d_n\) and \(\alpha_2 = d_0, a_{n+1}\ldots a_m, \) the concatenation of \(\alpha_1\) and \(\alpha_2\), denoted by \(\alpha_1 \cdot \alpha_2\), is defined as the following data path, \(d_0d_1\ldots a_n d_{n+1}\ldots a_m d_m\). 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 \(\{a \in \Sigma \mid \text{prj}(a) = \gamma\}\). Note that \((\text{prj}^{-1}(\gamma))\gamma\) 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 \( E \subseteq V \times \Sigma \times V \). For \( \epsilon = (v, a, v') \in E \), let \( \lambda (\epsilon) \) 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 \( (v_i, a_i^{-1}, v_{i+1}) \in E \). 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 \( G = (V, E, \eta) \), a node pair \( (v, v') \) in \( G \), decide whether \( (v, v') \in \xi (G) \).

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 \( G \). an \( \xi_1 (G) \subseteq \xi_2 (G) \).

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

A data graph \( G \) is a tuple \( (V, E, \eta) \), where \( (V, E) \) is a graph database and \( \eta : V \rightarrow \mathcal{D} \) 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 \( \eta (v_0)\eta (a_1)\eta (v_1) \cdots \eta (v_{n-1})\eta (a_n)\eta (v_n) \).

Let \( k \) be a natural number. A k-register data constraint is defined by the following rules:

\[
c := r_i \sim r_j \mid r_i \triangle r_j \mid c \lor c \mid c \land c,
\]

where \( 0 \leq i, j \leq k \), \( i \neq j \), and \( r_0 \) is a special register reserved for the current data value. Let \( C_k \) denote the set of data constraints.

Let \( c \) be a data constraint, \( \theta \in (\mathcal{D} \cup \{ 1 \})^{|k|} \) be the current state of registers (where \( \theta (i) = 1 \) 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 \( c = r_i \sim r_j \), then \( (\theta, d) \models c \) iff \( \theta (0 \leftarrow d)(i) \neq 1 \), \( \theta (0 \leftarrow d)(j) \neq 1 \), and \( \theta (0 \leftarrow d)(i) \neq \theta (0 \leftarrow d)(j) \), where \( \theta (0 \leftarrow d) \) is the function extending \( \theta \) by assigning \( d \) to 0. The semantics of \( c = r_i \triangle r_j \) 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, \( \text{NRA}_k \)) over \( \Sigma^* \) is a tuple \( (Q, k, \delta, I, F) \), where \( Q = Q_w \cup Q_d \) such that \( Q_w \) and \( Q_d \) are two finite disjoint sets of word states and data states, \( k \) is the number of registers, \( I \subseteq Q_d \) is the set of initial states, \( F \subseteq Q_w \) is the set of final states, \( \delta = \delta_w \cup \delta_d \) such that \( \delta_w \subseteq Q_w \times \Sigma^* \times Q_d \) is the word transition relation and \( \delta_d \subseteq Q_d \times C_k \times Q_w \times 2^{|k|} \) 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 \( Q_d \). 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 \( \alpha = d_0a_1d_1 \ldots a_nd_n \) 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 \( \theta \in (\mathcal{D} \cup \{ 1 \})^{|k|} \) is the current state of the registers. An initial configuration of \( A \) over \( \alpha \) is \( (q_0, 0, \theta_0) \), where \( q_0 \in I \), and \( \theta (0 \leftarrow d)(i) \neq 1 \) for every \( i \in [k] \). Let \( \alpha = (q_0, \ldots, q_k) \cdot (q'_{j+1}, \ldots, q'_{j+1}) \) be two configurations (where \( 0 \leq j < 2n \)). 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 \( (q, a, q') \in \delta_w, \theta' = \theta \).
- If the \( j \)-th symbol of \( \alpha \) is a data value \( d \), then there are \( c \in C_k, X \subseteq \Sigma \) such that \((q, c, q', X) \in \delta_d, (\theta, d) \models c \), and \( \theta' \) is obtained from \( \theta \) by assigning \( d \) to every \( i \in X \).

A data path \( \alpha = d_0a_1d_1 \ldots a_nd_n \) is accepted by a NRA \( A \) if there are \( q \in I, q' \in F \) and a data assignment \( \theta \) such that \( (q_0, \theta) \xrightarrow{\alpha} (q'_n, 2n + 1, \theta) \), where \( \xrightarrow{\alpha} \) is the reflexive and transitive closure of \( \xrightarrow{\alpha} \). The set of data paths accepted by \( A \) is denoted by \( \mathcal{L}(A) \). In addition, for every \( \theta, \theta' \in (\mathcal{D} \cup \{ 1 \})^{|k|} \), we use \( \mathcal{L}(A, \theta, \theta') \) to denote the set of data paths \( \alpha = d_0a_1d_1 \ldots a_nd_n \) such that there
are \( q \in I, q' \in F \) satisfying \((q, 0, \theta) \overset{\theta}{\rightarrow} (q', 2n + 1, \theta')\).

Let \( k \) be a natural number. Regular expressions with \( k \)-memory (REM, \[18\]) over \( \Sigma^* \) are defined by the following rules:

\[
e := \varepsilon \mid \emptyset \mid a \mid e \cdot e \mid e + e \mid e^+ \mid \downarrow X \ e \mid e[c],
\]

where \( a \in \Sigma^* \), \( c \in C_k \), and \( X \subseteq [k] \).

The semantics of REMs is defined by a relation \( \theta \overset{e}{\rightarrow} \theta' \), where \( e \) is a REM, \( \alpha \) is a data path, \( \theta, \theta' \in \mathcal{L}(\Sigma) \). 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 \( \theta \overset{e}{\rightarrow} \theta' \) if \( \theta_{X=\emptyset} \overset{e_1}{\rightarrow} \theta'_{X=\emptyset} \), where \( d \) is the first data value of \( \alpha \), and \( \theta_{X=\emptyset} \) is obtained from \( \theta \) by assigning \( d \) to all the registers in \( X \).

- If \( e = e_1[c] \), then \( \theta \overset{e}{\rightarrow} \theta' \) if \( \theta \overset{e_1}{\rightarrow} \theta' \) and \( (\theta', d) \models c \), where \( d \) is the last data value of \( \alpha \).

A data path \( \alpha \) is accepted by a REM \( e \) if there exists \( \theta \in \mathcal{L}(\Sigma) \) such that \( \theta \overset{e}{\rightarrow} \theta_\emptyset \). The set of data paths accepted by a REM \( e \) is denoted by \( \mathcal{L}(e) \). For every \( \theta, \theta' \in \mathcal{L}(\Sigma) \), we use \( \mathcal{L}(e, \theta, \theta') \) to denote the set of data paths \( \alpha \) such that \( \theta \overset{e}{\rightarrow} \theta' \).

A data path \( \alpha \) is obtained from \( \theta \) if \( \theta \overset{\alpha}{\rightarrow} \theta_\emptyset \).

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 determined 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,

\[
t := \text{cur} \mid \text{suc}(t) \mid \text{pred}(t) \mid \text{suc}_A(t) \mid \text{pred}_A(t),
\]

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 \( \text{suc}_a(t) \) or \( \text{pred}_a(t) \) (where \( a \in \Sigma^* \) ) are written as \( \text{suc}_a(t) \) or \( \text{pred}_a(t) \). In addition, we use \( \text{suc}_c \) to denote the repetitions of \( \text{suc} \) for \( i \) times. Similarly, we use the abbreviations \( \text{pred}_c \), \( \text{suc}_A \), and \( \text{pred}_A \).

The set of subterms of \( t \in T_p[\Sigma^*] \), denoted by \( \text{sub}(t) \), are defined in a natural way, e.g. \( \text{sub}(\text{suc}_A(t_1)) = \{ \text{suc}_A(t_1) \} \cup \text{sub}(t_1) \). We use \( t' \preceq t \) to denote the fact that \( t' \in \text{sub}(t) \), 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: Given a data path \( \alpha = d_0a_1d_1 \ldots a_n d_n \) and a position \( 2i \) (where \( 0 \leq i \leq n \) ), \( 2i \) is the position for the data value \( d_i \), 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.

- \( \text{cur}_\alpha[2i] = 2i \).

- If \( i < n \), then \( (\text{suc}(\text{cur}))_\alpha[2i] = 2(i+1) \). Otherwise, \( (\text{suc}(\text{cur}))_\alpha[2i] = 1 \).
If \( i > 0 \), then \((\text{pred}(\text{cur}))[a][2i] = 2(i - 1)\). Otherwise, \((\text{pred}(\text{cur}))[a][2i] = 1\).

• If \((t_1)[a][2i] ≠ 1\), then 
  \[(\text{suc}(t_1))[a][2i] = (\text{suc}(\text{cur}))[a][(t_1)[a][2i]].\]
  Otherwise, \((\text{suc}(t_1))[a][2i] = 1\).

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

If there exists \( j : i < j ≤ n \) such that \( a_j ∈ A \) and \( j \) is the minimum number satisfying this condition, that is, for every \( j' : i < j' < j \), we have \( a_{j'} ∉ A \), then \((\text{suc}_A(\text{cur}))[a][2i] = 2j\). Otherwise, \((\text{suc}_A(\text{cur}))[a][2i] = 1\).

If there exists \( j : j ≤ i \) such that \( a_j ∈ A \) and \( j \) is the maximum number satisfying this condition, that is, for every \( j' : j < j' ≤ i \), we have \( a_{j'} ∉ A \), then \((\text{pred}_A(\text{cur}))[a][2i] = 2(j - 1)\). Otherwise, \((\text{pred}_A(\text{cur}))[a][2i] = 1\).

If \((t_1)[a][2i] ≠ 1\), then 
\[(\text{suc}_A(t_1))[a][2i] = (\text{suc}_A(\text{cur}))[a][(t_1)[a][2i]].\]
Otherwise, \((\text{suc}_A(t_1))[a][2i] = 1\).

If \((t_1)[a][2i] ≠ 1\), then 
\[(\text{pred}_A(t_1))[a][2i] = (\text{pred}_A(\text{cur}))[a][(t_1)[a][2i]].\]
Otherwise, \((\text{pred}_A(t_1))[a][2i] = 1\).

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

A rigid data constraint \( c \) over \( \Sigma^* \) is defined by the following rules,
\[c := t_1 ∼ t_2 ∣ t_1 ∼ t_2 ∣ c ∨ c ∣ c ∧ c , \text{ where } t_1,t_2 ∈ T_p[\Sigma^*].\]

We use \( C_p[\Sigma^*] \) 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 \( c = t_1 ∼ t_2 \). The semantics of \( c = t_1 ∼ t_2 \) can be defined similarly. Moreover, the semantics of \( c = c_1 ∨ c_2 \) and \( c = c_1 ∧ c_2 \) can be defined in a standard way. Let \( c ∈ C_p[\Sigma^*] \) and \( \delta = d_0 a_1 d_1 ... a_n d_n \), and \( i : 0 ≤ i ≤ n \), then \((\alpha, 2i)\) is said to satisfy \( c = t_1 ∼ t_2 \), denoted by \((\alpha, 2i) = c_1\), if \((t_1)[a][2i] ≠ 1\), \((t_2)[a][2i] ≠ 1\), and \( d(t_1)[a][2i] = d(t_2)[a][2i] \).

Given a rigid data constraint \( c \), we use \( c \) to denote the negation of \( c \). More specifically, \( c \) is obtained from \( c \) by swapping \( ∼ \) for \( ∨ \) and \( ∨ \) for \( ∼ \). For instance, if \( c = \text{cur} ∼ \text{suc}_a(\text{cur}) ∨ \text{cur} ∼ \text{pred}(\text{cur}) \), then \( c = \text{cur} ∨ \text{suc}_a(\text{cur}) ∧ \text{cur} ∼ \text{pred}(\text{cur}) \).

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

### 3.2 Nondeterministic and deterministic rigid register automata

A nondeterministic rigid register automaton (NRRA) \( A \) over the alphabet \( \Sigma^* \) is a tuple \((Q, δ, I, F)\), where \( Q, I, F \) are as those in NRA, \( δ = δ_ω ∪ δ_δ \) such that \( δ_ω ≤ Q_w × \Sigma^* × Q_d \) and \( δ_δ ≤ Q_d × C_ω[\Sigma^*] × Q_w \).

A run of \( A \) over a data path \( α = d_0 a_1 d_1 ... a_n d_n \) is a state sequence \( q_0 q_0 a_1 q_2 ... q_2 n−1 a_n q_{2 n+1} \) such that \( q_0 ∈ I \), for every \( i: 0 ≤ i ≤ n \), \((q_{2 i}, c_i, q_{2 i+1}) ∈ δ_ω \) and \((α, 2 i) = c_i\), and for every \( i: 1 ≤ i ≤ n \), \((q_{2 i−1}, a_i, q_{2 i}) ∈ δ_δ\). A run \( ρ = q_0 q_0 a_1 q_2 ... q_{2 n−1} a_n q_{2 n+1} \) is accepting if \( q_{2 n+1} ∈ F \).

A deterministic rigid register automaton (DRA) over \( \Sigma^* \) is a NRA \( A = (Q, δ, I, F) \) such that \( I \) is a singleton, and \( δ \) satisfies that for every \( q ∈ Q_w, α ∈ \Sigma^* \), there is at most one \( q' ∈ Q_d \) such that \((q, a, q') ∈ δ_ω\), and for every \((q, c_1, q_1), (q, c_2, q_2) ∈ δ_δ\), if \( q_1 ≠ q_2 \), then \( c_1 ∧ c_2 \) is unsatisfiable.

Let \( A \) be a NRA. Then \( T_A \) is used to denote the minimal set of position terms satisfying that for every \( t_1 ∼ t_2 \) or \( t_1 ∼ t_2 \) occurring in \( A \), we have \( t_1, t_2 ∈ T_A \); moreover, if \( t ∈ T_A \) and \( t' \leq t \), then \( t', t[\text{\lnot \![\text{cur}]}] ∈ T_A \). In addition, \( C_A \) is used to denote the set of rigid data constraints occurring in \( A \).

**Example 2.** Let \( Σ = \{a, b\} \). Let \( L \) denote the language of data paths satisfying the sequence of word symbols on the data path belongs to \( ab^* \), 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 2, where \( \text{suc}_a \) is an abbreviation of \( \text{suc}_a(\text{cur}) \), \( Q_d = \{q_0, q_2, q_4, q_6\} \) 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 \( A = (Q, δ, I, F) \) be a NRA over the alphabet \( \Sigma^* \) and \( prj \) a letter projection from \( \Sigma^* \) to \( Γ \). Then the letter projection of \( A \), denoted by \( prj(A) \), is obtained
from $\mathcal{A}$ by replacing each transition $(q, a, q') \in \delta_w$ with 
$(q, prj(a), q')$, and each $suc_A$ (resp. $pred_A$) occurring 
in $\delta_d$, where $A \subseteq \Sigma^*$, with $suc_{prj(A)}$ (resp. $pred_{prj(A)}$).
Note that $prj(A)$ may not define $prj(\mathcal{L}(A))$, as wit-
nessed by the following result.

**Proposition 3.** The class of languages definable by 
NRRA's are not closed under letter projections.

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

Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRA over $\Sigma^*$, $prj$ be 
a letter projection from $\Sigma^*$ to $\Gamma$. Then $\mathcal{A}$ is said to 
be position-invariant under $prj$ if for every $suc_A$ (resp. $pred_A$) occurring 
in $\delta_d$, where $A \subseteq \Sigma^*$, there is $B \subseteq \Gamma$ 
such that $A = prj^{-1}(B)$. It is easy to observe that the 
position-invariance guarantees that for every $t \in T_\mathcal{A}$, 
every data path $\alpha$ and every position $2i$ of $\alpha$, 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 $\mathcal{A}$ is position invariant under $prj$, then 
$prj$ does not affect the interpretations of the rigid data 
constraints in $\mathcal{A}$. So we have the following result.

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

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 NR-
RA's can be simulated by an extension of NRAs with 
non-deterministic guessing.

Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRA over the alphabet 
$\Sigma^*$, $\alpha = d_0a_1d_1 \ldots a_n$ be a data path over $\Sigma^*$, $\rho =$

$q_0a_0q_1a_1q_2 \ldots q_{2n-1}a_nq_{2n}c_nq_{2n+1}$ be a run of $\mathcal{A}$ 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 
$\text{Pos}_\rho^1[\alpha, 2i]$, as 

$\{t_{\alpha}[2j] \mid j \leq i, t \text{ occurs in } c_j, t_{\alpha}[2j] \neq 1, t_{\alpha}[2j] > 2i\}$.

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

$\{t_{\alpha}[2j] \mid i \leq j, t \text{ occurs in } c_j, t_{\alpha}[2j] \neq 1, t_{\alpha}[2j] < 2i\}$.

**Lemma 1.** Let $\mathcal{A} = (Q, \delta, I, F)$ be a NRA over the 
alphabet $\Sigma^*$ and $\alpha = d_0a_1d_1 \ldots a_n$ be a data path. 
Then for every run $\rho$ of $\mathcal{A}$ over $\alpha$ and every $i : 0 \leq i \leq n$, 
$\text{Pos}_\rho^1[\alpha, 2i] \cup \text{Pos}_\rho^2[\alpha, 2i] \subseteq \{t_{\alpha}[2i] \mid t \in T_\mathcal{A}\}$.

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 $\rho$ (resp. before) reaching the position $2i$.

A nondeterministic register data path automaton with 
guessing (NRAG) $\mathcal{A}$ over $\Sigma^*$ is a tuple $(Q, \delta, I, F)$, 
where $Q, \delta, I, F$ are as those in the definition of NRA, 
$\delta \subseteq \delta_\alpha \cup \delta_\delta$ such that $\delta_\alpha \subseteq Q_\alpha \times \Sigma^* \times Q_\delta$ 
and $\delta_\delta \subseteq Q_\delta \times \Sigma \times Q_\delta$. The function $\delta$ satisfies 
that for every $(q, c, q', X, Y, c') \in \delta_\delta$, it holds that $X \cap Y = \emptyset$, 
and $c'$ does not contain $r_{k+1}$ with $r_i \notin Y$.

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 
$(q, \theta, r) \rightarrow (q', \theta', 1, \theta)$ can be defined, with the 
following adjustment for data transitions.

If the $j$-th symbol is a data value $d$, then there 
eexist $c, c'$ 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 $d'_i$ is guessed 
(thus $\theta'(i) = d'_i$), so that the guessed 
data values satisfy the following condition:

  - the function $\theta_g$ extending $\theta$ by assigning 
    $d'_i$ to $k + i$ for each $i \in Y$ satisfies 
    $\theta_g(d) = c'$,
- 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 e 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 NRAs can be captured by NRAGs.

**Theorem 3.** From a NRRA $A = (Q, \delta, I, F)$, an equivalent NRAG $B = (Q', k, \delta', I', F')$ 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 5 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 NRRA. In the following, we will construct a NRAG $B$ 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 $Pos_\rho(\alpha, 2i]$, guesses the data values in the positions belonging to $Pos_\rho(\alpha, 2i)$, and records the order for the positions in $Pos_\rho(\alpha, 2i]$ and $Pos_\rho(\alpha, 2i]$. We introduce some additional notations.

Let $\alpha = d_0a_1d_1...a_nd_n$ be a data path and $i: 0 \leq i \leq n$. The *profile* of the position $2i$ in $\alpha$, denoted by $prof_\alpha(2i)$, is defined as a triple $(S, \chi, \sim)$, where

- $S = \{ t \in T_A \mid t_\alpha(2i) \neq \bot \}$,
- $\chi$ is a sequence

$\ldots (b_{m_1}, T_{m_1}, b'_{m_1}, s_{m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})$

$\ldots (b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_m, T_m, b'_m, s_m)$

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 $S$, and $cur \in T_0$,
- 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 $t_\alpha(2i) = t'_\alpha(2i)$),
- for every $j : m_1 \leq j \leq m_2$, $b_j = a(t_\alpha(2i))/2$ for some $t \in T_j$, and $b_{m_1} = a(t_\alpha(2i))/2$ if $t_\alpha(2i) > 0$ for some $t \in T_{-m_1}$, otherwise, $b_{m_1} = \bot$,
- for every $j : m_1 \leq j < m_2$, $b'_j = a(t_\alpha(2i))/2+1$ for some $t \in T_j$, and $b'_{m_2} = a(t_\alpha(2i))/2+1$ if $t_\alpha(2i) < 2n$ for some $t \in T_{m_2}$, otherwise, $b'_{m_2} = \bot$,
- $s_{m_2} = \bot$, and for every $j : -m_1 \leq j \leq m_2$, 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 $d_{t_\alpha(2i)} = d_{t'_\alpha(2i)}$.

Let $\Sigma_{prof}$ denote the set of all triples $(S, \chi, \sim)$ such that $S \subseteq T_A$.

- $\chi$ is a sequence

$\ldots (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})$

$\ldots (b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_m, T_m, b'_m, s_m)$

such that

- for every $j : -m_1 \leq j \leq m_2$, $T_j \neq \emptyset$,
- $cur \in T_0$, and $T_{-m_1}, \ldots, T_{m_2}$ is a partition of $S$,
- $b_{-m_1} \in \Sigma^+ \cup \{ \bot \}$, and for every $j : -m_1 \leq j \leq m_2$,

$\ldots (b'_{m_2} \in \Sigma^+ \cup \{ \bot \}$, and for every $j : -m_1 \leq j \leq m_2$,
- $s_{m_2} = \bot$, 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 $(S, \chi, \sim) \in \Sigma_{prof}$, there may be no data paths $\alpha$ and a position in $\alpha$ such that the profile of the position in $\alpha$ is $(S, \chi, \sim)$. Nevertheless, we are able to define a consistency condition on the elements from $\Sigma_{prof}$ so that a consistent element from $\Sigma_{prof}$ indeed corresponds to the profile of a position in some data path. Moreover, for two consistent elements from $\Sigma_{prof}$, say $(S_1, \chi_1, \sim_1), (S_2, \chi_2, \sim_2)$, and $a \in \Sigma^*$, we are able to define a syntactic successor relation $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$, which mimics the changes from $prof_{\alpha}(2i)$ to $prof_{\alpha}(2(i + 1))$ by reading a word symbol $a$ in the position $2i+1$ of a data path. The details of the consistency condition and the successor relation are omitted due to the space limitation.

We are ready to construct the NRAG $B$.

There are $2|T_A| + 1$ registers in $B$, that is,

$r_1, \ldots, r_{|T_A|}, r_{|T_A| + 1}, \ldots, r_{2|T_A|}$.

Over a data path $\alpha = d_0a_1d_1...a_nd_n$, $B$ does the following.

- In each position $2i$ $(0 \leq i \leq n)$, $B$ guesses $\pi_i = (S_i, \chi_i, \sim_i) \in Prof_A$ (where $\pi_i$ is supposed to be $prof_{\alpha}(2i)$). In addition,
- if $i = 0$, then $\pi_0 = (S_0, \chi_0, \sim_0)$ is an initial profile, that is, for every $t \in T_A$ such that $pred(cur) \leq t$ or $pred_A(cur) \leq t$ for some $A \subseteq \Sigma^*$, $t \notin S_0$,
- if $i = n$, then $\pi_n = (S_n, \chi_n, \sim_n)$ is a final profile, that is, for every $t \in T_A$ such that $suc(cur) \leq t$ or $suc_A(cur) \leq t$ for some $A \subseteq \Sigma^*$, $t \notin S_n$. 7
• For every $i: 0 \leq i \leq n$, if
  \[
  \chi_i = (b_{i,m_i,1}, T_{i,m_i,1}, b'_{i,m_i,1}, s_{i,m_i,1}) \ldots
  (b_{i,1,m_i,2}, T_{i,1,m_i,2}, b'_{i,1,m_i,2}, s_{i,1,m_i,2})
  \]
  then after the position $2i$ is visited (that is, the reading head is in $2i+1$), for each $j: -m_i \leq j \leq m_i$, $B$ stores in the register $r_{j+|T_A|}$ the data value corresponding to $T_{i,j}$. In particular, $B$ stores the data value $d_i$ in $r_{i+|T_A|}$.

• Over each pair of positions $2i$ and $2(i+1)$ (where $0 \leq i < n$), $B$ checks that $\pi_i \overset{a_i}{\rightarrow} \pi_{i+1}$. 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 $(q_i, a_i, q'_i)$ over $a_i$, then $B$ checks that $b'_{i-1,0} = a_i$ and changes the state from $q$ to $q'_i$.
  - If $A$ makes a transition $(q_i, c, q'_i)$ over $d_i$, then $B$ checks that $\pi_i$ satisfies $c$, verifies that $d_i$ is equal to the data value stored in $r_{j+|T_A|}$ for each $j: -m_i \leq j \leq m_i$ such that there is $t \in T_j$ satisfying $\text{cur}_t t$ (in particular, $d_i$ should be equal to the data value in $r_{|T_A|}$), and changes the state from $q$ to $q'_i$.
  - $B$ 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 NIRA can be determined.

**Proposition 7.** For every NIRA, there is an equivalent DRA 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 NIRA.

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

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

- $\delta_d \subseteq Q \times C_{rd} \times Q \times \{+1,-1\}$.

Let $\alpha = d_0a_1d_1 \ldots a_md_m$ be a data path and $A$ be a 2NRA. 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 $\cdot \alpha - \cdot \in$ in the position $i_j$ is a word symbol $a \in \Sigma^* \cup \{-\cdot\}$, then there is $dir \in \{+1,-1\}$ such that $(q_j, a, q_{j+1}, dir) \in \delta_w$, $\theta_j = a$, and $i_{j+1} = i_j + dir$,

- for every $j: 0 \leq j < m$, if the symbol of $\cdot \alpha - \cdot \in$ in the position $i_j$ is a data value $d$, then there are $c \in C_{rd}$ and $dir \in \{+1,-1\}$ such that $(q_j, c, q_{j+1}, dir) \in \delta_d$, $(\alpha, i_j) = c$, $\theta_j = c$, and $i_{j+1} = i_j + dir$.

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

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

### 3.4 Rigid regular expressions with memory

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

\[
e := \varepsilon | a | [c] | e \cup e | e \cdot e | e^*,\text{ where } c \in C_{rd}[\Sigma^*] \]

Let $e$ be a RREM, $\alpha = d_0a_1d_1 \ldots a_md_m$ be a data path, and $i, j: 0 \leq i \leq j \leq 2n$. The semantics of $e$ is defined by a relation $(\alpha, i) \vdash_e (\alpha, j)$ as follows.

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

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

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

- 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 $\alpha = d_0a_1d_1 \ldots a_md_m$ is accepted by a RREM $e$ if $(\alpha, 0) \vdash_e (\alpha, 2n)$. Let $Z(e)$ 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 NIRA $A_e$ can be constructed in LOGSPACE such that $Z(e) = Z(A_e)$. 

8
• From a NRRA \( A \), a RREM \( e_A \) can be constructed in EXPTIME such that \( \mathcal{L}(A) = \mathcal{L}(e_A) \).

**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 (2RRDPQ) \( \xi \) 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 \( \Sigma^+ \).

Given a data graph \( G = (V, E, \eta) \) and a RDPQ \( \xi = (x, L, y) \), 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 2RRDPQs 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 \( \{ A_i \mid 1 \leq i \leq k \} \subseteq \Sigma \). Then the data-to-node \( k \)-transformation of \( G \), denoted by \( G_{dn,k} = (V_{dn,k}, E_{dn,k}, \eta_{dn,k}) \), 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 each \( v \in V \) and \( 1 \leq i \leq k \),
- for each \( v \in V \), \( \eta_{dn,k}(v) = \eta(v) \) and \( \eta_{dn,k}(n_v) = \xi(v) \).

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 2RDPQ \( \xi' = (x, L', y) \) over the alphabet \( \Sigma^* \cup \{ A_i, A_i^* \mid 1 \leq i \leq k \} \) 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 2RDPQs (C2RDPQ) can be defined similarly to 2RDPQs, 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 \( 2\text{EXPSPACE} \) and \( \text{EXPSPACE} \) 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 C2RPs in \[8\] and is the most technical part of this paper.

#### 5.1 Canonical data graph

Let \( \xi := \text{Ans}(z) \leftarrow \bigwedge_{1 \leq i \leq l} \{ y_{1i} \leftarrow 1, y_{2i} \leftarrow 2 \} \) be a C2RDPQ, \( G = (V, E, \eta) \) 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 \( \nu(y_{1i}) \) 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

\( \xi_i := \text{Ans}(z_1, \ldots, z_n) \leftarrow \bigwedge_{i \leq j \leq l_i} \{ y_{1i} \leftarrow 1, y_{2i} \leftarrow 2 \} \)

such that \( \{ y_{11}, \ldots, y_{12l_1} \} \cap \{ y_{21}, \ldots, y_{22l_2} \} \) 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 \( \mu : \{ y_{1j}, \ldots, y_{2j} \} \rightarrow V \) 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 \( j : 1 \leq j \leq l_2 \), \( \mu(y_{2j-1}) = \mu(y_{2j+1}) \) belongs to the evaluation result of \( (y_{2j-1}, L_2, y_{2j}) \) 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 C2RPs (Theorem 2 in \[8\]).
Proposition 12. Let $\xi_1, \xi_2$ be two $2$CRRDPQs. Then $\xi_1 \not= \xi_2$ 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

- $\mathcal{G}$ is $\nu$-canonical for $\xi_1$,
- and there are no $(\xi_1, \mathcal{G}, \nu)$-mappings for $\xi_2$.

5.2 Evaluating $2$RRDPQs over canonical data graphs

Let $\mathcal{G} = (V, E, \eta)$ be a $\nu$-canonical data graph for $\xi_1$ and $\xi = (X, L, y)$ be a $2$RRDPQ such that $L$ is defined by a NRRA $\mathcal{A} = (Q, \delta, I, F)$ over $\Sigma^\ast$. Then $\mathcal{G}$ consists of $l_1$-simple semipaths $\pi_1, ..., \pi_{l_1}$ such that for every $j$, $1 \leq j \leq l_1$, $\pi_j$ is a semipath from $\nu(y_{1,2j-1})$ to $\nu(y_{1,2j})$, 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 $2$RPOs over canonical graphs in [8]: the data graph $\mathcal{G}$ is first encoded into a data path $\alpha_{\mathcal{G}}$, then a $2$RRNA $\mathcal{A}_\xi$ is constructed from $\xi$ and $\mathcal{G}$ so that $\xi(\mathcal{G})$ is nonempty iff $\mathcal{A}_\xi$ accepts $\alpha_{\mathcal{G}}$.  

For every $i : 1 \leq i \leq l_1$, let $\text{ren}_i$ denote the renaming function that maps each $a \in \Sigma^\ast$ to $(a, i)$. For a data path $\alpha$, let $\text{ren}_1(\alpha)$ denote the data path obtained from $\alpha$ by replacing each $a \in \Sigma^\ast$ with $\text{ren}_1(a)$.  

Let $\mathcal{G}_{\xi_1} = \{\#\} \cup \{\nu(1,2j-1) \cup \{\#\})$. We represent $\mathcal{G}$ as a data path $\alpha_{\mathcal{G}}$ over the alphabet $\Sigma_{\xi_1}$ as follows.

$$\alpha_{\mathcal{G}} := d_1s_1\text{ren}_1(\eta(\pi_1))s_2d_2\#d_3s_3\text{ren}_2(\eta(\pi_2))s_4d_4\#...d_{2l_1-1}s_{2l_1-1}\text{ren}_{l_1}(\eta(\pi_{l_1}))s_{2l_1}d_{2l_1},$$

where $d_1, d_2, ..., d_{2l_1}$ are data values from $\mathbb{D}$ not occurring in $\mathcal{G}$ such that $d_i = d_j$ iff $\nu(y_{1,i}) = \nu(y_{1,j})$. Intuitively, for each $j : 1 \leq j \leq l_1$, the $j$-th semipath $\pi_j$ is represented by a data subpath $\alpha_{\mathcal{G}, \pi_j}$ in $\alpha_{\mathcal{G}}$, where $\alpha_{\mathcal{G}, \pi_j} = d_{2j-1}s_{2j-1}\text{ren}_j(\eta(\pi_j))s_{2j}d_{2j}$, and 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 $\alpha_{\mathcal{G}}$ corresponding to $(\pi_j, v)$, denoted by $\text{pos}_{\alpha_{\mathcal{G}}}((\pi_j, v))$. For instance, if $v = \nu(y_{1,2j}) = \nu(y_{1,2j-1})$, then $\text{pos}_{\alpha_{\mathcal{G}}}((\pi_j, v))$ is the position immediately before the symbol $s_{2j}$ and $\text{pos}_{\alpha_{\mathcal{G}}}((\pi_j, v))$ is the position immediately after $s_{2j-1}$.

The intuition of the $2$NRRA $\mathcal{A}_\xi$ is that for every semipath $\pi'$ of $\mathcal{G}$ and every run of $\mathcal{A}$ over $\eta(\pi')$, $\mathcal{A}_\xi$ goes through the trace of $\pi'$ in $\alpha_{\mathcal{G}}$ to simulate the run of $\mathcal{A}$ over $\eta(\pi')$.

Theorem 6. Let $\mathcal{G}$ be a $\nu$-canonical data graph for $\xi_1, \xi_2$ be a $2$RRDPQ. Then a $2$RRNA $\mathcal{A}_\xi$ can be constructed from $\xi$ and $\xi_1$ such that $\xi(\mathcal{G})$ is nonempty iff $\mathcal{A}_\xi$ accepts $\alpha_{\mathcal{G}}^{-1}$.

In the following, before giving a proof for Theorem 6, we first give the definition of traces of semipaths of $\mathcal{G}$ in $\alpha_{\mathcal{G}}$, then state and prove an important lemma.

Let $\pi' = v_0a_1v_1 ... v_{k-1}a_kv_k$ be a semipath in $\mathcal{G}$. The $\pi$-unraveling $(\pi = (\pi_1, ..., \pi_{l_1}))$ of $\pi'$, denoted by $\text{ur}_\pi(\pi')$, is defined as the sequence $\pi'_0 \# \pi'_1 \# \ldots \# \pi'_{k-1}$ satisfying the following conditions: There are $i_0, i_1, \ldots, i_{k+1}$ such that

- $0 = i_0 < i_1 < \ldots < i_r < i_{r+1} = \ell$,
- for every $0 \leq s \leq r$, $\pi'_s = v_i a_{i+1}v_{i+1} ... a_{i+s}v_{i+s}$,
- for every $s : 0 \leq s \leq r$, there is $j_s : 1 \leq j_s \leq l_1$ such that all the edges on $\pi'_s$ belong to $\pi_{j_s}$,
- and for every $s : 1 \leq s \leq r$, either $j_s \neq j_{s-1}$, or $j_s = j_{s-1}$ and one of the following conditions holds,
  - the last edge of $\pi'_{s-1}$ is the first edge of $\pi_{j_s}$, and the first edge of $\pi'_s$ is the last edge of $\pi_{j_s}$,
  - the last edge of $\pi'_{s-1}$ is the last edge of $\pi_{j_s}$, and the first edge of $\pi'_s$ is the first edge of $\pi_{j_s}$.

The last two conditions above correspond to the situation that the two endpoints of $\pi_{j_s}$ are in fact the same node and a semipath can jump from the first (resp. last) edge to the last (resp. first) edge of $\pi_{j_s}$.

For a semipath $\pi'$ in $\mathcal{G}$, define $\text{trc}_{\alpha_{\mathcal{G}}}((\pi'))$, the trace of $\pi'$ in $\alpha_{\mathcal{G}}$, as $\text{trc}_{\alpha_{\mathcal{G}}}((\pi'_0) \# \ldots \# \pi_{\alpha_{\mathcal{G}}}((\pi'))$, where $\pi'_0 \# \ldots \# \pi'_s$ is the $\pi$-unraveling of $\pi'$, and for every $s : 0 \leq s \leq r$,

$$\text{trc}_{\alpha_{\mathcal{G}}}(\pi'_s) := p_{\alpha_{\mathcal{G}}}((\pi_{j_s}, v_i)) a_{i+1}p_{\alpha_{\mathcal{G}}}((\pi_{j_s}, v_{i+1})) ... a_{i+s}p_{\alpha_{\mathcal{G}}}((\pi_{j_s}, v_{i+s})).$$

Note that although $\text{ur}_\pi(\pi')$ and $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$ are not data paths, they are of a similar structure, that is, nodes and position indices respectively separated by word symbols.

For brevity, later on, when $\alpha_{\mathcal{G}}$ is obvious from the context, we abbreviate $\text{trc}_{\alpha_{\mathcal{G}}}(\pi')$ as $\text{trc}(\pi')$.

It is easy to see that a run of $\mathcal{A}$ over $\eta(\pi')$ for a semipath $\pi'$ in $\mathcal{G}$ can be transformed into a run of a NRRA $\mathcal{A}'$ over $\text{ur}_\pi(\pi')$, if the interpretation of position terms over $\eta(\text{ur}_\pi(\pi'))$ is adjusted to jump over the additional $\#$ symbols as follows.

Since $\eta(\text{ur}_\pi(\pi')) = \eta(\pi'_0) \# \ldots \# \eta(\pi'_s)$, it follows that $\eta(\text{ur}_\pi(\pi')) = d_0b_1d_1 ... b_{i+s}d_{i+s}2r$, for $d_0, ..., d_{i+s}2r \in \mathbb{D}$ and $b_1, ..., b_{i+s} \in \Sigma^\ast \cup \{\#\}$. For a position term $t \in \Sigma^\ast \cup \{\#\}$
$T_p[\Sigma^+]$ and a position $2i : 0 \leq i \leq \ell + 2r$ on $\eta(\text{unw}^{\nu}(\pi'))$, define the \textit{adjusted position} represented by $t$ over $\eta(\text{unw}^{\nu}(\pi'))$ and $2i$, denoted by $t^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i)$, 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 $(t_1)^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = 1$ or $(t_1)^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = 2(\ell + 2r)$, then $(\text{suc}(t_1))^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = 1$; otherwise,

- if $(t_1)^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i)$ is not a position immediately before $\#$, then $(\text{suc}(t_1))^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = (t_1)^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) + 2$,
- otherwise, $(\text{suc}(t_1))^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = (t_1)^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) + 4$.

\textbf{Lemma 2.}\ Suppose $\pi' = v_0a_1v_1 \ldots a_nv_n$ is a semipath in $\mathcal{G}$ such that $\text{unw}^{\nu}(\pi') = \pi'_0# \ldots #\pi'_n$ and $\text{trc}(\pi') = p_0b_1p_1b_2 \ldots b_r$, where $(b_1, \ldots, b_r, p_r) \in \Sigma^* \cup \{\#\}$. Then for every $i : 0 \leq i \leq \ell + 2r$, there exists a function $\text{pos}_i \in (T_p[\Sigma^+] \cup \{\}_1)^T$ such that for every $t \in \mathcal{T}_A$, $\text{pos}_i(t) = 1$ iff $t^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = 1$; moreover, if $\text{pos}_i(t) = 1$ and $t^\text{adj}_{\text{unw}^{\nu}(\pi')}(2i) = 2$, then $(\text{pos}_i(t))_{\alpha} = \text{pos}_i$.

\textbf{Proof.}\ (Theorem 3) Let $\pi'$ be a path in $\mathcal{G}$, the $\pi'$-unraveling of $\pi$ be $\pi'_0 \ldots \pi'_n$. In addition, for every $s : 0 \leq s \leq r$, all the edges on $\pi'_s = v_s.a_{s+1}v_{s+1} \ldots v_{s+r}$ belong to $\pi_{j_s}$.

Our goal is to construct a 2NRRA $\mathcal{B}$ over $\alpha_{\pi}$ to simulate the runs of $\mathcal{A}'$ over $\eta(\text{unw}^{\nu}(\pi'))$. Similarly to the construction of NRAGs from NRRA in the proof of Theorem 3, the 2NRRA $\mathcal{B}$ goes through $\text{trc}(\pi')$ in $\alpha_{\pi}$ and guesses the profile of the current position of $\eta(\text{unw}^{\nu}(\pi'))$, in order to simulate $\mathcal{A}'$ over $\eta(\text{unw}^{\nu}(\pi'))$. The difference is that instead of storing and guessing the data values, $\mathcal{B}$ records and guesses position terms from $T_p[\Sigma^+]$ (interpreted over $\alpha_{\pi}$) for position terms occurring in the profile of the current position in $\eta(\text{unw}^{\nu}(\pi'))$. The most technical part of the construction is how to guarantee the consistency of the guessed position terms from $T_p[\Sigma^+]$ and how to update them during the simulation. Since the details of the consistency conditions and the updating of the guessed position terms is rather tedious, they are omitted due to the space limitation.

From the above description, we know that in its states, $\mathcal{B}$ should record the states of $\mathcal{A}'$, the guessed profiles, and the guessed position terms from $T_p[\Sigma^+]$. Because both the number of profiles and the number of possible guesses for the position terms from $T_p[\Sigma^+]$ are exponential over $|\mathcal{T}_A|$, it follows that the number of states of $\mathcal{B}$ is polynomial over $|Q|$ and exponential over $|\mathcal{T}_A|$. \hfill \Box

\textbf{5.3 Checking the non-containment}\n
We will construct a NRRA $\mathcal{A}' = (Q', \delta', I', F')$ to check the non-containment of $\xi_1$ over $\xi_2$ as follows.

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

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

(a) Construct a 2NRRA $\mathcal{B}_1$ to verify a $(\xi_1, \mathcal{G}, \nu)$-mapping for $\xi_2$ over $\alpha_{\pi}$ annotated with subsets of $\{y_{2,1}, \ldots, y_{2,2j}\}$. The intention is that the annotations encode an assignment of nodes in $\mathcal{G}$ to the variables from $\{y_{2,1}, \ldots, y_{2,2j}\}$. The alphabet of $\mathcal{B}_1$ is $\Sigma^\pi = \Sigma^\pi_{\xi_1} \times \{y_{2,1}, \ldots, y_{2,2j}\}$. If the word symbol immediately before a position $2i$ of the annotated $\alpha_{\pi}$ is $(a', \pi)$, then this means that each variable in $\pi$ is assigned to the node of $\mathcal{G}$ 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 2RDPQs $(y_{2,2j-1,1}, L_{2,j}, y_{2,2j})$ of $\xi_2$ over the annotated $\alpha_{\pi}$, the construction in the proof of Theorem is used. Note that since all the rigid data constraints in the RDPQs 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 $\mathcal{B}_1$, there is a $\mathcal{B} \subseteq \Sigma_{\xi_1}$ such that $\mathcal{B} = \mathcal{A} \times \{y_{2,1}, \ldots, y_{2,2j}\}$.

(b) Transform $\mathcal{B}_1$ into an equivalent NRRA $\mathcal{B}_2$ (cf. Proposition 3).

(c) Let $\text{prj} : \Sigma^\pi_{\xi_1} \rightarrow \Sigma_{\xi_1}$ such that $\text{prj}(a', \pi) = a'$. Construct $\mathcal{B}_3 = \text{prj}(\mathcal{B}_2)$. From the assumption above, we know that $\mathcal{B}_1$ and $\mathcal{B}_2$ are position-invariant under $\text{prj}$. Then from Proposition 3, we deduce that $\mathcal{Z}(\mathcal{B}_3) = \mathcal{Z}(\text{prj}(\mathcal{B}_2)) = \text{prj}(\mathcal{Z}(\mathcal{B}_2))$. So the NRRA $\mathcal{B}_3$ guesses and verifies a $(\xi_1, \mathcal{G}, \nu)$-mapping for $\xi_2$.

(d) Determine and complement $\mathcal{B}_3$ to get $\mathcal{A}'_2$ (cf. Proposition 7).

3. $\mathcal{A}'$ is the intersection of $\mathcal{A}'_1$ and $\mathcal{A}'_2$. There is a final remark for the above construction: As pointed out in [8], letter projections are only meaningful
The complexity analysis.

The size of $A_1'$ is polynomial over the size of $\xi_1$. From Theorem 5, the size of $B_1$ is exponential over the size of $\xi_2$. From Proposition 8, the size of $B_2$ is exponential over the size of $B_1$. The size of $B_3$ is the same as the size of $B_2$. Therefore, the size of $A_2'$ is triple-exponential over the size of $\xi_2$.

To check the nonemptiness of $A_1' \cap A_2'$, we can guess “on the fly” an accepting run of $A_1' \cap A_2'$ 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 2RPQs (§).

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 (NRRA), also introduced in this paper. We demonstrated the robustness of NRRA by showing that NRRA can be determinized and the two-way NRRA are expressively equivalent to NRRA. 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

[1] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: A deterministic class of timed automata. Theor. Comput. Sci., 211(1-2):253–273, 1999.
[2] R. Angles and C. Gutiérrez. Survey of graph database models. ACM Comput. Surv., 40(1):1:1–1:39, 2008.
[3] P. Barceló. Querying graph databases. In PODS, pages 175–188, 2013.
[4] P. Barceló, D. Figueira, and L. Libkin. Graph logics with rational relations and the generalized intersection problem. In LICS, pages 115–124, 2012.
[5] P. Barceló, C. Hurtado, L. Libkin, and P. Wood. Expressive languages for path queries over graph-structured data. In PODS, pages 3–14, 2010.
[6] P. Barceló, J. Pérez, and J. L. Reuter. Relative expressiveness of nested regular expressions. In AMW, pages 180–195, 2012.
[7] M. Benedikt, C. Ley, and G. Puppis. Automata vs. logics on data words. In CSL, pages 110–124, 2010.
[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In KR, pages 176–185, 2000.
[9] I. F. Cruz, A. O. Mendelzon, and P. T. Wood. A graphical query language supporting recursion. SIGMOD Rec., 16(3):323–330, 1987.
[10] S. Demri and R. Lazić. LTL with the freeze quantifier and register automata. ACM Trans. Comput. Logic, 10(3):16:1–16:30, 2009.
[11] A. Deutsch and V. Tannen. Optimization properties for classes of conjunctive regular path queries. In DBPL, pages 21–39, 2002.
[12] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In PODS, pages 139–148, 1998.
[13] M. Kaminski and N. Francez. Finite-memory automata. Theor. Comput. Sci., 134(2):329–363, 1994.
[14] M. Kaminski and D. Zeitlin. Extending finite-memory automata with non-deterministic reassignment (extended abstract). In AFL, pages 195–207, 2008.
[15] E. V. Kostylev, J. L. Reutter, and D. Vrgoč. Regular path queries on graphs with data. In ICDT, 2014. To appear.
[16] L. Libkin, J. Reutter, and D. Vrgoč. Trial for rdf: Adapting graph query languages for rdf data. In PODS, pages 201–212, 2013.
[17] L. Libkin and D. Vrgoč. Regular expressions for data words. In LPAR, pages 274–288, 2012.
[18] L. Libkin and D. Vrgoč. Regular path queries on graphs with data. In ICDT, pages 74–85, 2012.
[19] J. Pérez, M. Arenas, and C. Gutiérrez. nSPARQL: A navigational language for rdf. Web Semant., 8(4):255–270, 2010.
[20] J. C. Shepherdson. The reduction of two-way automata to one-way automata. IBM J. Res. Dev., 3(2):198–200, 1959.
[21] P. T. Wood. Query languages for graph databases. SIGMOD Rec., 41(1):50–60, 2012.
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 \( t_1 \not\sim t_2 \) occurring in \( c \), \( t_1, t_2 \in T_c \),
- for every \( t \in T_c \) and \( t' \in C_{\text{prj}}[\Sigma^*] \) such that \( t' \leq t \), we have \( t', [t'\,\text{cur}] \in T_c \).

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

\[
\chi = (b_{-1}, T_{-1}, b'_{-1}, s_{-1}) \ldots (b_1, T_1, b'_1, s_1)(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_m, T_m, b'_m, s_m).
\]

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 \( \text{NP} \). \( \square \)

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. \( \square \)

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 \( \text{prj}(L) \). \( \square \)

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 \( i : 0 \leq i \leq n \), \( t_a[2i] = (\text{prj}(t))_{\text{prj}(\alpha)}[2i] \).

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

Suppose \( \text{suc}_A(\text{cur})_{\alpha}(2i) = 2j \) 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
Lemma 1. The nonemptiness problem of NRAGs is PSPACE-complete.

Therefore, \( \mathcal{L}(\text{prj}(A)) \in \text{L}(\mathcal{L}(A)) \):

Suppose \( \beta = d_0q_1d_1 \ldots q_n \in \mathcal{L}(\text{prj}(A)) \). Then there is an accepting run of \( \text{prj}(A) \) over \( \beta \), say \( \rho = q_0q_1q_2 \ldots q_{2n-1}q_n \). Therefore, \( \text{prj}(A) \) is nonempty iff \( \mathcal{L}(\text{prj}(A)) \neq \emptyset \).

From the definition of \( \text{prj}(A) \), we know that

- for every \( i : 1 \leq i \leq n \), there is \( a_i \in \Sigma^* \) such that \( \text{prj}(a_i) = \gamma_i \) and \( \gamma_i \in \delta, v \).
- for every \( i : 0 \leq i \leq n \), there is \( c_i \in A \) such that \( \gamma_i c_i \in \delta_d \) and \( c_i = \text{prj}(c_i) \).

Let \( \alpha = d_0q_1d_1 \ldots a_n \in \mathcal{L}(A) \). Then \( \beta = \text{prj}(\alpha) \).

From the claim, we know that for every \( i : 0 \leq i \leq n \), \( (\alpha, 2i) \vdash c_i \) iff \( (\text{prj}(\alpha), 2i) \vdash \text{prj}(c_i) \).

Therefore, \( \text{prj}(\alpha) \vdash \text{prj}(\alpha) \) is an accepting run of \( \mathcal{L}(A) \) and \( \beta = \text{prj}(\alpha) \in \mathcal{L}(\text{prj}(A)) \).

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 \( \alpha = d_0q_1d_1 \ldots a_n \) be a data path. Then for every run \( \rho \) of \( A \) over \( \alpha \) and every \( i : 0 \leq i \leq n \), \( \text{Pos}_\rho[\alpha, 2i] \cup \text{Pos}_\rho[\alpha, 2i] \in \{ t_\alpha[2i] | t \in \mathcal{T}_A \} \).

Proof. Let \( A = (Q, \delta, I, F) \) be a NRRA, \( \alpha = d_0q_1d_1 \ldots a_n \) be a data path, \( \rho = q_0q_1q_2 \ldots q_{2n-1}q_n \) be a run of \( A \) over \( \alpha \), and \( i : 0 \leq i \leq n \).

Let \( t_\alpha[2j] \in \text{Pos}_\rho[\alpha, 2i] \) such that \( j \leq i, t \) occurs in \( c_j, t_\alpha[2j] \neq \perp \) and \( t_\alpha[2j] > 2i \). Then there is \( t' \in \mathcal{T}_B[\Sigma^*] \) such that \( t' \leq t, t_\alpha'[2j] \leq 2i \), and for every \( t'' \), \( t' < t'' \leq t, t_\alpha''[2j] > 2i \). It follows that \( \text{suc}(t') \leq t \) or \( \text{suc}(t') \leq t \) for some \( A \in \Sigma^* \).

If \( \text{suc}(t') \leq t \), then \( t_\alpha'[2j] = 2i \), since \( \text{suc}(t')_{\alpha[2j]} = t_\alpha'[2j] + 1 > 2i \) and \( t_\alpha'[2j] < 2i \). Thus, \( t_\alpha[2j] = (\text{t}(\text{t}(\text{cur}))_{\alpha[2j]} = (\text{t}(\text{t}(\text{cur}))_{\alpha[2j]} \).

If \( \text{suc}(t') \leq t \), then \( t_\alpha'[2j] < 2i \) and \( \text{suc}(A(t'))_{\alpha[2j]} > 2i \). It follows that \( \text{suc}(A(t'))_{\alpha[2j]} > 2i \). From this, it is deduced that \( \text{suc}(A(t'))_{\alpha[2j]} = \text{suc}(A(t'))_{\alpha[2j]} = (\text{t}(\text{t}(\text{cur}))_{\alpha[2j]} \).

From the above argument, it follows that \( \text{Pos}_\rho[\alpha, 2i] \in \{ t_\alpha[2i] | t \in \mathcal{T}_A \} \). Similarly, we can show that \( \text{Pos}_\rho[\alpha, 2i] \in \{ t_\alpha[2i] | t \in \mathcal{T}_A \} \). □

Theorem 3. From a NRRA \( A = (Q, \delta, I, F) \), an equivalent NRAG \( B = (Q', k, \delta', I', F') \) can be constructed such that \( |Q'| \) is polynomial over \( |Q| \) and exponential over \( |T_A| \) and \( k \) is polynomial over \( |T_A| \).
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 $\rho$ 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 $i : 0 \leq i \leq n$, $Pos^p_\rho[\alpha, 2i] \cup Pos^p_\rho[\alpha, 2i]$ 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 $\rho$ as follows: In the position $2i$,

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

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 $prof_\alpha(2i)$, is defined as a triple $(S, \chi, \sim)$, where

- $S = \{ t \in T_A \mid t_0[2i] \neq 1 \}$,
- $\chi$ is a sequence
  \[
  (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})
  \]

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 $t \in T_{j_1}$ and $t' \in T_{j_2}$, then $j_1 \leq j_2$ iff $t_0[2i] \leq t'_0[2i]$ (in particular, $j_1 = j_2$ iff $t_0[2i] = t'_0[2i]$),
- $cur \in T_0$,
- for every $j : -m_1 \leq j \leq m_2$, $b_j = a(t_0[2i])/2$ for some $t \in T_{j_1}$ 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 $j : -m_1 \leq j < m_2$, $b'_j = a(t_0[2i])/2+1$ for some $t \in T_{j_1}$, 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 $j : -m_1 \leq j < m_2$, if $t'_0(2i) = t_0(2i) + 1$ for some $t \in T_{j_1}$ and $t' \in T_{j_1+1}$, 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 $d_{t_0[2i]} = d_{t'_0[2i]}$.

Let $\Sigma_{prof}$ denote the set of all triples $(S, \chi, \sim)$ 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 $j : -m_1 \leq j \leq m_2$, $T_j \neq \emptyset$,
- $cur \in T_0$,
- $T_{-m_1}, \ldots, T_{m_2}$ 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 $(S, \chi, \sim) \in \Sigma_{prof}$, there may be no data paths $\alpha$ and a position in $\alpha$ such that the profile of the position in $\alpha$ is $(S, \chi, \sim)$. Nevertheless, we are able to define a consistency condition on the elements from $\Sigma_{prof}$ so that a consistent element from $\Sigma_{prof}$ indeed corresponds to the profile of a position in some data path. Moreover, for two consistent elements from $\Sigma_{prof}$, say $(S_1, \chi_1, \sim_1), (S_2, \chi_2, \sim_2)$, and $a \in \Sigma^*$, we are able to define a syntactic successor relation $(S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2)$, which mimics the changes from $prof_\alpha(2i)$ to $prof_\alpha(2(i+1))$ by reading a word symbol $a$ in the position $2i + 1$ of a data path.

For $A \subseteq \Sigma^*$, a sequence
\[
\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_{-1}, T_{-1}, b'_{-1}, s_{-1})(b_0, T_0, b'_0, s_0)(b_1, T_1, b'_1, s_1) \ldots (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})
\]
and $j: -m_1 \leq j \leq m_2$. $A$ is said to occur after (resp. before) $T_j$ in $\chi$ if $b_j \in A$ for some $j': j < j' \leq m_2$ or $b'_j \in A$ for some $j': j < j' \leq m_2$. $b_j \in A$ for some $j': m_1 \leq j \leq j'$ or $b_j \in A$ for some $j': -m_1 \leq j' \leq j$.

Let $(S, \chi, \sim) \in \Sigma_{pro}$ and

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

Then $(S, \chi, \sim)$ is said to be consistent if $\chi$ satisfies the following conditions.

- For every $suc(t) \in \mathcal{T}_A$ (resp. $pred(t) \in \mathcal{T}_A$), and every $j: -m_1 \leq j \leq m_2$ (resp. $j: -m_1 < j \leq m_2$), $t \in T_j$ iff $suc(t) \in T_{j+1}$ (resp. $t \in T_j$ iff $pred(t) \in T_{j-1}$).
- For every $suc(t) \in \mathcal{T}_A$ and every $j: -m_1 \leq j < m_2$, if $t \in T_j$ and $suc(t) \in T_{j+1}$, then $b'_j = b_{j+1}$ and $s_j = 1$.
- For every $pred(t) \in \mathcal{T}_A$ and every $j: -m_1 < j \leq m_2$, if $t \in T_j$ and $pred(t) \in T_{j-1}$, then $b'_j = b_j$ and $s_{j-1} = 1$.
- For every $suc(a(t) \in \mathcal{T}_A$, if $suc(a(t) \in T_j$ for some $j: -m_1 < j \leq m_2$, then $b_j \in A$.
- For every $pred(a(t) \in \mathcal{T}_A$, if $pred(a(t) \in T_j$ for some $j: -m_1 \leq j < m_2$, then $b'_j \in A$.
- For every $suc(a(t) \in \mathcal{T}_A$ (resp. $pred(a(t) \in \mathcal{T}_A$), if $suc(a(t) \in T_j$ (resp. $pred(a(t) \in T_j$), then $t \in T_j$ for some $j' < j$ (resp. $j' > j$).
- For every $suc(a(t) \in \mathcal{T}_A$, if $t \in T_{j_1}$ and $suc(a(t) \in T_{j_2}$ $(j_1 < j_2$, then for every $j_3: j_1 < j_3 < j_2$, $b_{j_3} \notin A$, and for every $j_3: j_3 \leq j_3 < j_2 - 1$, $b_{j_3'} \notin A$; in addition, $b'_{j_2-1} \notin A$ implies $s_{j_2-1} = 1$.
- For every $pred(a(t) \in \mathcal{T}_A$, if $pred(a(t) \in T_{j_1}$ and $t \in T_{j_2}$ $(j_1 < j_2$, then for every $j_3: j_1 < j_3 < j_2$, $b_{j_3} \notin A$, and for every $j_3: j_3 + 1 \leq j_3 < j_2$, $b_{j_3} \notin A$; in addition, $b_{j_2+1} \notin A$ implies $s_{j_1} = 1$.
- For every $t, suc(a(t) \in \mathcal{T}_A$, if $t \in T_j$ for $j: -m_1 \leq j \leq m_2$, and $A$ occurs after $T_j$, then $suc(a(t) \in T_{j'}$ for some $j' > j$.
- For every $t, pred(a(t) \in \mathcal{T}_A$, if $t \in T_j$ for $j: -m_1 < j \leq m_2$, and $A$ occurs before $T_j$, then $pred(a(t) \in T_{j'}$ for some $j' < j$.

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

**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 $t \sim t'$, 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 $b_{-m_1} \notin A$, then let $\alpha_{-m_1} = dB_{-m_1}d_{-m_1}$,
  - if $s_{-m_1} = 1$ and $b_{-m_1} = 1$, then let $\alpha_{-m_1} = dB_{-m_1}$,
  - if $s_{-m_1} = 0$ and $b_{-m_1} \notin A$, then let $\alpha_{-m_1} = dB_{-m_1}d_{-m_1}b'_{-m_1}$,
  - if $s_{-m_1} = 0$ and $b_{-m_1} = 1$, then let $\alpha_{-m_1} = dB_{-m_1}b'_{-m_1}$.

- For $j: -m_1 < j < m_2$,
  - if $s_{j-1} = 1$ and $s_j = 1$, then let $\alpha_j = d_{j-1}bd_jd_j$,
  - if $s_{j-1} = 1$ and $s_j = 0$, then let $\alpha_j = d_{j-1}bdjd'_jd_j$,
  - if $s_{j-1} = 0$ and $s_j = 1$, then let $\alpha_j = dbd_jd_j$,
  - if $s_{j-1} = 0$ and $s_j = 0$, then let $\alpha_j = dbd_jd_jd_j$.

- For $j = m_2$,
  - if $s_{m_2-1} = 1$ and $b'_{m_2} \notin A$, then let $\alpha_{m_2} = dB_{m_2}d_{m_2}d_{m_2}b'_{m_2}$,
  - if $s_{m_2-1} = 1$ and $b'_{m_2} = 1$, then let $\alpha_{m_2} = dB_{m_2}d_{m_2}b'_{m_2}$,
  - if $s_{m_2-1} = 0$ and $b'_{m_2} \notin A$, then let $\alpha_{m_2} = dB_{m_2}d_{m_2}b'_{m_2}$,
  - if $s_{m_2-1} = 0$ and $b'_{m_2} = 1$, then let $\alpha_{m_2} = dB_{m_2}d_{m_2}$. 

16
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 \( t \in T_A \) and every \( j : -m_1 \leq j \leq m_2, \ t \in T_j \iff t[A][2i_0] = 2i_j \).

Let us take \( t = suc_A(cur) \) as an example to illustrate the proof.

Suppose \( suc_A(cur) \in T_j \), then \( b_j \in A \) and for every \( j' \) s.t. \( 0 \leq j' < j, \ b_{j'} \notin A \) 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 \( suc_A(cur)[2i_0] = 2i_j \), 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 \( suc_A(cur)[2i_0] = 2i_j \), 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, \( b_{j-1}' \in A \) implies \( s_{j-1} = 1 \) and \( b_{j-1}' = b_j \). From this, we conclude that \( suc_A(cur) \in T_j \).

From the result (\( * \)), we conclude that \( prof_A[2i_0] = (S,\chi,\sim) \). \( \square \)

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

\[
\chi = (b_{-m_1}, T_{-m_1}, b'_{-m_1}, s_{-m_1}) \ldots (b_1, T_1, b'_1, s_1) (b_{m_2}, T_{m_2}, b'_{m_2}, s_{m_2})
\]

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 \( a \in \Sigma^+, (S,\chi_1,\sim_1), (S,\chi_2,\sim_2) \in Prof_A \), for \( i = 1, 2 \),

\[
\chi_i = (b_{i,-m_i}, T_{i,-m_i}, b'_{i,-m_i}, s_{i,-m_i}) \ldots (b_{i,-1}, T_{i,-1}, b'_{i,-1}, s_{i,-1}) (b_0, T_{0,0}, b'_0, s_0) (b_{i,1}, T_{i,1}, b'_{i,1}, s_1) \ldots (b_{i,m_i}, T_{i,m_i}, b'_{i,m_i}, s_{i,m_i})
\]

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

For every \( j : -m_{1,1} \leq j \leq m_{1,2} \), construct \( T'_{1,j} \subseteq T_A \) from \( T_{1,j} \) as follows.

- For every \( t \in T_{1,j} \) such that \( suc(cur) \leq t \), let \( t[suc(cur) \setminus \{cur\}] \in T'_{1,j} \).
- For every \( t \in T_{1,j} \) and \( A \subseteq \Sigma^+ \) such that \( a \in A \) and \( suc_A(cur) \leq t \), let \( t[suc_A(cur) \setminus \{cur\}] \in T'_{1,j} \).
- For every \( t \in T_{1,j} \) such that \( t[cur\setminus pred_A(cur)] \in T_A \), let \( t[cur\setminus pred_A(cur)] \in T'_{1,j} \).
- For every \( t \in T_{1,j} \) and \( A \subseteq \Sigma^+ \) such that \( a \in A \) and \( t[cur\setminus pred_A(cur)] \in T_A \), let \( t[cur\setminus pred_A(cur)] \in T'_{1,j} \).
- For every \( t \in T_{1,j} \) such that \( a \notin A \), \( suc_A(cur) \leq t \) or \( pred_A(cur) \leq t \), let \( t \in T'_{1,j} \).
- If \( s_{1,1} = 0 \), let \( cur \in T'_{1,1} \).

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

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

- \( b'_{1,0} = a \).
- Let \( j_1j_2 \ldots j_r \) : \( -m_{1,1} \leq j_1 < j_2 < \ldots < j_r \leq 0 \) 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

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

is equal to the sequence \((b_{1,j_1}, T_{1,j_1}, b'_{1,j_1}, s_{1,j_1}) \ldots (b_{1,j_r}, T_{1,j_r}, b'_{1,j_r}, s_{1,j_r})\).

- There is a partial mapping \( f \) from \( \{1, \ldots, m_{1,2}\} \) to \( \{0, \ldots, m_{2,2}\} \) such that
  - for every \( j : 1 \leq j \leq m_{1,2} \), \( f(j) \) is undefined iff \( T_{1,j} = \emptyset \),
  - \( f \) is increasing, that is, if \( j_1 < j_2 \) and \( f(j_1), f(j_2) \) are defined, then \( f(j_1) < f(j_2) \),
  - for every \( j : 1 \leq j \leq m_{1,2} \), if \( f(j) \) is defined, then \( T_{1,j} \subseteq T_{2,f(j)} \), \( b_{1,j} = b_{2,f(j)} \) and \( b'_{1,j} = b'_{2,f(j)} \),
  - if \( f(1) \) is defined, then \( f(1) = 0 \) iff \( s_{1,0} = 1 \),
  - for every \( j : 1 \leq j < m_{1,2} \), if \( f(j), f(j + 1) \) are both defined, then \( s_{1,j} = 1 \) implies \( f(j + 1) = f(j) + 1 \) and \( s_{2,f(j)} = 1 \).
– for every \( j_1, j_2 : j_1 < j_2 \), if \( f(j_1), f(j_2) \) are both defined, then for every \( t \in T_{2, f(j_1)}, t' \in T_{2, f(j_2)} \), \( t \sim t' \) iff there are \( t_1 \in T_{1, j_1}, t'_1 \in T_{1, j_2} \) such that \( t_1 \sim t'_1 \).

Intuitively, \((S_1, \chi_1, \sim_1)\) is rotated one-position to the left to get the profile \((S_2, \chi_2, \sim_2)\). The \( T_{1, j}'s\) together with \( f \) above define the information that should be inherited during the rotation.

We are ready to construct the NRAG \( B \).

There are \( 2|T_A| + 1 \) registers in \( B \), that is,

\[
r_1, \ldots, r_{|T_A|}, r_{|T_A|+1}, \ldots, r_{2|T_A|}.
\]

Over a data path \( \alpha = d_0a_1d_1 \ldots a_na_n \), \( B \) does the following.

- In each position \( 2i \) (\( 0 \leq i \leq n \)), \( B \) guesses \( \pi_i = (S_i, \chi_i, \sim_i) \in \text{Prof}_A \) (where \( \pi_i \) is supposed to be \( \text{prof}_A[2i] \)). In addition,
  - if \( i = 0 \), then \( \pi_0 = (S_0, \chi_0, \sim_0) \) is an initial profile, that is, for every \( t \in T_A \) such that \( \text{pred}(\text{cur}) \leq t \) or \( \text{pred}(\text{cur}) \leq t \) for some \( A \subseteq \Sigma^* \), \( t \in S_0 \),
  - if \( i = n \), then \( \pi_n = (S_n, \chi_n, \sim_n) \) is a final profile, that is, for every \( t \in T_A \) such that \( \text{suc}(\text{cur}) \leq t \) or \( \text{suc}(\text{cur}) \leq t \) for some \( A \subseteq \Sigma^* \), \( t \notin S_n \).
- For every \( i : 0 \leq i \leq n \), if
  \[
  \chi_i = (b_{i,-m_i,1}, T_{i,-m_i,1}, b'_{i,-m_i,1}, s_{i,-m_i,1}) \ldots
  \]

then after the position \( 2i \) is visited (that is, the reading head is in \( 2i + 1 \)), for each \( j : -m_i,1 \leq j \leq m_i,2 \), \( B \) stores in the register \( r_{j+i|T_A|} \) the data value corresponding to \( T_{1,j} \). In particular, \( B \) stores the data value \( d_i \) in \( r_{|T_A|} \).
- Over each pair of positions \( 2i \) and \( 2(i + 1) \) (\( 0 \leq i < n \)), \( B \) checks that \( \pi_i \overset{a_{i+1}}{\rightarrow} \pi_{i+1} \). 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 \( r_{j+i|T_A|} \) for each \( j : -m_i,1 \leq j \leq m_i,2 \) such that there is \( t \in T_j \) satisfying \( \text{cur} \sim_t t \) (in particular, \( d \) 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 \( |T_A| \). \( \square \)

**Proposition 6.** The nonemptiness of NRRAAs and DRRAs is \( \text{PSPACE-complete} \).

**Proof.** The upper bound:
From Theorem 3, given a NRRA \( A = (Q, \delta, I, F) \), an equivalent NRAG \( B = (Q', k, \delta', I', F') \) can be constructed such that \( \mathcal{L}(A) \) is nonempty iff \( \mathcal{L}(B) \) is nonempty. Moreover, \( B \) satisfies that \( |Q'| \) is polynomial over \( |Q| \) and exponential over \( |T_A| \), and \( k \) is polynomial over \( |T_A| \).

From the proof of Proposition 5, we know that a NFA \( B' \) can be constructed from \( B \) such that \( \mathcal{L}(B') \) is nonempty iff \( \mathcal{L}(B) \) is nonempty. Since the number of states of \( B' \) is polynomial over \( |Q'| \) and exponential over \( k \), it follows that the number of states of \( B' \) is polynomial over \( |Q| \) and exponential over \( |T_A| \). To decide the nonemptiness of \( A \), an accepting run of \( B' \) can be guessed nondeterministically in polynomial space. The \( \text{PSPACE} \) upper bound then follows from Savitch’s theorem.

The lower bound:
The reduction from the membership problem of polynomial space Turing machines to the nonemptiness problem of NRAs or DRAs (10) can be adapted to a reduction to the nonemptiness problem of NRRAAs or DRRAs. \( \square \)

**Proposition 7.** For every NRRA \( A \), there is an equivalent DRRRA of exponential size.

**Proof.** Let \( A = (Q, \delta, I, F) \) be a NRRA. We construct a DRRRA \( A' = (Q', \delta', q'_0, F') \) as follows:
- \( Q' = Q'' \cup Q'_d \), where \( Q'' = 2^Q \), \( Q'_d = 2^{Q_0} \).
- \( q'_0 = I, F' = \{S \in Q'_d \mid S \cap F \neq \emptyset \} \).
\[ \delta' = \delta'_w \cup \delta'_d \text{ is defined as follows:} \]
\[
\delta'_w = \{(S, a, S') \mid S \in Q'_w, S' \in Q'_d \},
\]
\[
\delta'_d \text{ is defined as follows:}
\]

For every \( S \in Q'_d \), let \( C \) denote the set of rigid data constraints occurring in the tuples \((q, c, q') \in \delta_d \) such that \( q \in S \). Then \( \delta'_d \) contains all tuples \((S, c', S') \) such that there exists \( C' \subseteq C \) satisfying that \( c' = \bigwedge_{c \in C'} c \land \bigwedge_{c \in C \setminus C'} \bar{c} \), and \( S' = \{(q' \mid \exists q \in S, c \in C'(q, c, q') \in \delta_d) \}. \]

Note that the transitions \((S, c', S') \) may be non-applicable if \( c' \) is unsatisfiable.

If \((S, c'_1, S'_1), (S, c'_2, S'_2) \) \( \in \delta'_d \) such that \( S'_1 \neq S'_2 \), then there are \( C'_1, C'_2 \) such that \( C'_1 \neq C'_2, c'_1 = \bigwedge_{c \in C'_1} c \land \bigwedge_{c \in C \setminus C'_1} \bar{c} \) and \( c'_2 = \bigwedge_{c \in C'_2} c \land \bigwedge_{c \in C \setminus C'_2} \bar{c} \). It is easy to observe that if \( C'_1 \neq C'_2 \), then \( c_1 \land c_2 \) is unsatisfiable. Therefore, \( A' \) is a DRRA.

**Corollary** 2. 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.

- Determinize and complement \( \mathcal{B} \), let \( \mathcal{C} \) be the resulting DRRA.
- Construct the product of \( \mathcal{A} \) and \( \mathcal{B} \), 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 \( \mathcal{D}_A \) and \( \mathcal{D}_B \).

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 \( |D_A|, |D_B| \).

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 2NRRA, there is an equivalent NRRA 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 \( \mathcal{A} = (Q, \tau, \sigma, I, F) \) be a 2NRRA. We construct a NRRA \( \mathcal{A}' = (Q', \delta', I', F') \) as follows.

- \( Q' = Q_d \cup Q_w \), where
  - \( Q'_d \) is the set of all tuples \((C, f) \in 2^{C \times Q \cup \{\bot\}}^{Q_d \cup \{\bot\}} \) such that \( \bigwedge_{c \in C} c \) is satisfiable, \( f(\cdot) \in Q_d \), and for every \( q \in Q_d \), \( f(q) \in Q_w \cup \{\bot\} \),
  - \( Q'_w \) is the set of all tuples \((a, f) \in (\{\pm\} \times \{\tau \})^{Q_w \cup \{\bot\}} \) such that \( f(\cdot) \in Q_w \), and for every \( q \in Q_w \), \( f(q) \in Q_d \cup \{\bot\} \).
- \( I' \) 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 \((q_i, c_i, p_i, +1, -1) \in \delta_d \), and for every \( i : 1 \leq i \leq k \), \((p_i, +1, q_i, +1) \in \delta_w \),
    - there is \( c \in C \) such that \((q_k, c, q', +1) \in \delta_d \).
- \( \delta' \) are defined as follows.
  - \( (a, f) \in Q'_w, (C, f') \in Q'_d \). 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,
      - for every \( 0 \leq i < k \), there exists \( c \in C \) such that \((q_i, c, p_i, +1) \in \delta_d \),
      - for every \( 1 \leq i \leq k \), \( f(p_i) = q_i \),

19
3. there is $c \in C$ such that $(q_k, c, q', +1) \in \delta$.

Let $(C, f) \in Q_w^d \cap (a, f') \in Q_w'$. Then $((C, f), \wedge c, (a, f')) \in \delta_k'$ iff the following conditions hold.

For every pair $(q, q')$ such that $q, q' \in Q$ and $f'(q) = q'$, there exist $q_0, q_1, \ldots, q_k \in Q_w$ and $p_1, \ldots, p_k \in Q_d$ satisfying that $q_0 = q$, and

1. for every $0 \leq i < k$, $(q_i, a, p_{i+1}, -1) \in \delta_w$, and for every $1 \leq i \leq k$, $f(p_i) = q_i$,
2. $(q_k, a, q', +1) \in \delta_w$.

- $F'$ consists of all $(\cdot, f) \in Q'_w$ 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 $((C, f'), \wedge c, f) \in \delta_k'$, 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 $i : 0 \leq i \leq k$, $f(p_i) = q_i$, and $q_k \in F$.

Now we prove the correctness of the construction, that is, for every data path $\alpha = d_0 a_1 d_1 \ldots a_n d_n$, $A$ accepts $\alpha$ iff $A'$ accepts $\alpha$.

"Only if direction":

Suppose $A$ accepts $\alpha$. Then there is an accepting run of $A$ over $\alpha$, say $(q_0, i_0)\theta_0(q_1, i_1)\theta_1 \ldots \theta_{m-1}(q_m, i_m)$, such that

- $q_0 \in I$, $q_m \in F$,
- $i_0 = 0$, $i_m = 2n + 2$,
- for every $j : 0 \leq j < m$, if $i_j$ is even, then there is $d r e i \in \{+1, -1\}$ such that $(q_j, a_i, j, q_{j+1}, d r e i) \in \delta_w$ (where $a_0 = -, a_{n+1} = +$), $q_{j+1} = a_{j+2}$ and $i_{j+1} = i_j + d r e i$,
- for every $j : 0 \leq j \leq m$, if $i_j$ is odd, then there are $c \in C_{rgd}$ and $d r e i \in \{+1, -1\}$ such that $(q_j, c, q_{j+1}, d r e i) \in \delta_d$, $(\alpha, i, j - 1) = c$, $q_j = c$, and $i_{j+1} = i_j + d r e i$.

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 $q_{i_{j_1}} \neq q_{i_{j_2}}$.\[ (*)\]

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

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

Then

$$(C_0, f_1)(\bigwedge_{c \in C_0} c)(a_1, f_2) a_1(C_1, f_3)(a_2, f_2) a_2(C_2, f_3) \ldots (a_n, f_{2n}) a_n(C_n, f_{2n+1})(\bigwedge_{c \in C_n} c)(\cdot, f_{2n+2})$$

is an accepting run of $A'$ over $\alpha$.

"If direction":

Suppose $(C_0, f_0)c_0(a_1, f_1) a_1(C_1, f_2) \ldots (a_n, f_{2n-1}) a_n(C_n, f_{2n}) c_n(\cdot, f_{2n+1})$ is an accepting run of $A'$ over $\alpha$. Since $f_{2n+1} \in F'$, 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 $i : 1 \leq i \leq k$, $f_{2n}(p_i) = q_i$.\[ 20\]
• $q_k \in F$.

From the fact that $f_{2n+1}(\cdot) = q_0 \neq \bot$, we deduce from the definition of $\delta'$ in $A'$ that $f_i(\cdot) \neq \bot$ 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 $f_0(\cdot) \neq \bot$, there is $p_0 \in I$ such that $(p_0, \tau, f_0(\cdot), +1) \in \delta_w$.

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

Consider the composition of the following subruns,

$$(p_0, 0) \vdash (f_0(\cdot), 1), (f_0(\cdot), f_0(\cdot), f_0(\cdot)), \ldots, (f_0(\cdot), f_0(\cdot), f_0(\cdot), \ldots, (q_0, 2n + 2) \vdash (p_1, 2n + 1), \rho_{p_1, q_1}^{2n}, (q_1, 2n + 2) \vdash (p_2, 2n + 1), \rho_{p_2, q_2}^{2n}, \ldots, (q_{k-1}, 2n + 2) \vdash (p_k, 2n + 1), \rho_{p_k, q_k}^{2n}.\)$$

Let $\rho$ denote this composition. Then $\rho$ is an accepting run of $A$ over $\alpha$. □

D. PROOFS IN SECTION 4

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

Proof. The upper bound:

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

Let $\mathcal{G} = (V, E, \eta)$ be a data graph, $\xi = (x, L, y)$ be a 2RRDPQ, and $(v_1, v_2) \in V \times V$. Suppose $L$ is given by a NRRA $A = (Q, \delta, I, F)$ 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 $(v_1)_s$ and final state $(v_2)_s$ over the alphabet $\Sigma^* \cup D$ as follows.

• $Q' = \{ v_s \mid v \in V \}$,
• $\delta' = \{ (v_s, a, v'_s), (v'_s, a', v_s) \mid (v, a, v') \in E \} \cup \{ (v_s, d, v_t) \mid v \in V, \eta(v) = d \}$,
• $I' = \{ v_1 \}$, $F' = \{ v_2 \}$.

From the proof of Theorem 3, we know that from $A$, an equivalent NRAG $B = (Q', \delta', I', F')$ 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 NRAG $B$ 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 $\mathcal{L}(A_{\mathcal{G}, (v_1, v_2)} \cap B') \neq \emptyset$. 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 NRRA is in 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 NRRA is NLOGSPACE.

The PSPACE lower bound is obtained by an easy reduction from the emptiness of NRRA. The NLOGSPACE lower bound of data complexity is from that of RPQs. □

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 $\Sigma^* \cup \{ A_i, A_i^c \mid 1 \leq i \leq k \}$ 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 NRRA $A' = (Q', \delta', I', F')$ over $\Sigma^* \cup \{ A_i, A_i^c \mid 1 \leq i \leq k \}$ 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 $d$ is stored into the $i$-th register in $A$, the sequence $da_1da_2 \ldots d$ is read by $A'$. Later on, we can refer to the data values stored in the $i$-th register by using the position terms $\text{pred}_{A_i}^d$.

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

• $Q' = Q'_w \cup Q'_d$ such that
  - $Q'_w$ is the union of $Q_w$ and $\delta_d \times \{ A_i, A_i^c \mid 1 \leq i \leq k \}$,
  - $Q'_d$ is the union of $Q_d$ and $\delta_d \times \{ \#_i, \#_i^c \mid 1 \leq i \leq k \}$.
E. PROOFS IN SECTION 5

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

Proof. The PSPACE lower bound follows from that of 2RRDPQs. The NLOGSPACE lower bound follows from that of RFQs.

The upper bound:
Let $\xi := \text{Ans}(\bar{z}) \leftarrow \land_{1 \leq i \leq l} (y_{2i-1}, L_i, y_{2i})$ be a C2RRDPQ, $\mathcal{G} = (V, E, \eta)$ a data graph, and $\pi$ is a tuple of nodes of the same arity as $\mathcal{G}$. 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 $\Sigma^*$. Then for every $i : 1 \leq i \leq l$, $L_i$ is given by a 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 each $A_i$, an equivalent NRAG $B_i = (Q'_i, k_i, I'_i, F'_i)$ 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 $Q'_i \times D^{[l]}$. 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 $\bar{v} \in \xi(\mathcal{G})$, 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 $(\nu(2y_{2i-1}), \nu(y_{2i}))$, the data graph $\mathcal{G}$ together with $(\nu(2y_{2i-1}), \nu(y_{2i}))$ can be seen as an NFA $A_{\mathcal{G}, i}$ over the alphabet $\Sigma^* \cup D$ with the initial state $(\nu(y_{2i-1})), M$ 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 $M \in \xi(\mathcal{G})$, 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 C2RRDPQs 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 C2RRDPQs has the NLOGSPACE data complexity. □

Lemma 2. Suppose $\pi' = v_0 a_1 v_1 \ldots a_t v_t$ is a semipath in $\mathcal{G}$ such that $ur_{\pi}(\pi') = \pi_0 \# \ldots \# \pi_r$ and $trc(\pi') = p_0 b_1 p_1 \ldots b_{t+2r} p_{t+2r}$ (where $b_1, \ldots, b_{t+2r} \in \Sigma^* \cup \{\#\}$). Then for every $i : 0 \leq i \leq t + 2r$, there exists a function $pos_i : (\mathcal{T}_p[\Sigma_i] \cup \{1\})^T_A$ such that for every $t \in \mathcal{T}_A$, $pos_i(t) = 1$ iff $t_{ur\pi}(\pi')(2i) = 1$; moreover, if $pos_i(t) \neq 1$ and $t_{ur\pi}(\pi')(2i) = 1$, then $pos(t)_{\pi_i}[p_i] = pos_i(t)_{\pi_i}[p_i] = pos_i(t)_{\pi_i}[p_i]$.

Proof. Let $\pi' = v_0 a_1 \ldots a_t v_t$ be a semipath in $\mathcal{G}$, $ur_{\pi}(\pi') = \pi_0 \# \ldots \# \pi_r$ such that for every $s : 0 \leq s \leq r$, all the edges on $\pi_s'$ belonging to $\pi_j$, for some $j : 1 \leq j \leq k$, and $trc(\pi') = p_0 b_1 p_1 \ldots b_{t+2r} p_{t+2r}$.

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

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

Induction step:...
Let us first consider the case $t = \text{suc}(t_1)$.
Let $i : 0 \leq i \leq \ell + 2r$.
If $t^{adj}_{\text{urv}_\pi}(2i) = 1$, then let $pos_i(t) = 1$. Otherwise, $t^{adj}_{\text{urv}_\pi}(2i) = 2i'$ for some $i'$. From $t = \text{suc}(t_1)$, it follows that
$$t^{adj}_{\text{urv}_\pi}(2i) = 2i''$$ for some $i''$ such that $2i'' = 2i'' + 2$ or $2i'' = 2i'' + 4$.
According to the induction hypothesis, there exists $pos_i(t_1) \in \mathcal{T}_p[\Sigma_{\xi_1}]$ such that $(pos_i(t_1))_{\alpha_{\xi_0}}[p_i] = p_{i''}$.
- If $2i'' = 2i'' + 2$, then let $pos_i(t) = pos_i(suc(t_1)) = suc(pos_i(t_1))$ if $p_{i''} < p_{i''}$, otherwise, let $pos_i(t) = pos_i(suc(t_1)) = \text{pred}(pos_i(t_1))$.
- 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 $s_{j_2}$, $p_{i''}$ is the third position before $s_{j_2}$ in $\alpha_{\xi_0}$, and $j_1 \neq j_2$,
  2. $p_{i''}$ is the position immediately before $s_{j_1}$, $p_{i''}$ is the third position after $s_{j_2}$, $p_{i''}$ is the third position before $s_{j_2}$,
  3. $p_{i''}$ is the position immediately after $s_{j_1}$, $p_{i''}$ is the third position after $s_{j_2}$, and $j_1 \neq j_2$.

We illustrate the argument by considering the second situation above. The arguments for the other three situations are similar.
- if $j_1 < j_2$, then let $pos_i(suc(t_1)) = suc(suc_{j_2-1}(pos_i(t_1)))$,
- if $j_1 > j_2$, then let $pos_i(suc(cur)) = suc_{j_2-1}(pos_i(t_1)))$.

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

Now consider the case $t = \text{suc}_A(t_1)$.
Let $i : 0 \leq i \leq \ell + 2r$.
If $t^{adj}_{\text{urv}_\pi}(2i) = 1$, let $pos_i(t) = 1$. Otherwise, let $t^{adj}_{\text{urv}_\pi}(2i) = 2i'$. From $t = \text{suc}_A(t_1)$, we know that
$$t^{adj}_{\text{urv}_\pi}(2i) = 2i''$$ for some $i''$ such that $2i'' < 2i'$.

From the induction hypothesis, $(pos_i(t_1))_{\alpha_{\xi_0}}(p_i) = p_{i''}$.

If there are no symbols in the subpath of $\text{urv}_\pi(\pi')$ from the position $2i''$ to $2i'$, then the position $2i''$ and $2i'$ both belong to $\pi'$ for some $s : 0 \leq s \leq r$. It follows that $p_{i''}$ and $p_i$ are two positions between $s_{j_2-1}$ and $s_{j_2}$ in $\alpha_{\xi_0}$. Define $pos_i(t)$ as follows.
- If $p_i < p_{i''}$, let $pos_i(suc_A(t_1)) = \text{pred}_{A \times (j_2)}(pos_i(t_1))$.
- If $p_{i''} < p_i$, let $pos_i(suc_A(t_1)) = suc_{A \times (j_2)}(pos_i(t_1))$.

Otherwise (that is, there are symbols from $2i''$ to $2i'$), let $2i''$ be the position before the position $2i'$ on $\text{urv}_\pi(\pi')$ 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 $\pi'_\xi$. Then $p_{2i''}$ is the position immediately after $s_{j_2-1}$ or the position immediately before $s_{j_2}$ in $\alpha_{\xi_0}$. We illustrate the argument by considering the situation that $p_{2i''}$. The discussion for the latter situation is similar. Define $pos_i(t)$ as follows.
- If $p_i < p_{i''}$, let $pos_i(suc_A(t_1)) = suc_A(\times (j_2))(\text{pred}_{s_{j_2-1} \rightarrow 1}(pos_i(t_1)))$.
- If $p_{i''} < p_i$, let $pos_i(suc_A(t_1)) = suc_A(\times (j_2))(\text{pred}_{s_{j_2-1} \rightarrow 1}(pos_i(t_1)))$.

The case $t = \text{pred}(t_1)$ can be discussed similarly as $t = \text{suc}_A(t_1)$.

In summary, for every $t = \text{op}(t_1) \in \mathcal{T}_p[\Sigma_{\xi_1}]$ such that $pos_i(t) \neq 1$ (where $op = suc, pred, suc_A, pred_A$), there is $t_{op} \in \mathcal{T}_p[\Sigma_{\xi_1}]$ such that $pos_i(t) = t_{op} \text{cur}(pos_i(t_1))$.

**Theorem 6** Let $\mathcal{G}$ be a $\nu$-canonical data graph for $\xi_1$, $\xi$ be a 2RRDPQ. Then a 2NRA $A_\xi$ can be constructed from $\xi$ and $\xi_1$ such that $(\mathcal{G})$ is nonempty iff $A_\xi$ accepts $\nu_{\alpha_{\xi_0}}$.

**Proof.** Let $\pi'$ be a path in $\mathcal{G}$, the $\pi$-unraveling of $\pi$ be $\pi_0' \cdots \pi_{r'}'$, where for every $s : 0 \leq s \leq r$, all the edges on $\pi_s' = v_{i_s}a_{i_s+1}v_{i_s+1} \cdots v_{i_{s+1}}$ belong to $p_{i_s}$.

Our goal is to construct a 2NRA $B$ to simulate the runs of $A'$ over $\eta(\text{urv}_\pi(\pi'))$.

Similarly to the transformation from NRRAs to NRAGs in Theorem 3, the 2NRA $B$ goes through $\text{trc}(\pi')$ in $\alpha_{\xi_0}$ and guesses the profile of the current position of $\eta(\text{urv}_\pi(\pi'))$, in order to simulate $A'$ over $\eta(\text{urv}_\pi(\pi'))$. The difference is that instead of storing and guessing the data values, $B$ records and guesses a position term from $\mathcal{T}_p[\Sigma_{\xi_1}]$ (interpreted over $\alpha_{\xi_0}$) for each position term occurring in the profile of the current position in $\eta(\text{urv}_\pi(\pi'))$. The intricacy of the construction is how to guarantee the consistency of the guessed position terms $\mathcal{T}_p[\Sigma_{\xi_1}]$ and how to update them during the simulation.
A locating profile \( \text{loc} \) of \( A' \) over \( \alpha_g \), is defined as a pair \( ((S, \chi, \sim), \text{pos}) \), where \( (S, \chi, \sim) \in \text{Prof}_A \) (cf. proof of Theorem 3).

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

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

- \( \text{pos}(t) = \perp \) for every \( t \in T_A \setminus S \),
- \( \text{pos}(\text{cur}) = \text{cur} \),
- for every \( t, t' \in S \) such that \( t \leq t' \), we have \( \text{pos}(t) \leq \text{pos}(t') \),
- 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 \( \text{op} \in \{ \text{suc}, \text{pred}, \text{suc}_A, \text{pred}_A \mid A \subseteq \Sigma^* \} \), then there is \( t_m \in T_p[\Sigma_{\xi_1}] \) of the form as those in the proof of Lemma 2 (e.g. \( t_{\text{suc}} = \text{suc}(\text{suc}_{k_2,1} (\text{cur})) \)) such that \( \text{pos}(\text{op}(t_j)) = t_{\text{op}}(\text{cur}' \text{pos}(t_j)) \) for \( j = 1, 2 \).

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

Similar to the construction of NRAGs from NRRAs, we define two successor relations between locating profiles.

Let \( ((S_1, \chi_1, \sim_1), \text{pos}_1), ((S_2, \chi_2, \sim_2), \text{pos}_2) \in \Sigma_{\text{loc}}, a \in \Sigma^*, j : 1 \leq j \leq l_1, 1 \leq k_1, k_2 \leq 2l_1, \text{ and } \text{dir} \in \{+1, -1\} \).

In the following, we will define two relations \( ((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a,j),\text{dir}} ((S_2, \chi_2, \sim_2), \text{pos}_2) \) and \( ((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a,s_{k_1},s_{k_2})} ((S_2, \chi_2, \sim_2), \text{pos}_2) \). The latter relation corresponds to the situation that the run of \( A' \) is jumping over \# on \( \text{urv}(\pi') \), and the former relation corresponds to the situation that the run of \( A' \) is not.

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

- \( (S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2) \).
- If \( \text{suc}(\text{cur}) \in S_1 \), then \( \text{pos}_1(\text{suc}(\text{cur})) = \text{suc}(\text{cur}) \) if \( \text{dir} = +1 \), and \( \text{pos}_1(\text{suc}(\text{cur})) = \text{pred}(\text{cur}) \) otherwise.
- If \( a \in A \) and \( \text{suc}_A(\text{cur}) \in S_1 \), then \( \text{pos}_1(\text{suc}_A(\text{cur})) = \text{suc}_A \times (j) (\text{cur}) \) if \( \text{dir} = +1 \), and \( \text{pos}_1(\text{suc}_A(\text{cur})) = \text{pred}_A \times (j) (\text{cur}) \) otherwise.
- For every \( t \in S_1 \) such that \( \text{suc}(\text{cur}) \preceq t \), if \( \text{dir} = +1 \), then \( \text{pos}_2(t[\text{suc}(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{suc}(\text{cur}) \setminus \text{cur}] \), otherwise, \( \text{pos}_2(t[\text{suc}(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}(\text{cur}) \setminus \text{cur}] \).
- For every \( t \in S_1 \) such that \( \text{suc}_A(\text{cur}) \preceq t \) and \( a \in A \), if \( \text{dir} = +1 \), then \( \text{pos}_2(t[\text{suc}_A(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{suc}_A \times (j) (\text{cur}) \setminus \text{cur}] \), otherwise, \( \text{pos}_2(t[\text{suc}_A(\text{cur}) \setminus \text{cur}]) = \text{pos}_1(t)[\text{pred}_A \times (j) (\text{cur}) \setminus \text{cur}] \).
- For every \( t \in S_2 \) such that \( \text{pred}(\text{cur}) \preceq t \), \( \text{pos}_2(t) = \text{pos}_1(t[\text{pred}(\text{cur}) \setminus \text{cur}]) \).
- For every \( t \in S_2 \) such that \( \text{pred}_A(\text{cur}) \preceq t \) and \( a \in A \), \( \text{pos}_2(t) = \text{pos}_1(t[\text{pred}_A(\text{cur}) \setminus \text{cur}]) \).
- For every \( t \in S_1 \) such that \( \text{suc}(\text{cur}) \preceq t \) and \( a \notin A \), \( \text{pos}_2(t) = \text{pos}_1(t) \).
- For every \( t \in S_1 \) such that \( \text{suc}_A(\text{cur}) \preceq t \) and \( a \notin A \), \( \text{pos}_2(t) = \text{pos}_1(t) \).

In the following, we will define \( ((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a,s_{k_1},s_{k_2})} ((S_2, \chi_2, \sim_2), \text{pos}_2) \) 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. \( k_1 = 2k_1' - 1, k_2 = 2k_2' - 1, \text{ and } k_1' \neq k_2' \).

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 \( ((S_1, \chi_1, \sim_1), \text{pos}_1) \xrightarrow{(a,s_{k_1}',s_{k_2}')} ((S_2, \chi_2, \sim_2), \text{pos}_2) \) if the following conditions hold.

- \( (S_1, \chi_1, \sim_1) \xrightarrow{a} (S_2, \chi_2, \sim_2) \).
- If \( \text{suc}(\text{cur}) \in S_1 \), then \( \text{pos}_1(\text{suc}(\text{cur})) = \text{pred}(\text{suc}_{k_2}(\text{cur})) \) if \( k_1' < k_2' \), and \( \text{pos}_1(\text{suc}(\text{cur})) = \text{pred}(\text{pred}_{k_2}(\text{cur})) \) otherwise.
If \( a \in A \) and \( \text{suc}_A(\text{cur}) \in S_1 \), then \( \text{pos}_1(\text{suc}_A(\text{cur})) = \text{pred}_A(\text{suc}_2(\text{cur})) \) if \( k'_1 < k'_2 \), and \( \text{pos}_1(\text{suc}_A(\text{cur})) = \text{pred}_A(\text{suc}_2(\text{cur})) \) otherwise.

For every \( t \in S_1 \) such that \( \text{suc}_A(\text{cur}) \leq t \), if \( k'_1 < k'_2 \), then
\[
\text{pos}_2(t[\text{suc}_A(\text{cur}) \backslash \text{cur}]) = \text{pos}_1(t[\text{pred}^2(\text{suc}_2(\text{cur})) \backslash \text{cur}]),
\]
otherwise,
\[
\text{pos}_2(t[\text{suc}_A(\text{cur}) \backslash \text{cur}]) = \text{pos}_1(t[\text{pred}(\text{suc}_2(\text{cur})) \backslash \text{cur}]).
\]

For every \( t \in S_1 \) such that \( \text{suc}_A(\text{cur}) \leq t \) and \( a \in A \), if \( k'_1 < k'_2 \), then
\[
\text{pos}_2(t[\text{suc}_A(\text{cur}) \backslash \text{cur}]) = \text{pos}_1(t[\text{pred}_A(\text{suc}_2(\text{cur})) \backslash \text{cur}]),
\]
otherwise,
\[
\text{pos}_2(t[\text{suc}_A(\text{cur}) \backslash \text{cur}]) = \text{pos}_1(t[\text{pred}_A(\text{suc}_2(\text{cur})) \backslash \text{cur}]).
\]

For every \( t \in S_2 \) such that \( \text{pred}(\text{cur}) \leq t \), \( \text{pos}_2(t) = \text{pos}_1(t[\text{pred}(\text{cur}) \backslash \text{cur}]). \)

For every \( t \in S_2 \) such that \( \text{pred}(\text{cur}) \leq t \) and \( a \in A \), \( \text{pos}_2(t) = \text{pos}_1(t[\text{pred}(\text{cur}) \backslash \text{cur}]). \)

For every \( t \in S_1 \) such that \( \text{pred}(\text{cur}) \leq t \) and \( \text{pred}(\text{cur}) \leq t \), \( \text{pos}_2(t) = \text{pos}_1(t). \)

For every \( t \in S_1 \) such that \( \text{pred}(\text{cur}) \leq t \) and \( a \notin A \), \( \text{pos}_2(t) = \text{pos}_1(t). \)

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

Suppose \( \pi' = v_0a_1v_1 \ldots a_nv_n \) is a semimath in \( \alpha_g \), \( \text{urw}_{\pi}(\pi') = \pi'_0 \# \ldots \# \pi'_r \), for every \( s : 0 \leq s \leq r \), all the edges on \( \pi'_s \) belong to \( \pi_{j_s} \), \( (1 \leq j_s \leq l) \), and \( \text{trc}(\pi') = d_0a'_1p_1 \ldots a'_{l+2r},p_{l+2r} \) (where for every \( j : 1 \leq j \leq l + 2r \), \( a'_j \in \Sigma^* \cup \{\#\} \)). Then \( B \) does the following.

In each position \( p_i \) \( (0 \leq i \leq \ell + 2r) \) of \( \alpha_g \), \( B \) guesses a locating profile \( \text{loc}_i = ((S_i, \chi_i, \sim_i), \text{pos}_i) \in \Sigma_{loc} \) with
\[
\chi_i = \begin{cases} 
(b_{i,0,m_i,1}, T_{i,0,m_i,1}, b'_{i,0,m_i,1}, s_{i,1,1}) \cdots (b_{i,1,1,1}, T_{i,1,1,1}, b'_{i,1,1,1}, s_{i,1,1}) \cdots (b_{i,m_i,2}, T_{i,m_i,2}, b'_{i,m_i,2}, s_{i,m_i,2})
\end{cases}.
\]

In addition,
- if \( i = 0 \), then \( \text{loc}_0 = (S_0, \chi_0, \sim_0) \) is an initial locating profile, that is, for every \( t \in T_A \) such that \( \text{pred}(\text{cur}) \leq t \) or \( \text{pred}_A(\text{cur}) \leq t \) for some \( A \in \Sigma^* \), \( t \notin T_0 \),
- if \( i = \ell + 2r \), then \( \text{loc}_{\ell+2r} = (S_{\ell+2r}, \chi_{\ell+2r}, \sim_{\ell+2r}) \) is a final profile, that is, for every \( t \in T_A \) such that \( \text{succ}(\text{cur}) \leq t \) or \( \text{ succ}_A(\text{cur}) \leq t \) for some \( A \in \Sigma^* \), \( t \notin T_{\ell+2r} \).

Over each pair of positions \( p_i \) and \( p_{i+1} \) \( (0 \leq i < \ell + 2r) \) of \( \alpha_g \),
- if \( i_s + 2s \leq 2i < i_{s+1} + 2s \) for some \( s : 0 \leq s \leq r \), then \( B \) checks that \( \text{loc}_i \xleftarrow{(a'_{i+1,j})_{\text{dir}}} \text{loc}_{i+1} \), where \( \text{dir} = +1 \) if \( p_{i+1} = p_i + 2 \), and \( \text{dir} = -1 \) otherwise,
- if \( 2i = i_s + 2(s - 1) \) for some \( 1 \leq s \leq r \) (that is, \( 2i \) is the position immediately before \( \# \) in \( \text{urw}_{\pi}(\pi') \)), then \( B \) jumps from \( p_i \) to \( p_{i+1} \), then to \( p_{i+2} \), and checks that \( \text{loc}_i \xrightarrow{(a'_{i+2,j},b'_{i+2,j})} \text{loc}_{i+2} \), where \( k = 2j_{s-1} \) if \( p_{a,g}(\pi_{j_{s-1}}, \text{v}_{i_s}) \) is the position immediately before \( \text{v}_{2j_{s-1}} \), and \( k = 2j_{s-1} - 1 \) if \( p_{a,g}(\pi_{j_{s-1}}, \text{v}_{i_s}) \) is the position immediately after \( \text{v}_{2j_{s-1}} \), \( k' = 2j_s \) if \( p_{a,g}(\pi_{j_s}, \text{v}_{i_s}) \) is the position immediately before \( \text{v}_{2j_s} \), and \( k' = 2j_s - 1 \) if \( p_{a,g}(\pi_{j_s}, \text{v}_{i_s}) \) is the position immediately after \( \text{v}_{2j_s} \).

At the same time, \( B \) simulates the run of \( \mathcal{A}' \) over \( \eta(\text{urw}_{\pi}(\pi')) \) as follows.
- If \( \mathcal{A}' \) 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 \( \mathcal{A}' \) makes a transition \((q,c,q') \) in the position \( 2i \) of \( \eta(\text{urw}_{\pi}(\pi')) \), 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 \( \text{pos}_2(t) \) for every \( t \in S_i \) such that \( \text{cur} \sim_i t \), 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 \( \text{pos} \) in locating profiles are exponential over \(|T_A|\), it follows that the number of states of \( B \) is polynomial over \(|Q|\) and exponential over \(|T_A|\).