On First-Order Query Rewriting for Incomplete Database Histories [*]

Véronique Bruyère Alexandre Decan Jef Wijsen
Université de Mons, Belgium

Abstract

Multiwords are defined as words in which single symbols can be replaced by nonempty sets of symbols. Such a set of symbols captures uncertainty about the exact symbol. Words are obtained from multiwords by selecting a single symbol from every set. A pattern is certain *in a multiword W if it occurs in every word that can be obtained from W. For a given pattern, we are interested in finding a logic formula that recognizes the multiwords in which that pattern is certain.*

This problem can be seen as a special case of consistent query answering (CQA). We show how our results can be applied in CQA on database histories under primary key constraints.

1 Motivation

An *incomplete database* DB is a set of *possible* databases. Under the *certain answer* semantics, a Boolean query q evaluates to true on such an incomplete database DB if q evaluates to true on every possible database in DB; otherwise q evaluates to false.

A database that violates primary key constraints naturally gives rise to an incomplete database: every possible database is obtained by selecting a maximal number of tuples from each relation without ever selecting two distinct tuples that agree on their primary key. In this setting, possible databases are called *repairs*, as they are obtained by "repairing" constraint violations [1].

Recently, progress has been made in computing certain answers to conjunctive queries under primary key constraints [3, 5, 10, 11]. This article makes a first step in extending these works to database histories.

Assume a discrete, linear time scale. The following database history shows the WorksFor relation at four successive time points t_0, t_1, t_2, t_3. The primary key Name is violated at times t_1 and t_2.

t_0

<u>Name</u>	Company
Ed	IBM

t_1

<u>Name</u>	Company
Ed	MS
Ed	IBM

t_2

<u>Name</u>	Company
Ed	MS
Ed	IBM

t_3

<u>Name</u>	Company
Ed	MS

To make this history consistent, we have to delete one of the two tuples at t_1, and one of the two tuples at t_2. Thus, there are four repairs for this database history; one such repair is shown next.

t_0

<u>Name</u>	Company
Ed	IBM

t_1

<u>Name</u>	Company
Ed	IBM

t_2

<u>Name</u>	Company
Ed	IBM

t_3

<u>Name</u>	Company
Ed	MS

The query *"Did MS recruit an IBM employee?"* is expressed by the following formula in first-order temporal logic (FOTL):

$$q_0 \quad = \quad \exists x(\mathsf{WorksFor}(\underline{x}, \mathrm{IBM}) \land \bigcirc \mathsf{WorksFor}(\underline{x}, \mathrm{MS}))$$

The primary key arguments are underlined. It can be easily checked that q_0 evaluates to true on each of the four repairs.

We are interested in the following problem: find a FOTL query q_0' such that for every (possibly inconsistent) database history HIS, q_0' evaluates to true on HIS if and only if q_0 evaluates to true on every repair of HIS. Such a query q_0', if it exists, is called a *consistent FOTL-rewriting* of q_0. The interest of consistent FOTL-rewritings should be clear: they provide certain answers on inconsistent databases, without the need for computing repairs. Moreover, such rewritings may be amenable to implementation in practical database languages like (temporal extensions of) SQL.

It can be verified that the following query q_0' is a consistent FOTL-rewriting of q_0:

$$q_0' \quad = \quad \exists x(\varphi_{\mathrm{IBM}}(x) \land \bigcirc(\varphi_{\{\mathrm{IBM},\mathsf{MS}\}}(x) \ \mathbf{until} \ \varphi_{\mathsf{MS}}(x)))$$

[*]A previous version of this article has been presented at the Workshop on Logic in Databases (LID 2008, `http://conferenze.dei.polimi.it/lid2008/`). This workshop had informal proceedings that were distributed only to the participants.

where:

$$\begin{aligned}\varphi_{\text{IBM}}(x) &= \exists y(\mathsf{WorksFor}(\underline{x}, y)\wedge\\ &\quad \forall y(\mathsf{WorksFor}(\underline{x}, y) \rightarrow y = \text{IBM}))\\ \varphi_{\mathsf{MS}}(x) &= \exists y(\mathsf{WorksFor}(\underline{x}, y)\wedge\\ &\quad \forall y(\mathsf{WorksFor}(\underline{x}, y) \rightarrow y = \text{MS}))\\ \varphi_{\{\text{IBM},\mathsf{MS}\}}(x) &= \exists y(\mathsf{WorksFor}(\underline{x}, y)\wedge\\ &\quad \forall y(\mathsf{WorksFor}(\underline{x}, y) \rightarrow \quad y = \text{IBM}\\ &\qquad\qquad\qquad\qquad \vee y = \text{MS}\ \ ))\end{aligned}$$

Intuitively, it is certain that some employee x moved directly from IBM to MS if in the (possibly inconsistent) database history, there is one state where x worked certainly for IBM, some later state where x certainly worked for MS, and between these two states, x was employed by either IBM or MS.

Clearly, on databases that satisfy the primary key Name, q_0 is equivalent to the following query $\tilde{q}_0$:

$$q_0 \equiv \tilde{q}_0 \quad = \quad \exists x(\varphi_{\text{IBM}}(x) \wedge \bigcirc\varphi_{\mathsf{MS}}(x))$$

The queries $\varphi_{\text{IBM}}(x)$, $\varphi_{\{\text{IBM},\mathsf{MS}\}}(x)$, and $\varphi_{\mathsf{MS}}(x)$ can be obtained using recent results on first-order query rewriting under primary keys [3, 10, 11]. In this article, the focus is on the rewriting from $\tilde{q}_0$ into q_0'. To study this rewriting, we use the abstraction explained in the next paragraph.

Assume a finite alphabet Σ. A *multiword* is a word over the *powerset alphabet* $2^{\Sigma} \setminus \{\emptyset\}$. For the moment, take $\Sigma = \{a, b\}$, so the powerset alphabet is the three-letter alphabet $\{\{a\}, \{b\}, \{a, b\}\}$. Every multiword W gives rise to a set of possible words (or repairs) obtained from W by selecting one representative from each set. For example, the multiword $W_0 = \langle\{a\}, \{a, b\}, \{a, b\}, \{b\}\rangle$ has four possible words: $aaab$, $aabb$, $abab$, and $abbb$. A word w is *certain* in a multiword if it is a factor of every possible word. For example, ab is certain in W_0, but aa is not (because aa is not a factor of $abbb$). Given a word w, let $\mathsf{CERTAIN}(w)$ denote the set of multiwords in which w is certain. We are interested in the following problem: given a word w, is $\mathsf{CERTAIN}(w)$ FO-definable?

The example of the preceding paragraph is our propositional abstraction of the WorksFor example introduced before. Think of a as IBM, and b as MS. The word ab corresponds to the temporal formula $\psi_0 = a \wedge \bigcirc b$, which abstracts $\tilde{q}_0$. Let W be a multiword over the powerset alphabet $\{\{a\}, \{b\}, \{a, b\}\}$. Then, ψ_0 evaluates to true on every possible word of W if and only if W satisfies $\psi_0' = \{a\}\wedge\bigcirc(\{a, b\}\ \mathbf{until}\ \{b\})$; the latter formula abstracts q_0'.

The article is organized as follows. Section 2 recalls some fundamental results about logics over words. Section 3 formalizes the problem we are interested in. Section 4 gives a straightforward algorithm for deciding $\mathsf{CERTAIN}(w)$ if w contains no variables. The algorithm is interesting because it immediately leads to the result that $\mathsf{CERTAIN}(w)$ is regular, as shown in Section 5. That section also provides sufficient conditions for FO-definability of $\mathsf{CERTAIN}(w)$. Section 6 deals with words that contain variables, which are placeholders for symbols of Σ. For example, axx is certain in W_0 if either aaa or abb is certain in W_0. Finally, Section 7 concludes the article.

2 Related Work

Multiwords can be seen as classical words over some powerset alphabet. Some fundamental results on standard words will be used in this article:

- Over words, linear temporal logic has the same expressive power as FO [4, 6].
- A regular language is FO-definable if and only if it is aperiodic [8, 9].
- A language is definable in MSO (monadic second-order logic) if and only if it is regular (Büchi's theorem).

Recall that a regular language L is aperiodic if there exists an integer $k \geq 0$ such that for all words p, u, q,

$$p \cdot u^k \cdot q \in L \text{ if and only if } p \cdot u^{k+1} \cdot q \in L.$$

Multiwords can be seen as a syntactic generalization of *partial words*, which have been extensively studied by Blanchet-Sadri and others (a recent article is [2]). Partial words are strings over a finite alphabet that may contain one or more occurrences of the "do not know" symbol $\Diamond$. A partial word could be encoded as a multiword, by replacing each symbol a with $\{a\}$, and $\Diamond$ with the entire alphabet. For example, over the alphabet $\{a, b, c\}$, the partial word $a\Diamond b$ is encoded by the multiword $\{a\} \cdot \{a, b, c\} \cdot \{b\}$. However, the problems studied for partial words are quite different from the problems studied in this article, and semantics diverge.

3 Problem Statement

We assume a finite alphabet $\Sigma = \{a, b, c, \ldots\}$ of *symbols*.

Definition 1 A *word* of *length* $n \geq 0$ is a sequence $a_1 a_2 \ldots a_n$ of symbols. The length of a word w is denoted by $|w|$. The *empty word* has length 0 and is denoted by ϵ. The *concatenation* of words v and w is denoted by $v \cdot w$. The concatenation operator naturally extends to sets S, T of words: $S \cdot T = \{v \cdot w \mid v \in S, w \in T\}$. A word v is a *factor* of w, denoted $w \Vdash v$, if there exist (possibly empty) words p and q such that $w = p \cdot v \cdot q$. $\lhd$

Multiwords capture the notion of inconsistency and are defined next.

Definition 2 A *multiword* (over Σ) is a sequence $W = \langle A_1, \ldots, A_n \rangle$ where for each $i \in \{1, \ldots, n\}$, $A_i \subseteq \Sigma$ and $A_i \neq \emptyset$. Thus, a multiword can be conceived as a word (in the sense of Def. 1) relative to the alphabet $2^\Sigma \setminus \{\emptyset\}$, called *powerset alphabet*..

For the multiword $W = \langle A_1, \ldots, A_n \rangle$, we define:

$$\mathsf{words}(W) = \{a_1 a_2 \ldots a_n \mid \forall i \in \{1, \ldots, n\} : a_i \in A_i\} \ .$$

If v is a word, then we define:

$$W \Vdash_{\mathsf{certain}} v \text{ iff for every } w \in \mathsf{words}(W),\ w \Vdash v.$$

We write $\mathcal{M}_\Sigma$ for the set of all multiwords (over Σ). $\lhd$

For example, the following multiword W_0 contains two uncertain positions with values $\{a, b\}$ and $\{c, d\}$. Curly braces are omitted at positions that are certain; for example, $\{a\}$ is written as a.

$$W_0 = \langle a, b, d, a, b, c, a, \{a, b\}, b, d, a, b, \{c, d\}, a, b, c, a, b \rangle$$

We have

$$\begin{aligned} \mathsf{words}(W_0) \ = \{ \ & abdabca\underline{abdabcab}cab \\ & abdabcaabd\underline{abdabcab} \\ & \underline{abdabcab}bdabcabcab \\ & \underline{abdabcab}bd\underline{abdabcab} \ \} \ . \end{aligned}$$

The underlined positions show that $abdabcab$ is a factor of each word in $\mathsf{words}(W_0)$. Hence, $W_0 \Vdash_{\mathsf{certain}} abdabcab$.

For a word w, we are interested in (the complexity of) the following language:

$$\mathsf{CERTAIN}(w) := \{W \in \mathcal{M}_\Sigma \mid W \Vdash_{\mathsf{certain}} w\} \ .$$

In particular, we want to answer the question:

Given w, is $\mathsf{CERTAIN}(w)$ FO-definable?

That is, is there a FO formula φ_w such that

$$\mathsf{CERTAIN}(w) = \{W \in \mathcal{M}_\Sigma \mid W \models \varphi_w\} \ ,$$

where the logic formalism is FO over words [7, p. 124]?

For example, for the alphabet $\Sigma = \{a, b\}$, multiwords can be regarded as words with symbols taken in the alphabet $\{\{a\}, \{b\}, \{a, b\}\}$. This gives rise to three predicate symbols $P_{\{a\}}$, $P_{\{b\}}$, and $P_{\{a,b\}}$. For instance, the multiword $W = \langle \{a\}, \{a, b\}, \{a, b\}, \{b\} \rangle$ of length 4 is represented by the first-order structure $(\{1, 2, 3, 4\}, <, P_{\{a\}}, P_{\{b\}}, P_{\{a,b\}})$ where $<$ is the natural order and each P_A contains the positions in W at which A occurs: $P_{\{a\}} = \{1\}$, $P_{\{b\}} = \{4\}$, $P_{\{a,b\}} = \{2, 3\}$.

The following formula defines all the multiwords that contain a subsequence among $\langle \{a\}, \{b\} \rangle$, $\langle \{a\}, \{a, b\}, \{b\} \rangle$, $\langle \{a\}, \{a, b\}, \{a, b\}, \{b\} \rangle$,...

$$\begin{aligned} &\exists x \exists y (\ x < y \wedge P_{\{a\}}(x) \wedge P_{\{b\}}(y) \wedge \\ &\qquad \forall z ((x < z \wedge z < y) \rightarrow P_{\{a,b\}}(z))) \end{aligned}$$

It can be verified that this formula defines the language $\mathsf{CERTAIN}(ab)$. Recall that there exists an equivalent formula in linear time logic (LTL), since over words, LTL and FO have the same expressive power.

4 Deciding $\mathsf{CERTAIN}(w)$

Given w, we give a procedure for deciding membership of $\mathsf{CERTAIN}(w)$. The procedure uses a construction defined in the statement of Lemma 2 and its correctness is shown in Theorem 1. The decision procedure is interesting because it can be used to show that $\mathsf{CERTAIN}(w)$ is regular (see Section 5).

Definition 3 Let w be a word. For all (possibly empty) words p and q, if $w = p \cdot q$, then p is a *prefix* of w, and q a *suffix*. If u is a word, then $\mathsf{sufpre}(u, w)$ denotes the maximal suffix of u that is a prefix of w. For a set S of words, we define $\mathsf{sufpre}(S, w) = \{\mathsf{sufpre}(u, w) \mid u \in S\}$. $\lhd$

Example 1 $\mathsf{sufpre}(abcd, cde) = cd$ and $\mathsf{sufpre}(ab, c) = \epsilon$. Clearly, $\mathsf{sufpre}(w, w) = w$.

Lemma 1 *If* $\mathsf{sufpre}(u, w) = q$, *then* $\mathsf{sufpre}(u \cdot a, w) = \mathsf{sufpre}(q \cdot a, w)$.

PROOF. Assume $\mathsf{sufpre}(u, w) = q$. We can assume a (possibly empty) word u' such that:

- $u = u' \cdot q$,
- q is a prefix of w, and
- *Maximality:* for every suffix $s \neq \epsilon$ of u', $s \cdot q$ is not a prefix of w.

We need to show $\mathsf{sufpre}(u' \cdot q \cdot a, w) = \mathsf{sufpre}(q \cdot a, w)$. Obviously, $\mathsf{sufpre}(q \cdot a, w)$ is a suffix of $\mathsf{sufpre}(u' \cdot q \cdot a, w)$. Assume $|\mathsf{sufpre}(u' \cdot q \cdot a, w)| > |\mathsf{sufpre}(q \cdot a, w)|$. Then, there exists a suffix $s \neq \epsilon$ of u' such that $s \cdot q \cdot a$ is a prefix of w, contradicting the above *Maximality* condition. We conclude by contradiction $\mathsf{sufpre}(u' \cdot q \cdot a, w) = \mathsf{sufpre}(q \cdot a, w)$. $\lhd$

The construction for deciding membership of $\mathsf{CERTAIN}(w)$ is described in the statement of Lemma 2. For a given multiword $W = \langle A_1, \ldots, A_n \rangle$, we construct a sequence $\langle S_0, \ldots, S_n \rangle$ such that $S_0 = \{\epsilon\}$ and for each subsequent S_i, $u \in S_i$ if and only if $u \neq w$ and $u = \mathsf{sufpre}(p \cdot a, w)$ for some $p \in S_{i-1}$ and $a \in A_i$. By Theorem 1, $W \in \mathsf{CERTAIN}(w)$ if and only if the last element S_n is the empty set.

The construction is illustrated next for the multiword

$$W_0 = \langle a, b, d, a, b, c, a, \{a, b\}, b, d, a, b, \{c, d\}, a, b, c, a, b \rangle$$

shown earlier (see equation (3) on page 3) and the word $w = abdabcab$.

	$S_0 = \{\epsilon\}$
$A_1 = \{a\}$	$S_1 = \{a\}$
$A_2 = \{b\}$	$S_2 = \{ab\}$
$A_3 = \{d\}$	$S_3 = \{abd\}$
$A_4 = \{a\}$	$S_4 = \{abda\}$
$A_5 = \{b\}$	$S_5 = \{abdab\}$
$A_6 = \{c\}$	$S_6 = \{abdabc\}$
$A_7 = \{a\}$	$S_7 = \{abdabca\}$
$A_8 = \{a, b\}$	$S_8 = \{a\}$
$A_9 = \{b\}$	$S_9 = \{ab\}$
$A_{10} = \{d\}$	$S_{10} = \{abd\}$
$A_{11} = \{a\}$	$S_{11} = \{abda\}$
$A_{12} = \{b\}$	$S_{12} = \{abdab\}$
$A_{13} = \{c, d\}$	$S_{13} = \{abdabc, abd\}$
$A_{14} = \{a\}$	$S_{14} = \{abdabca, abda\}$
$A_{15} = \{b\}$	$S_{15} = \{abdab\}$
$A_{16} = \{c\}$	$S_{16} = \{abdabc\}$
$A_{17} = \{a\}$	$S_{17} = \{abdabca\}$
$A_{18} = \{b\}$	$S_{18} = \{\}$

Lemma 2 *Let $W = \langle A_1, \ldots, A_n \rangle$ be a multiword. Let w be a nonempty word. Let $\langle S_0, S_1, \ldots, S_n \rangle$ be a sequence such that $S_0 = \{\epsilon\}$ and for every $i \in \{1, \ldots, n\}$, $S_i = \mathsf{sufpre}(S_{i-1} \cdot A_i, w) \setminus \{w\}$.*

Let $m \in \{1, 2, \ldots, n\}$. For every word $u \in \mathsf{words}(\langle A_1, \ldots, A_m \rangle)$, either $\mathsf{sufpre}(u, w) \in S_m$ or $u \Vdash w$.

PROOF. Proof by induction on m.

Basis $m = 1$. Let $u = b \in \mathsf{words}(\langle A_1 \rangle)$ such that $b \nVdash w$ (i.e. $w \neq b$). Then, S_1 contains $\mathsf{sufpre}(\epsilon \cdot b, w) = \mathsf{sufpre}(b, w)$.

Step. Assume w.l.o.g. $S_m = \{p_1, \ldots, p_k\}$ and $A_{m+1} = \{a_1, \ldots, a_l\}$. Then, $S_{m+1} = \{\mathsf{sufpre}(p_1 \cdot a_1, w), \ldots, \mathsf{sufpre}(p_i \cdot a_j, w), \ldots, \mathsf{sufpre}(p_k \cdot a_l, w)\} \setminus \{w\}$. Let $u \in \mathsf{words}(\langle A_1, \ldots, A_{m+1} \rangle)$. We can assume $v \in \mathsf{words}(\langle A_1, \ldots, A_m \rangle)$ and $j \in \{1, \ldots, l\}$ such that $u = v \cdot a_j$. Two cases can occur:

- $v \Vdash w$. Obviously, $u \Vdash w$.
- $v \nVdash w$. By the induction hypothesis, $\mathsf{sufpre}(v, w) \in S_m$. We can assume w.l.o.g. $i \in \{1, \ldots, k\}$ such that $p_i = \mathsf{sufpre}(v, w)$. By Lemma 1, $\mathsf{sufpre}(u, w) = \mathsf{sufpre}(v \cdot a_j, w) = \mathsf{sufpre}(p_i \cdot a_j, w)$. If $\mathsf{sufpre}(p_i \cdot a_j, w) = w$, then $u \Vdash w$; otherwise S_{m+1} contains $\mathsf{sufpre}(u, w)$.

This concludes the proof. $\lhd$

Lemma 3 *Let $W = \langle A_1, \ldots, A_n \rangle$, w, and $\langle S_0, S_1, \ldots, S_n \rangle$ be as in Lemma 2.*

Let $i \in \{1, \ldots, n\}$. For every $p \in S_i$, there exists $u \in \mathsf{words}(\langle A_1, \ldots, A_i \rangle)$ such that $u \nVdash w$ and $\mathsf{sufpre}(u, w) = p$.

PROOF. Proof by induction on i.

Basis $i = 1$. Easy.

Step. Assume $p \in S_{i+1}$. We can assume $q \in S_i$ and $a \in A_{i+1}$ such that $p = q \cdot a$. By the induction hypothesis, there exists $u \in \mathsf{words}(\langle A_1, \ldots, A_i \rangle)$ such that $u \nVdash w$ and $\mathsf{sufpre}(u, w) = q$. Then, $u \cdot a \in \mathsf{words}(\langle A_1, \ldots, A_{i+1} \rangle)$. It suffices to show that $u \cdot a \nVdash w$ and that $\mathsf{sufpre}(u \cdot a, w) = p$.

- Assume $u \cdot a \Vdash w$. Then necessarily, $w = q \cdot a$. But then $p = w$, a contradiction. We conclude by contradiction that $u \cdot a \nVdash w$.
- By Lemma 1, $\mathsf{sufpre}(u \cdot a, w) = \mathsf{sufpre}(q \cdot a, w) = \mathsf{sufpre}(p, w)$. Since p is a prefix of w, $\mathsf{sufpre}(p, w) = p$.

This concludes the proof. $\lhd$

Theorem 1 *Let $W = \langle A_1, \ldots, A_n \rangle$, w, and $\langle S_0, S_1, \ldots, S_n \rangle$ be as in Lemma 2. Then, $W \in \mathsf{CERTAIN}(w)$ if and only if $S_n = \{\}$.*

PROOF. *If part.* Assume $S_n = \{\}$. Lemma 2 implies that $W \in \mathsf{CERTAIN}(w)$.
Only-if part. Assume $S_n \neq \{\}$. Lemma 3 implies that $W \notin \mathsf{CERTAIN}(w)$. $\lhd$

As a small optimization, it is not hard to verify that Theorem 1 remains valid if every S_i that contains ϵ is replaced by $\{\epsilon\}$. Formally, define:

$$\lfloor S_i \rfloor = \begin{cases} \{\epsilon\} & \text{if } \epsilon \in S_i \ ; \\ S_i & \text{otherwise.} \end{cases}$$

Then, the sequence $\langle S_0, \ldots, S_n \rangle$ defined by $S_0 = \{\epsilon\}$ and for each $i \in \{1, \ldots, n\}$, $S_i = \lfloor \mathsf{sufpre}(S_{i-1} \cdot A_i, w) \setminus \{w\} \rfloor$ satisfies: $W \in \mathsf{CERTAIN}(w)$ if and only if $S_n = \{\}$.

5 Defining $\mathsf{CERTAIN}(w)$ in FO

Assume a nonempty word w over alphabet Σ. We show that $\mathsf{CERTAIN}(w)$ is regular and, under certain conditions, aperiodic. Theorem 1 suggests the following construction of a deterministic finite automaton that accepts $\mathsf{CERTAIN}(w)$.

Let P' be the set of prefixes of w, and let $P = P' \setminus \{\epsilon, w\}$. We define $\mathsf{DFA}_\Sigma(w)$ as the following deterministic finite automaton $(Q, \Omega, S_0, F, \delta)$:

1. The finite set Q of states is $2^P \cup \{\{\epsilon\}\}$. Note that $\{\epsilon\}$ is the only state containing ϵ.
2. The alphabet Ω is the powerset alphabet $2^\Sigma \setminus \{\emptyset\}$.
3. The start state is $S_0 = \{\epsilon\}$ and $F = \{\emptyset\}$ is the set of accepting states.
4. The transition function $\delta : Q \times \Omega \to Q$ is defined by:

$$\delta(S, A) = \lfloor \mathsf{sufpre}(S \cdot A, w) \setminus \{w\} \rfloor \ .$$

For example, Fig. 1 shows $\mathsf{DFA}_\Sigma(abb)$ for the alphabet $\Sigma = \{a, b\}$. Theorem 1 and the construction of $\mathsf{DFA}_\Sigma(\cdot)$ immediately lead to the following result.

Theorem 2 *Let w be a nonempty word and $W = \langle A_1, \dots, A_n\rangle$ a multiword (both relative to the alphabet Σ). Then, $W \in \mathsf{CERTAIN}(w)$ if and only if $\mathsf{DFA}_\Sigma(w)$ accepts W. Hence, $\mathsf{CERTAIN}(w)$ is regular.*

A regular language is FO-definable if and only if it is aperiodic [8, 9]. Since our driving motivation is consistent FO-rewriting, we are interested in aperiodicity results for $\mathsf{CERTAIN}(w)$. The following theorem shows that $\mathsf{CERTAIN}(w)$ is aperiodic if the first (or the last) symbol of w occurs only once in w. Our proof technique explicitly relies on the condition that the first (or the last) symbol of w is unique. This uniqueness condition is not necessary for FO-definability, however; for example, $\mathsf{CERTAIN}(aa)$ is FO-definable. It is an open conjecture that $\mathsf{CERTAIN}(w)$ is aperiodic for any word w.

Theorem 3

1. *If the last symbol of w occurs only once in w, then $\mathsf{CERTAIN}(w)$ is aperiodic.*
2. *If the first symbol of w occurs only once in w, then $\mathsf{CERTAIN}(w)$ is aperiodic.*

PROOF. We prove the first item; the proof of the second item is symmetrical. Let $w = w' \cdot a$ where a does not occur in w'. Let $k = |w|$. It suffices to show that for all multiwords P, U, Q:

$$P \cdot U^{k+1} \cdot Q \in \mathsf{CERTAIN}(w) \text{ if and only if } P \cdot U^{k+2} \cdot Q \in \mathsf{CERTAIN}(w).$$

That is, for all multiwords P, U, Q:

$$P \cdot U^{k+1} \cdot Q \not\Vdash_{\mathsf{certain}} w \iff P \cdot U^{k+2} \cdot Q \not\Vdash_{\mathsf{certain}} w \ .$$

Let $U = \langle A_1, \dots, A_n\rangle$. The proof is obvious if $n = 0$. Next assume $n > 0$.

$\boxed{\Rightarrow}$ Assume $P \cdot U^{k+1} \cdot Q \not\Vdash_{\mathsf{certain}} w$. We can assume the existence of $p \in \mathsf{words}(P)$, $u_1, \dots, u_{k+1} \in \mathsf{words}(U)$, and $q \in \mathsf{words}(Q)$ such that the word

$$m = p \cdot u_1 \cdot u_2 \cdot \dots \cdot u_{k+1} \cdot q$$

satisfies $m \not\Vdash w$. Two cases can occur:

1. The symbol a does not occur in $u_2 \cdot \dots \cdot u_{k+1}$. Let m' be the word obtained from m by "duplicating" the factor u_2 as follows:

$$m' \quad = \quad \overbrace{p \cdot u_1 \cdot u_2}^{s_1} \cdot \overbrace{\underbrace{u_2 \cdot \dots \cdot u_{k+1}}_{t_2} \cdot q}^{t_1}$$

   Assume $m' \Vdash w$. Since $m \not\Vdash w$, any factor w of m' must "start in" s_1 and "end in" t_1. Since $|w| = k$, w must actually end in t_2. But then a must occur in $u_2 \cdot \dots \cdot u_{k+1}$, a contradiction. We conclude by contradiction that $m' \not\Vdash w$. Since $m' \in \mathsf{words}(P \cdot U^{k+2} \cdot Q)$, $P \cdot U^{k+2} \cdot Q \not\Vdash_{\mathsf{certain}} w$.

2. a occurs in $u_2 \cdot \dots \cdot u_{k+1}$. Then, we can assume a word v with $|v \cdot a| = |U|$ and words o and r such that:

$$m \quad = \quad p \cdot o \cdot v \cdot a \cdot r \cdot q$$

   Let m' be the word obtained from m by "duplicating" the factor $v \cdot a$ as follows:

$$m' \quad = \quad \overbrace{p \cdot o \cdot v \cdot a}^{s_3} \cdot \overbrace{v \cdot a \cdot r \cdot q}^{t_3}$$

   Assume $m' \Vdash w$. Since $m \not\Vdash w$, any factor w of m' must start in s_3 and end in t_3. But then w contains two occurrences of a, a contradiction. We conclude by contradiction that $m' \not\Vdash w$. It is easy to verify that $m' \in \mathsf{words}(P \cdot U^{k+2} \cdot Q)$. It follows $P \cdot U^{k+2} \cdot Q \not\Vdash_{\mathsf{certain}} w$.

$\boxed{\Leftarrow}$ Assume $P \cdot U^{k+2} \cdot Q \not\Vdash_{\mathsf{certain}} w$. We can assume $p \in \mathsf{words}(P)$, $u_1, \dots, u_{k+2} \in \mathsf{words}(U)$, and $q \in \mathsf{words}(Q)$ such that the word

$$m = p \cdot u_1 \cdot \dots \cdot u_{k+1} \cdot u_{k+2} \cdot q$$

satisfies $m \not\Vdash w$. Let m' be the word obtained from m by "omitting" the factor u_1 as follows:

$$m' \quad = \quad \overbrace{p}^{s_4} \cdot \overbrace{\underbrace{u_2 \cdot \dots \cdot u_k}_{t_5} \cdot u_{k+1} \cdot u_{k+2} \cdot q}^{t_4}$$

Clearly, $m' \in \mathsf{words}(P \cdot U^{k+1} \cdot Q)$. Two cases can occur:

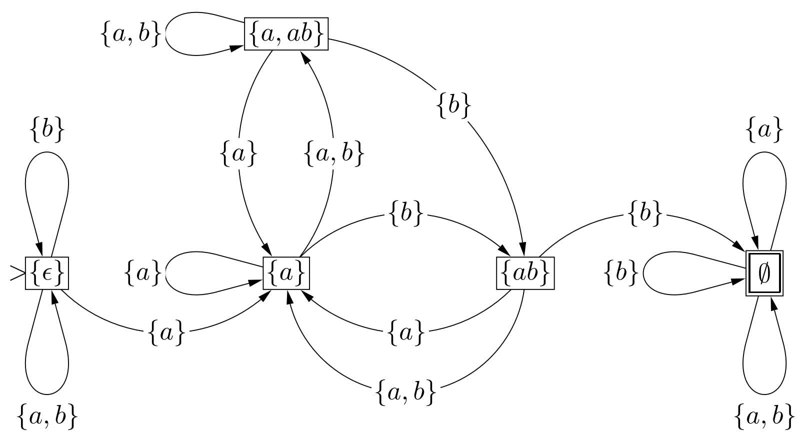

Figure 1. $\mathsf{DFA}_\Sigma(abb)$ **with** $\Sigma = \{a,b\}$**.**

1. $m' \not\Vdash w$. Then, $P \cdot U^{k+1} \cdot Q \not\Vdash_{\mathsf{certain}} w$.

2. $m' \Vdash w$. Since $m \not\Vdash w$, any factor w of m' must start in s_4 and end in t_4. Since $|w| = k$, w must actually end in t_5. Then a must occur in $u_2 \cdot \ldots \cdot u_k$. We can assume a word v with $|v| = |U|$ and words o and r such that:

$$m = p \cdot u_1 \cdot o \cdot a \cdot v \cdot r \cdot q$$

Let m'' be the word obtained from m by "omitting" the factor v as follows:

$$m'' = \overbrace{p \cdot u_1 \cdot o \cdot a}^{s_6} \cdot \overbrace{r \cdot q}^{t_6}$$

Assume $m'' \Vdash w$. Since $m \not\Vdash w$, the factor w of m'' must start in s_6 and end in t_6. But then w contains two occurrences of a, a contradiction. We conclude by contradiction that $m'' \not\Vdash w$. It is easy to verify that $m'' \in \mathsf{words}(P \cdot U^{k+1} \cdot Q)$. Consequently, $P \cdot U^{k+1} \cdot Q \not\Vdash_{\mathsf{certain}} w$.

This concludes the proof. $\lhd$

Since aperiodic regular languages are expressible in FO [8, 9], we have the following result.

Corollary 1 *If the symbol a does not occur in w, then* $\mathsf{CERTAIN}(a \cdot w)$ *and* $\mathsf{CERTAIN}(w \cdot a)$ *are FO-definable.*

6 Words with Variables

We define v-words as words in which variables can replace symbols. These variables are placeholders for symbols. For example, the v-word xax represents any word of length 3 in which the first and the last symbol are equal, and the second symbol is a.

We assume a countable set $\mathbf{vars} = \{x, y, z, x_1, y_1, z_1, \ldots\}$ of *variables* disjoint from the alphabet Σ.

Definition 4 A *v-word* is a sequence $s_1 s_2 \ldots s_n$ where for each $i \in \{1, \ldots, n\}$, $s_i \in \Sigma \cup \mathbf{vars}$. The notions of length and concatenation (see Def. 1) naturally extend to v-words.

The relation $\Vdash$ is extended as follows. A *valuation* is a total function $\theta : \mathbf{vars} \cup \Sigma \to \Sigma$ that is the identity on Σ. For the v-word $v = s_1 s_2 \ldots s_n$, we define $\theta(v) = \theta(s_1) \cdot \theta(s_2) \cdot \ldots \cdot \theta(s_n)$. Note that if v contains no variables (i.e. v is a word), then $\theta(v) = v$. If w is a word and v a v-word, then we define:

$$w \Vdash v \text{ iff for some valuation } \theta, \theta(v) \text{ is a factor of } w.$$

The relation $\Vdash_{\mathsf{certain}}$ directly carries over to v-words: if W is a multiword and v a v-word, then $W \Vdash_{\mathsf{certain}} v$ if and only if for every $w \in \mathsf{words}(W)$, $w \Vdash v$. $\lhd$

The problem statement in Section 3 carries over to v-words. For example, $\mathsf{CERTAIN}(xx)$ contains some multiword W if each $w \in \mathsf{words}(W)$ contains two successive occurrences of the same symbol. Assume a binary alphabet $\Sigma = \{a, b\}$. Clearly, $\mathsf{CERTAIN}(aa) \cup \mathsf{CERTAIN}(bb) \subseteq \mathsf{CERTAIN}(xx)$, and the inclusion is strict: for $W_1 = \langle a, \{a,b\}, b\rangle$, we have $W_1 \notin \mathsf{CERTAIN}(aa) \cup \mathsf{CERTAIN}(bb)$ and $W_1 \in \mathsf{CERTAIN}(xx)$.

Theorem 2 stated that $\mathsf{CERTAIN}(w)$ is regular if w is variable-free. We now show that $\mathsf{CERTAIN}(w)$ remains regular if w contains variables. Notice from the example in the previous paragraph that an automaton for $\mathsf{CERTAIN}(xx)$ is *not* simply the union of the automata for $\mathsf{CERTAIN}(aa)$ and $\mathsf{CERTAIN}(bb)$. Rather than taking an

$$\forall X_a \forall X_b \left(\left(\begin{array}{ll} & \forall y(X_a(y) \lor X_b(y)) \\ \land & \neg\exists y(X_a(y) \land X_b(y)) \\ \land & \forall y(X_a(y) \rightarrow P_{\{a\}}(y) \lor P_{\{a,b\}}(y)) \\ \land & \forall y(X_b(y) \rightarrow P_{\{b\}}(y) \lor P_{\{a,b\}}(y)) \end{array} \right) \rightarrow \exists z_1 \exists z_2 \left(\begin{array}{ll} & S(z_1,z_2) \\ \land & X_a(z_1) \\ \land & X_b(z_2) \end{array} \right) \right)$$

Table 1. MSO definition of $\mathsf{CERTAIN}(ab)$ **for alphabet** $\Sigma = \{a,b\}$**.**

$$\forall X_a \forall X_b \left(\left(\begin{array}{ll} & \forall y(X_a(y) \lor X_b(y)) \\ \land & \neg\exists y(X_a(y) \land X_b(y)) \\ \land & \forall y(X_a(y) \rightarrow P_{\{a\}}(y) \lor P_{\{a,b\}}(y)) \\ \land & \forall y(X_b(y) \rightarrow P_{\{b\}}(y) \lor P_{\{a,b\}}(y)) \end{array} \right) \rightarrow \exists z_1 \exists z_2 \left(\begin{array}{ll} & S(z_1,z_2) \\ \land & X_a(z_1) \leftrightarrow X_a(z_2) \\ \land & X_b(z_1) \leftrightarrow X_b(z_2) \end{array} \right) \right)$$

Table 2. MSO definition of $\mathsf{CERTAIN}(xx)$ **for alphabet** $\Sigma = \{a,b\}$**.**

automaton approach as in Section 5, we will show that for any v-word v, $\mathsf{CERTAIN}(v)$ can be defined by a formula in monadic second-order logic (MSO). Regularity then follows by Büchi's theorem.

We first look at MSO formulas defining $\mathsf{CERTAIN}(w)$ if w is variable-free. We know from Theorem 2 and Büchi's theorem that such formula exists (there exists even an $\exists$MSO definition since MSO=$\exists$MSO over words). Table 1 shows an MSO formula (actually a $\forall$MSO formula) defining $\mathsf{CERTAIN}(ab)$ when the alphabet is $\Sigma = \{a,b\}$. That is, for any multiword W, $W \Vdash_{\mathsf{certain}} ab$ if and only if W satisfies the MSO formula of Table 1. The monadic second-order variables X_a and X_b encode all possible words of W: if $X_a(y)$ is true, then a is chosen at position y, and if $X_b(y)$ is true, then b is chosen at position y. The premise states that for each position in multiword W, either a or b has to be chosen; a can only be chosen at position y if the multiword W contains $\{a\}$ or $\{a,b\}$ at position y; b can only be chosen at position y if the multiword W contains $\{b\}$ or $\{a,b\}$ at y. Notice that the premise depends on the alphabet only. The consequence states that two successive positions contain a and b in that order; the successor relation $S(z_1,z_2)$ means that position z_2 is immediately after z_1.

After the example of Table 1, it is now easy to construct an MSO formula that defines $\mathsf{CERTAIN}(w)$ for any word w over a fixed alphabet Σ. Moreover, it is an easy exercise to generalize this construction to v-words, as illustrated by Table 2 for the v-word xx. So we have the following result.

Theorem 4 *For every v-word v,* $\mathsf{CERTAIN}(v)$ *is regular.*

We are particularly interested in conditions on v that guarantee aperiodicity of $\mathsf{CERTAIN}(v)$, or equivalently, FO-definability of $\mathsf{CERTAIN}(v)$. For words without variables, such condition was settled by Theorem 3, and it is an open question whether there exists a variable-free word w such that $\mathsf{CERTAIN}(w)$ is not FO-definable. On the other hand, the following theorem shows the existence of a v-word v such that $\mathsf{CERTAIN}(v)$ is not FO-definable.

Theorem 5 *Let Σ be a finite alphabet with $|\Sigma| \geq 2$. Then,*

1. $\mathsf{CERTAIN}(xx)$ *is in* **P***;*
2. $\mathsf{CERTAIN}(xx)$ *is not FO-definable.*

PROOF. For the first item, let $W = \langle A_1, \ldots, A_n\rangle$. We construct, in polynomial time, a sequence $\langle B_1, \ldots, B_n\rangle$ where $B_1 = A_1$ and for each $k \in \{2,\ldots,n\}$,

$$B_k = \begin{cases} \{\} & \text{if } B_{k-1} = \{\} \\ A_k \setminus B_{k-1} & \text{if } |B_{k-1}| = 1 \\ A_k & \text{otherwise} \end{cases}$$

It is easy to show, by induction on increasing j, that for each $j \in \{1,\ldots,n\}$,

$$B_j = \{a \in A_j \mid \langle A_1, \ldots, A_{j-1}, \{a\}\rangle \nVdash_{\mathsf{certain}} xx\} \ .$$

Then, $W \in \mathsf{CERTAIN}(xx)$ if and only if $B_n = \{\}$.

For the second item, for every $i \geq 0$, let

$$W_i = \langle \{a\}, \overbrace{\{a,b\}, \ldots, \{a,b\}}^{i \text{ times}}, \{b\}\rangle \ .$$

It is easy to verify that $W_i \in \mathsf{CERTAIN}(xx)$ if and only if i is odd. The latter condition is not FO-definable. ◁

7 Conclusion

Every multiword defines a set of possible words. A word w is "certain" in a multiword W if it is a factor of every possible word of W. Given w, a significant question is whether $\mathsf{CERTAIN}(w)$, i.e. the set of multiwords in which w is certain, is FO-definable. We showed the following results:

1. $\mathsf{CERTAIN}(w)$ is regular if w is variable-free.

2. $\mathsf{CERTAIN}(w)$ is FO-definable if w is variable-free and the first or the last symbol of w occurs only once in w.

3. $\mathsf{CERTAIN}(xx)$ is not FO-definable.

These results are useful in consistent first-order rewriting under primary keys in historical databases. For example, our results imply that the following FOTL query q_1 (*"Did some employee remain in the same company for two successive time points?"*) has no consistent first-order rewriting:

$$q_1 \quad = \quad \exists x \exists y (\mathsf{WorksFor}(\underline{x}, y) \wedge \bigcirc \mathsf{WorksFor}(\underline{x}, y)) \ .$$

On the other hand, the following FOTL query q_2 (*"Did MS recruit an IBM employee who had worked for IBM for more than k time instants?*) has a consistent first-order rewriting:

$$\begin{aligned} q_2 = \exists x(& \mathsf{WorksFor}(\underline{x}, \mathrm{IBM}) \\ & \wedge \bigcirc^1 \mathsf{WorksFor}(\underline{x}, \mathrm{IBM}) \\ & \wedge \ldots \\ & \wedge \bigcirc^k \mathsf{WorksFor}(\underline{x}, \mathrm{IBM}) \\ & \wedge \bigcirc^{k+1} \mathsf{WorksFor}(\underline{x}, \mathrm{MS})) \ , \end{aligned}$$

where $\bigcirc^i \varphi = \overbrace{\bigcirc\bigcirc \ldots \bigcirc}^{i \text{ times}} \varphi$. Moreover, there exists an effective procedure for constructing a consistent first-order rewriting of q_2. This follows from Theorem 3 and the fact that the translation from an aperiodic regular language into FO is effective. So far, we have not studied algorithmic simplifications that may apply in our framework.

The following tasks remain for future research:

1. Show the conjecture that $\mathsf{CERTAIN}(w)$ is FO-definable if w is variable-free.

2. Find a syntactic characterization of the v-words w for which $\mathsf{CERTAIN}(w)$ is FO-definable.

References

[1] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *PODS*, pages 68–79. ACM Press, 1999.

[2] F. Blanchet-Sadri, R. Mercas, and G. Scott. A generalization of Thue freeness for partial words. *Theor. Comput. Sci.*, 410(8-10):793–800, 2009.

[3] A. Fuxman and R. J. Miller. First-order query rewriting for inconsistent databases. *J. Comput. Syst. Sci.*, 73(4):610–635, 2007.

[4] D. M. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal basis of fairness. In *POPL*, pages 163–173, 1980.

[5] L. Grieco, D. Lembo, R. Rosati, and M. Ruzzi. Consistent query answering under key and exclusion dependencies: algorithms and experiments. In O. Herzog, H.-J. Schek, N. Fuhr, A. Chowdhury, and W. Teiken, editors, *CIKM*, pages 792–799. ACM, 2005.

[6] J. A. W. Kamp. *Tense Logic and The Theory of Linear Order*. PhD thesis, University of California, Los Angeles, 1968.

[7] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[8] R. McNaughton and S. Papert. *Counter-free Automata*. MIT Press, Cambridge, MA, 1971.

[9] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[10] J. Wijsen. On the consistent rewriting of conjunctive queries under primary key constraints. In M. Arenas and M. I. Schwartzbach, editors, *DBPL*, volume 4797 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2007. A journal version will appear in Information Systems.

[11] J. Wijsen. Consistent query answering under primary keys: a characterization of tractable queries. In R. Fagin, editor, *ICDT*, volume 361 of *ACM International Conference Proceeding Series*, pages 42–52. ACM, 2009.