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 \( \text{WorksFor} \) relation at four successive time points \( t_0, t_1, t_2, t_3 \). The primary key \( \text{Name} \) is violated at times \( t_1 \) and \( t_2 \).

\[
\begin{array}{|c|c|} \hline
\text{Name} & \text{Company} \\ \hline
\text{Ed} & \text{IBM} \\ \hline
\text{Ed} & \text{MS} \\ \hline
\text{Ed} & \text{IBM} \\ \hline
\text{Ed} & \text{MS} \\ \hline
\end{array}
\]

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.

\[
\begin{array}{|c|c|} \hline
\text{Name} & \text{Company} \\ \hline
\text{Ed} & \text{IBM} \\ \hline
\text{Ed} & \text{IBM} \\ \hline
\text{Ed} & \text{MS} \\ \hline
\text{Ed} & \text{MS} \\ \hline
\end{array}
\]

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

\[
q_0 = \exists x (\text{WorksFor}(x,\text{IBM}) \land \circ \text{WorksFor}(x,\text{MS}))
\]

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 \( \text{HIS} \), \( q'_0 \) evaluates to true on \( \text{HIS} \) if and only if \( q_0 \) evaluates to true on every repair of \( \text{HIS} \). 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 \( q_0 \):

\[
q'_0 = \exists x (\varphi_{\text{IBM}}(x) \land \circ (\varphi_{\text{IBM,MS}}(x) \text{ until } \varphi_{\text{MS}}(x)))
\]

---

*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:
\[
\varphi_{\text{IBM}}(x) = \exists y (\text{WorksFor}(x, y) \land \\
\forall y (\text{WorksFor}(x, y) \rightarrow y = \text{IBM})) \\
\varphi_{\text{MS}}(x) = \exists y (\text{WorksFor}(x, y) \land \\
\forall y (\text{WorksFor}(x, y) \rightarrow y = \text{MS})) \\
\varphi_{\{\text{IBM, MS}\}}(x) = \exists y (\text{WorksFor}(x, y) \land \\
\forall y (\text{WorksFor}(x, y) \rightarrow y = \text{IBM} \lor y = \text{MS}))
\]

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 \(q_0\):
\[
q_0 \equiv \exists x (\varphi_{\text{IBM}}(x) \land \varphi_{\text{MS}}(x))
\]
The queries \(\varphi_{\text{IBM}}(x)\), \(\varphi_{\{\text{IBM, MS}\}}(x)\), and \(\varphi_{\text{MS}}(x)\) 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 \(q_0\) into \(q'_0\). To study this rewriting, we use the abstraction explained in the next paragraph.

Assume a finite alphabet \(\Sigma\). 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 \(W_0 = \{\{a\}, \{a, b\}, \{a, b\}, \{b\}\}\) 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 \(\text{CERTAIN}(w)\) denote the set of multiwords in which \(w\) is certain. We are interested in the following problem: given a word \(w\), is \(\text{CERTAIN}(w)\) 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 \(\psi_0 = a \land \Diamond b\), which abstracts \(q_0\). Let \(W\) be a multiword over the powerset alphabet \(\{\{a\}, \{b\}, \{a, b\}\}\). Then, \(\psi_0\) evaluates to true on every possible word of \(W\) if and only if \(W\) satisfies \(\psi_0 = \{a\} \land \Diamond \{a, b\}\) 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 \(\text{CERTAIN}(w)\) if \(w\) contains no variables. The algorithm is interesting because it immediately leads to the result that \(\text{CERTAIN}(w)\) is regular, as shown in Section 5. That section also provides sufficient conditions for FO-definability of \(\text{CERTAIN}(w)\). Section 6 deals with words that contain variables, which are placeholders for symbols of \(\Sigma\). For example, \(a x x\) is certain in \(W_0\) if either \(a a a\) or \(a b b\) 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 \(\epsilon\).

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 \(w \triangleright v\), if there exist (possibly empty) words \(p\) and \(q\) such that \(w = p \cdot v \cdot q\).

Multiwords capture the notion of inconsistency and are defined next.
In particular, we want to answer the question: That is, is there a FO formula for the following language: It can be verified that this formula defines the language $\text{CERTAIN}(ab)$. 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 $\text{CERTAIN}(w)$

Given $w$, we give a procedure for deciding membership of $\text{CERTAIN}(w)$. 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 $\text{CERTAIN}(w)$ 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 $\text{suffix}(u, w)$ denotes the maximal suffix of $u$ that is a prefix of $w$. For a set $S$ of words, we define $\text{suffix}(S, w) = \{ \text{suffix}(u, w) \mid u \in S \}$.

Example 1 $\text{suffix}(\text{abcd}, \text{cd}) = \text{cd}$ and $\text{suffix}(\text{ab}, \text{c}) = \epsilon$. Clearly, $\text{suffix}(w, w) = w$.

Lemma 1 If $\text{suffix}(u, w) = q$, then $\text{suffix}(u \cdot a, w) = \text{suffix}(q \cdot a, w)$.

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

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

We need to show $\text{suffix}(u' \cdot q \cdot a, w) = \text{suffix}(q \cdot a, w)$. Obviously, $\text{suffix}(q \cdot a, w)$ is a suffix of $\text{suffix}(u' \cdot q \cdot a, w)$. Assume $|\text{suffix}(u' \cdot q \cdot a, w)| > |\text{suffix}(q \cdot a, w)|$. 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 $\text{suffix}(u' \cdot q \cdot a, w) = \text{suffix}(q \cdot a, w)$.

The construction for deciding membership of $\text{CERTAIN}(w)$ 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 $u = \text{suffix}(p \cdot a, w)$ for some $p \in S_{i-1}$ and $a \in A_i$. By Theorem 1, $W \in \text{CERTAIN}(w)$ 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
\( w = \text{abdalcab} \).

\[ S_0 = \{ \epsilon \} \]
\[ A_1 = \{ a \} \]
\[ A_2 = \{ b \} \]
\[ A_3 = \{ d \} \]
\[ A_4 = \{ a \} \]
\[ A_5 = \{ b \} \]
\[ A_6 = \{ c \} \]
\[ A_7 = \{ a \} \]
\[ A_8 = \{ a, b \} \]
\[ A_9 = \{ b \} \]
\[ A_{10} = \{ d \} \]
\[ A_{11} = \{ a \} \]
\[ A_{12} = \{ b \} \]
\[ A_{13} = \{ c, d \} \]
\[ A_{14} = \{ a \} \]
\[ A_{15} = \{ b \} \]
\[ A_{16} = \{ c \} \]
\[ A_{17} = \{ a \} \]
\[ A_{18} = \{ b \} \]

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

Let \( m \in \{ 1, 2, \ldots, n \} \). For every word \( u \in \text{words}(\langle A_1, \ldots, A_m \rangle) \), either \( \text{supfre}(u, w) \in S_m \) or \( u \not\vdash w \).

Proof. Proof by induction on \( m \).

Basis \( m = 1 \). Let \( u = b \in \text{words}(\langle A_1 \rangle) \) such that \( b \not\vdash w \) (i.e. \( w \neq b \)). Then, \( S_1 \) contains \( \text{supfre}(\epsilon, b, w) = \text{supfre}(b, w) \).

Step. Assume w.l.o.g. \( S_m = \{ p_1, \ldots, p_k \} \) and \( A_{m+1} = \{ a_1, \ldots, a_l \} \). Then, \( S_{m+1} = \{ \text{supfre}(p_1 \cdot a_1, w), \ldots, \text{supfre}(p_k \cdot a_j, w), \ldots, \text{supfre}(p_k \cdot a_l, w) \} \setminus \{ w \} \).

Let \( u \in \text{words}(\langle A_1, \ldots, A_{m+1} \rangle) \). We can assume \( v \in \text{words}(\langle A_1, \ldots, A_m \rangle) \) and \( j \in \{ 1, \ldots, l \} \) such that \( u = v \cdot a_j \). Two cases can occur:

- \( v \not\vdash w \). Obviously, \( u \not\vdash w \).
- \( v \not\vdash w \). By the induction hypothesis, \( \text{supfre}(v, w) \in S_m \). We can assume w.l.o.g. \( i \in \{ 1, \ldots, k \} \) such that \( p_i = \text{supfre}(v, w) \).

By Lemma 1, \( \text{supfre}(u, w) = \text{supfre}(v, a_j, w) = \text{supfre}(p_i \cdot a_j, w) \).

If \( \text{supfre}(p_i \cdot a_j, w) = w \), then \( u \vdash w \); otherwise \( S_{m+1} \) contains \( \text{supfre}(u, w) \).

This concludes the proof.

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 \( u \in \text{words}(\langle A_1, \ldots, A_i \rangle) \) such that \( u \not\vdash w \) and \( \text{supfre}(u, w) = p \).

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 \( u \in \text{words}(\langle A_1, \ldots, A_i \rangle) \) such that \( u \not\vdash w \) and \( \text{supfre}(u, w) = q \). Then, \( a \cdot u \in \text{words}(\langle A_1, \ldots, A_{i+1} \rangle) \). It suffices to show that \( u \cdot a \not\vdash w \) and that \( \text{supfre}(u \cdot a, w) = p \).

- Assume \( u \cdot a \vdash w \). Then necessarily, \( w = q \cdot a \). But then \( p = w \), a contradiction. We conclude by contradiction that \( u \cdot a \not\vdash w \).
- By Lemma 1, \( \text{supfre}(u \cdot a, w) = \text{supfre}(q \cdot a, w) = \text{supfre}(p, w) \). Since \( p \) is a prefix of \( w \), \( \text{supfre}(p, w) = p \).

This concludes the proof.

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, \( W \in \text{CERTAIN}(w) \) if and only if \( S_n = \{ \} \).

Proof. If part. Assume \( S_n = \{ \} \). Lemma 2 implies that \( W \in \text{CERTAIN}(w) \).

Only-if part. Assume \( S_n \neq \{ \} \). Lemma 3 implies that \( W \not\in \text{CERTAIN}(w) \).

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

\[ [S_i] = \begin{cases} \{ \epsilon \} & \text{if } \epsilon \in S_i \\ S_i & \text{otherwise} \end{cases} \]

Then, the sequence \( \langle S_0, \ldots, S_n \rangle \) defined by \( S_0 = \{ \} \) and for each \( i \in \{ 1, \ldots, n \} \), \( S_i = [\text{supfre}(S_{i-1} \cdot A_i, w) \setminus \{ w \}] \) satisfies: \( W \in \text{CERTAIN}(w) \) if and only if \( S_n = \{ \} \).

5 Defining \text{CERTAIN}(w) in FO

Assume a nonempty word \( w \) over alphabet \( \Sigma \). We show that \text{CERTAIN}(w) is regular and, under certain conditions, aperiodic. Theorem 1 suggests the following construction of a deterministic finite automaton that accepts \text{CERTAIN}(w).
Let \( P' \) be the set of prefixes of \( w \), and let \( P = P' \setminus \{ \epsilon, w \} \). We define \( \text{DFA}_\Sigma(w) \) 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 \( \epsilon \).
2. The alphabet \( \Omega \) 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 \( \delta : Q \times \Omega \rightarrow Q \) is defined by:
   \[
   \delta(S, a) = \{ s' \in Q \mid a \in \delta(s, a) \setminus \{ \} \}.
   \]

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

**Theorem 2** Let \( w \) be a nonempty word and \( W = \langle A_1, \ldots, A_n \rangle \) a multiword (both relative to the alphabet \( \Sigma \)). Then, \( W \in \text{CERTAIN}(w) \) if and only if \( \text{DFA}_\Sigma(w) \) accepts \( W \). Hence, \( \text{CERTAIN}(w) \) 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 \( \text{CERTAIN}(w) \). The following theorem shows that \( \text{CERTAIN}(w) \) 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, \( \text{CERTAIN}(aa) \) is FO-definable. It is an open conjecture that \( \text{CERTAIN}(w) \) is aperiodic for any word \( w \).

**Theorem 3**

1. If the last symbol of \( w \) occurs only once in \( w \), then \( \text{CERTAIN}(w) \) is aperiodic.
2. If the first symbol of \( w \) occurs only once in \( w \), then \( \text{CERTAIN}(w) \) 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 \( P, U, Q \):

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

That is, for all multiwords \( P, U, Q \):

\[
P \cdot U^{k+1}, Q \not\in \text{CERTAIN}(w) \iff P \cdot U^{k+2}, Q \not\in \text{CERTAIN}(w).
\]

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

Assume \( P \cdot U^{k+1}, Q \not\in \text{CERTAIN}(w) \). We can assume the existence of \( p \in \text{words}(P), u_1, \ldots, u_{k+1} \in \text{words}(U) \), and \( q \in \text{words}(Q) \) such that the word

\[
m = p \cdot u_1 \cdot u_2 \cdot \ldots \cdot u_{k+1} \cdot q
\]

satisfies \( m \not\in \text{w} \). Two cases can occur:

1. The symbol \( a \) does not occur in \( u_2 \ldots u_{k+1} \). Let \( m' \) be the word obtained from \( m \) by “duplicating” the factor \( u_2 \) as follows:

\[
m' = p \cdot u_1 \cdot u_2 \cdot u_2 \cdot \ldots \cdot u_{k+1} \cdot q
\]

Assume \( m' \models w \). Since \( m \not\in w \), any factor \( w' \) of \( m' \) must “start in” \( s_1 \) and “end in” \( t_1 \). Since \( |w'| = k \), \( w \) must actually end in \( t_2 \). Then \( m' \) is the word obtained from \( m \) by “omitting” the factor \( u_1 \cdot a \cdot r \cdot q \):

\[
m = p \cdot o \cdot v \cdot a \cdot r \cdot q
\]

Let \( m' \) be the word obtained from \( m \) by “duplicating” the factor \( v \cdot a \) as follows:

\[
m' = p \cdot o \cdot v \cdot a \cdot v \cdot a \cdot r \cdot q
\]

Assume \( m' \models w \). Since \( m \not\in w \), 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 \( m' \not\in w \). It is easy to verify that \( m' \in \text{words}(P \cdot U^{k+2}, Q) \). It follows \( P \cdot U^{k+2}, Q \not\in \text{CERTAIN}(w) \).

Assume \( P \cdot U^{k+2}, Q \not\in \text{CERTAIN}(w) \). We can assume \( p \in \text{words}(P), u_1, \ldots, u_{k+2} \in \text{words}(U) \), and \( q \in \text{words}(Q) \) such that the word

\[
m = p \cdot u_1 \cdot \ldots \cdot u_{k+1} \cdot u_{k+2} \cdot q
\]

satisfies \( m \not\in w \). Let \( m' \) be the word obtained from \( m \) by “omitting” the factor \( u_1 \) as follows:

\[
m' = p \cdot u_2 \cdot \ldots \cdot u_{k+1} \cdot u_{k+2} \cdot q
\]

Clearly, \( m' \in \text{words}(P \cdot U^{k+1}, Q) \). Two cases can occur:
This concludes the proof. \( \Box \)

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 \( \text{CERTAIN}(a \cdot w) \) and \( \text{CERTAIN}(w \cdot a) \) 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 \( xaax \) 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 \( \text{vars} = \{x, y, z, x_1, y_1, z_1, \ldots \} \) of variables disjoint from the alphabet \( \Sigma \).

**Definition 4** A v-word is a sequence \( s_1s_2 \ldots s_n \) where for each \( i \in \{1, \ldots, n\} \), \( s_i \in \Sigma \cup \text{vars} \). The notions of length and concatenation (see Def. 1) naturally extend to v-words.

The relation \( \models \) is extended as follows. A *valuation* is a total function \( \theta : \text{vars} \cup \Sigma \rightarrow \Sigma \) that is the identity on \( \Sigma \). For the v-word \( v = s_1s_2 \ldots s_n \), 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:

\[
\text{if } w \models v \text{ iff for some valuation } \theta, \theta(v) \text{ is a factor of } w.
\]

The relation \( \models_{\text{CERTAIN}} \) directly carries over to v-words: if \( W \) is a multitword and \( v \) a v-word, then \( W \models_{\text{CERTAIN}} v \) if and only if for every \( w \in \text{words}(W) \), \( w \models v \).

The problem statement in Section 3 carries over to v-words. For example, \( \text{CERTAIN}(xx) \) contains some multitword \( W \) if each \( w \in \text{words}(W) \) contains two consecutive occurrences of the same symbol. Assume a binary alphabet \( \Sigma = \{a, b\} \). Clearly, \( \text{CERTAIN}(aa) \cup \text{CERTAIN}(bb) \subseteq \text{CERTAIN}(xx) \), and the inclusion is strict: for \( W_1 = \langle a, \{a, b\}, b \rangle \), we have \( W_1 \not\in \text{CERTAIN}(aa) \cup \text{CERTAIN}(bb) \) and \( W_1 \in \text{CERTAIN}(xx) \).

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

![Figure 1. DFA\(_\Sigma(abb)\) with \( \Sigma = \{a, b\} \).](image)
We are particularly interested in conditions on $v$ that guarantee aperiodicity of $\text{CERTAIN}(v)$, or equivalently, FO-definability of $\text{CERTAIN}(v)$. 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 $\text{CERTAIN}(w)$ is not FO-definable. On the other hand, the following theorem shows the existence of a v-word $v$ such that $\text{CERTAIN}(v)$ is not FO-definable.

\section{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 $\text{CERTAIN}(w)$, i.e., the set of multiwords in which $w$ is certain, is FO-definable. We showed the following results:

1. $\text{CERTAIN}(w)$ is regular if $w$ is variable-free.
2. 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. 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:

\[
q_1 = \exists x \exists y (\text{WorksFor}(x, y) \land \bigcirc \text{WorksFor}(x, y))
\]

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:

\[
q_2 = \exists x (\text{WorksFor}(x, \text{IBM}) \land \bigcirc^i \text{WorksFor}(x, \text{IBM}) \land \ldots \land \bigcirc^{k+1} \text{WorksFor}(x, \text{IBM}) \land \bigcirc^{k+1} \text{WorksFor}(x, \text{MS}))
\]

where \(\bigcirc^i \varphi = \bigcirc \ldots \bigcirc \varphi\). 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 CERTAIN\((w)\) is FO-
definable if \(w\) is variable-free.

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

References


