Generation of a Linear Time Query Processing Algorithm Based on Well-Quasi-Orders

Mizuhito Ogawa[1]

Japan Science and Technology Corporation, PRESTO, and
NTT Communication Science Laboratories
3-1 Morinosato-Wakamiya Atsugi Kanagawa, 243-0198 Japan
mizuhito@theory.brl.ntt.co.jp
http://www.brl.ntt.co.jp/people/mizuhito/

Abstract. This paper demonstrates the generation of a linear time query processing algorithm based on the constructive proof of Higman's lemma described by Murthy-Russell (IEEE LICS 1990). A linear time evaluation of a fixed disjunctive monadic query in an indefinite database on a linearly ordered domain, first posed by Van der Meyden (ACM PODS 1992), is used as an example. Van der Meyden showed the existence of a linear time algorithm, but an actual construction has, until now, not been published elsewhere.

1 Introduction

Temporal databases, which are the databases of events on the linearly ordered domain, have attracted attention from early 90's, and the complexity of evaluating various queries has been analyzed [1, 6]. One of important issues is indefinite database, i.e., processing a query whether all possible models of incomplete (partial) information hold it. Van der Meyden showed that the complexity of evaluating a conjunctive (n-ary) query containing inequalities is Π_2^p-complete [16], solving an open problem.

He also investigated tractable subclasses; for instance, a fixed monadic query can be evaluated in linear time (to the size of a database)[1]. However, for a fixed monadic disjunctive query, he only showed the existence of a linear time query processing algorithm, and an actual construction was not given. His (non-constructive) proof of the existence is based on Higman's lemma, which states that the embedding relation on finite words is a Well-Quasi-Order (WQO). Under a WQO, minimal elements of any set are guaranteed to be finitely many, and the linear time algorithm is constructed as to the comparison with these finitely many minimal elements (called *minors*).

This situation frequently appears in the 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 [12], and this implies the existence of square time algorithms for a wide range of graph problems [3, 13,

[1] Either fixing a query or restricting to a monadic query remains in co-NP.

10]. Difficulty is; one can compute minors by brute-horse manner, but how one can know whether all have been found. Fortunately, we know constructive proofs of Higman's lemma [7, 11, 2], and what we expect is how to apply them.

In this paper, we describe the generation of a linear time query processing algorithm of a fixed disjunctive monadic query in an indefinite database over a linearly ordered domain. This problem is first posed by Van der Meyden in [16], and has, until now, not been published elsewhere. Our method will effectively compute minors for a given disjunctive monadic query, using the regular expression techniques appeared in Murthy-Russell's constructive proof of Higman's lemma [7], and thus will lead a linear time query processing algorithm.

This paper is organized as follows: Section 2 briefly outlines the problem. Section 3 reviews the results of query processing in an indefinite database [16]. Section 4 gives the constructive proof of Higman's lemma [7] and its extension. Section 5 proposes linear time algorithm generation for fixed disjunctive monadic query processing in an indefinite database. Section 6 concludes the paper.

2 A tour

In this section, to get a basic feeling for our method, we introduce a simpler version of the target example. A complete description is included in Section 3.

Let x and y be lists, and let $sublst(x, y)$ be a predicate which returns *true* if x is a sublist of y. It returns *false* otherwise. More rigorously, we can write in Haskell as:

```
sublst :: [a] -> [a] -> Bool
sublst    []     ys  = true
sublst  x:xs     []  = false
sublst (x:xs) (y:ys) = if x == y then sublst xs ys
                                 else sublst (x:xs) ys
```

Let us consider an easy problem. Fix a list x.

- **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 problem can be regarded a query as follows: we know partial information on events (which exclusively occur), and this partial information is represented as a set of lists y_j's. Then, *can we decide whether there exists an event sequence represented as x?*

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

Now we consider two extensions: (simplified versions of) conjunctive query and disjunctive query. The conjunctive query is as follows: fix 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, since this problem is decomposed into the check on each x_i, i.e., whether for each x_i, $sublst(x_i, y_j)$ for some y_j holds.

However, the disjunctive query is much harder. The disjunctive query is formalized as follows: fix 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 problem is not as easy as it appears. To illustrate, consider $x_1 = [P, Q, R]$, $x_2 = [Q, R, P]$, and $x_3 = [R, P, Q]$. This holds for $y_1 = [P, Q]$, $y_2 = [Q, R]$, and $y_3 = [R, P]$, even though none of x_i's and y_j's hold $sublst(x_i, y_j)$.

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

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

$$\begin{aligned}\mathcal{M} = \{&\{[P,Q],[Q,R],[R,P]\},\{[R,Q,P],[Q,P,R],[P,R,Q]\},\\ &\{[P,Q,R]\},\{[Q,R,P]\},\{[R,P,Q]\},\\ &\{[P,Q,P],[Q,R]\},\{[Q,R,Q],[R,P]\},\{[R,P,R],[P,Q]\}\}.\end{aligned}$$

Then the disjunctive query for input $\bar{y} = \{y_1, \cdots, y_t\}$ is reduced as to whether there exists a minor $\bar{m}$ in $\mathcal{M}$ such that for each $m \in \bar{m}$ there exists y_j satisfying $sublst(m, y_j)$.

By Higman's lemma, minors are guaranteed to be finitely many. When generating minors, the most difficult aspect is knowing whether all have been found. To do this, we apply the regular expression techniques, called *sequential r.e.'s*, used in Murthy-Russell's constructive proof of Higman's lemma [7]. Then along generating minors, we explicitly estimate the resting possibility of minors, that is represented by sequential r.e.'s. Then, we will eventually find that the resting possibility is empty. This means that all minors have been found.

3 Disjunctive monadic query on indefinite database

As a target example, we used the linear time fixed disjunctive monadic query processing of indefinite database, proposed by Van der Meyden [16]. He posed the following unsolved problem:

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

In this section, we briefly review his results. For details, please refer to [16].

Proper atoms are of the form $P(a)$, where P is a predicate symbol, and a is a tuple of constants or variables. *Order atoms* are of the form $u < v$, where u and v are order constants or variables. An *indefinite database* D is a set of ground atoms. The atoms are either proper atoms or order atoms. A model $\mathcal{D}$ of D is a linearly ordered domain (such as time) satisfying D. D is a collection of partial facts on a linearly ordered domain, and thus is called indefinite. A model $\mathcal{D}'$ is an extension of a model $\mathcal{D}$ if some subset of $\mathcal{D}'$ is isomorphic to $\mathcal{D}$.

We concentrate on *monadic query processing*, (i.e., database and queries contain only monadic predicate symbols except for $<$). A predicate symbol is *monadic* if its arity is less than or equal to one. The class of monadic queries is restrictive, but contains nontrivial problems, such as a comparison between two gene alignments (regarding C, G, A, T as monadic predicates).

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 $\wedge$. For simplicity, queries are expressed in disjunctive normal forms.

For an indefinite database D and a query φ, we define $D \models \varphi$ if φ is valid in any model of D. For instance, let $D = \{P(a), Q(b), a < b, Q(c), R(d), c < d, R(e), P(f), e < f\}$, 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 neither $D \models \psi_1$, $D \models \psi_2$, nor $D \models \psi_3$.

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

$$\exists t_1 t_2 \cdots t_n \ [P_1(t_1) \wedge \cdots \wedge P_n(t_n) \wedge \ t_1 < t_2 < \cdots < t_n],$$

Let $Pred$ be a set of monadic predicates, and let $\Sigma = \mathcal{P}(Pred)$ be the power set of $Pred$. Σ^* is the set of all the finite words of the symbols in Σ. Without losing generality, we can assume that a monadic query does not contain constants. Thus, up to variable-renaming, sequential monadic queries correspond one-to-one with words in Σ^*. For instance, $\exists t_1 t_2 t_3 \ [P(t_1) \wedge Q(t_1) \wedge P(t_2) \wedge R(t_3) \wedge t_1 < t_2 < t_3]$ corresponds to $\{P, Q\}\{P\}\{R\}$. If ψ is a conjunctive monadic query, a *path* in ψ is a maximal (wrt implication) sequential subquery of ψ.

We use the expression $Paths(\psi)$ for the subset of Σ^* corresponding to paths of ψ, and $length(\psi)$ for the sum of the lengths of all the paths.

Lemma 1. Let D be a monadic database and ψ 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 the indefinite database $D = \{P_1, P_2, \cdots, P_n\}$ as a 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 as $Paths(D)$. Note that the paths in an indefinite database can be computed in linear time wrt the size of the database.

Lemma 2. Let ψ be a sequential query, and let $\preceq$ be a subword relation on Σ^* constructed from $\subseteq$ on Σ (i.e., $u \preceq v$ if there is an order preserving injection f from u to v s.t. $u_i \subseteq v_{f(i)}$ for each i). Then $D \models \psi$ if, and only if, there is a path $\psi' \in Paths(D)$ s.t. $\psi \preceq \psi'$.

For a disjunctive query φ, $D \models \varphi$ may be *true* even if $D \models \psi$ does not for each conjunctive component ψ of φ. This makes it difficult to judge whether $D \models \varphi$. For the indefinite databases, D_1 and D_2, $D_1 \sqsubseteq D_2$ if $Paths(D_1) \leq_m Paths(D_2)$, where $U \leq_m V$ if $\forall u \in U \exists v \in V$ s.t. $u \preceq v$.

Theorem 1. For any disjunctive monadic query φ, if $D_1 \models \varphi$ and $D_1 \sqsubseteq D_2$, then $D_2 \models \varphi$.

At the end of this section, we remark on the existence of the linear time algorithm to decide whether $D \models \varphi$ for a fixed disjunctive query φ. A *quasi order* (QO) $(\Sigma, \leq)$ is a reflexive and transitive binary relation on Σ

Definition 2. For a QO $(\Sigma, \leq)$, a 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 any infinite sequence $x_1, x_2, x_3, \cdots$ in Σ is not bad (i.e., there exist i and j such that $i < j$ and $x_i \leq x_j$). When Σ is clear from the context, we simply denote as $\leq$.

Elements in $Pred$ of interest is elements in the monadic queries. Thus, without loss of generality, we can assume that $Pred$ is finite, and the set inclusion $\subseteq$ in $\Sigma = \mathcal{P}(Pred)$ is a WQO. Then, according to Higman's lemma, $(\Sigma^*, \preceq)$ and $(\mathcal{F}(\Sigma^*), \leq_m)$ are WQOs. Based on Theorem 1, the set of indefinite databases which hold a fixed disjunctive query φ is upward closed wrt $\sqsubseteq$. Thus the problems of judging whether $D \models \varphi$ is reduced to a comparison of D with minimal indefinite databases wrt $\sqsubseteq$. The judgment can be made in linear in the size of D. From this observation, the next theorem follows.

Theorem 2. Let us fix a disjunctive monadic query φ. Then, there exists a linear time algorithm to decide $D \models \varphi$ for a monadic database D.

Note that if a disjunctive monadic query varies, the complexity becomes co-NP. This theorem only says the existence of a linear time algorithm, and the construction, which is reduced to the generation of all the minimal indefinite databases wrt $\sqsubseteq$, will be shown in Section 5.

4 Higman's lemma and the constructive proof

In this section, we will briefly explain the constructive proof of Higman's lemma. Higman's lemma states that any bad sequence has finite length, and the constructive proof of Higman's lemma is presented by constructing the effective well-founded-order (WFO) among bad sequences.

The basic idea is as follows: for a bad sequence, we first assign a union of special regular expressions which approximate possible choice of the next element to enlarge a bad sequence. Next, we construct an WFO on sets of special regular expressions such that for each bad sequence its prefix is strictly larger. Thus, this means that any extension of bad sequences eventually terminates. For details, please refer [7]. We also show an extension of the proof.

4.1 Constructive proof by Murthy-Russell

Lemma 3 (Higman's lemma). [5] If $(\Sigma, \leq)$ is a WQO, then $(\Sigma^*, \preceq)$ is a WQO, where $\preceq$ is a subword relation constructed based on $\leq$ (i.e., $u \preceq v$ if there is an order preserving injection f from u to v s.t. $u_i \leq v_{f(i)}$ for each i).

The standard proof by Nash-Williams [8] 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 impredicative. An example is universal quantification over all bad sequences. A minimal bad sequence is a bad sequence which is minimal wrt the lexicographical order of sizes.

Murthy-Russell, Richman-Stolzenberg, and Coquand-Fridlender independently gave constructive proofs for Higman's lemma [7, 11, 2][2]. For a constructive proof, we must make the following assumptions.

1. Let A and B be bad sequences of Σ, 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 a well founded induction scheme.
2. The WQO $\leq$ on Σ is decidable.

Classically, the first assumption is obviously based on the WQO property of $\leq$, but constructively it is not. The WQO that satisfies the assumptions above is called a *constructive well-quasi-order* (*CWQO*) [14].

We will briefly review the techniques used in [7]. We will refer to an empty word as ϵ and an upward closure of words which contains w (i.e., $\{x \in \Sigma^* \mid w \preceq x\}$) as w°.

As a convention, we will refer to the symbols in Σ as $a, b, c, \cdots$, the words in Σ^* as $u, v, w, \cdots$, the finite sequences in Σ as $A, B, \cdots$, the finite sequences in Σ^* as $V, W, \cdots$, the subsets of Σ^* as $L, L', \cdots$, the finite subsets of Σ^* as $\alpha, \beta, \cdots$, the subsets of finite subsets of Σ^* as $\mathcal{L}, \mathcal{L}', \cdots$, the special periodic

[2] Similar idea to [11] is also found in [14].

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 $\bar{\sigma}, \bar{\theta}, \cdots$, and the finite sets of base expressions as $\Phi, \Phi_1, \Phi_2, \cdots$.

Definition 3. Let $b \in \Sigma$, and let $A = a_1, a_2, \cdots, a_k$ be a bad sequence in Σ. The constant expression $(b - A)$ denotes a subset of Σ 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 Σ^* defined by

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

The concatenation of A and $a \in \Sigma$ is $A|a$.

A *sequential regular expression* (sequential r.e.) σ is a (possibly empty) concatenation of either *constant* or *starred expressions*. The size $size(\sigma)$ of σ is the number of the concatenation. For a finite set Θ of sequential expressions, we define $L(\Theta) = \cup_{\sigma \in \Theta} \sigma$.

Let w_1, w_2, w_3, $\cdots$ be a bad sequence of elements in Σ^*. We will explicitly construct a finite set Θ_k of sequential r.e.'s for $w_1, w_2, \cdots, w_k$ such that $\Sigma^* \setminus (w_1^\circ \cup \cdots \cup w_k^\circ) \subseteq L(\Theta_k)$. For a word not to be a superword of w, it can only contain a proper subword of w. So what we do is write down the sequential r.e.'s which accept classes of words containing different proper subwords of w.

Definition 4. 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. Let σ be a sequential r.e. and let $w \in \sigma$. We will define $\Theta(\sigma, w)$ as follows.

1. When σ is a constant expression $(b - A)$[3], we can identify w as a single symbol in Σ because $\sigma \subseteq \Sigma$. Then $\Theta(\sigma, w) = (b - A|w) + \epsilon$.
2. When σ is a starred expression $(\Sigma - A)^*$[4], let $w = c_1 c_2 \cdots c_l$ with $c_j \in \Sigma$ for each j. Then

$$\begin{aligned}\Theta(\sigma, w) = \cup_{j=1}^{l} \ & (\Sigma - A|c_1)^*((c_1 - A) + \epsilon) \cdots (\Sigma - A|c_{j-1})^*((c_{j-1} - A) + \epsilon) \\ & (\Sigma - A|c_j)^* \\ & ((c_{j+1} - A) + \epsilon)(\Sigma - A|c_{j+1})^* \cdots ((c_l - A) + \epsilon)(\Sigma - A|c_l)^*.\end{aligned}$$

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

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

[3] Ref. [7] has a flaw that $\Theta(\sigma, w)$ is simply defined as $(b - A|w)$.

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

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

Lemma 4. Let $L \subseteq \Sigma^*$. Assume that there is a finite set Θ of sequential r.e.'s such that $L \subseteq L(\Theta)$. For any $w \in L$, $\sigma \in \Theta$ with $w \in \sigma$,

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

Thus, for a bad sequence $w_1, w_2, \cdots$, we can construct Θ_k by starting from Σ^* and repeating the applications of Lemma 4. If this process terminates, Θ_k eventually empties, and $\preceq$ is a WQO. For termination, we construct a well-founded order $\sqsubset_{setexp}$ which strictly decreases when Lemma 4 is applied. This gives a constructive proof of Higman's lemma.

Definition 5. For the finite sequences A, B in Σ, $A \sqsubset_{seq} B$ if B is a proper prefix of A. For any pair of constant expressions $(a-A)$ and $(b-B)$, $(a-A) \sqsubset_{const} (b-B)$ if $a = b \wedge A \sqsubset_{seq} B$. For any pair of starred expressions $(\Sigma - A)^*$ and $(\Sigma - B)^*$, $(\Sigma - A)^* \sqsubset_* (\Sigma - B)^*$ if $A \sqsubset_{seq} B$. Let $\sqsubset_{exp} = \sqsubset_{const} \cup \sqsubset_* \cup \{(a-A)\} \times \{(\Sigma - B)\}$ (i.e., all the constant expressions are below the starred expressions). Let $\sqsubset_{setexp}$ be a multiset extension [9] of $\sqsubset_{exp}$.

We define an ordering $\sqsubset_{re}$ of sequential r.e.'s by $\sigma \sqsubset_{re} \theta \Leftrightarrow \uplus_{i=1}^{k}\{\sigma_i\} \sqsubset_{setexp} \uplus_{j=1}^{l}\{\theta_j\}$, for $\sigma = \sigma_1 \cdots \sigma_k$ and $\theta = \theta_1 \cdots \theta_l$, where the σ_is and θ_js are either constant or starred expressions. We also denote a multiset extension of $\sqsubset_{re}$ with $\sqsubset_{setre}$.

Theorem 3. Let $W = w_1, w_2, \cdots, w_k$ be a finite bad sequence in Σ^*. One can effectively compute a finite set Θ_i of sequential r.e.'s for $i \leq k$ such that

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

and $\Theta_{i+1} \sqsubset_{setre} \Theta_i$ for $i < k$.

Corollary 1. If $(\Sigma, \leq)$ is a CWQO, then $(\Sigma^*, \preceq)$ is a CWQO.

Example 1. Let $\Sigma = \{a, b\}$. Consider the bad sequence $ab, bbaa, ba, bb, a, b$ wrt $\preceq$.

$$\begin{aligned}
\Theta_0 &= (\Sigma - \epsilon)^* \\
\Theta_1 &= (\Sigma - a)^*(b + \epsilon)(\Sigma - b)^* \cup (\Sigma - a)^*(a + \epsilon)(\Sigma - b)^* \\
&= \{b^*a^*\} \\
\Theta_2 &= (b + \epsilon)(\Sigma - b) \cup (\Sigma - a)(a + \epsilon) \\
&= \{ba^*, b^*a\} \\
\Theta_3 &= (\Sigma - b) \cup (b + \epsilon) \cup (a) \cup (\Sigma - a) \\
&= \{a^*, b^*\} \\
\Theta_4 &= (\Sigma - b) \cup (b + \epsilon) \\
&= \{a^*, b\} \\
\Theta_5 &= \{\epsilon, b\} \\
\Theta_6 &= \{\epsilon\}
\end{aligned}$$

4.2 An Extension

For our purposes, we need further extension to the sets of finite sets of finite words (which is not included in [7]). Let $\mathcal{F}(\Sigma^*)$ be the set of all finite sets of Σ^*. Assume that $(\Sigma^*, \leq)$ satisfies the CWQO assumptions. Note that an embedding $(\Sigma^*, \preceq)$ satisfies them. We define $\alpha \leq_m \beta$ for $\alpha, \beta \in \mathcal{F}(\Sigma^*)$ if, for each $x \in \alpha$, there exists $y \in \beta$ such that $x \leq y$. We also denote the upward closure of α in $\mathcal{F}(\Sigma^*)$ (i.e., $\{\gamma \in \mathcal{F}(\Sigma^*) \mid \alpha \leq_m \gamma\}$) with α°.

Definition 6. Let $W = w_1, w_2, \cdots, w_k$ be a finite bad sequence in Σ^*. The *base expression* is

$$(\Sigma^* \ominus W) = \mathcal{F}(\{u \in \Sigma^* \mid w_i \not\leq u \text{ for each } \ i \leq k\})$$

We define $\Sigma^* \ominus V \sqsubset_{base} \Sigma^* \ominus W$ if $V \sqsubset_{seq} W$. For a finite set Φ of base expressions, we define $\mathcal{L}(\Phi) = \cup_{\bar{\sigma} \in \Phi} \bar{\sigma}$.

Let $\alpha_1, \alpha_2, \cdots$ be a bad sequence of elements in $\mathcal{F}(\Sigma^*)$. We will construct a finite set Φ_k of base expressions for $\alpha_1, \cdots, \alpha_k$ such that $\mathcal{F}(\Sigma^*) \setminus (\alpha_1^\circ \cup \cdots \cup \alpha_k^\circ) \subseteq \mathcal{L}(\Phi_i)$. For a finite set not to be a superset of α, it must contain one of the elements in α. What we do is write down base expressions which accept finite sets not containing some element of α.

Definition 7. Let $(\Sigma^* \ominus V)$ be the base expression for a finite bad sequence V in Σ^*, and let $\alpha \in (\Sigma^* \ominus V)$. We then define $\Phi(\Sigma^* \ominus V, \alpha) = \{\Sigma^* \ominus V | v \mid v \in \alpha \ \wedge \ v_i \not\preceq v \text{ for each } v_i \in V\}$.

Lemma 5. Let $\mathcal{L} \subseteq \mathcal{F}(\Sigma^*)$. Assume that there is a finite set Φ of base expressions such that $\mathcal{L} \subseteq \cup_{\bar{\sigma} \in \Phi} \bar{\sigma}$. For any $\alpha \in L$ and $\bar{\sigma} \in \Phi$ with $\alpha \in \bar{\sigma}$,

$$\mathcal{L} \setminus \alpha^\circ \subseteq \mathcal{L}((\Phi \setminus \{\bar{\sigma}\}) \cup \Phi(\bar{\sigma}, \alpha)).$$

Let $\sqsubset_{base}$ be the multiset extension of $\sqsubset_{seq}$. Since $\preceq$ is a WQO on Σ^*, $\sqsubset_{seq}$ is well founded, and so is $\sqsubset_{base}$. Thus, by using a constructive proof similar to Higman's lemma, we obtain the next theorem.

Theorem 4. Let $\mathcal{A} = \alpha_1, \alpha_2, \cdots, \alpha_k$ be a finite bad sequence in $\mathcal{F}(\Sigma^*)$. Then one can effectively compute a finite set Φ_i of base expressions for $i \leq k$ such that

$$\mathcal{F}(\Sigma^*) \setminus (\alpha_1^\circ \cup \alpha_2^\circ \cup \cdots \cup \alpha_i^\circ) \subseteq \mathcal{L}(\Phi_i)$$

and $\Phi_{i+1} \sqsubset_{base} \Phi_i$ for $i < k$.

Corollary 2. If $(\Sigma^*, \leq)$ is a CWQO, then $(\mathcal{F}(\Sigma^*), \leq_m)$ is a CWQO.

Example 2. Let $\Sigma = \{a, b\}$. Consider the bad sequence $\{ab, bbaa\}, \{ba, bb\}, \{a, b\}$ wrt $\leq_m$.

$$
\begin{aligned}
\Phi_0 &= \{(\Sigma^* \ominus \epsilon)\} \\
\Phi_1 &= \{(\Sigma^* \ominus (ab)),\ (\Sigma^* \ominus (bbaa))\} \\
&= \{\mathcal{F}(b^*a^*\}),\ \mathcal{F}(\{a^*(b+\epsilon)a^*b^*(a+\epsilon)b^*\})\} \\
\Phi_2 &= \{(\Sigma^* \ominus (ab, ba)),\ (\Sigma^* \ominus (ab, bb)), \\
&\quad (\Sigma^* \ominus (bbaa, ab)),\ (\Sigma^* \ominus (bbaa, bb))\} \\
&= \{\mathcal{F}(\{a^*, b^*\}),\ \mathcal{F}(\{(b+\epsilon)a^*, b^*(a+\epsilon)\}), \\
&\quad \mathcal{F}(\{a^*(b+\epsilon)a^*\})\} \\
\Phi_3 &= \{(\Sigma^* \ominus (ab, ba, a)),\ (\Sigma^* \ominus (ab, ba, b)), \\
&\quad (\Sigma^* \ominus (ab, bb, a)),\ (\Sigma^* \ominus (ab, bb, b)), \\
&\quad (\Sigma^* \ominus (bbaa, ab, a)),\ (\Sigma^* \ominus (bbaa, ab, b)), \\
&\quad (\Sigma^* \ominus (bbaa, bb, a)),\ (\Sigma^* \ominus (bbaa, bb, b))\} \\
&= \{\mathcal{F}(\{a^*\}),\ \mathcal{F}(\{b^*\})\}
\end{aligned}
$$

5 Algorithm generation based on WQO techniques

Throughout the section, we use the symbol D for an indefinite database, and we fix a disjunctive monadic query $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_m$, where the ψ_i's are conjunctive components (i.e., conjunctive monadic queries).

5.1 Design of disjunctive query processing algorithm

We say *minors* for minimal indefinite databases wrt $\sqsubseteq$, which are valid for φ, and a set of all minors is denoted by $\mathcal{M}_\varphi$. From the observation in Section 3, we know that the essence of linear time algorithm generation for deciding $D \models \varphi$ is reduced to generating $\mathcal{M}_\varphi$. Thus, our aim is to generate $\mathcal{M}_\varphi$.

Let $Pred$ be the set of monadic predicate symbols appearing in φ, and let $\Sigma = \mathcal{P}(Pred)$. Then $Paths(D) \in \mathcal{F}(\Sigma^*)$, and we identify D and $Paths(D)$. (Thus, we also identify $\sqsubseteq$ and $\leq_m$.)

Since Σ is finite, $\mathcal{F}(\Sigma^*)$ is enumerable. We assume the enumeration function $\texttt{Enumerate} : \mathbf{N} \to \mathcal{F}(\Sigma^*)$ such that $i < j$ implies $\texttt{Enumerate}(i) \not>_m \texttt{Enumerate}(j)$. Such `Enumerate` is easily obtained by enumerating objects from smaller to larger (in size). Then, the algorithm to generate minors is presented in Figure 1 with the aid of the following predicates and functions.

- `Exclude`$(\bar{\Theta}, \alpha)$: For a finite set $\bar{\Theta}$ of finite sets of sequential r.e.'s and $\alpha \in \mathcal{F}(\Sigma^*)$, construct a finite set $\bar{\Theta}'$ of finite sets of sequential r.e.'s such that $\cup_{\Theta \in \bar{\Theta}} L(\mathcal{F}(\Theta)) \setminus \alpha^\circ \subseteq \cup_{\Theta' \in \bar{\Theta}'} L(\mathcal{F}(\Theta'))$ and $\bar{\Theta} >_m \bar{\Theta}'$.
- `QueryTest`(α): For $\alpha \in \mathcal{F}(\Sigma^*)$, decide whether $D \models \alpha$ implies $D \models \varphi$.
- `ExistsMinor`$(\bar{\Theta})$: For a finite set $\bar{\Theta}$ of finite sets of sequential r.e.'s, decide whether there exists $\alpha \in \cup_{\Theta \in \bar{\Theta}} L(\mathcal{F}(\Theta))$ satisfying `QueryTest`(α).
- `Minimize(M)`: For a finite subset M of $\mathcal{F}(\Sigma^*)$, minimize `M` wrt $\leq_m$.

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

Fig. 1. The algorithm to detect minors $\mathcal{M}_\varphi$

The implementation of these predicates and functions is as follows: `QueryTest(`α`)` is decidable, because this is specified in the monadic second order logic S1S [15]. To illustrate, let $\varphi = \psi_1 \vee \psi_2 \vee \psi_3$, where

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

and let $\alpha = \{PQ, QR, RP\}$. `QueryTest`(α) is represented in S1S as

$$\begin{aligned} (\exists xyzuvw.\ & P(x) \wedge Q(y) \wedge x < y \ \wedge Q(z) \wedge R(u) \wedge z < u \ \wedge R(v) \wedge P(w) \wedge v < w \\ & \wedge x \neq y \ \wedge \ x \neq z \ \wedge \ x \neq u \ \wedge \ x \neq v \ \wedge \ x \neq w \\ & \wedge y \neq z \ \wedge \ y \neq u \ \wedge \ y \neq v \ \wedge \ y \neq w \ \wedge \ z \neq u \\ & \wedge z \neq v \ \wedge \ z \neq w \ \wedge \ u \neq v \ \wedge \ u \neq w \ \wedge \ v \neq w) \\ & \rightarrow (\psi_1 \vee \psi_2 \vee \psi_3) \end{aligned} .$$

This is valid and `QueryTest`(α) is *true*.

`Exclude` is constructed by repeating the applications of Theorem 3 and Theorem 4. `Minimize` is easily computed by using $\leq_m$.

Note that if `ExistsMinor(L)` then eventually `QueryTest(Enumerate(n)) and In(Enumerate(n),L)` becomes *true*. Thus, the test of `ExistsMinor(L)` ensures termination of the algorithm, and there are difficulties associated with `ExistsMinor`.

5.2 Construction of `ExistsMinor(L)`

Let Φ be a finite set of base expressions.

Definition 8. For the constant expression $(b-A)$ and the starred expression $(\Sigma-A)^*$, we define $\psi((b-A))$ as the maximal element in $(b-A)$, and $\psi((\Sigma-A)^*)$ as the maximal element in $(\Sigma-A)^*$. (Since $\Sigma = \mathcal{P}(Pred)$ is a lattice wrt $\subseteq$, we know that such maximal elements exist.)

For the sequential r.e. $\theta = \sigma_1 \cdots \sigma_l$, we define $\Psi(\theta, n)$ as $\psi(\sigma_1)^{n_1} \cdots \psi(\sigma_l)^{n_l}$ where $n_i = 1$ if σ_i is a constant expression, and $n_i = n$ otherwise.

Lemma 6.

1. Let θ be a sequential r.e. Then, $\forall w \in \theta\ \exists n$ s.t. $w \preceq w'$ for some path $w' \in Paths(\Psi(\theta, n))$.
2. Let θ be a sequential r.e. Then, $\forall S \subseteq \mathcal{F}(\theta)\ \exists n$ s.t. $S \leq_m Paths(\Psi(\theta, n))$.
3. Let Θ be a finite set of sequential r.e.'s. Then, $\forall T \in L(\Theta)\ \exists n$ s.t. $T \leq_m \{\Psi(\theta, n) \mid \theta \in \Theta\}$.

Definition 9. Let Θ be a finite set of sequential r.e.'s $\theta_1, \cdots, \theta_s$. Let $\varphi = \psi_1 \vee \psi_2 \vee \cdots \vee \psi_m$ where $\psi_1, \cdots, \psi_t$ are conjunctive components, and let $l(\varphi) = max(\{length(\psi_i) \mid 1 \leq i \leq t\})$. $\Delta(\Theta)$ is defined as $l(\varphi)^{2s-1} \times \Pi_{j=1}^{s}(size(\theta_j)+1)$.

Lemma 7. Let Θ be a finite set of sequential r.e.'s $\theta_1, \cdots, \theta_s$. For each $n \geq \Delta(\Theta)$, $\{\Psi(\theta_j, n) \mid 1 \leq j \leq s\} \models \varphi$ if, and only if, $\{\Psi(\theta_j, \Delta(\Theta)) \mid 1 \leq j \leq s\} \models \varphi$.

Sketch of proof We prove the stronger statement below by induction on s.

> *Let $m_j = l(\varphi)^{2(s-j)+1} \times \Pi_{i=j+1}^{s}(size(\theta_i)+1)$ and $m'_j \geq m_j$. Then for any model $\mathcal{D}$ of $\wedge_{j=1}^{s}\Psi(\theta_j, m_j)$, $\mathcal{D}$ holds φ if, and only if, each model $\mathcal{D}'$ of $\wedge_{j=1}^{s}\Psi(\theta_j, m'_j)$, which is an extension of $\mathcal{D}$, holds φ.*

For $s = 1$, since the n-times multiplication of the same starred expression is equivalent for $n \geq l(\varphi)$, it is easy. Assume that the statement holds for s.

Let σ be a starred expression in some θ_j. Note that the continuous sequence $\psi(\sigma)$ of lengths greater than $l(\varphi)$ in a model is equivalent to length $l(\varphi)$.

Let $\bar{\mathcal{D}}$ (resp. $\bar{\mathcal{D}}'$) be a model by multiplying each occurrence of $\psi(\sigma)$ for each starred expression σ in $\mathcal{D}$ (resp. $\mathcal{D}'$) $l(\varphi)^2 \times (size(\theta_{s+1})+1)$-times. Thus, $\bar{\mathcal{D}}$ (resp. $\bar{\mathcal{D}}'$) holds $\wedge_{j=1}^{s}\Psi(\theta_j, m_j \times l(\varphi)^2 \times (size(\theta_{s+1})+1))$ (resp. $\wedge_{j=1}^{s}\Psi(\theta_j, m'_j \times l(\varphi)^2 \times (size(\theta_{s+1})+1))$). Let us consider the extension $\bar{\mathcal{D}}^+$ of $\bar{\mathcal{D}}$, which is a model of $\Psi(\theta_{s+1}, l(\varphi))$ and $\wedge_{j=1}^{s} xs\Psi(\theta_j, m_j \times l(\varphi)^2 \times (size(\theta_{s+1})+1))$.

Note that the *fairest* distribution of $\Psi(\theta_{s+1}, l(\varphi))$ in the continuous sequence of $\psi(\sigma)$ has segments of length $l(\varphi)$ between each adjacent elements in $\Psi(\theta_{s+1}, l(\varphi))$.

Thus $\bar{\mathcal{D}}^+$ does not hold φ, if and only if there exists an extension of $\bar{\mathcal{D}}'^+$, which is a model of $\wedge_{j=1}^{s} \Psi(\theta_j, m'_j \times l(\varphi)^2 \times (size(\theta_{s+1}) + 1))$ and $\Psi(\theta_{s+1}, m'_{j+1})$ with $m'_{j+1} \geq l(\varphi)$, such that $\bar{\mathcal{D}}'^+$ does not hold φ. ■

Theorem 5. Let $\bar{\Theta}$ be a finite set of finite sets of sequential r.e.'s. Then,

$$\texttt{ExistsMinor}(\bar{\Theta}) = \vee_{\Theta \in \bar{\Theta}} \texttt{QueryTest}(\{\Psi(\theta, \Delta(\Theta)) \mid \theta \in \Theta\}).$$

Theorem 6. The algorithm (in Figure 1) to detect a set of minors $\mathcal{M}_\varphi$ terminates.

Proof From Theorem 4, for each iteration of `while ExistsMinor(L)`, `L` strictly decreases wrt $\leq_m$, and $\leq_m$ is an WFO. ■

Thus, we can effectively compute a set of minors $\mathcal{M}_\varphi$, and we can give a simple algorithm to decide $D \models \varphi$.

Corollary 3. For a fixed disjunctive monadic query φ, a linear time algorithm to decide whether $D \models \varphi$ for an indefinite database D is as follows:

```
begin
  Flag:= false;
  for each m in M_φ do Flag:= Flag or [m ≤_m D];
  return flag;
end
```

6 Concluding Remarks

This paper described a generation of a linear time query-processing algorithm for a fixed disjunctive monadic query in an indefinite database on a linearly ordered domain. This problem was first posed by Van der Meyden [16] and had, until now, not been published elsewhere. There are several future directions:

1. Our method is based on the regular expression techniques appeared in Murthy-Russell's constructive proof of Higman's lemma [7]. Among its known constructive proofs [7, 11, 2], [2] would be the most simple and is implemented on Coq prover (see `http://coq.inria.fr/contribs/logic-eng.html`). This could be applied to the simpler method to algorithm generation.
2. We are designing an automatic generator based on MONA (available at `http://www.brics.dk/mona`). MONA, which runs on Linux, efficiently decides the satisfiability of formulae in monadic second order logic S1S/S2S. Based on initial experiments, we expect reasonably fast calculations.
3. The next extension may be to use the constructive proof of Kruskal's theorem [8]. Gupta demonstrated the constructive proof of the weaker form [4]. This would correspond to, for instance, query-processing in an indefinite database over partial ordered domains (i.e., events on branching time).

Acknowledgments

The author would like to thank Ronald van der Meyden for his private lectures on indefinite database theory during his stay at NTT. He would also like to thank the members of the Tokyo CACA seminar for their guidance.

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. available at `http://www.md.chalmers.se/~coquand/intuitionism.html`.
3. R.M. Fellows and M.A. Langston. Nonconstructive tools for proving polynomial-time decidability. *Journal of the ACM*, 35(3):727–739, 1988.
4. A. Gupta. A constructive proof that tree 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.
5. G. Higman. Ordering by divisibility in abstract algebras. *Proc. London Mathematical Society*, 2:326–336, 1952.
6. M. Koubarakis. The complexity of query evaluation in indefinite temporal constraint databases. *Theoretical Computer Science*, 171:25–60, 1997.
7. C.R. Murthy and J.R. Russell. A constructive proof of Higman's lemma. In *Proc. 5th IEEE sympo. on Logic in Computer Science*, pages 257–267, 1990.
8. C.ST.J.A. Nash-Williams. On well-quasi-ordering finite trees. *Proc. Cambridge Phil. Soc.*, 59:833–835, 1963.
9. N.Dershowitz and Z.Manna. Proving termination with multiset orderings. *Communications of the ACM*, 22(8):465–476, 1979.
10. L. Perković and B. Reed. An improved algorithm for finding tree decompositions of small width. In et.al. Widmayer, editor, *WG'99*, pages 148–154, 1999. Lecture Notes in Computer Science, Vol. 1665, Springer-Verlag.
11. F. Richman and G. Stolzenberg. Well quasi-ordered sets. *Advances in Mathematics*, 97:145–153, 1993.
12. N. Robertson and P.D. Seymour. Graph minors XX. Wagner's conjecture, 1988. Manuscript.
13. N. Robertson and P.D. Seymour. Graph minors XIII. the disjoint path problem. *Journal of Combinatorial Theory Series B*, 63:65–110, 1995.
14. S.G. Simpson. Ordinal numbers and the Hilbert basis theorem. *Journal of Symbolic Logic*, 53(3):961–974, 1988.
15. 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.
16. 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 11th ACM sympo. on Principles of Database Systems, pp.331–345, 1992.