# A Linear Time Algorithm for Monadic Querying of Indefinite Data over Linearly Ordered Domains

Mizuhito Ogawa
Japan Science and Technology Corporation, PRESTO
University of Tokyo, Tokyo, 113-8656 Japan
E-mail: mizuhito@acm.org

This paper demonstrates the generation of a linear-time query-answering algorithm based on the constructive proof of Higman's lemma by Murthy and Russell (*A constructive proof of Higman's lemma, 5th IEEE Symposium on Logic in Computer Science, 1990*). The target problem is linear-time evaluation of a fixed disjunctive monadic query on an indefinite database over linearly ordered domains, first posed by van der Meyden (*The complexity of querying indefinite information about linearly ordered domains, in 11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, 1992*). Van der Meyden showed the existence of a linear-time algorithm, but an explicit construction has, until now, not been reported.

*Key Words:* Well-quasi-ordering, Higman's lemma, Indefinite data, Query answering, linear-time algorithm.

## 1. INTRODUCTION

Temporal databases, databases of events in an ordered domain, have attracted attention since the early 90's, and the complexity of evaluating various queries on such databases has been analyzed [1, 10]. One important issue is an efficient algorithm for a query on an indefinite data over a linearly ordered domain, i.e., efficiently answering a query as to whether all possible models of incomplete (partial) information satisfy it. Ref. [20] gives the nice overview of the applications.

In general, query answering on an indefinite database is a hard problem by any measure: combined complexity, expression complexity, and data complexity [21]. Combined complexity is complexity in the usual sense, expression complexity is the complexity when the database is fixed, and data complexity is the complexity when the query is fixed. The complexity of query answering using a search engine on the Web would be measured using expression complexity, and the complexity of finding the best-fit gene to several characteristic alignments would be measured using data complexity.

Van der Meyden estimated the precise complexity of query answering on an indefinite database, as illustrated in Table 1 [20]. The table includes the answer to an open problem that the combined complexity of evaluating a conjunctive ($n$-ary) query containing inequalities is $\Pi_2^p$-complete [9].

He also investigated tractable subclasses and demonstrated that the restriction to monadic queries on monadic indefinite databases reduces the complexity drastically, as shown in Table 2. While the class of monadic queries is restrictive, still contains nontrivial problems, such as a comparison between two gene alignments (assuming $C, G, A$, and $T$ to be monadic predicates).

TABLE 1
Complexity of query answering on an indefinite database

| Complexity type | | |
|---|---|---|
| Combined | Expression | Data |
| $\Pi_2^p$-complete | NP-complete | co-NP-complete |

TABLE 2
Complexity of monadic query answering on monadic indefinite databases

| Complexity type | | |
|---|---|---|
| Combined | Expression | Data |
| co-NP-complete | *linear* | *linear* |

He showed explicit constructions of algorithms for combined and expression complexity; however, for data complexity of a fixed monadic disjunctive query, he only proved the existence of a linear-time algorithm. His (non-constructive) proof is based on Higman's lemma, which states that the embedding relation over finite words is a well-quasi-ordering (WQO). With a WQO, minimal elements of any set are guaranteed to be finitely many, and the linear-time algorithm is reduced to a comparison using these finitely many minimal elements (called *minors*).

This situation frequently appears in upper bound complexity estimation based on WQO techniques. For instance, the graph minor theorem states that the embedding relation on finite graphs is a WQO [16], implying the existence of square time algorithms [14] for a wide range of graph problems [4, 17, 13]. While one could compute minors by brute force, one could not tell whether all had been found. Fortunately, we can apply constructive proofs of Higman's lemma [11, 15, 2] in our setting; the intuition tells us that the Curry-Howard isomorphism automatically realizes a linear-time algorithm.

In this paper, the generation of a linear-time query-answering algorithm for a fixed disjunctive monadic query on an indefinite database over a linearly ordered domain is described. This problem was first posed by van der Meyden [20], and its solution has, until now, not been reported. Our method effectively computes the minors for a given disjunctive monadic query, using the regular expression-like techniques in Murthy and Russell's constructive proof of Higman's lemma [11], thus leading to a linear-time query-answering algorithm.

This paper is organized as follows: Section 2 briefly outlines the problem. Section 3 reviews the results of query answering on an indefinite database [20]. Section 4 gives the constructive proof of Higman's lemma [11] and its variation. Section 5 describes how to generate a linear-time algorithm for fixed disjunctive monadic query answering on an indefinite database. Section 6 concludes with a brief summary and a look at future directions.

## 2. OVERVIEW

In this section, so the reader may get a basic understanding of the target problem, we introduce a simpler version. A complete description is included in Section 3.

Let $x$ and $y$ be lists of symbols $P, Q, R, \cdots$, and let $sublst(x, y)$ be a predicate that returns *true* if $x$ is a *sublist* of $y$. It returns *false* otherwise. More rigorously, if, and only if, there is an order-preserving one-to-one map from the elements in $x$ to those in $y$, $sublst(x, y)$ is *true*; otherwise it is *false*. For instance,

$$sublst([P, P, Q, R], [P, Q, P, Q, Q, P, R]) = true,$$

but

$$sublst([P, P, Q, R], [P, Q, R, Q, Q, P, R]) = false.$$

Given a list $x$, consider an easy query, which corresponds to a *sequential query* described in Section 3.2.

- **Input**: A finite set of lists $\bar{y}(= \{y_1, \cdots, y_t\})$.
- **Output**: A decision as to whether $sublst(x, z)$ holds for each list $z$ with $\wedge_{j=1}^{t} sublst(y_j, z)$.

This query can be regarded as follows. We have partial information about events, and this partial information is represented as a set of lists, $y_j$'s. *Can we then decide whether there is an event sequence that can be represented as $x$?*

This query is answered simply by computing $sublst(x, y_j)$ for each $y_j$, and if some $sublst(x, y_j)$ returns *true*, it holds; otherwise, it does not.

Now consider two extensions: (simplified versions of) a conjunctive query and a disjunctive query. The conjunctive query is formalized as follows: fix a finite number of lists, $x_1, \cdots, x_s$.

- **Input**: A finite set of lists $\bar{y}(= \{y_1, \cdots, y_t\})$.
- **Output**: A decision as to whether $\wedge_{i=1}^{s} sublst(x_i, z)$ holds for each list $z$ with $\wedge_{j=1}^{t} sublst(y_j, z)$.

This is still easy. The query is decomposed into a check on each $x_i$, i.e., whether for each $x_i$, $sublst(x_i, y_j)$ for some $y_j$ holds [20].

However, the disjunctive query is much harder. It is formalized as follows: fix a finite number of lists, $x_1, \cdots, x_s$.

- **Input**: A finite set of lists $\bar{y}(= \{y_1, \cdots, y_t\})$.
- **Output**: A decision as to whether $\vee_{i=1}^{s} sublst(x_i, z)$ holds for each list $z$ with $\wedge_{j=1}^{t} sublst(y_j, z)$.

Finding an efficient solution (a linear-time algorithm) for this query is not as easy as it appears.

EXAMPLE 1. Consider $x_1 = [P, Q, R]$, $x_2 = [Q, R, P]$, and $x_3 = [R, P, Q]$. This holds for $y_1 = [P, Q, P]$ and $y_2 = [R, P]$, even though none of the $x_i$'s and $y_j$'s hold for $sublst(x_i, y_j)$.

Of course, if one computes every possible combination of $z$ (which is computed by interleaving the $y_i$'s), a decision is possible, but this requires an exponential amount of time. For instance, for lists $y_1, \cdots, y_t$ of lengths $n_1, \cdots, n_t$, the number of combinations is $(n_1 + \cdots + n_t)!/(n_1! \times \cdots \times n_t!)$, which grows exponentially.

The aim here is to generate a linear-time algorithm for a given disjunctive query. In my method, suitable finite set $\mathcal{M}$ of finite sets of lists, called *minors* corresponding to the given $x_i$'s, is generated. Namely, for the example $x_1 = [P, Q, R]$, $x_2 = [Q, R, P]$, and $x_3 = [R, P, Q]$ above,

$$\mathcal{M} = \left\{ \begin{array}{l} \{[P,Q,R]\}, \{[Q,R,P]\}, \{[R,P,Q]\}, \{[P,Q],[Q,R],[R,P]\}, \\ \{[P,Q,P],[Q,R]\}, \{[Q,R,Q],[R,P]\}, \{[R,P,R],[P,Q]\}, \\ \{[P,R,P],[Q,R]\}, \{[Q,P,Q],[R,P]\}, \{[R,Q,R],[P,Q]\}, \\ \{[P,Q,P,Q],[R]\}, \{[Q,R,Q,R],[P]\}, \{[R,P,R,P],[Q]\}, \\ \{[Q,P,Q,P],[R]\}, \{[R,Q,R,Q],[P]\}, \{[P,R,P,R],[Q]\} \end{array} \right\}.$$

The disjunctive query for input $\bar{y}$ reduces to whether there is minor $\bar{m}$ in $\mathcal{M}$ such that for each $m \in \bar{m}$ there is a $y_j$ satisfying $sublst(m, y_j)$.

By Higman's lemma (and its variation), minors are guaranteed to be finitely many. This shows the existence of a linear-time algorithm. However, the generation of minors is a different matter. When generating minors, the most difficult aspect is knowing whether all have been found. To know this, we apply the special regular expressions, called *sequential r.e.'s*, used in Murthy and Russell's constructive proof of Higman's lemma [11]. Then, as the minors are generated, we explicitly compute the remaining candidates of minors, which are represented by finite sets of sequential r.e.'s. We eventually find that there are no remaining candidates, meaning that all the minors have been found.

## 3. MONADIC QUERY ON INDEFINITE DATABASE

### 3.1. Indefinite database over linearly ordered domains

Our target problem is the explicit construction of a linear-time algorithm to answer a fixed disjunctive monadic query on an indefinite database. Van der Meyden posed the following problem [20] :

> *In a fixed disjunctive monadic query, there is an algorithm answering the query, which is linear wrt the size of the indefinite database over a linearly ordered domain. What is the algorithm?*

In this section, we briefly review his results. The details are given elsewhere [20].

*Proper atoms* are of the form $P(t)$, where $P$ is a predicate symbol, and $t$ is a tuple of constants or variables. *Order atoms* are of the form $u < v$ or $u \leq v$, where $u$ and $v$ are constants or variables. The atoms are either proper atoms or order atoms. We do not assume the *unique name* assumption, i.e., different constants may represent the same point in a linearly ordered domain.

*Indefinite database* $D$ is a set of ground atoms. Model $M$ of $D$ is a linearly ordered domain (such as time) satisfying $D$. More rigorously, the set of models of $D$ is

$$Mod_{\mathcal{O}}(D) = \{M \mid M \models D \text{ and } <_M \text{ is of type } \mathcal{O}\},$$

where $\mathcal{O}$ is a class of linear order types, typically finite linear orders (**Fin**), linear orders isomorphic to integers (**Z**), and dense linear orders isomorphic to the rationals (**Q**). Van der Meyden showed that these order types do not affect the result,

$M \models D$ (see Corollary 2.9 in Ref. [20]); thus we consider only the order type **Fin**. We regard $D$ as a collection of partial facts over a linearly ordered domain and thus refer to it as indefinite.

A *disjunctive query* (or, simply a *query*) is a positive existential first-order clause constructed from proper and order atoms using only $\exists$, $\wedge$, and $\vee$. A *conjunctive query* is a first-order clause constructed from proper atoms and order atoms using only $\exists$ and $\wedge$. We assume that queries are expressed in disjunctive normal forms, i.e., disjunctions of conjunctive queries. Each conjunctive query in a disjunctive normal form is called a *conjunctive component*. For indefinite database $D$ and query $\varphi$, we define $D \models \varphi$ if $\varphi$ is valid in any model of $D$.

Hereafter, we focus on *monadic queries* (i.e., the database and queries contain only monadic predicate symbols, except for $<$ and $\leq$). A predicate symbol is *monadic* if its arity is less than or equal to one. Without loss of generality, we can assume that a monadic query does not contain constants, i.e., if query $\varphi$ contains constant $u$, then $u \in \varphi$ is replaced with $x$, $\varphi$ is replaced with $\exists x\ [P_u(x) \wedge \varphi]$, and $P_u(u)$ is added to the database with a new predicate symbol, $P_u$.

The next example formalizes Example 1 in terms of monadic queries on an indefinite database.

EXAMPLE 2. Let $D = \{P(a), Q(b), P(c), a < b < c, R(d), P(e), d < e\}$, and let $\varphi = \psi_1 \vee \psi_2 \vee \psi_3$, where

$$\begin{cases} \psi_1 &= \exists xyz[P(x) \wedge Q(y) \wedge R(z) \wedge\ x < y < z], \\ \psi_2 &= \exists xyz[Q(x) \wedge R(y) \wedge P(z) \wedge\ x < y < z], \text{ and} \\ \psi_3 &= \exists xyz[R(x) \wedge P(y) \wedge Q(z) \wedge\ x < y < z]. \end{cases}$$

As a result, $D \models \varphi$. Note that $D \not\models \psi_1$, $D \not\models \psi_2$, and $D \not\models \psi_3$.

### 3.2. Conjunctive monadic query on indefinite database

DEFINITION 1. A conjunctive query is *sequential* if its form is

$$\exists x_1 x_2 \cdots x_n\ [P_{1,1}(x_1) \wedge \cdots \wedge P_{1,k_1}(x_1) \wedge P_{2,1}(x_2) \wedge \cdots \wedge P_{n,k_n}(x_n) \wedge x_1 r_1 x_2 r_2 \cdots r_{n-1} x_n],$$

where $r_1, \cdots, r_{n-1} \in \{<, \leq\}$.

For instance, $\psi_1, \psi_2$, and $\psi_3$ in Example 2 are sequential. Note that each conjunctive query $\psi$ can be transformed into conjunction $\psi_1 \wedge \cdots \wedge \psi_m$ of sequential queries $\psi_i$'s such that $D \models \psi$ if, and only if, $D \models \psi_1 \wedge \cdots \wedge \psi_m$ for each database $D$. We denote the set of all finite subsets of set $X$ by $\mathcal{F}(X)$, and the set of all non-empty subsets of set $X$ by $\mathcal{P}(X)$.

DEFINITION 2. Let $Pred$ be a set of monadic predicates, and let $\Sigma = \mathcal{P}(Pred)$. We define set $FW(\Sigma)(= \Sigma \cdot (\{<, \leq\} \cdot \Sigma)^*)$ of *flexi-words* over $Pred$ to be the set of all finite sequences of the form

$$a_1 r_1 a_2 r_2 \cdots r_{n-1} a_n,$$

where for each $i$, $a_i \in \Sigma$ and $r_i \in \{<, \leq\}$.

For sequential query $\psi$, we denote the set of predicates that hold at point $x$ by $\psi[x]$. Similarly, for database $D$ and model $M$, we denote the set of predicates that hold at point $t$ by $D[t]$ and $M[t]$, respectively. Then, up to variable renaming,

sequential monadic query $\psi$ corresponds to flexi-word $\psi[x_1]r_1\psi[x_2]r_2\cdots r_{n-1}\psi[x_n]$ in $FW(\Sigma)$. For instance,

$$\exists x_1x_2x_3\ [P(x_1)\wedge Q(x_1)\wedge P(x_2)\wedge R(x_3)\wedge\ x_1<x_2<x_3]$$

corresponds to $\{P,Q\}<\{P\}<\{R\}$ with $\Psi[x_1]=\{P,Q\}$, $\Psi[x_2]=\{P\}$, and $\Psi[x_3]=\{R\}$. Similarly, a model corresponds to a flexi-word.

This correspondence is naturally extended to conjunctive queries, i.e., correspondence from a conjunctive query to a finite set of flexi-words in $\mathcal{F}(FW(\Sigma))$. For instance,

$$\begin{array}{rl} & \exists x_1x_2x_3\ [P(x_1)\wedge Q(x_1)\wedge P(x_2)\wedge R(x_3)\wedge\ x_1<x_2<x_3] \\ \wedge & \exists x_1x_2x_4\ [P(x_1)\wedge Q(x_1)\wedge P(x_2)\wedge S(x_4)\wedge\ x_1<x_2\leq x_4] \end{array}$$

corresponds to $\{\ \{P,Q\}<\{P\}<\{R\},\ \{P,Q\}<\{P\}\leq\{S\}\ \}$. If $\psi$ is a conjunctive monadic query, a *path* in $\psi$ is a maximal (wrt implication) sequential subquery of $\psi$. We use the expression $Paths(\psi)$ for the subset of $FW(\Sigma)$ corresponding to the paths of $\psi$. Similarly, database $D$ corresponds to a finite set of flexi-words, and we use the expression $Paths(D)$.

LEMMA 1. *Let $D$ be a monadic database and $\psi$ be a conjunctive monadic query. Then, $D\models\psi$ if, and only if, $D\models p$ for every path $p\in Paths(\psi)$.*

Let $P_1,P_2,\cdots,P_n$ be either proper or order atoms. By regarding indefinite database $D=\{P_1,P_2,\cdots,P_n\}$ as conjunctive monadic formula $P_1\wedge P_2\wedge\cdots\wedge P_n$, the paths of the database are similarly defined. We denote the set of paths of $D$ as $Paths(D)$.

We switch at our convenience among presentations of flexi-words such as models and sequential queries and among those of finite sets of flexi-words such as databases and conjunctive queries. For instance, we define the order $\models$ on flexi-words and $\models_m$ on finite sets of flexi-words by freely interchanging the presentations.

DEFINITION 3. For flexi-words $p$ and $q$, if $p$ holds in $q$ by interpreting $q$ as a model and $p$ as a sequential query, we write $q\models p$.

For finite sets $a$ and $b$ of flexi-words, if for each flexi-word $p\in a$ there is a flexi-word $q\in b$ such that $q\models p$, we write $b\models_m a$.

LEMMA 2. *Let $\psi$ be a sequential query and $\{p\}=Paths(\psi)$. $D\models\psi$ if, and only if, there is path $q\in Paths(D)$ such that $q\models p$.*

From Lemmas 1 and 2, the next corollary is immediate.

COROLLARY 1. *Let $\psi$ be a conjunctive query. Then, $D\models\psi$ if, and only if, $Paths(D)\models_m Paths(\psi)$.*

Note that the number of paths in an indefinite database can grow exponentially wrt the size of the database. For instance, consider the indefinite database

$$\left\{\begin{array}{c} P(a_1),\cdots,P(a_n),\ Q(b_1),\cdots,Q(b_n),\ R(c_1),\cdots,R(c_n), \\ a_1<b_1<\cdots<a_n<b_n,\ a_1<c_1<\cdots<a_n<c_n \end{array}\right\}.$$

It has $2^n$ paths, as shown in Fig. 1. Nevertheless, a conjunctive monadic query on an indefinite database can be answered in linear-time.

LEMMA 3. *For sequential query $p$, $D\models p$ is decided in time $O(|D|\cdot|p|\cdot|Pred|)$.*

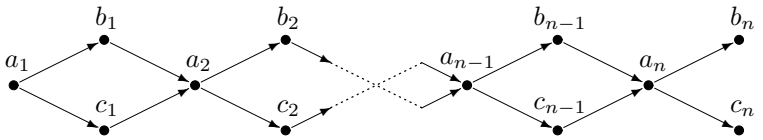


**FIG. 1** Example of exponential growth in number of paths

```
1:    begin
2:      if p is empty then return true;
3:      if D is empty then return false;
4:      if p = ap' and there is minimal constant t
        with a ⋢ D[t] then return SEQ(D \ {t}, p);
5:      if p = a < p' then return SEQ(D \ Pref(D), p');
6:      if p = a ≤ p' then return SEQ(D, p');
7:    end
```

**FIG. 2** Linear-time recursive procedure $SEQ(D, p)$ to decide $D \models p$

The algorithm is given in Fig. 2. In it, for database $D$ and constant $t$, $D \setminus \{t\}$ means the database consisting of all atoms in $D$ except for those containing $t$. $Pref(D)$ is the set of atoms in $D$ corresponding to all $<$-free prefixes of flexi-words in $Paths(D)$.[1] Thus, the next theorem holds.

THEOREM 1. (Corollary 4.4 in Ref. [20]) *For fixed conjunctive monadic query $\varphi$, $D \models \varphi$ for monadic database $D$ is decided in linear-time (wrt the size of $D$).*

### 3.3. Disjunctive monadic query on indefinite database

Let $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_t$, where each $\psi_i$ is a conjunctive query. For disjunctive query $\varphi$, $D \not\models \psi_i$ for each $i$ does not refer to $D \not\models \varphi$, as shown in Example 2. This makes it difficult to decide whether $D \models \varphi$. For the indefinite databases $D_1$ and $D_2$, we define

$$D_1 \sqsubseteq D_2 \text{ if } Paths(D_2) \models_m Paths(D_1).$$

LEMMA 4. *For any disjunctive monadic query $\varphi$, if $D_1 \models \varphi$ and $D_1 \sqsubseteq D_2$, $D_2 \models \varphi$.*

A *quasi order* (QO) $(\Sigma, \leq)$ is a reflexive and transitive binary relation on $\Sigma$.

DEFINITION 4. For a QO $(\Sigma, \leq)$, sequence $x_1, x_2, x_3, \cdots$ (either finite or infinite) is *bad* if $x_i \not\leq x_j$ for all $i, j$ with $i < j$. A $(\Sigma, \leq)$ is a WQO if all infinite sequences $x_1, x_2, x_3, \cdots$ in $\Sigma$ are not bad (i.e., there exist $i$ and $j$ such that $i < j$ and $x_i \leq x_j$). When $\Sigma$ is clear from the context, we simply denote the WQO as $\leq$.

The next lemma is immediate from the definition and is the key to the existence of a linear-time algorithm.

[1] Van der Meyden called this a *minor* [20], but to avoid confusion in terminology in Section 5, we denote it by $Pref(D)$.

LEMMA 5. *Let $(\Sigma, \leq)$ be a WQO. Any non-empty subset $X$ of $\Sigma$ has a finite set of (inequivalent) minimal elements, which are called* minors *of $X$.*

Based on Lemma 4, the set of indefinite databases that satisfy fixed disjunctive query $\varphi$ is upward closed wrt $\sqsubseteq$. Thus, the problem of judging whether $D \models \varphi$ is reduced to a comparison of $D$ with minimal (wrt $\sqsubseteq$) indefinite databases $\{D_i\}$ with $D_i \models \varphi$. Van der Meyden showed that $\models$ over flexi-words is a WQO, so $\models_m$ over finite sets of flexi-words is also a WQO (Lemma 6.3 [20]), assuming that $Pred$ is finite.[2] Elements in the $Pred$ of interest are elements in the monadic queries. Thus, without loss of generality, we can assume that $Pred$ is finite, and that $\sqsubseteq$ is a WQO. This means that minimal indefinite databases $\{D_i\}$ are finitely many from Lemma 5.

The next theorem follows by regarding conjunctive query $\psi_i$ as a query with $Paths(\psi_i) = Paths(D_i)$. Note that this query is unique up to variable renaming.

THEOREM 2. *We fix disjunctive monadic query $\varphi$. There are finitely many conjunctive queries $\{\psi_i\}$ such that $D \models \varphi$ if, and only if, $D \models \psi_i$ for some $i$.*

From Theorems 1 and 2, the next corollary immediately follows.

COROLLARY 2. (Theorem 6.5 in Ref. [20]) *We fix disjunctive monadic query $\varphi$. There is a linear-time algorithm that can decide $D \models \varphi$ for monadic database $D$.*

This corollary means that the data complexity is linear. From Lemma 3, the expression complexity is also linear (with the explicit construction of a linear-time algorithm), though the combined complexity remains co-NP.

Note that Corollary 2 only states the existence of a linear-time algorithm. The construction, which is reduced to the generation of all the minimal indefinite databases wrt $\sqsubseteq$ (or, characteristic queries), will be described in Section 5.

## 4. CONSTRUCTIVE PROOF OF HIGMAN'S LEMMA

In this section, the constructive proof of Higman's lemma is explained. Higman's lemma states that any bad sequence has a finite length. The constructive proof of the lemma is presented by constructing an effective well-founded-order (WFO) among bad sequences.

The basic idea is as follows: for each bad sequence, we first assign a union of special regular expressions that approximate the possible choice of the next element to enlarge the bad sequence. Next, we construct a WFO on sets of special regular expressions such that the regular expression associated with a bad sequence strictly decreases in size when the bad sequence is enlarged. This means that any extension of bad sequences eventually terminates. This is explained in detail elsewhere [11]. Section 4.2 and 4.3 show a variation of Higman's lemma that better fits to our situation.

### 4.1. Constructive proof by Murthy and Russell

Let $\Sigma^*$ be the set of all finite words consisting of symbols in $\Sigma$.

DEFINITION 5. The subword relation $\preceq$ on $\Sigma^*$ is $u \preceq v$ for $u = u_1 \cdots u_m$ and $v = v_1 \cdots v_n$ if there is an order-preserving one-to-one map, $f$, from $[1..m]$ to $[1..n]$ such that $u_i \leq v_{f(i)}$ for each $i$.

[2] An alternative constructive proof of this will be given in Section 5.2 as a by-product.

LEMMA 6 (Higman's lemma). [8] *If* $(\Sigma, \leq)$ *is a WQO,* $(\Sigma^*, \preceq)$ *is a WQO.*

The standard proof by Nash-Williams [12] is non-constructive, especially the reasoning called *minimal bad sequence*, in which (1) the proof proceeds based on contradictions, (2) the existence of a minimal bad sequence is a result of Zorn's lemma, and (3) the arguments on a minimal bad sequence are heavily higher order. An example is universal quantification over all bad sequences. A minimal bad sequence is a bad sequence that is minimal wrt the lexicographical order of the size of each element in a sequence.

Murthy and Russell, Richman and Stolzenberg, and Coquand and Fridlender independently gave constructive proofs for Higman's lemma [11, 15, 2][3]. For a constructive proof, we must make the following assumptions.

1. Let $A$ and $B$ be bad sequences of $\Sigma$, and let $A \sqsubset_{seq} B$ if, and only if, $A$ is a proper extension of $B$; $\sqsubset_{seq}$ is well founded and equipped with an induction scheme.

2. The WQO $\leq$ on $\Sigma$ is decidable.

In the classical sense, the first assumption is obvious from the WQO property of $\leq$ by direct reduction to absurdity. However, in a constructive sense, this is not enough. The WQO that satisfies the assumptions above is called a *constructive well-quasi-ordering* (*CWQO*) [18]. In practice, this assumption is not a serious restriction; a WQO is frequently constructed by embedding relation from a WQO on a simpler data structure, and the base case is frequently either an order on a finite set or a well-ordering on a discrete set (such as integers). Such a WQO is also a CWQO.

We will briefly review the techniques used in Ref. [11]. We refer to an empty word as $\epsilon$ and an upward closure of words that contains $w$ (i.e., $\{x \in \Sigma^* \mid w \preceq x\}$) as $w^\circ$.

For convention, we refer to the symbols in $\Sigma$ as $a, b, c, \cdots$, the words in $\Sigma^*$ as $u, v, w, \cdots$, the finite sequences in $\Sigma$ as $A, B, \cdots$, the finite sequences in $\Sigma^*$ as $V, W, \cdots$, the subsets of $\Sigma^*$ as $L, L', \cdots$, the finite subsets of $\Sigma$ or $\Sigma^*$ as $\alpha, \beta, \cdots$, the subsets of finite subsets of $\Sigma^*$ as $\mathcal{L}, \mathcal{L}', \cdots$, the special periodic expressions called *sequential regular expressions* as $\sigma, \theta, \cdots$, the finite sets of sequential regular expressions as $\Theta, \Theta_1, \Theta_2, \cdots$, the special power set expressions called *base expressions* as $\mathfrak{s}, \mathfrak{t}, \cdots$, and the finite sets of base expressions as $\mathfrak{S}, \mathfrak{S}_1, \mathfrak{S}_2, \cdots$.

DEFINITION 6. Let $b \in \Sigma$, and let $A = a_1, a_2, \cdots, a_k$ be a bad sequence in $\Sigma$. The concatenation of $A$ and $a \in \Sigma$ is $A|a$.

The constant expression $(b - A)$ denotes a subset of $\Sigma$ defined by

$$\{x \in \Sigma \mid b \leq x \ \wedge \ a_i \not\leq x \text{ for each } \ i \leq k\},$$

and the starred expression $(\Sigma - A)^*$ denotes a subset of $\Sigma^*$ defined by

$$\{w = c_1 c_2 \cdots c_n \in \Sigma^* \mid a_j \not\leq c_i \text{ for each } i \leq n, j \leq k\}.$$

Note that $\epsilon \in (\Sigma - A)^*$.

*Sequential regular expression* (sequential r.e.) $\sigma$ is a (possibly empty) concatenation of either *constant* or *starred expressions*. Each sequential r.e. is identified

[3] An idea similar to that in Ref. [15] is found in Ref. [18].

with a set of finite words that are a concatenation of elements in either the constant or starred expressions. For finite multiset $\Theta$ of sequential expressions, we define $L(\Theta) = \cup_{\sigma \in \Theta} \sigma$.[4]

Let $W = w_1, w_2, \cdots$ be a bad sequence in $\Sigma^*$. We explicitly construct finite multiset $\Theta_k(W)$ of sequential r.e.'s for $w_1, w_2, \cdots, w_k$ such that $\Sigma^* \setminus (w_1^\circ \cup \cdots \cup w_k^\circ) \subseteq L(\Theta_k(W))$. For describing $\Theta_k(W)$, we define $\Theta(\sigma, w)$, where $w \in \Sigma^*$. The basic idea of $\Theta(\sigma, w)$ is that, for a word not to be a superword of $w$, it can contain only a proper subword of $w$. We therefore write down the sequential r.e.'s that accept classes of words containing different proper subwords of $w$.

DEFINITION 7. For sequential r.e.'s $\sigma_1, \cdots, \sigma_n$, we define their concatenation $\sigma_1 \cdots \sigma_n$ as $\{w_1 \cdots w_n \mid w_i \in \sigma_i \text{ for } i \leq n\}$, and denote $+$ for the union operation.[5] Let $\sigma$ be a sequential r.e., and let $w \in \sigma$. We define $\Theta(\sigma, w)$ as follows.

1. When $\sigma$ is a constant expression $(b - A)$[6], we can identify $w$ as a single symbol in $\Sigma$ because $w \in (b - A) \subseteq \Sigma$. Therefore, $\Theta(\sigma, w) = (b - A|w) + \epsilon$.

2. When $\sigma$ is starred expression $(\Sigma - A)^*$[7], if $w$ is empty word $\epsilon$, $\Theta(\sigma, \epsilon) = \phi$; otherwise, $w = c_1 c_2 \cdots c_l$ with $c_j \in \Sigma$ for each $j$. Thus, $\Theta(\sigma, w)$ is

$$\uplus_{j=1}^{l} \left\{ \begin{array}{l} (\Sigma - A|c_1)^*((c_1 - A) + \epsilon) \cdots (\Sigma - A|c_{j-1})^*((c_{j-1} - A) + \epsilon) \\ (\Sigma - A|c_j)^*((c_{j+1} - A) + \epsilon)(\Sigma - A|c_{j+1})^* \cdots ((c_l - A) + \epsilon)(\Sigma - A|c_l)^* \end{array} \right\}.$$

3. When $\sigma = \sigma_1 \sigma_2 \cdots \sigma_n$, where $\sigma_i$ is either a constant or starred expression, we fix some decomposition of $w$ into $\sigma_i$'s (i.e., $w = w_1 w_2 \cdots w_n$) with $w_i \in \sigma_i$ for each $i \leq n$.[8] Consequently,

$$\Theta(\sigma, w) = \uplus_{i=1}^{n} \{\sigma_1 \cdots \sigma_{i-1} \theta \sigma_{i+1} \cdots \sigma_n \mid \theta \in \Theta(\sigma_i, w_i)\}.$$

Let $\Theta$ be a finite multiset of sequential r.e.'s. The following lemma shows that if we remove sequential r.e. $\sigma$ from $\Theta$ and replace it with multiset $\Theta(\sigma, w)$ with $w \in \sigma$, the resulting (finite) multiset of sequential r.e.'s includes all the finite words in $L(\Theta)$ not containing $w$.

LEMMA 7. *For sequential r.e. $\sigma$ with $w \in \sigma$, $\sigma \setminus w^\circ \subseteq L(\Theta(\sigma, w))$.*

LEMMA 8. *Let $L \subseteq \Sigma^*$. Assume there is finite multiset $\Theta$ of sequential r.e.'s such that $L \subseteq L(\Theta)$. For any $w \in L$ and $\sigma \in \Theta$ with $w \in \sigma$,*

$$L \setminus w^\circ \subseteq L((\Theta \setminus \{\sigma\}) \uplus \Theta(\sigma, w)).$$

Thus, for bad sequence $W = w_1, w_2, \cdots$, we can construct $\Theta_i(W)$ by starting from $\Sigma^*$ and repeatedly applying Lemma 8. That is,

$$\left\{ \begin{array}{lcl} \Theta_0(W) & = & (\Sigma - ())^* \\ \Theta_{i+1}(W) & = & (\uplus_{w_{i+1} \notin \sigma \in \Theta_i(W)} \{\sigma\}) \ \uplus \ (\uplus_{w_{i+1} \in \sigma \in \Theta_i(W)} \Theta(\sigma, w_{i+1})). \end{array} \right.$$

[4] In Ref. [11], $\Theta$ is defined as a finite set.

[5] Strictly speaking, $+$ is not allowed in a sequential r.e., but we use it as an abbreviation for representing multiple sequential r.e.'s.

[6] In Ref. [11], $\Theta(\sigma, w)$ is defined simply as $(b - A|w)$; for sequential r.e. $\sigma$ consisting of only constant expressions $(a_i - A)$, this $\Theta(\sigma, a_1 \cdots a_l)$ is the empty set.

[7] In Ref. [11], $\Theta(\sigma, w)$ is defined simply as $(\Sigma - A|c_1)^*(c_1 + \epsilon) \cdots (c_{l-1} + \epsilon)(\Sigma - A|c_l)^*$.

[8] The choice of the decomposition is arbitrary with $w_i \in \sigma_i$.

When this process terminates, $\Theta_i(W)$ eventually empties. This means that $\preceq$ is a WQO. For termination, we construct well-founded order $\sqsubset_{setexp}$, which strictly decreases when Lemma 8 is applied. This concludes our constructive proof of Higman's lemma.

DEFINITION 8. Let $A$ and $B$ be finite sequences in $\Sigma$, $(a - A)$ and $(b - B)$ be constant expressions, and $(\Sigma - A)^*$ and $(\Sigma - B)^*$ be starred expressions. We define orders $\sqsubset_{seq}, \sqsubset_{const}$, and $\sqsubset_*$ by

$$\begin{array}{rl} A \sqsubset_{seq} B & \text{if } B \text{ is a proper prefix of } A, \\ (a - A) \ \sqsubset_{const} \ (b - B) & \text{if } a = b \ \wedge \ A \sqsubset_{seq} B, \text{ and} \\ (\Sigma - A)^* \ \sqsubset_* \ (\Sigma - B)^* & \text{if } A \sqsubset_{seq} B. \end{array}$$

Let $\sqsubset_{exp} = \sqsubset_{const} \cup \sqsubset_* \cup \{(a - A)\} \times \{(\Sigma - B)\}$ (i.e., all the constant expressions are below the starred expressions), and let $\sqsubset_{setexp}$ be its multiset extension [3].[9]

For sequential r.e.'s $\sigma = \sigma_1 \cdots \sigma_k$ and $\theta = \theta_1 \cdots \theta_l$, where the $\sigma_i$s and $\theta_j$s are either constant or starred expressions, we define order $\sqsubset_{re}$ as

$$\sigma \sqsubset_{re} \theta \quad \text{if} \quad \uplus_{i=1}^{k} \{\sigma_i\} \sqsubset_{setexp} \uplus_{j=1}^{l} \{\theta_j\},.$$

and define the multiset extension of $\sqsubset_{re}$ as $\sqsubset_{setre}$.

Since the construction of $\Theta$ always enlarges the bad sequences in (either constant or starred) expressions, the next lemma holds.

LEMMA 9. *For sequential r.e. $\sigma$ with $w \in \sigma$, $\Theta(\sigma, w) \sqsubset_{setre} \{\sigma\}$.*

THEOREM 3. *Let $W = w_1, w_2, \cdots$ be a bad sequence in $\Sigma^*$. One can effectively compute finite multiset $\Theta_i(W)$ of sequential r.e.'s such that*

$$\Sigma^* \setminus (w_1^\circ \cup w_2^\circ \cup \cdots \cup w_i^\circ) \subseteq L(\Theta_i(W))$$

*and $\Theta_{i+1}(W) \sqsubset_{setre} \Theta_i(W)$ for each $i$.*

Since $\sqsubset_{setre}$ is well-founded, $W$ must be finite.

COROLLARY 3. *If $(\Sigma, \leq)$ is a CWQO, $(\Sigma^*, \preceq)$ is a CWQO.*

EXAMPLE 3. Let $\Sigma = \{a, b\}$, and consider bad sequence $ab, bbaa, ba, bb, a, b$. For simplicity, we use some optimization rules for sequential r.e.'s:

- reduce constant expressions $(\Sigma - (a, b)), (\Sigma - (b, a))$ to $\epsilon$,
- reduce $\sigma\sigma$ to $\sigma$ for each starred expression $\sigma$,
- avoid sequential r.e.'s including self-deleting constant expressions, such as $(a - (a)), (a - (b, a))$, and
- avoid sequential r.e. $\sigma$ if there is $\sigma'$ with $\sigma \subseteq \sigma'$ that is inferred by either $\sqsubseteq_*$, $\sqsubseteq_{const}$, or $\epsilon \in \theta$ for starred expression $\theta$.

We also use the symbol $\Sigma$ for a starred expression $(\Sigma - ())$ and $a$ for a constant expression $(a - ())$.

Consider bad sequence $ab, bbaa, ba, bb, a, b$ wrt $\preceq$. The construction of the sequential r.e's is shown in Fig. 3. For instance, $(\Sigma - (a))^* b (\Sigma - (b))^* \in \Theta_1$ contains $bbaa$, and its decomposition is

[9] For convention, we denote the multiset extension by adding *set* as the prefix of the index.

$$
\begin{array}{lcl}
\Theta_0 & = & \{(\Sigma-())^*\} \\
\Theta_1 & = & \{\underline{(\Sigma-(a))^*b(\Sigma-(b))^*},\ (\Sigma-(a))^*a(\Sigma-(b))^*,\ (\Sigma-(a))^*(\Sigma-(b))^*\} \\
\Theta_2 & = & \{\underline{b(\Sigma-(b))^*},\ \underline{(\Sigma-(a))^*(\Sigma-(b))^*},\ \underline{(\Sigma-(a))^*b(a-(b))}, \\
 & & \underline{(\Sigma-(a))^*b},\ \underline{(b-(a))a(\Sigma-(b))^*},\ a(\Sigma-(b))^*,\ (\Sigma-(a))^*a\} \\
\Theta_3 & = & \{(\Sigma-(b))^*,\ b,\ (\Sigma-(a))^*,\ (\Sigma-(a))^*(a-(b)),\ (\Sigma-(a))^*b, \\
 & & a(\Sigma-(b))^*,\ (b-(a))(\Sigma-(b))^*,\ a\} \\
\Theta_4 & = & \{(\Sigma-(b))^*,\ b,\ b(a-(b)),\ (\Sigma-(a))^*,\ a(\Sigma-(b))^*, \\
 & & (b-(a))(\Sigma-(b))^*,\ a\} \\
\Theta_5 & = & \{b,\ (\Sigma-(a))^*,\ (\Sigma-(b))^*\} \\
\Theta_6 & = & \{(\Sigma-(b))^*\}
\end{array}
$$

**FIG. 3** Construction of $\Theta_i$ for bad sequence $ab, bbaa, ba, bb, a, b$.

$$b \in (\Sigma-(a)),\ b \in b = (b-()),\ \text{and}\ aa \in (\Sigma-(b))^*.$$

Then, corresponding to each part of the decomposition, $\Theta((\Sigma-(a))^*b(\Sigma-(b))^*, bbaa)$ is constructed as $(\Sigma-(a,b))^*b(\Sigma-(b))^*$, $(\Sigma-(a))^*((b-(b))+\epsilon)(\Sigma-(b))^*$, and $(\Sigma-(a))^*b(\Sigma-(b,a))^*((a-(b))+\epsilon)(\Sigma-(b,a))^*$ (see the underlined part in Fig. 3), and they are optimized to

$$b(\Sigma-(b))^*,\ (\Sigma-(a))^*(\Sigma-(b))^*,\ (\Sigma-(a))^*b(a-(b)),\ \text{and}\ (\Sigma-(a))^*b.$$

Note that the sequence $L(\Theta_1), L(\Theta_2), L(\Theta_3), L(\Theta_4), L(\Theta_5), L(\Theta_6)$ is equal to

$$\{b^*a^*\}, \{b^*a^*\}, \{ba^*, a^*, b^*\}, \{ba^*, a^*, b^*\}, \{a^*, b^*\}, \{a^*\},$$

where the sequence of real complements for bad sequence $ab, bbaa, ba, bb, a, b$ is

$$\{b^*a^*\}, \{ba^*, b^*a\}, \{a^*, b^*\}, \{a^*, b\}, \{\epsilon, b\}, \{\epsilon\}.$$

The sequence $L(\Theta_1), L(\Theta_2), L(\Theta_3), L(\Theta_4), L(\Theta_5), L(\Theta_6)$ is an approximation and has a plateau, but

$$\Theta_1 \sqsupset_{setre} \Theta_2 \sqsupset_{setre} \Theta_3 \sqsupset_{setre} \Theta_4 \sqsupset_{setre} \Theta_5 \sqsupset_{setre} \Theta_6.$$

### 4.2. A Variation

For our purposes, we need a variation (not included in Ref. [11]) of Higman's lemma.

DEFINITION 9. For $\alpha, \beta \subseteq \Sigma$, $\alpha \leq_m \beta$ if $\forall u \in \alpha \exists v \in \beta$ such that $u \preceq v$.

Assume that $(\Sigma, \leq)$ satisfies the CWQO assumptions. Let $\mathcal{F}(\Sigma)$ be the set of all finite sets of $\Sigma$. As a variation of Higman's lemma, $(\mathcal{F}(\Sigma), \leq_m)$ is a WQO. In this section, we further show that $(\mathcal{F}(\Sigma), \leq_m)$ is a CWQO.

A base expression represents the set of all finite sets of elements not greater-than-or-equal to any element in a bad sequence.

DEFINITION 10. Let $A = a_1, a_2, \cdots, a_k$ be a finite bad sequence in $\Sigma$. The corresponding *base expression* is

$$(\Sigma \ominus A) = \mathcal{F}(\{x \in \Sigma \mid a_i \not\leq x \text{ for each } i \leq k\}).$$

We define $(\Sigma \ominus A) \sqsubset_{base} (\Sigma \ominus B)$ if $A \sqsubset_{seq} B$, and define $\sqsubset_{setbase}$ as its multiset extension. For finite multiset $\mathfrak{S}$ of the base expressions, we define $\mathcal{L}(\mathfrak{S}) = \cup_{\mathfrak{s} \in \mathfrak{S}} \mathfrak{s}$.

Let $\alpha_1, \alpha_2, \cdots$ be a bad sequence of elements in $\mathcal{F}(\Sigma)$. We denote the upward closure of $\alpha$ in $\mathcal{F}(\Sigma)$ (i.e., $\{\gamma \in \mathcal{F}(\Sigma) \mid \alpha \leq_m \gamma\}$) with $\alpha^\circ$, and explicitly construct finite multiset $\mathfrak{S}_k$ of base expressions for $\alpha_1, \cdots, \alpha_k$ such that

$$\mathcal{F}(\Sigma) \setminus (\alpha_1^\circ \cup \cdots \cup \alpha_k^\circ) \subseteq \mathcal{L}(\mathfrak{S}_i).$$

To describe $\mathfrak{S}_k$, we define $\mathfrak{S}(\Sigma \ominus A, \alpha)$. The basic idea of $\mathfrak{S}(\Sigma \ominus A, \alpha)$ is, for a finite set not to be a superset of $\alpha$, it must not contain an element in $\alpha$. What we do is write down base expressions that accept finite sets not containing some element of $\alpha$.

Definition 11. Let $(\Sigma \ominus A)$ be the base expression for finite bad sequence $A = a_1, \cdots, a_k$ in $\Sigma$, and let $\alpha \in \Sigma \ominus A$. We then define

$$\mathfrak{S}(\Sigma \ominus A, \alpha) = \{\Sigma \ominus A|a \mid a \in \alpha\}.$$

Lemma 10. *For base expression $\mathfrak{s}$ and $\alpha \in \mathfrak{s}$,*

$$\mathfrak{s} \setminus \alpha^\circ \subseteq \mathcal{L}(\mathfrak{S}(\mathfrak{s}, \alpha)) \text{ and } \mathfrak{S}(\mathfrak{s}, \alpha) \sqsubset_{setbase} \{\mathfrak{s}\}.$$

*Proof.* Let $A = a_1, a_2, \cdots, a_k$ be a bad sequence in $\Sigma$ such that $\mathfrak{s} = (\Sigma \ominus A)$. By the definition of $\mathfrak{S}$, each base expression $\mathfrak{t}$ in $\mathfrak{S}(\mathfrak{s}, \alpha)$ has the form $\mathfrak{t} = (\Sigma \ominus A|a)$ for some $a \in \alpha$. Since $\alpha \in (\Sigma \ominus A)$, $a_i \not\leq a$ for each $i \leq k$, and $A|a$ is a bad sequence. Thus, $\mathfrak{t} = (\Sigma \ominus A|a) \sqsubset_{base} (\Sigma \ominus A)$ and $\mathfrak{S}(\mathfrak{s}, \alpha) \sqsubset_{setbase} \{(\Sigma \ominus A)\}$.

For $\beta \in \mathfrak{s} \setminus \alpha^\circ$, some element in $\alpha$, say $b$, satisfies $b \not\leq x$ for each $x \in \beta$. Thus, $\beta \in (\Sigma \ominus A|b) \in \mathfrak{S}(\mathfrak{s}, \alpha)$. ∎

From Lemma 10, the next lemma and theorem immediately follow.

Lemma 11. *Let $\mathcal{L} \subseteq \mathcal{F}(\Sigma)$. Assume that there is finite multiset $\mathfrak{S}$ of base expressions such that $\mathcal{L} \subseteq \mathcal{L}(\mathfrak{S})$. For any $\alpha \in \mathcal{L}$ and $\mathfrak{s} \in \mathfrak{S}$ with $\alpha \in \mathfrak{s}$,*

$$\mathcal{L} \setminus \alpha^\circ \subseteq \mathcal{L}((\mathfrak{S} \setminus \{\mathfrak{s}\}) \uplus \mathfrak{S}(\mathfrak{s}, \alpha)) \text{ and } (\mathfrak{S} \setminus \{\mathfrak{s}\}) \uplus \mathfrak{S}(\mathfrak{s}, \alpha) \sqsubset_{setbase} \mathfrak{S}.$$

Thus, for bad sequence $\mathcal{A} = \alpha_1, \alpha_2, \cdots$, we can construct $\mathfrak{S}_i(\mathcal{A})$ by starting from $\mathcal{F}(\Sigma)$ and repeating the applications of Lemma 11. That is,

$$\begin{array}{lcl} \mathfrak{S}_0(\mathcal{A}) & = & \mathcal{F}(\Sigma) \\ \mathfrak{S}_{i+1}(\mathcal{A}) & = & (\uplus_{\alpha_{i+1} \notin \mathfrak{s} \in \mathfrak{S}_i(\mathcal{A})} \{\mathfrak{s}\}) \ \uplus \ (\uplus_{\alpha_{i+1} \in \mathfrak{s} \in \mathfrak{S}_i(\mathcal{A})} \mathfrak{S}(\mathfrak{s}, \alpha_{i+1})). \end{array}$$

If this process terminates, $\mathfrak{S}_i(\mathcal{A})$ is empty. This means that $\leq_m$ is a WQO. For termination, we construct well-founded order $\sqsubset_{setbase}$ that strictly decreases when Lemma 11 is applied.

Theorem 4. *Let $\mathcal{A} = \alpha_1, \alpha_2, \cdots$ be a bad sequence in $\mathcal{F}(\Sigma)$. We can then effectively compute finite multiset $\mathfrak{S}_i(\mathcal{A})$ of base expressions such that*

$$\mathcal{F}(\Sigma) \setminus (\alpha_1^\circ \cup \alpha_2^\circ \cup \cdots \cup \alpha_i^\circ) \subseteq \mathcal{L}(\mathfrak{S}_i(\mathcal{A}))$$

*and $\mathfrak{S}_{i+1}(\mathcal{A}) \sqsubset_{setbase} \mathfrak{S}_i(\mathcal{A})$ for each $i$.*

Since $\sqsubset_{setbase}$ is well-founded, $\mathcal{A}$ must be finite.

Corollary 4. *If $(\Sigma, \leq)$ is a CWQO, $(\mathcal{F}(\Sigma), \leq_m)$ is a CWQO.*

*Proof.* Since $\preceq$ is a (C)WQO on $\Sigma$, $\sqsubset_{seq}$ is well-founded and has a well-founded induction scheme, as do $\sqsubset_{base}$ and $\sqsubset_{setbase}$. ∎

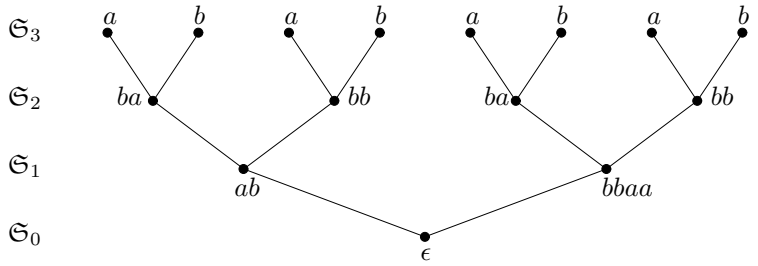


**FIG. 4** Tree of excluding elements for bad sequence $\{ab, bbaa\}, \{ba, bb\}, \{a, b\}$

EXAMPLE 4. Let $\Sigma$ be the set of finite words constructed from $a$ and $b$, and let $\preceq$ be an embedding on $\Sigma$ (i.e., $\preceq$ is a WQO). Consider bad sequence $\{ab, bbaa\}, \{ba, bb\}, \{a, b\}$ wrt $\preceq_m$.

$$
\begin{array}{lcl}
\mathfrak{S}_0 & = & \{(\Sigma \ominus \epsilon)\} \\
\mathfrak{S}_1 & = & \{(\Sigma \ominus (ab)),\ (\Sigma \ominus (bbaa))\} \\
\mathfrak{S}_2 & = & \{(\Sigma \ominus (ab, ba)),\ (\Sigma \ominus (ab, bb)),\ (\Sigma \ominus (bbaa, ab)),\ (\Sigma \ominus (bbaa, bb))\} \\
\mathfrak{S}_3 & = & \{(\Sigma \ominus (ab, ba, a)),\ (\Sigma \ominus (ab, ba, b)),\ (\Sigma \ominus (ab, bb, a)),\ (\Sigma \ominus (ab, bb, b)), \\
 & & (\Sigma \ominus (bbaa, ab, a)),\ (\Sigma \ominus (bbaa, ab, b)), \\
 & & (\Sigma \ominus (bbaa, bb, a)),\ (\Sigma \ominus (bbaa, bb, b))\}
\end{array}
$$

Each basic expression in $\mathfrak{S}_i$ corresponds to a node at the $i$-th level in the tree shown in Fig. 4. They are interpreted as, for instance, $\mathcal{L}(\mathfrak{S}_0) = \mathcal{F}(\Sigma^*)$ and $\mathcal{L}(\mathfrak{S}_3) = \mathcal{F}(\{a^*\}) \cup \mathcal{F}(\{b^*\})$.

### 4.3. Combination

For our purposes, we need constructive proof that $\preceq_m$ over the set of finite sets of finite words is a WQO. By Theorems 3 and 4, we can identify each basic expression as a finite multiset of sequential r.e.'s, and we obtain the next lemma.

LEMMA 12. *Let $W = w_1, \cdots, w_k$ be a bad sequence in $\Sigma^*$ (wrt $\preceq$). Then,*

$$(\Sigma^* \ominus W) \subseteq \mathcal{F}(L(\Theta_k(W))).$$

LEMMA 13. *Let $W = w_1, \cdots, w_k$ be a bad sequence in $\Sigma^*$ (wrt $\preceq$), and let $\alpha \in \Sigma^* \ominus W$. Then,*

$$\mathcal{L}(\mathfrak{S}(\Sigma^* \ominus W, \alpha)) \subseteq \cup_{v \in \alpha} \mathcal{F}(L(\Theta_{k+1}(W|v))).$$

For bad sequence $\mathcal{A} = \alpha_1, \alpha_2, \cdots$ in $\mathcal{F}(\Sigma^*)$, we construct finite multiset $\mathcal{T}_k(\mathcal{A})$ of the finite multisets of the sequential r.e.'s such that $\mathcal{F}(\Sigma^*) \setminus (\alpha_1^\circ \cup \cdots \cup \alpha_k^\circ) \subseteq \cup_{\Theta \in \mathcal{T}_k(\mathcal{A})} \mathcal{F}(L(\Theta))$. That is,

$$\mathcal{T}_i(\mathcal{A}) = \{\Theta_i((v_1, \cdots, v_i)) \mid v_1 \in \alpha_1, \cdots, v_i \in \alpha_i,\ (v_1, \cdots, v_i) \text{ is a bad sequence.}\}.$$

THEOREM 5. *Let $\mathcal{A} = \alpha_1, \alpha_2, \cdots$ be a bad sequence in $\mathcal{F}(\Sigma^*)$ (wrt $\preceq_m$). We can then effectively compute finite multiset $\mathcal{T}_i(\mathcal{A})$ of the multisets of the sequential r.e.'s such that*

$$\mathcal{F}(\Sigma^*) \setminus (\alpha_1^\circ \cup \alpha_2^\circ \cup \cdots \cup \alpha_i^\circ) \subseteq \cup_{\Theta \in \mathcal{T}_i(\mathcal{A})} \mathcal{F}(L(\Theta))$$

*and $\mathcal{T}_{i+1}(\mathcal{A}) \sqsubset_{setsetre} \mathcal{T}_i(\mathcal{A})$ for each $i$, where $\sqsubset_{setsetre}$ is the multiset extension of $\sqsubset_{setre}$.*

EXAMPLE 5. Let $\mathcal{A} = \{ab, bbaa\}, \{ba, bb\}, \{a, b\}$, as in Example 4. Then,

$$\mathcal{F}(\Sigma^*) \setminus \{ab, bbaa\}^\circ \subseteq \mathcal{F}(\Sigma^* \ominus (ab)) \cup \mathcal{F}(\Sigma^* \ominus (bbaa))$$

and

$$\begin{array}{lcl} \mathcal{F}(\Sigma^* \ominus (ab)) & \subseteq & \mathcal{F}(L(\Theta_1)) \\ \mathcal{F}(\Sigma^* \ominus (bbaa)) & \subseteq & \mathcal{F}(L(\Theta_2)), \end{array}$$

where $\Theta_1$ and $\Theta_2$ are as below. Thus, $\mathcal{T}_1(\mathcal{A}) = \{\Theta_1, \Theta_2\}$.

$$\Theta_1 = \{(\Sigma - (a))^* b (\Sigma - (b))^*, (\Sigma - (a))^* a (\Sigma - (b))^*, (\Sigma - (a))^* (\Sigma - (b))^*\},$$

$$\Theta_2 = \left\{ \begin{array}{l} (\Sigma - (b))^* b (\Sigma - (b))^* a (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (a))^* a (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (b))^* b (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (b))^* b (\Sigma - (b))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* (\Sigma - (a))^* a (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (b))^* (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (b))^* b (\Sigma - (b))^* (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (b))^* (\Sigma - (a))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* b (\Sigma - (a))^* a (\Sigma - (a))^*, \ (\Sigma - (b))^* b (\Sigma - (b))^* a (\Sigma - (a))^*, \\ (\Sigma - (b))^* (\Sigma - (a))^* a (\Sigma - (a))^*, \ (\Sigma - (b))^* b (\Sigma - (b))^* (\Sigma - (a))^*, \\ (\Sigma - (b))^* a (\Sigma - (a))^*, \ (\Sigma - (b))^* b (\Sigma - (a))^*, \ (\Sigma - (b))^* (\Sigma - (a))^* \end{array} \right\}$$

## 5. GENERATING LINEAR-TIME ALGORITHM BASED ON WQO

Theorem 2 states that for disjunctive monadic query $\varphi$, there are finitely many conjunctive queries $\{\psi_i\}$ such that $D \models \varphi$ if, and only if, $D \models \psi_i$ for some $i$. Theorem 1 guarantees that $D \models \psi_i$ can be decided in linear-time. Thus, the remaining task is detection of all conjunctive queries $\psi_i$'s, the *minors*. In this section, an algorithm to detect minors is described. The key is the function `ExistsMinor(`$\mathcal{L}$`)`, which decides whether undetected minors remain in $\mathcal{L}$.

For convention, we refer to the disjunctive monadic queries as $\varphi$, the conjunctive monadic queries as $\psi, \psi_1, \psi_2, \cdots$, the sequential queries as $p, q, \cdots$, the indefinite database as $D, D', \cdots$, the models as $M, M', \cdots$, the minors as $\mathcal{M}, \mathcal{M}', \cdots$, the variables as $x, y, z, u, v, x_1, x_2, \cdots$, the constants as $t_1, t_2, \cdots$, the sets of predicates as $a, b, c, a_1, a_2, \cdots$, the finite sets in $FW(\Sigma)$ as $\alpha, \beta, \gamma, \cdots$, the power sets in $\mathcal{F}(FW(\Sigma))$ as $\mathcal{L}, \mathcal{L}', \cdots$, the sequential r.e.'s as $\theta, \theta_1, \theta_2, \cdots$, the either constant or starred expressions as $\sigma, \sigma_1, \sigma_2, \cdots$, the finite multisets of sequential r.e.'s as $\Theta, \Theta_1, \Theta_2, \cdots$, and the finite multisets of finite multisets of sequential r.e.'s as $\mathfrak{T}, \mathfrak{T}', \cdots$.

### 5.1. Design of algorithm to detect minors

We define *minors* as the minimal indefinite databases wrt $\sqsubseteq$ that are valid for $\varphi$, and denote the set of all minors as $\mathcal{M}$. Our aim is to detect $\mathcal{M}$.

Let *Pred* be the set of monadic predicate symbols appearing in $\varphi$, and let $\Sigma = \mathcal{P}(Pred)$. The *ideal* algorithm for detecting minors is presented in Fig. 5. Since $\models_m$ is a WQO, `ExistsMinor(`$\mathcal{L}$`)` will eventually be *false*, so this algorithm is guaranteed to terminate. The algorithm has the following predicates and functions.

```
1:    begin
2:      M:={ };
3:      L:=F(FW(Σ));
4:      n=0;
5:      begin
6:        while ExistsMinor(L) do
7:          begin
8:            NotFound:= true;
9:            while NotFound do
10:           begin
11:             α:=Enumerate(n);
12:             if QueryTest(α) and In(α,L) then
13:               begin
14:                 add α to M;
15:                 L:= Exclude(L,α);
16:                 NotFound:= false;
17:               end
18:             n:= n+1;
19:           end
20:         end
21:       return M;
22:     end
23:   end
```

**FIG. 5** Ideal algorithm for detecting minors $\mathcal{M}$ for disjunctive query $\varphi$

- `Enumerate(n)`. Enumerates all elements of $\mathcal{F}(FW(\Sigma))$ (i.e., a one-to-one map from $\mathbf{N}$ onto $\mathcal{F}(FW(\Sigma))$ such that `Enumerate(j)` $\models_m$ `Enumerate(i)` implies $i \leq j$).
- `Exclude(`$\mathcal{L}$`,`$\alpha$`)`. Computes the subset in $\mathcal{L}(\subseteq \mathcal{F}(FW(\Sigma))$ consisting of all finite sets *not* greater-than-or-equal to $\alpha$ wrt $\models_m$.
- `QueryTest(`$\alpha$`)`. For $\alpha \in \mathcal{F}(FW(\Sigma))$, decides whether, for each model $M$, $M \models \alpha$ implies $M \models \varphi$.
- `In(`$\alpha$`,`$\mathcal{L}$`)`. Decides whether element $\alpha$ is in $\mathcal{L}$.
- `ExistsMinor(`$\mathcal{L}$`)`. For $\mathcal{L} \subseteq \mathcal{F}(FW(\Sigma))$, decides whether there is $\alpha \in \mathcal{L}$ satisfying `QueryTest(`$\alpha$`)`.

The implementation of `QueryTest(`$\alpha$`)` is as follows.

`QueryTest(`$\alpha$`)` is decidable because this feature is specified in the monadic second order logic S1S [19]. To illustrate, assume $\varphi = \psi_1 \vee \psi_2 \vee \psi_3$, where

$$\begin{cases} \psi_1 &= \exists xyz[P(x) \wedge Q(y) \wedge R(z) \wedge\ x < y < z], \\ \psi_2 &= \exists xyz[Q(x) \wedge R(y) \wedge P(z) \wedge\ x < y < z], \text{ and} \\ \psi_3 &= \exists xyz[R(x) \wedge P(y) \wedge Q(z) \wedge\ x < y < z], \end{cases}$$

and let $\alpha = \{\{P\} < \{Q\} < \{P\}, \{R\} < \{P\}\}$, which corresponds to indefinite database

$$\{P(t_1), Q(t_2), P(t_3), t_1 < t_2 < t_3, R(t_4), P(t_5), t_4 < t_5\}.$$

The corresponding conjunctive query $\psi$ is represented in $S1S$ as

$$\exists xyzuv.\ P(x) \wedge Q(y) \wedge P(z) \wedge x < y < z \ \wedge \ Q(u) \wedge R(v) \wedge u < v.$$

Thus, `QueryTest`$(\alpha)$ is represented in S1S as $\varphi \rightarrow (\psi_1 \vee \psi_2 \vee \psi_3)$. In this example, it is valid, so `QueryTest`$(\alpha)$ is *true*.

The test of `ExistsMinor(`$\mathcal{L}$`)` ensures termination of the algorithm; note that if `ExistsMinor(`$\mathcal{L}$`)` is true, `QueryTest(`$\alpha$`) and In(`$\alpha$`,`$\mathcal{L}$`)` eventually become *true*.

The difficult parts are at `Exclude(`$\mathcal{L}$`,`$\alpha$`)` and `ExistsMinor(`$\mathcal{L}$`)`. The modified algorithm to solve these difficulties will be shown in the following.

### 5.2. Implementation of algorithm to detect minors

Instead of precisely computing `Exclude(`$\mathcal{L}$`,`$\alpha$`)`, we use the approximation given by the regular expression-like construction, i.e., multiset $\mathfrak{T}$ of finite multisets of sequential r.e.'s (Section 4.3, Theorem 5), satisfying

$$\texttt{Exclude(}\mathcal{L}\texttt{,}\alpha\texttt{)} \subseteq \cup_{\Theta \in \texttt{ApproxExclude(}\mathfrak{T}\texttt{,}\alpha\texttt{)}} \mathcal{F}(L(\Theta))$$

for $\alpha \in \mathcal{L} \subseteq \cup_{\Theta \in \mathfrak{T}} \mathcal{F}(L(\Theta))$. Corresponding to this setting, we introduce a WQO $\preceq_m$ with $\succeq_m \subseteq \models_m$. Roughly speaking, minors $\mathcal{M}$ wrt $\preceq_m$ are first detected, and at the end of the algorithm $\mathcal{M}$ is minimized wrt $\models_m$.

Let $Pred$ be the set of all predicate symbols appearing in $\varphi$, and let $\Sigma = \mathcal{P}(Pred)$ and $\Sigma_+ = \Sigma \cup \{<\}$ ($\Sigma$ is a lattice wrt set inclusion $\subseteq$).

Definition 12. We define the order $\leq$ on $\Sigma_+$ as the extension of $\subseteq$ on $\Sigma$ by adding the symbol $<$ such that $<$ is incomparable to any element in $\Sigma$.

Lemma 14. *$(\mathcal{F}(\Sigma_+^*), \preceq_m)$ is a CWQO.*

*Proof.* Since $\Sigma_+$ is finite, $\leq$ is a CWQO. Thus, $(\Sigma_+^*, \preceq)$ is a CWQO from Corollary 3, and $(\mathcal{F}(\Sigma_+^*), \preceq_m)$ is a CWQO from 4. ∎

Definition 13. We define mapping $\rho : FW(\Sigma) \rightarrow \Sigma_+^*$ by omitting the symbol $\leq$ in the flexi-word. Furthermore, $\rho$ is naturally extended to $\rho : \mathcal{F}(FW(\Sigma)) \rightarrow \mathcal{F}(\Sigma_+^*)$.

We define mapping $\tau : \Sigma_+^* \rightarrow FW(\Sigma)$ as the mapping to the normal form of the rewriting rules:

$$\begin{cases} < < & \rightarrow & < \\ a \ \ b & \rightarrow & a \ \leq \ b \quad \text{if } a, b \notin \{<, \leq\}. \end{cases}$$

Further, $\tau$ is naturally extended to $\tau : \mathcal{F}(\Sigma_+^*) \rightarrow \mathcal{F}(FW(\Sigma))$.

Example 6.

$$\rho : \begin{cases} \{P,Q\} < \{P\} < \{R\} & (\in FW(\Sigma)) & \rightarrow & \{P,Q\} < \{P\} < \{R\} & (\in \Sigma_+^*) \\ \{P,Q\} < \{P\} \leq \{S\} & (\in FW(\Sigma)) & \rightarrow & \{P,Q\} < \{P\}\ \{S\} & (\in \Sigma_+^*) \end{cases}$$

$$\tau : \begin{cases} \{P,Q\} <<< \{P\} < \{R\} & (\in \Sigma_+^*) & \rightarrow & \{P,Q\} < \{P\} < \{R\} & (\in FW(\Sigma)) \\ \{P,Q\} < \{P\}\ \{S\} & (\in \Sigma_+^*) & \rightarrow & \{P,Q\} < \{P\} \leq \{S\} & (\in FW(\Sigma)) \end{cases}$$

```
1:    begin
2:      M:={ };
3:      T:={{(Σ − ())*}};
4:      n=0;
5:      begin
6:        while ExistsMinorExp(T) do
7:          begin
8:            NotFound:= true;
9:            while NotFound do
10:           begin
11:             α:=Enumerate(n);
12:             if QueryTest(α) and
                    InExp(α,T) and β ⋠m α for each β ∈ τ(M) then
13:               begin
14:                 add α to M;
15:                 T:= ApproxExclude(T,α);
16:                 NotFound:= false;
17:               end
18:             n:= n+1;
19:           end
20:       end
21:       M:= Minimize(τ(M));
22:       return M;
23:     end
24:   end
```

**FIG. 6** Revised algorithm for detecting minors $\mathcal{M}$ for disjunctive query $\varphi$

Note that $\tau\rho$ is the identity, and $\rho\tau$ is an idempotent, i.e., $(\rho\tau)(\rho\tau) = (\rho\tau)$. We naturally extend $\preceq_m$ to $\mathcal{F}(FW(\Sigma))$ such that $\alpha \preceq_m \beta$ for $\alpha, \beta \in \mathcal{F}(FW(\Sigma))$ if $\rho(\alpha) \preceq_m \rho(\beta)$.

LEMMA 15. *For $p, q \in FW(\Sigma)$, $\rho(q) \succeq \rho(p)$ implies $q \models p$. For $\alpha, \beta \in \mathcal{F}(FW(\Sigma))$, $\rho(\beta) \succeq_m \rho(\alpha)$ implies $\beta \models_m \alpha$.*

Since $\preceq$ and $\preceq_m$ are WQOs, this lemma gives an alternative proof that $\sqsubseteq$ is a WQO (see Section 3.3).

LEMMA 16. *For $\alpha \in \mathcal{L} \subseteq \mathcal{F}(FW(\Sigma))$, $\mathcal{L} \setminus \alpha^\circ \subseteq \tau(\rho(\mathcal{L}) \setminus \rho(\alpha)^\circ)$.*

*Proof.* $\rho(\alpha^\circ) \supseteq \rho(\alpha)^\circ \cap \rho(FW(\Sigma))$ by Lemma 15, and $\rho(\mathcal{L} \setminus \alpha^\circ) \subseteq \rho(\mathcal{L}) \setminus \rho(\alpha)^\circ$. ∎

Now, the algorithm presented in Fig. 5 is modified as in Fig. 6. The modifications are at L6, L12, L15, and L21 (in *italics*). Corresponding to the change of the representation from $\mathcal{F}(FW(\Sigma))$ to $\mathcal{F}(\Sigma_+^*)$, `ApproxExclude(`$\mathfrak{T}, \alpha$`)`, `InExp(`$\alpha$`,`$\mathfrak{T}$`)`, and `ExistsMinorExp(`$\mathfrak{T}$`)` are substituted for `Exclude(`$\mathcal{L}$`,`$\alpha$`)`, `In(`$\alpha$`,`$\mathcal{L}$`)`, and `ExistsMinor(`$\mathcal{L}$`)`, respectively.

- `ApproxExclude(`$\mathfrak{T}, \alpha$`)`. For finite multiset $\mathfrak{T}$ of the finite multisets of the sequential r.e.'s and $\alpha \in \mathcal{F}(FW(\Sigma))$, construct finite multiset $\mathfrak{T}'$ of the

finite multisets of the sequential r.e.'s such that $\cup_{\Theta\in\mathfrak{T}} L(\mathcal{F}(\Theta)) \setminus \rho(\alpha)^\circ \subseteq \cup_{\Theta\in\mathfrak{T}'} L(\mathcal{F}(\Theta))$ and $\mathfrak{T} \sqsupset_{setsetre} \mathfrak{T}'$.

- `InExp(`$\alpha$`,`$\mathfrak{T}$`)`. For finite multiset $\mathfrak{T}$ of the finite multisets of the sequential r.e.'s, decide whether $\rho(\alpha) \in \cup_{\Theta\in\mathfrak{T}} \mathcal{F}(L(\Theta))$.

- `ExistsMinorExp(`$\mathfrak{T}$`)`. For finite multiset $\mathfrak{T}$ of the finite multisets of the sequential r.e.'s, decide whether there is an $\alpha \in \cup_{\Theta\in\mathfrak{T}} \tau(\mathcal{F}(L(\Theta)))$ satisfying `QueryTest(`$\alpha$`)`.

`ApproxExclude(`$\mathfrak{T}, \alpha$`)` is realized as $\mathcal{T}_{|\mathcal{M}|}(\mathcal{M})$ by regarding the detected $\mathcal{M}$ as a bad sequence (see Theorem 5 and Lemma 16). `InExp(`$\alpha$`,`$\mathfrak{T}$`)` is computed by checking whether each element in $\rho(\alpha)$ is contained in one of the sequential r.e.'s in $\Theta(\in \mathfrak{T})$. The decision procedure for `ExistsMinorExp(`$\mathfrak{T}$`)` will be described in Section 5.3.

Note that there are two ways for garbages to be added to minors:

1. Since `Exclude(`$\mathcal{L}$`,`$\alpha$`)` $\subseteq \cup_{\Theta\in\texttt{ApproxExclude(}\mathfrak{T}\texttt{,}\alpha\texttt{)}} \mathcal{F}(L(\Theta))$, there may be some element $\beta$ such that $\beta \succeq_m \gamma$ for some $\gamma \in \tau(\mathcal{M})$, so $\beta$ eventually satisfies the condition in L12.

2. Since $\succeq_m \subseteq \models_m$ over finite sets of flexi-words, there may be some $\alpha, \beta \in FW(\Sigma)$ such that $\beta \models_m \alpha$ but $\rho(\beta) \not\succeq_m \rho(\alpha)$.

The former possibility is removed at L12, and the latter is removed at L21 by the newly introduced function `Minimize(`$\mathcal{M}$`)`.

- `Minimize(`$\mathcal{M}$`)`. Minimize $\mathcal{M}$ wrt $\models_m$.

Then, assuming the decision procedure for `ExistsMinorExp(`$\mathfrak{T}$`)` (which will be explained in Section 5.3), the next theorem holds.

THEOREM 6. *The algorithm (in Fig. 6) to detect the set of minors $\mathcal{M}$ for disjunctive query $\varphi$ terminates.*

*Proof.* From Theorem 4, for each iteration of `while ExistsMinorExp(`$\mathfrak{T}$`)`, $\mathfrak{T}$ strictly decreases wrt WFO $\sqsupset_{setsetre}$. ∎

### 5.3. Decision procedure for `ExistsMinorExp(`$\mathfrak{T}$`)`

During execution of the algorithms in Figs. 5 and 6, invariant

$$\mathcal{L} \subseteq \cup_{\Theta\in\mathfrak{T}} \tau(\mathcal{F}(L(\Theta)))$$

holds at each stage. Thus, if `ExistsMinor(`$\mathcal{L}$`)` is *true*, `ExistsMinorExp(`$\mathfrak{T}$`)` is *true*. Since the construction of $\mathfrak{T}$ is well-founded (Theorem 4), `ExistsMinorExp(`$\mathfrak{T}$`)` eventually becomes *false* as does `ExistsMinor(`$\mathcal{L}$`)`. In this section, we describe the decision procedure of `ExistsMinorExp(`$\mathfrak{T}$`)`, a substitute for `ExistsMinor(`$\mathcal{L}$`)`.

Without loss of generality, we can assume that disjunctive query $\varphi$ is $\leq$-free by changing $x \leq y$ to $x < y \vee x = y$. For instance,

$$\exists x_1 x_2 x_3\ [P(x_1) \wedge Q(x_1) \wedge P(x_2) \wedge R(x_3) \wedge\ x_1 < x_2 \leq x_3]$$

has an equivalent $\leq$-free form:

$$\begin{array}{ll} & \exists x_1x_2x_3\ [P(x_1) \wedge Q(x_1) \wedge P(x_2) \wedge R(x_3) \wedge\ x_1 < x_2 < x_3] \\ \vee & \exists x_1x_2\ [P(x_1) \wedge Q(x_1) \wedge P(x_2) \wedge R(x_2) \wedge\ x_1 < x_2]. \end{array}$$

By definition, $\texttt{ExistsMinorExp}(\mathfrak{T}) = \vee_{\Theta\in\mathfrak{T}}\texttt{ExistsMinorExp}(\{\Theta\})$. We will construct an *upper bound* on indefinite database $D$ using $Paths(D) \in \tau(\mathcal{F}(L(\Theta)))$ such that $\texttt{ExistsMinorExp}(\mathfrak{T})$ is reduced to the query $D \models \varphi$.

The basic idea is to construct database $D_{\Theta,n}$ such that $Paths(D_{\theta,n})$ is the maximal wrt $\models_m$ under the upper bound $n$, which is the number of unfolding each starred expression in a sequential r.e. $\theta \in \Theta$. We will show that $D_{\Theta,n} \models \varphi$ and $D_{\Theta,n'} \models \varphi$ are equivalent for sufficiently large $n, n'$. What we will do is find such an upper bound $n$; actually, $n$ is decided only by disjunctive query $\varphi$.

DEFINITION 14. Let $A$ be bad sequence $a_1, a_2, \cdots, a_k$ in $\Sigma_+$ and $b \in \Sigma_+$. For constant expression $(b - A)$, we define a set of regular expressions:

$$\partial(b - A) = max(b^\circ \setminus a_1^\circ \cup \cdots \cup a_k^\circ).$$

Let $\{c_1, \cdots, c_m\} = max(\Sigma_+ \setminus a_1^\circ \cup \cdots \cup a_k^\circ)$ (with suitable numeration of the $c_i$'s). For starred expression $(\Sigma_+ - A)^*$, we define the set of regular expressions:

$$\partial(\Sigma_+ - A)^* \quad = \quad \left\{ \begin{array}{ll} \{(c_1 \cdots c_m)*\} & \text{if } < \in A, \\ \{(< c_1 < \cdots < c_m <)^*\} & \text{if } < \notin A. \end{array} \right.$$

DEFINITION 15. For sequential r.e. $\theta = \sigma_1 \cdots \sigma_l$, we define the set of regular expressions on $\Sigma_+$:

$$\partial\theta = \{w_1 \cdots w_l \mid w_i \in \partial\sigma_i, 1 \leq i \leq l\}.$$

For $w = w(1) \cdots w(l) \in \partial\theta$ with $w(i) \in \partial\sigma_i$, where $w(i)$ is the $i$-th component of $w$, let

$$base(w(i)) = \left\{ \begin{array}{ll} w(i) & \text{if } \sigma_i \text{ is a constant expression,} \\ v & \text{if } \sigma_i \text{ is a starred expression and } w(i) = v^*. \end{array} \right.$$

For $\vec{\mathbf{n}} = (n_1, \cdots, n_l)$, define $w(\vec{\mathbf{n}}) = base(w(1))^{n_1} \cdots base(w(l))^{n_l}$ under the constraints that $n_i = 1$ if $\sigma_i$ is a constant expression.

DEFINITION 16. For finite multiset $\Theta$ of sequential r.e.'s, we define $\partial\Theta = \cup_{\theta\in\Theta}\partial\theta$ and $D_{\Theta,n}$ as a database with $Paths(D_{\Theta,n}) = \tau(\{w(\vec{\mathbf{n}}) \mid w \in \partial\theta, \theta \in \Theta\})$.

EXAMPLE 7. Let $Pred = \{P, Q, R\}$, $b = \{Q\}$, $A = (\{P, Q\}, \{Q, R\})$, and $A' = (\{P\}, <, \{Q, R\})$. Then,

$$\begin{array}{lcl} \partial(b - A) & = & \{\{Q\}, <\} \\ \partial(\Sigma_+ - A)^* & = & \{\{P, R\}, \{Q\}, <\} \\ \partial(\Sigma_+ - A')^* & = & \{\{Q\}, \{R\}\}. \end{array}$$

For $\theta = (\Sigma_+ - A)^*(b - A)(\Sigma_+ - A')^*$,

$$\partial\theta = \left\{ \begin{array}{l} \{P, R\}^*\{Q\}\{Q\}^*, \{P, R\}^*\{Q\}\{R\}^*, \{P, R\}^* < \{Q\}^*, \{P, R\}^* < \{R\}^*, \\ \{Q\}^*\{Q\}\{Q\}^*, \{Q\}^*\{Q\}\{R\}^*, \{Q\}^* < \{Q\}^*, \{Q\}^* < \{R\}^*, \\ <^*\ \{Q\}\{Q\}^*, <^*\ \{Q\}\{R\}^*, <^*< \{Q\}^*, <^*< \{R\}^* \end{array} \right\}.$$

For $w = \{P, R\}^* < \{Q\}^* \in \partial\theta$, $w(2, 1, 3) = \{P, R\}\{P, R\} < \{Q\}\{Q\}\{Q\}$.

DEFINITION 17. Let $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_t$, where $\psi_1, \cdots, \psi_t$ are conjunctive queries. Then, $l(\varphi) = max\{length(p) \mid p \in Paths(\psi_i), 1 \leq i \leq t\}$ and $\Delta(\varphi) = 2^{|Pred|} \cdot t \cdot l(\varphi)^2$.

For model $M$, we use the following notations: $M_{\leq t}$ is a submodel of $M$ consisting of the atoms that contains only constants smaller than or equal to $t \in M$, and $M_{>t}$ is a submodel of $M$ consisting of the atoms that contains only constants greater than $t \in M$.

LEMMA 17. *Let $D$ be a database with $Paths(D) \in \tau(\mathcal{F}(L(\Theta)))$ for finite multiset $\Theta$ of sequential r.e.'s. There then exists $n$ such that $D \models \varphi$ implies $D_{\Theta,n} \models \varphi$.*

*Proof.* Let $n = max(\{length(p) \mid p \in Paths(D)\})$. Then, for each $p \in Paths(D)$ there exists $q \in Paths(D_{\Theta,n})$. Assume that $D_{\Theta,n} \not\models \varphi$, i.e., there is a model $M$ such that $M \not\models \varphi$. Since $M$ is also a model of $D$, this contradicts $D \models \varphi$. ∎

LEMMA 18. *Fix disjunctive query $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_t$, where $\psi_1, \cdots, \psi_t$ are conjunctive queries. Let $\Theta$ be a finite multiset of sequential r.e.'s. For each $n \geq \Delta(\varphi) + 1$, $D_{\Theta,n} \models \varphi$ if, and only if, $D_{\Theta,\Delta(\varphi)+1} \models \varphi$.*

*Proof.* The *if-part* is obvious, so we will prove only the *only-if* part. Assume that $D_{\Theta,\Delta(\varphi)+1} \not\models \varphi$ and $D_{\Theta,n} \models \varphi$ for some $n > \Delta(\varphi) + 1$. Let $\Theta = \{\theta_1, \cdots, \theta_s\}$.

Let $\vec{\mathbf{n}}_1, \cdots, \vec{\mathbf{n}}_{|\partial\Theta|}$ be the minimum sequence of tuples of integers (wrt the product of the product of the order on integers) such that $D \models \varphi$ with $Paths(D) = \{\tau(w_i(\vec{\mathbf{n}}_i)) \mid w_i \in \partial\Theta\}$. Since $D_{\Theta,\Delta(\varphi)+1} \not\models \varphi$, there is some $\vec{\mathbf{n}}_j = (n_1, \cdots, n_l)$ such that some $n_k$ is larger than $\Delta(\varphi) + 1$ (i.e., $n_k > \Delta(\varphi) + 1$).

Let $\vec{\mathbf{n}}'_j = (n_1, \cdots, n_{k-1}, n_k - 1, n_{k+1}, \cdots n_l)$. Let $w_j \in \theta = \sigma_1 \cdots \sigma_l \in \Theta$. By definition, $\sigma_k$ must be starred expression $(\Sigma_+ - A)^*$. From the minimality of $D$, $D' \not\models \varphi$ with $Paths(D') = \{\tau(w_i(\vec{\mathbf{n}}_i)) \mid w_i \in \partial\Theta \setminus \{w_j\}\} \cup \{\tau(w_j(\vec{\mathbf{n}}'_j))\}$. Then there exists model $M$ of $D'$ such that $M \not\models \varphi$.

Let $u$ be the number of elements other than $<$ in $base(w_j)$. Let

$$t_{1,1} \leq \cdots \leq t_{1,u} \leq t_{2,1} \leq \cdots \leq t_{n_k-1,u}$$

be the constants in $M$ corresponding to each element in $w_j$ other than $<$.

If $t_{i,1} = \cdots = t_{i,u}$ for some $i$, $M$ is a model of $D$ as well as $D'$, which contradicts $M \not\models \varphi$. Thus, for each $i$ with $1 \leq i \leq n_k - 1$, there are $m_i$ such that $t_{i,m_i} < t_{i,m_i+1}$ (or $t_{i-1,u} < t_{i,1}$). Since $n_k - 1 > \Delta(\varphi)$ and $u \leq 2^{|Pred|}$, from the pigeon-hole principle, there are $m$ and more than $t \cdot l(\varphi)^2$ $t_{i,m}$'s with $t_{i,m} < t_{i,m+1}$.

Let $M'$ be a model of $base(w_j)$, and let $M_{t_{i,m}}(M')$ be a model extended by inserting $M'$ into $M$ just after $t_{i,m}$. Since $M_{t_{i,m}}(M')$ is a model of $D$, $M_{t_{i,m}}(M') \models \varphi$. Since $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_t$, again from the pigeon-hole principle, there are some $\psi_s$ such that there are more than $l(\varphi)^2$ $t_{i,m}$'s satisfying $M_{t_{i,m}}(M') \models \psi_s$ with $t_{i,m} < t_{i,m+1}$. We consider only such $t_{i,m}$'s.

Since $M \not\models \psi_s$, there is path $p \in Paths(\psi_i)$ such that $M \not\models p$. For such $p$, again from the pigeon-hole principle, there are $t_{i,m}$ and $t_{j,m}$ (with $i < j$), and the decomposition of $p = p_1 < q < p_2$ with $\tau(base(w_j)) \models q$, such that

$$M_{\leq t_{i,m}} \models p_1, M_{>t_{i,m}} \models p_2, M_{\leq t_{j,m}} \models p_1, M_{>t_{j,m}} \models p_2, \text{ and } M' \models q.$$

Since $t_{i,m} < t_{i,m+1}$, $M_{>t_{i,m}} \cap M_{\leq t_{j,m}} \models base(w_j)$; thus, $M_{>t_{i,m}} \cup M_{\leq t_{j,m}} \models q$. This leads to the contradiction $M \models p$. ∎

The next theorem is immediate from Lemmas 17 and 18.

THEOREM 7. *Let $\Theta$ be a finite multiset of sequential r.e.'s. Then,*

$$\texttt{ExistsMinorExp}(\{\Theta\}) = \texttt{QueryTest}(D_{\Theta,\Delta(\varphi)+1}).$$

COROLLARY 5. *Let $\mathfrak{T}$ be a finite multiset of finite multisets of sequential r.e.'s. Then,*

$$\texttt{ExistsMinorExp}(\mathfrak{T}) = \vee_{\Theta\in\mathfrak{T}}\ \texttt{QueryTest}(D_{\Theta,\Delta(\varphi)+1}).$$

Thus, we can effectively compute a set of minors $\mathcal{M}$ for disjunctive monadic query $\varphi$. Let the $\psi_i$'s be conjunctive queries such that $Paths(\psi) = Paths(D)$ for some $D \in \mathcal{M}$. We can then obtain a simple algorithm to decide $D \models \varphi$.

COROLLARY 6. *For fixed disjunctive monadic query $\varphi$, we can effectively compute finitely many conjunctive queries $\{\psi_i\}$ such that $D \models \varphi$ if, and only if, there is $i$ with $D \models \psi_i$ for monadic database $D$.*

Since $D \models \psi$ for conjunctive query $\psi$ and monadic database $D$ is decided in linear-time (see Theorem 1), this Corollary shows that a linear-time algorithm can be generated for fixed disjunctive query answering on an indefinite database over linearly ordered domains.

## 6. CONCLUSION

This paper described the generation of a linear-time query answering algorithm for a fixed disjunctive monadic query on an indefinite database over a linearly ordered domain, using the constructive proof of Higman's Lemma [11]. This problem was first posed by van der Meyden [20], and its solution had, until now, not been reported. Unfortunately, the solution given here remains rather theoretical because of the potentially huge constant factor. That is, as the example in Section 2 shows, the number of minors may explode, and the constant factor may become huge. This phenomena frequently appears when WQO techniques are applied to the upper bound estimation of the complexity.

There are several future directions, including the following two.

- Our method is based on the regular expression techniques in Murthy and Russell's constructive proof of Higman's lemma [11]. Among its known constructive proofs [11, 15, 2] (or intuitionistic proofs [6, 5]), that of Coquand and Fridlender [2] would be one of the most simple and was implemented on Coq prover. This could lead to a simpler method of algorithm generation, in combination with well-developed proof-extraction techniques.

- Kruskal's theorem [12] is an extension of Higman's Lemma to the tree structure. Gupta demonstrated the constructive proof of the weaker form [7], and Veldman presented an intuitionistic proof of Kruskal's theorem [22]. The next extension would be to apply these proofs to a more general class of problems.

## ACKNOWLEDGMENTS

I am grateful to Ronald van der Meyden for his private lectures on indefinite database theory during his stay at NTT and to the members of the Tokyo Programming Seminar (ToPS) for our fruitful discussions. Last but not least, thanks to the anonymous referees for helpful comments; they greatly improved the draft.

REFERENCES


[1] J. Chomicki. Temporal query languages: a survey. In D.M. Gabbay and H.J. Ohlbach, editors, *Temporal Logic: (ICTL'94)*, pages 506–534, 1994. Lecture Notes in Artificial Intelligence, Vol.827, Springer-Verlag.

[2] T. Coquand and D. Fridlender. A proof of Higman's lemma by structural induction, November 1993. manuscript.

[3] N. Dershowitz and Z. Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.

[4] R.M. Fellows and M.A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35(3):727–739, 1988.

[5] D. Fridlender. Higman's lemma in type theory. In E. Gimnez and C. P.-Mohring, editors, *Types for Proofs and Programs, TYPES'96*, pages 112–133, 1996. Lecture Notes in Computer Science, Vol. 1512, Springer-Verlag.

[6] A. Geser. A proof of Higman's lemma by open induction. Technical Report MIP-9606, Passau University, April 1996.

[7] A. Gupta. A constructive proof that trees are well-quasi-ordered under minors. In A. Nerode and M. Taitslin, editors, *Logical foundations of computer science - Tver'92*, pages 174–185, 1992. Lecture Notes in Computer Science, Vol. 620, Springer-Verlag.

[8] G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Mathematical Society*, 2:326–336, 1952.

[9] A. Klug. On conjunctive queries containing inequalities. *Journal of the ACM*, 35(1):146–160, 1988.

[10] M. Koubarakis. The complexity of query evaluation in indefinite temporal constraint databases. *Theoretical Computer Science*, 171:25–60, 1997.

[11] C.R. Murthy and J.R. Russell. A constructive proof of Higman's lemma. In *Proceedings of the 5th IEEE Symposium on Logic in Computer Science*, pages 257–267, 1990.

[12] C.ST.J.A. Nash-Williams. On well-quasi-ordering finite trees. *Proc. Cambridge Phil. Soc.*, 59:833–835, 1963.

[13] L. Perković and B. Reed. An improved algorithm for finding tree decompositions of small width. In P. Widmayer, G. Neyer, and S. Eidenbenz, editors, *WG'99*, pages 148–154, 1999. Lecture Notes in Computer Science, Vol. 1665, Springer-Verlag.

[14] L. Perković and B. Reed. An improved algorithm for finding tree decompositions of small width. *International Journal of Foundations of Computer Science*, 11(3):365–371, 2000.

[15] F. Richman and G. Stolzenberg. Well quasi-ordered sets. *Advances in Mathematics*, 97:145–153, 1993.

[16] N. Robertson and P.D. Seymour. Graph minors XX. Wagner's conjecture, 1988. Manuscript.

[17] N. Robertson and P.D. Seymour. Graph minors XIII. The disjoint path problem. *Journal of Combinatorial Theory Series B*, 63:65–110, 1995.

[18] S.G. Simpson. Ordinal numbers and the Hilbert basis theorem. *Journal of Symbolic Logic*, 53(3):961–974, 1988.

[19] W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, chapter 4, pages 133–192. Elsevier Science Publishers, 1990.

[20] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *Journal of Computer and System Science*, 54(1):113–135, 1997. Previously presented at the *11th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pp.331–345, 1992.

[21] M.Y. Vardi. The complexity of relational query languages. In *Proc. 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[22] W. Veldman. An intuitionistic proof of Kruskal's theorem. Technical Report 17, Department of Mathematics, University of Nijmegen, April 2000.