INFORMATION AND COMPUTATION 83, 245–263 (1989)

Decision Problems of Object Histories*

YONGKYUN CHO AND SEYMOUR GINSBURG

Computer Science Department, University of Southern California, Los Angeles, California 90089-0782

In an earlier paper, a record-based computation-oriented data model was introduced to describe historical data (here called "object history" and represented by a sequence of "computation tuples"). The major construct in the model is a computation-tuple sequence scheme (CSS), which specifies the set of all possible "valid" histories for the object of interest. In subsequent papers, a number of properties of the model were identified and studied in their own right. The present investigation considers decision problems related to two of these properties, namely "local" constraints and "bad-subsequence" constraints, as well as decision problems for several set-theoretic relations concerning the sets of object histories described by CSS. All of the decision problems considered are shown to be recursively unsolvable. The technique employed in each case is to exhibit a special class of CSS and use a reduction argument based on a known undecidable problem for context-free languages.

INTRODUCTION

In Ginsburg and Tanaka (1986) a record-based computation-oriented data model was introduced to describe historical data (here called "object history" and represented by a sequence of "computation tuples"). The major construct in the model is a computation-tuple sequence scheme (CSS), which specifies the set of all possible "valid" histories for the object of interest. In subsequent papers (Ginsburg and Tanaka, 1984; Ginsburg and Gyssens, 1987; Ginsburg and Tang, 1986, 1989; Dong and Ginsburg, 1986), various properties of the model, in particular, projection, "local" constraints, and "bad-subsequence" constraints, were investigated. From time to time, various researchers, including a referee of Ginsburg and Gyssens (1987), asked one of the authors about decision problems for object histories. One such problem related to projection was discussed and shown to be recursively unsolvable in Ginsburg and Tang (1986). The purpose of the present paper is to consider decision problems related to local constraints and bad-subsequence constraints, as well as decision problems

* This work was supported in part by the National Science Foundation under Grant DCR-831-8752.

0890-5401/89 $3.00

for several set-theoretic relations concerning the set of object histories described by CSS.

To deal in a reasonable way with decision problems, we limit our attention to those CSS in which every component is recursive. All the decidability problems considered (except whether or not an arbitrary computation-tuple sequence is valid) turn out to be recursively unsolvable. Furthermore, our recursive unsolvability results are obtained for proper subfamilies of CSS. None of the proofs is particularly profound, each being based on reduction of a known unsolvable problem for context-free languages.

The paper itself consists of five sections (besides this Introduction). The first section reviews the object history model (for a simplified version). Section 2 deals mainly with unsolvability results about equivalence and inclusion of the sets of valid computation-tuple sequences determined by CSS. Sections 3 and 4 establish unsolvability results related to CSS with local constraints and bad-subsequence constraints, respectively. The last section presents two open decision questions.

1. Preliminaries

In this section, we review the model of "object histories" introduced in Ginsburg and Tanaka (1986), to which the reader is refered for additional details and examples.

Informally, an object history is a historical record of an object (such as a person's checking account, a company's sales record of an item, etc.). An object history is a sequence of occurrences, each occurrence consisting of some input data and, possibly, some calculation. (For example, in a checking account history, one occurrence might be, in part, the amount to be deposited or withdrawn, together with the computation of the new balance.) In the model, each object history is represented as a sequence of tuples (over the same attributes), called a "computation-tuple sequence." A computation-tuple sequence scheme (abbreviated CSS) is a construct which defines the set of all possible "valid" computation-tuple sequences. (For example, a CSS for objects of the type "checking accounts" specifies the set of all possible "valid" individual checking account histories.) A CSS consists of:

($\Delta 1$) a set of attributes, partitioned into state, input, and evaluation attributes, according to their roles;

($\Delta 2$) functions which calculate values for state and evaluation attributes;

($\Delta 3$) semantic constraints whose satisfaction is to hold uniformly throughout a computation-tuple sequence; and

($\Delta 4$) a set of specific computation-tuple sequences of some bounded length with which to start a valid computation-tuple sequence, until all state and evaluation functions can be applied.

As mentioned in the Introduction, our concern in this paper is with certain decision problems. All of those considered here will turn out to be recursively unsolvable. In establishing our results, we shall show that these problems are unsolvable for CSS which have exactly one state attribute and no evaluation attributes (and thus no evaluation functions). Furthermore, these problems are only meaningful within an environment in which every component is recursive. Accordingly, the formal model we now present is a special case of the more general one given in Ginsburg and Tanaka (1986), in that *there will be exactly one state attribute, no evaluation attributes, no evaluation functions, and all given components will be recursive.*[1]

Turning to our simplified version of the model, Dom_∞ is an infinite set of elements (called *domain* values) and U_∞ is an infinite set of symbols (called *attributes*). For each A in U_∞, $Dom(A)$ (called the *domain* of A) is a recursive subset of Dom_∞ of at least two elements. All attributes occurring are assumed to be elements of U_∞.

Let X be a finite nonempty subset of U_∞ and $A_1, ..., A_n$ some fixed listing of the distinct elements of X. Then $\langle X \rangle$ denotes the sequence (written without commas) $A_1 \ldots A_n$ and $Dom(\langle X \rangle)$ the Cartesian product $Dom(A_1) \times \cdots \times Dom(A_n)$.

We now formalize the notions of occurrence and sequence of occurrences as used earlier. (Instead of "occurrence" and "sequence of occurrences," we shall use the terms "computation tuple" and "computation-tuple sequence.")

DEFINITION. Let $\langle U \rangle$ be a sequence of attributes. A *computation tuple* over $\langle U \rangle$ is an ordered pair $(\langle U \rangle, u)$, or u when $\langle U \rangle$ is understood, where u is an element of $Dom(\langle U \rangle)$. A *computation-tuple sequence* over $\langle U \rangle$ is a nonempty finite sequence of computation tuples over $\langle U \rangle$. The set of all computation-tuple sequences over $\langle U \rangle$ is denoted by $\mathrm{SEQ}(\langle U \rangle)$.

Unless otherwise stated, u and v (possibly subscripted or primed) denote computation tuples. Similarly, $\bar{u}$ and $\bar{v}$ denote computation-tuple sequences.

Using the previous notation, we now formalize ($\Delta 1$) and ($\Delta 2$).

[1] We assume that the reader is familiar with the basic notions of recursive function theory, as found, for example, in Hopcroft and Ullman (1979) and Machtey and Young (1978).

DEFINITION. A *computation scheme* (abbreviated CS) over $\langle U \rangle$ is a triple $\mathscr{C} = (A, \langle I \rangle, f_A)$, where

(i) $\langle U \rangle = A\langle I \rangle$, with A in U and I a nonempty subset of $U - \{A\}$. A is called a *state attribute* and each element of I is called an *input attribute*. (Given A and $\langle I \rangle = B_1 \ldots B_n$, $A\langle I \rangle = AB_1 \ldots B_n$.)

(ii) f_A is a partial recursive function (called a *state function*) from $Dom(\langle U \rangle)$ into $Dom(A)$ such that $\{u \mid f_A(u) \text{ is defined}\}$ is recursive.

We illustrate the above with the following:

EXAMPLE 1.1 (*Apartment rental*). Consider the sequences of rental records for a particular apartment, each rental record consisting of the four attributes SEQ-NO, DATE, TENANT, and AMOUNT. Each apartment rental occurrence is represented as a 4-tuple u. Here, (a) u(SEQ-NO) is the sequential number of the record; (b) u(DATE) is the year, month, and day on which the record is being listed; (c) u(TENANT) is the name of the tenant; and (d) u(AMOUNT) is the amount of the (monthly) rent received. Also, *Dom*(SEQ-NO) is the set of positive integers, *Dom*(DATE) the set of all date values in which the day of the month is either 1 or 15[2], *Dom*(TENANT) the set of people names plus the value VACANT, and *Dom*(AMOUNT) the set of nonnegative numbers. SEQ-NO is the state attribute, and DATE, TENANT, and AMOUNT are input attributes. Thus, $\langle U \rangle = A\langle I \rangle$, where $A = \text{SEQ-NO}$ and $\langle I \rangle = \text{DATE TENANT AMOUNT}$. The state function $f_{\text{SEQ-NO}}$ is defined by $f_{\text{SEQ-NO}}(u) = u(\text{SEQ-NO}) + 1$ for all u in $Dom(\langle U \rangle)$.

The purpose of a computation scheme is to select those computation-tuple sequences whose values for the state attributes are determined by the corresponding state functions. More formally, we have:

Notation. Let $\mathscr{C} = (A, \langle I \rangle, f_A)$ be a CS over $\langle U \rangle$. For each A in S, let

$$\text{VSEQ}(\mathscr{C}) = \{u_1 \ldots u_m \text{ in SEQ}(\langle U \rangle) \mid u_i(A) = f_A(u_{i-1}) \text{ for each } i,\ 2 \leqslant i \leqslant m\}.$$

Thus, VSEQ($\mathscr{C}$) is the set of computation-tuple sequences over $\langle U \rangle$ "consistent" with the computation scheme. Note that VSEQ($\mathscr{C}$) is a recursive set.

Turning to ($\Delta 3$), we have

DEFINITION. A *constraint* σ over $\langle U \rangle$ is a total recursive mapping from SEQ($\langle U \rangle$) into {true, false}. If $\sigma(\bar{u}) = \text{true}$, also denoted by $\bar{u} \models \sigma$, then $\bar{u}$

[2] This domain is selected because later we insist that rent always be paid on either the 1st or 15th of the month.

is said to *satisfy* σ. For each set Σ of constraints over $\langle U \rangle$, the set $\{\bar{u}$ in $\mathrm{SEQ}(\langle U \rangle) | \bar{u} \models \sigma$ for each σ in $\Sigma\}$ is denoted by $\mathrm{VSEQ}(\Sigma)$.

Note that each $\mathrm{VSEQ}(\sigma)$, as well as $\mathrm{VSEQ}(\Sigma)$, is a recursive set.

Without further limitations, constraints can permit highly pathological situations. To avoid these undesirable cases and to retain the intuitive feeling that intervals of history are also history in some sense, we restrict ourselves to a class called *uniform*.

DEFINITION. A constraint σ over $\langle U \rangle$ is *uniform* if $u_1 \ldots u_m \models \sigma$ implies $u_i \ldots u_j \models \sigma$ for all $u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U \rangle)$ and all i and j, $1 \leqslant i, j \leqslant m$.

Clearly, $\mathrm{VSEQ}(\Sigma)$ is interval closed if each σ in Σ is uniform.

Uniform constraints are natural, mathematically tractable, cover most situations arising in practice, and eliminate many pathological cases.

EXAMPLE 1.1 (*Continued*). The conditions involved in the tenancy are as follows. Rent is due once a month, always on the day of the month (either the 1st or 15th) the current tenant took possesion of the apartment. Because of rent control, rent for a continuing tenant cannot be changed more than once every 12 months (by a variable percentage determined by law). When a tenant vacates the premises, then the next tenant's rent becomes negotiable.

The above conditions translate into the following five constraints: For each $\bar{u} = u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U \rangle)$,

(a) $\bar{u} \models \sigma_1$ iff for each i, $1 \leqslant i < m$, $u_{i+1}(\mathrm{DATE}) \sqsupseteq u_i(\mathrm{DATE})$, where $\sqsupseteq$ denotes calendar-wise ordering;

(b) $\bar{u} \models \sigma_2$ iff for each i, $1 \leqslant i < m$, $u_{i+1}(\mathrm{TENANT}) = u_i(\mathrm{TENANT})$ implies $u_{i+1}(\mathrm{DATE}) = u_i(\mathrm{DATE}) \oplus 1$ month, $\oplus$ denoting calendar-wise addition;

(c) $\bar{u} \models \sigma_3$ iff for each i, $1 \leqslant i \leqslant m$, $u_i(\mathrm{TENANT}) = \mathrm{VACANT}$ implies $u_i(\mathrm{AMOUNT}) = 0$;

(d) $\bar{u} \not\models \sigma_4$ iff there exist i and j, $1 \leqslant i < j < m$, such that $u_i(\mathrm{AMOUNT}) \neq u_{i+1}(\mathrm{AMOUNT})$, $u_j(\mathrm{AMOUNT}) \neq u_{j+1}(\mathrm{AMOUNT})$, $u_i(\mathrm{TENANT}) = u_{i+1}(\mathrm{TENANT}) = \cdots = u_j(\mathrm{TENANT}) = u_{j+1}(\mathrm{TENANT})$, and $j - i < 12$; and

(e) $\bar{u} \not\models \sigma_5$ iff there exist i and j, $1 \leqslant i < j < m$, such that $u_i(\mathrm{TENANT}) \neq u_{i+1}(\mathrm{TENANT})$, $u_{i+1}(\mathrm{TENANT}) = \cdots = u_{j+1}(\mathrm{TENANT})$, $u_{j+1}(\mathrm{AMOUNT}) \neq u_{i+1}(\mathrm{AMOUNT})$, and $j - i < 12$.

Thus, σ_1 requires the tuples to follow each other in calendar-wise order, σ_2 says that the tenant payments occur exactly one month apart, σ_3 declares that there is no rent money when the apartment is unoccupied, σ_4

insists that no continuing tenant has the rent changed twice in any 12 month period, and σ_5 asserts that the rent stay fixed at least 12 months after a tenant moves in. Clearly, each constraint is uniform.

Finally, we formalize ($\Delta 4$), the notion of how to get started or "initialization,"

DEFINITION. Given a finite set Σ of uniform constraints over $\langle U \rangle$, an *initialization (with respect to Σ)* is any recursive subset $\mathscr{I}$ of $\mathrm{VSEQ}(\Sigma) \cap Dom(\langle U \rangle)$. For each initialization $\mathscr{I}$, let $\mathrm{VSEQ}(\mathscr{I}) = \mathscr{I} \cup \{u\bar{u} \mid u \text{ in } \mathscr{I}, \bar{u} \text{ in } \mathrm{SEQ}(\langle U \rangle)\}$.

Note that $\mathrm{VSEQ}(\mathscr{I})$ is recursive.

EXAMPLE 1.1 (*Continued*). We shall assume that the apartment is initially vacant. Thus, the initialization is

$$\mathscr{I} = \{(1, d, \mathrm{VACANT}, 0) \mid d \text{ in } Dom(\mathrm{DATE})\}.$$

Using the concepts already defined, we are now able to formalize the fundamental notion of "computation-tuple sequence scheme."

DEFINITION. A *computation-tuple sequence scheme* (abbreviated CSS) over $\langle U \rangle$ is a triple $T = (\mathscr{C}, \Sigma, \mathscr{I})$, where

(i) $\mathscr{C}$ is a computation scheme over $\langle U \rangle$;

(ii) Σ is a finite set of uniform constraints over $\langle U \rangle$; and

(iii) $\mathscr{I}$ is an initialization with respect to Σ.

A CSS determines a set of "valid" computation-tuple sequences as follows:

DEFINITION. For each CSS $T = (\mathscr{C}, \Sigma, \mathscr{I})$ over $\langle U \rangle$,

$$\mathrm{VSEQ}(T) = \mathrm{VSEQ}(\mathscr{C}) \cap \mathrm{VSEQ}(\Sigma) \cap \mathrm{VSEQ}(\mathscr{I}).$$

A computation-tuple sequence is said to be *valid* (for T) if it is in $\mathrm{VSEQ}(T)$.

Note that $\mathrm{VSEQ}(T)$ is recursive.

EXAMPLE 1.1 (*Continued*). A CSS for the apartment rental situation is $T = ((A, \langle I \rangle, f_A), \{\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5\}, \mathscr{I})$, where the individual components are as already defined. One valid computation-tuple sequence is given in Table I.

TABLE I

SEQ-NO	DATE	TENANT	AMOUNT	SEQ-NO	DATE	TENANT	AMOUNT
1	1-15-85	Vacant	0	11	10-15-85	Smith	575
2	2-1-85	Jones	550	12	11-15-85	Smith	575
3	3-1-85	Jones	550	13	12-15-85	Smith	575
4	4-1-85	Jones	550	14	1-15-86	Smith	575
5	5-1-85	Vacant	0	15	2-15-86	Smith	575
6	5-15-85	Smith	575	16	3-15-86	Smith	575
7	6-15-85	Smith	575	17	4-15-86	Smith	575
8	7-15-85	Smith	575	18	5-15-86	Smith	603
9	8-15-85	Smith	575	19	6-15-86	Smith	603
10	9-15-85	Smith	575	20	7-15-86	Smith	603

2. Set-Theoretic Properties

In this section, we consider the decision problems for the set-theoretic properties of emptiness (non-emptiness), infiniteness, equivalence, and inclusion. We shall establish the recursive unsolvability of these problems by using reductions of known unsolvability results in context-free language theory.[3]

We start with the problems for emptiness (non-emptiness), infiniteness, and non-empty finiteness. First though, we recall several well-known results (Ginsburg, 1966; Harrison, 1978; Hopcroft and Ullman, 1979) from formal language theory.

THEOREM 2A. (a) *Each context-free language is recursive.*

(b) *It is recursively unsolvable to decide for an arbitrary context-free grammar (CFG) G over an alphabet Δ of at least two elements whether or not $L(G) = \Delta^*$.*

(c) *It is recursively unsolvable to determine for arbitrary CFG G_1 and G_2 over an alphabet Δ of at least two elements whether or not $L(G_1) \cap L(G_2) = \varnothing$.*

(d) *It is recursively unsolvable to determine for arbitrary CFG G_1 and G_2 over an alphabet Δ of at least two elements whether or not $L(G_1) = L(G_2)$, resp., $L(G_1) \subseteq L(G_2)$.*

[3] We assume that the reader is familiar with the terminology and basic notions of context-free language theory, as found, e.g., in Ginsburg (1966), Harrison (1978), and Hopcroft and Ullman (1979).

THEOREM 2.1. *It is recursively unsolvable to determine for an arbitrary CSS T whether or not*

(a) $\mathrm{VSEQ}(T) = \varnothing$;

(b) $\mathrm{VSEQ}(T)$ *is infinite*;

(c) $\mathrm{VSEQ}(T)$ *is nonempty finite.*

Proof. Let $Dom(A) = \{1, 2, 3, \ldots\}$ and $f_A(u) = u(A) + 1$ for all u in $Dom(\langle U \rangle)$, U defined below.

Consider (a). Let $\Delta = \{a, b\}$ and $\langle U \rangle = AB$, where $Dom(B) = \Delta^*$.[4] For each CFG G over Δ, let $T_G = (\mathscr{C}, \{\sigma_G\}, \mathscr{I})$ be defined as follows:

(i) $\mathscr{C} = (A, B, f_A)$;

(ii) σ_G is the constraint over $\langle U \rangle$ defined by $\mathrm{VSEQ}(\sigma_G) = \{u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U \rangle) \mid m \geqslant 1$ and for all i, $1 \leqslant i \leqslant m$, $u_i(B)$ is not in $L(G)\}$; and

(iii) $\mathscr{I} = Dom(\langle U \rangle) \cap \mathrm{VSEQ}(\sigma_G)$.

By (a) of Theorem 2A and the fact that $\mathrm{VSEQ}(\sigma_G) = \mathrm{SEQ}(AB')$, where $Dom(B') = \Delta^* - L(G)$, it follows that σ_G is recursive. Also, σ_G is obviously uniform. Thus T_G is a CSS over $\langle U \rangle$. From the construction of T_G, it is readily seen that

(1) $\mathrm{VSEQ}(T_G) = \varnothing$ iff $L(G) = \Delta^*$.

From (1) and (b) of Theorem 2A, (a) is recursively unsolvable.[5]

Consider (b). Let $\Delta = \{a, b\}$ and c be a new symbol. Let $\langle U \rangle = AB$, where $Dom(B) = \Delta^* \cup \{c\}$. For arbitrary CFG G_1 and G_2 over Δ, let $T_{G_1G_2} = (\mathscr{C}, \{\sigma_{G_1G_2}\}, \mathscr{I})$ be defined as follows:

(iv) $\mathscr{C} = (A, B, f_A)$;

(v) $\mathrm{VSEQ}(\sigma_{G_1G_2}) = Dom(\langle U \rangle) \cup \{u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U \rangle) \mid u_i(B)$ is in $L(G_1) \cap L(G_2)$ for all i, $2 \leqslant i \leqslant m$, $m \geqslant 2\}$; and

(vi) $\mathscr{I} = \{(1, c)\}$.

Obviously, $\sigma_{G_1G_2}$ is a uniform constraint. Hence, $T_{G_1G_2}$ is a CSS over $\langle U \rangle$. Also, it is easy to see that

(2) $\mathrm{VSEQ}(T_{G_1G_2})$ is infinite iff $L(G_1) \cap L(G_2) \neq \varnothing$.

From (2) and (c) of Theorem 2A, (b) is recursively unsolvable.

[4] Consistent with the definition of a domain value, we regard each word in Δ^* as a distinguished symbol. This artifice is used throughout the paper.

[5] Throughout this paper, we omit from the proofs the ordinary reduction arguments standard in recursive function theory.

Finally, consider (c). Let $T_{G_1G_2}$ be as in (b). Since $(1, c)$ is in $\mathrm{VSEQ}(T_{G_1G_2})$, $\mathrm{VSEQ}(T_{G_1G_2}) \neq \varnothing$. By (2), we have

(3) $\mathrm{VSEQ}(T_{G_1G_2})$ is nonempty finite iff $L(G_1) \cap L(G_2) = \varnothing$.

From (3) and (c) of Theorem 2A, (c) is recursively unsolvable. ▮

Letting $T_1 = T_2 = T$, we immediately get

COROLLARY. *It is recursively unsolvable to decide for arbitrary CSS* $T_1 = (\mathscr{C}, \Sigma_1, \mathscr{I}_1)$ *and* $T_2 = (\mathscr{C}, \Sigma_2, \mathscr{I}_2)$ *whether or not*

(a) $\mathrm{VSEQ}(T_1) \cap \mathrm{VSEQ}(T_2) = \varnothing$;

(b) $\mathrm{VSEQ}(T_1) \cap \mathrm{VSEQ}(T_2)$ *is infinite*;

(c) $\mathrm{VSEQ}(T_1) \cap \mathrm{VSEQ}(T_2)$ *is nonempty finite.*

We now turn to the equality and containment problems between $\mathrm{VSEQ}(T)$.

THEOREM 2.2. *It is recursively unsolvable to decide for arbitrary CSS* $T_1 = (\mathscr{C}, \Sigma_1, \mathscr{I}_1)$ *and* $T_2 = (\mathscr{C}, \Sigma_2, \mathscr{I}_2)$ *whether or not*

(a) $\mathrm{VSEQ}(T_1) = \mathrm{VSEQ}(T_2)$;

(b) $\mathrm{VSEQ}(T_1) \subseteq \mathrm{VSEQ}(T_2)$.

Proof. Let A, B, and f_A be as in (a) of Theorem 2.1, and let $\langle U \rangle = AB$. For each CFG G over Δ, let $T_G = (\mathscr{C}, \{\sigma_G\}, \mathscr{I})$ be defined as follows:

(i) $\mathscr{C} = (A, B, f_A)$;

(ii) $\mathrm{VSEQ}(\sigma_G) = \{u_1 \ldots u_m \text{ in } \mathrm{SEQ}(\langle U \rangle) \mid u_i(B) \text{ is in } L(G) \text{ for all } i, 1 \leqslant i \leqslant m,\ m \geqslant 1\}$; and

(iii) $\mathscr{I} = Dom(\langle U \rangle) \cap \mathrm{VSEQ}(\sigma_G)$.

It is obvious that T_G is a CSS over $\langle U \rangle$ for each CFG G. It is also clear that, for arbitrary CFG G_1 and G_2 over Δ,

(1) $\mathrm{VSEQ}(T_{G_1}) = \mathrm{VSEQ}(T_{G_2})$ iff $L(G_1) = L(G_2)$, and

(2) $\mathrm{VSEQ}(T_{G_1}) \subseteq \mathrm{VSEQ}(T_{G_2})$ iff $L(G_1) \subseteq L(G_2)$.

The recursive unsolvability of (a) and (b) then follows from (1), (2), and (d) of Theorem 2A. ▮

3. LOCAL CSS

In this section, we focus our attention on decision problems related to the special class of uniform constraints called "local." We start by recalling some notions.

DEFINITION. A constraint σ over $\langle U \rangle$ is *k-local* ($k \geqslant 1$) if, for all $\bar{u}$ of length at least k, $\bar{u} \models \sigma$ iff $\bar{v} \models \sigma$ for all intervals $\bar{v}$ of $\bar{u}$ of length k. That is, for all $\bar{u} = u_1 \dots u_m$, $m \geqslant k$, $\bar{u} \models \sigma$ iff $u_i \dots u_{i+k-1} \models \sigma$ for each i, $1 \leqslant i \leqslant m-k+1$. A constraint is *local* if it is k-local for some k.

k-local constraints are important for two reasons:

(1) Many real-life constraints are of this type. In particular, for the constraints in Example 1.1, σ_1 and σ_2 are 2-local, σ_3 is 1-local, and σ_4 and σ_5 are 13-local. See Ginsburg and Tanaka (1986); Ginsburg and Gyssens (1987); Ginsburg and Tang (1986); Dong and Ginsburg (1986) for more examples. One major exception is most functional dependencies.

(2) They have the property that satisfaction by a computation-tuple sequence under addition of a computation tuple can be maintained by merely examining satisfaction of the last k tuples in the new sequence. That is, suppose σ is k-local and $\bar{u} = u_1 \dots u_m$, $m \geqslant k$, satisfies σ. Then, $\bar{u}u$ satisfies σ iff $u_{m-k+2} \dots u_m u$ satisfies σ.

DEFINITION. A set Σ of constraints over $\langle U \rangle$ is *k-local* (resp. *local*), if each constraint in Σ is k-local (resp. local). A CSS $T = (\mathscr{C}, \Sigma, \mathscr{I})$ is *k-local* (resp. *local*), if Σ is k-local (resp. local).[6]

It is known (Ginsburg and Tanaka, 1986) that if σ is k-local, then it is k'-local for all $k' \geqslant k$. Thus, a finite set Σ of local constraints is k-local for some k. The set of constraints in Example 1.1 is 13-local.

Prior to considering decision problems related to localness, note that the unsolvability results in Section 2 (i.e., Theorem 2.1 and its corollary and Theorem 2.2) still hold if the CSS are restricted to local CSS. Indeed, all the constraints arising in their proofs are 2-local.

We now turn to the results of Section 3. Our first theorem considers the decision problems for T being local.

THEOREM 3.1. *It is recursively unsolvable to decide for an arbitrary CSS T whether or not*

(a) *T is k-local for some given $k \geqslant 1$;*[7]

(b) *T is local.*

Proof. Let A and f_A be as in Theorem 2.1. Let $\Delta = \{a, b\}$ and c and d

[6] In the more extended version of a CSS $T = (\mathscr{C}, \Sigma, \mathscr{I})$, that is, the one including evaluation attributes and evaluation functions, one has the extended notion of T being (k_1, k_2)-*local* if k_1 is greater than the "rank" of $\mathscr{C}$ and Σ is k_2-local.

[7] The version for (a) in the extended model becomes "T is (k_1, k_2)-local for given k_1 and k_2." The proof for the simpler model also holds for the extended model.

be two new symbols. Let $\langle U\rangle = AB_1B_2$, where $Dom(B_1) = \Delta^* \cup \{c\}$ and $Dom(B_2) = \Delta^* \cup \{d\}$. For arbitrary CFG G_1 and G_2 over Δ, let $T_{G_1G_2} = (\mathscr{C}, \{\sigma_{G_1G_2}\}, \mathscr{I})$ be defined as follows:

(i) $\mathscr{C} = (A, B_1B_2, f_A)$;

(ii) $\mathrm{VSEQ}(\sigma_{G_1G_2}) = \{u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U\rangle) \mid$ for all i and j, $1 \leqslant i, j \leqslant m$, $m \geqslant 1$, $u_i(B_1)$ is in $L(G_1) \cup \{c\}$, $u_j(B_2)$ is in $L(G_2) \cup \{d\}$, and $u_i(B_1) \neq u_j(B_2)\}$; and

(iii) $\mathscr{I} = Dom(\langle U\rangle) \cap \mathrm{VSEQ}(\sigma_{G_1G_2})$.

Obviously, $T_{G_1G_2}$ is a CSS over $\langle U\rangle$.

We first note that

(1) if $L(G_1) \cap L(G_2) = \varnothing$, then $T_{G_1G_2}$ is 1-local (and thus k-local for all $k \geqslant 1$);

(2) if $L(G_1) \cap L(G_2) \neq \varnothing$, then $T_{G_1G_2}$ is not local; and

(3) $T_{G_1G_2}$ is local iff $L(G_1) \cap L(G_2) = \varnothing$.

Clearly (1) is true, and (3) follows from (1) and (2). Consider (2). Suppose $L(G_1) \cap L(G_2) \neq \varnothing$, say w is in $L(G_1) \cap L(G_2)$. Assume $\sigma_{G_1G_2}$ is local, say k-local for some k. Let $\bar{u} = u_1 \ldots u_{k+1}$, where $u_1 = (1, w, d)$, $u_{k+1} = (k+1, c, w)$, and $u_i = (i, c, d)$ for all i, $1 < i \leqslant k$. Then, $\bar{v} \models \sigma_{G_1G_2}$ for each interval $\bar{v}$ of $\bar{u}$ of length k. However, $\bar{u} \not\models \sigma_{G_1G_2}$ (since $u_1(B_1) = u_{k+1}(B_2) = w$), a contradiction. Thus, $\sigma_{G_1G_2}$ is not local, whence $T_{G_1G_2}$ is not local.

From (1), (2), and (c) of Theorem 2A, it follows that (a) is recursively unsolvable. Similarly, the recursive unsolvability of (b) follows from (3) and (c) of Theorem 2A. ∎

Our next result shows the recursive unsolvability for finding the smallest k for which a local CSS is k-local.

THEOREM 3.2. *For each $k \geqslant 2$, it is recursively unsolvable to decide for an arbitrary k-local CSS T whether or not there exists some k', $1 \leqslant k' < k$, such that T is k'-local.*

Proof. Let A and f_A be as in Theorem 2.1. Let $\Delta = \{a, b\}$, c and d be two new symbols, and $k \geqslant 2$. Let $\langle U\rangle = AB$, where $Dom(B) = \Delta^* \cup \{c, d\}$. For arbitrary CFG G_1 and G_2 over Δ, let $T_{G_1G_2k}$ (abbreviated $T_{G_1G_2}$) $= (\mathscr{C}, \{\sigma_{G_1G_2k}\}, \mathscr{I}_k)$ (abbreviated $(\mathscr{C}, \{\sigma_{G_1G_2}\}, \mathscr{I})$), be defined as follows:

(i) $\mathscr{C} = (A, B, f_A)$;

(ii) $\mathrm{VSEQ}(\sigma_{G_1G_2}) = \{u_1 \ldots u_m$ in $\mathrm{SEQ}(\langle U\rangle) \mid$ for each i, $1 \leqslant i \leqslant m$, if $u_i(A) = 0 \bmod k$, then $u_i(B)$ is in $L(G_1) \cup \{c\}$ and $u_i(B) \neq u_j(B)$ for each j,

$i < j \leqslant \min\{i+k-1, m\}$; if $u_i(A) \neq 0 \bmod k$, then $u_i(B)$ is in $L(G_2) \cup \{d\}\}$; and

(iii) $\mathscr{I} = Dom(\langle U \rangle) \cap \mathrm{VSEQ}(\sigma_{G_1G_2})$.

It is readily seen that $T_{G_1G_2}$ is a k-local CSS. It is also readily observed that

(1) If $L(G_1) \cap L(G_2) = \varnothing$, then $\sigma_{G_1G_2}$ is 1-local and thus k'-local for all $k' \geqslant 1$; and

(2) If $L(G_1) \cap L(G_2) \neq \varnothing$, then there is no k', $1 \leqslant k' < k$ such that $\sigma_{G_1G_2}$ is a k'-local.

Indeed, (1) is clearly true. Consider (2). Suppose $L(G_1) \cap L(G_2) \neq \varnothing$. Then there exists a word w in $L(G_1) \cap L(G_2)$. Let $\bar{u} = u_1 \ldots u_k$, where $u_1 = (1, w)$, $u_k = (k, w)$ and $u_i = (i, d)$ for all i, $1 < i < k$. Each interval of length $k-1$ of $\bar{u}$ is in $\mathrm{VSEQ}(\sigma_{G_1G_2})$. However, $\bar{u}$ does not satisfy $\sigma_{G_1G_2}$, since $u_1(B) = u_k(B) = w$. Hence, $\sigma_{G_1G_2}$ is not $(k-1)$-local, and therefore not k'-local for all $k' < k$, i.e., (2) holds.

From (1), (2), and (c) of Theorem 2A, the theorem holds. ∎

COROLLARY. *It is recursively unsolvable to determine for arbitrary $k \geqslant 2$ and k-local CSS T whether or not there exists some k', $1 \leqslant k' < k$, such that T is k'-local.*

It is known (Ginsburg and Tanaka, 1986) that there exist CSS $T = (\mathscr{C}, \Sigma, \mathscr{I})$ and $T' = (\mathscr{C}, \Sigma', \mathscr{I})$ such that T is not local, T' is local, and $\mathrm{VSEQ}(T) = \mathrm{VSEQ}(T')$. Such a situation is of interest for design considerations because of the ease of maintaining validity of computation-tuple sequences for local CSS. This leads to the following:

DEFINITION. A CSS $T = (\mathscr{C}, \Sigma, \mathscr{I})$ is said to be *k-locally representable*, $k \geqslant 1$ (resp. *locally representable*) if there exists a k-local (resp. local) CSS $T' = (\mathscr{C}, \Sigma', \mathscr{I})$ such that $\mathrm{VSEQ}(T') = \mathrm{VSEQ}(T)$.

Given a CSS, it would be helpful for design purposes to know if the CSS is (k-)locally representable. Unfortunately, as we now show, the problem is recursively unsolvable.

THEOREM 3.3. *It is recursively unsolvable to decide for an arbitrary CSS T whether or not*

(a) *T is k-locally representable for given $k \geqslant 1$;*

(b) *T is locally representable.*

Proof. Let $T_{G_1G_2} = (\mathscr{C}, \{\sigma_{G_1G_2}\}, \mathscr{I})$ be the same as in the proof of Theorem 3.1. We claim that

(1) If $L(G_1) \cap L(G_2) = \varnothing$, then $T_{G_1G_2}$ is 1-locally representable (and thus k-locally representable for all $k \geqslant 1$); and

(2) If $L(G_1) \cap L(G_2) \neq \varnothing$, then there is no $k \geqslant 1$ such that $T_{G_1G_2}$ is k-locally representable.

Consider (1). Suppose $L(G_1) \cap L(G_2) = \varnothing$. Then, $T_{G_1G_2}$ is 1-local. Hence, $T_{G_1G_2}$ is 1-locally representable. Consider (2). Suppose $L(G_1) \cap L(G_2) \neq \varnothing$, say w is in $L(G_1) \cap L(G_2)$. Assume $T_{G_1G_2}$ is k-locally representable for some $k \geqslant 1$. Then there exists a CSS $T'_{G_1G_2} = (\mathscr{C}, \Sigma'_{G_1G_2}, \mathscr{I})$ such that

(3) $\Sigma'_{G_1G_2}$ is k-local, and

(4) $\mathrm{VSEQ}(T'_{G_1G_2}) = \mathrm{VSEQ}(T_{G_1G_2})$.

Let $\bar{u} = u_1 \ldots u_{k+1}$, where $u_1 = (1, w, d)$, $u_{k+1} = (k+1, c, w)$, and $u_i = (i, c, d)$ for all i, $1 < i \leqslant k$. Clearly,

(5) $\bar{u}$ is in $\mathrm{VSEQ}(\mathscr{C})$,

(6) $\bar{u}$ is in $\mathrm{VSEQ}(\mathscr{I})$, and

(7) $\bar{u}$ is not in $\mathrm{VSEQ}(T_{G_1G_2})$.

For each interval $\bar{v}$ of $\bar{u}$ of length k, $\bar{v}$ is in $\mathrm{VSEQ}(T_{G_1G_2})$. By (4), each $\bar{v}$ is in $\mathrm{VSEQ}(T'_{G_1G_2})$ and therefore in $\mathrm{VSEQ}(\Sigma'_{G_1G_2})$. By (3),

(8) $\bar{u}$ is in $\mathrm{VSEQ}(\Sigma'_{G_1G_2})$.

By (5), (6), and (8),

(9) $\bar{u}$ is in $\mathrm{VSEQ}(T'_{G_1G_2})$.

From (7) and (9), we have

$$\mathrm{VSEQ}(T'_{G_1G_2}) \neq \mathrm{VSEQ}(T_{G_1G_2}),$$

which contradicts (4). Hence, there is no $k \geqslant 1$ such that T is k-locally representable.

Now consider (a). By (1) and (2), $T_{G_1G_2}$ is k-locally representable for given $k \geqslant 1$ iff $L(G_1) \cap L(G_2) = \varnothing$. By (c) of Theorem 2A, (a) is recursively unsolvable. Consider (b). From (1) and (2), $T_{G_1G_2}$ is locally representable iff $L(G_1) \cap L(G_2) = \varnothing$. Hence, by (c) of Theorem 2A, (b) is recursively unsolvable. ∎

In passing, we note that for each $k \geqslant 2$, it is recursively unsolvable to decide for an arbitrary k-locally representable CSS T whether or not there exists some k', $1 \leqslant k' < k$, such that T is k'-locally representable. Indeed, let $T_{G_1G_2k} = (\mathscr{C}, \{\sigma_{G_1G_2k}\}, \mathscr{I})$ be the same as in the proof of Theorem 3.2. An argument similar to that in Theorem 3.3 shows that for each $k \geqslant 2$, $T_{G_1G_2k}$ is k'-locally representable for some k', $1 \leqslant k' < k$, iff $L(G_1) \cap L(G_2) = \varnothing$. The standard reduction argument then yields the stated result.

4. Bad Subsequence CSS

In this section, we consider decision problems related to the notion of bad subsequence constraints, a class of constraints considered extensively in Ginsburg and Gyssens (1987). We start with some relevant concepts.

Definition. For each recursive set $\mathscr{B} \subseteq \mathrm{SEQ}(\langle U \rangle)$, let $c(\mathscr{B})$ be the constraint over $\langle U \rangle$ defined by $\bar{u} \models c(\mathscr{B})$ if there is no $\bar{v}$ in $\mathscr{B}$ such that $\bar{v}$ is a subsequence of $\bar{u}$. A constraint σ is a *bad subsequence* constraint if $\mathrm{VSEQ}(\sigma) = \mathrm{VSEQ}(c(\mathscr{B}))$ for some $\mathscr{B}$. A bad subsequence constraint σ is *k-bounded*, $k \geq 1$, if $\mathrm{VSEQ}(\sigma) = \mathrm{VSEQ}(c(\mathscr{B}))$ for some $\mathscr{B}$ such that the length of $\bar{v}$ is at most k for all $\bar{v}$ in $\mathscr{B}$. A bad subsequence constraint is *bounded* if it is k-bounded for some k.

The class of bad subsequence constraints is quite extensive. It includes σ_1 and σ_3 of Example 1.1, these being 2-bounded and 1-bounded, respectively.[8] [In particular, $\sigma_1 = c(\mathscr{B}_1)$ and $\sigma_3 = c(\mathscr{B}_3)$, where $\mathscr{B}_1 = \{u_1 u_2 \mid u_1(\mathrm{DATE}) \sqsupset u_2(\mathrm{DATE})\}$ and $\mathscr{B}_3 = \{u \mid u(\mathrm{TENANT}) = \mathrm{VACANT}$ and $u(\mathrm{AMOUNT}) \neq 0\}$.] Other examples of (bounded) bad subsequence constraints are all functional dependencies (Ginsburg and Gyssens, 1987). More generally, each equality generating dependency (Beeri and Vardi, 1984), defined in any of several different ways to take into account the order between the computation tuples in a computation-tuple sequence, is a (bounded) bad subsequence constraint. Similarly, order dependencies (Ginsburg and Hull, 1983) are (2-bounded) bad subsequence constraints.

Applying these concepts to a CSS, we get

Definition. A CSS $(\mathscr{C}, \Sigma, \mathscr{I})$ is a (*k-bounded*, *bounded* resp.) *bad subsequence CSS*, abbreviated *k-bounded* (resp. *bounded*) *b-CSS*, if Σ is a set of (k-bounded, bounded, resp.) bad subsequence constraints.

Our first major result in this section concerns the unsolvability of determining each of the above types of CSS.

Theorem 4.1. *It is recursively unsolvable to decide for an arbitrary (local) CSS T whether or not*

(a) *T is a b-CSS;*

(b) *T is a k-bounded b-CSS for given* $k \geq 1$;

(c) *T is a bounded b-CSS.*

Proof. Let $\Delta = \{a, b\}$, c and d be new symbols, and $\langle U \rangle = AB$, where

[8] It is easily seen that σ_2, σ_4, and σ_5 are not bad subsequence constraints.

$Dom(A) = \{1, 2\}$ and $Dom(B) = \Delta^* \cup \{c, d\}$. For arbitrary CFG G_1 and G_2 over Δ, let $T_{G_1G_2} = (\mathscr{C}, \{\sigma_{G_1G_2}\}, \mathscr{I})$ be defined as follows:

(i) $\mathscr{C} = (A, B, f_A)$, where $f_A(u) = 1$ if $u(A) = 2$ and $f_A(u) = 2$ if $u(A) = 1$;

(ii) $\text{VSEQ}(\sigma_{G_1G_2}) = \{u_1 \ldots u_m \mid 1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant m-1$, $u_i(B)$ is in $L(G_1) \cup \{c\}$ if $u_i(A) = 1$, $u_i(B)$ is in $L(G_2) \cup \{d\}$ if $u_i(A) = 2$, and $u_j(B) \neq u_{j+1}(B)$ if $u_j(A) \neq u_{j+1}(A)\}$; and

(iii) $\mathscr{I} = Dom(\langle U \rangle) \cap \text{VSEQ}(\sigma_{G_1G_2})$.

Then $T_{G_1G_2}$ is a (local) CSS. We first show that

(1) $\sigma_{G_1G_2}$ is a 1-bounded bad subsequence constraint if $L(G_1) \cap L(G_2) = \varnothing$; and

(2) $\sigma_{G_1G_2}$ is not a bad subsequence constraint if $L(G_1) \cap L(G_2) \neq \varnothing$.

Indeed, suppose that $L(G_1) \cap L(G_2) = \varnothing$. Then $\text{VSEQ}(\sigma_{G_1G_2}) = \text{VSEQ}(c(\mathscr{B}_{G_1G_2}))$, where

$$\mathscr{B}_{G_1G_2} = \{u \text{ in } Dom(\langle U \rangle) \mid u(A) = 1 \text{ and } u(B) \text{ is not in } L(G_1) \cup \{c\}\} \cup \{u \text{ in } Dom(\langle U \rangle) \mid u(A) = 2 \text{ and } u(B) \text{ is not in } L(G_2) \cup \{d\}\}.$$

Thus, (1) holds. Now suppose $L(G_1) \cap L(G_2) \neq \varnothing$. Let w be a word in $L(G_1) \cap L(G_2)$, $\bar{u} = (1, w)(2, d)(1, c)(2, w)$, and $\bar{v} = (1, w)(2, w)$. Suppose that $\sigma_{G_1G_2}$ is a bad subsequence constraint, i.e., there exists a subset $\mathscr{B}_{G_1G_2} \subseteq \text{SEQ}(\langle U \rangle)$ such that $\text{VSEQ}(\sigma_{G_1G_2}) = \text{VSEQ}(c(\mathscr{B}_{G_1G_2}))$. Two cases arise.

(α) $\bar{v}$ is in $\mathscr{B}_{G_1G_2}$. Clearly, $\bar{u}$ is in $\text{VSEQ}(\sigma_{G_1G_2})$. Thus, $\bar{u}$ is in $\text{VSEQ}(c(\mathscr{B}_{G_1G_2}))$. Since $\bar{v}$ is a subsequence of $\bar{u}$, $\bar{v}$ is not in $\mathscr{B}_{G_1G_2}$ (by definition of $c(\mathscr{B}_{G_1G_2})$), a contradiction.

(β) $\bar{v}$ is not in $\mathscr{B}_{G_1G_2}$. Clearly, $\bar{v}$ is not in $\text{VSEQ}(\sigma_{G_1G_2})$. Hence, $\bar{v}$ is not in $\text{VSEQ}(c(\mathscr{B}_{G_1G_2}))$. By definition of $c(\mathscr{B}_{G_1G_2})$, at least one subsequence of $\bar{v}$ is in $\mathscr{B}_{G_1G_2}$. Since both $(1, w)$ and $(2, w)$ are in $\text{VSEQ}(\sigma_{G_1G_2}) = \text{VSEQ}(c(\mathscr{B}_{G_1G_2}))$, neither $(1, w)$ nor $(2, w)$ is in $\mathscr{B}_{G_1G_2}$. Thus, $(1, w)(2, w) = \bar{v}$ is in $\mathscr{B}_{G_1G_2}$, a contradiction.

Since both cases yield contradictions, $\sigma_{G_1G_2}$ is not a bad subsequence constraint, and (2) holds.

From (1) and (2), it is obvious that

(3) $T_{G_1G_2}$ is a b-CSS iff $L(G_1) \cap L(G_2) = \varnothing$;

(4) $T_{G_1G_2}$ is a k-bounded b-CSS for given $k \geqslant 1$ iff $L(G_1) \cap L(G_2) = \varnothing$; and

(5) $T_{G_1G_2}$ is a bounded b-CSS iff $L(G_1) \cap L(G_2) = \varnothing$.

Standard reduction arguments involving (3), (4), (5), and (c) of Theorem 2A yield (a), (b), and (c) of the theorem. ∎

Analogous to Theorem 3.2, our next result shows the recursive unsolvability for finding the smallest k for which a bounded b-CSS is k-bounded.

THEOREM 4.2. *For each $k \geq 2$, it is recursively unsolvable to decide for an arbitrary k-bounded b-CSS T whether or not there exists some k', $1 \leq k' < k$, such that T is a k'-bounded b-CSS.*

Proof. Let A and f_A be as in Theorem 2.1. Let $\Delta = \{a, b\}$ and $\langle U \rangle = AB$, where $Dom(B) = \Delta^*$. For each $k \geq 2$ and each CFG G over Δ, let $T_{Gk} = (\mathscr{C}, \{\sigma_{Gk}\}, \mathscr{I}_k)$ be defined as follows:

(i) $\mathscr{C} = (A, B, f_A)$;

(ii) $\text{VSEQ}(\sigma_{Gk}) = \{u_1 \ldots u_m \mid$ for each i, $1 \leq i \leq m$, $u_i(B)$ is in $\Delta^* - L(G)$, and there is no $j_1, \ldots, j_k$, $1 \leq j_1 < \cdots < j_k \leq m$, such that $u_{j_1}(B) = \cdots = u_{j_k}(B)\}$; and

(iii) $\mathscr{I}_k = Dom(\langle U \rangle) \cap \text{VSEQ}(\sigma_{Gk})$.

Clearly, T_{Gk} is a CSS. And T_{Gk} is a k-bounded b-CSS, since $\sigma_{Gk} = c(\mathscr{B}_{Gk})$, where

$$\mathscr{B}_{Gk} = \{u \text{ in } Dom(\langle U \rangle) \mid u(B) \text{ is in } L(G)\} \cup \{u_1 \ldots u_k \mid u_1(B) = \cdots = u_k(B)\}.$$

We shall see that

(*) T_{Gk} is a k'-bounded b-CSS for some k', $1 \leq k' < k$, iff $L(G) = \Delta^*$.

By the standard reduction argument employing (b) of Theorem 2A, (*) implies the theorem.

Suppose $L(G) = \Delta^*$. Then $\text{VSEQ}(\sigma_{Gk}) = \varnothing$ and $\sigma_{Gk} = c(\mathscr{B}'_{Gk})$, where $\mathscr{B}'_{Gk} = Dom(\langle U \rangle)$. Hence, T_{Gk} is a 1-bounded b-CSS. Now suppose $L(G) \neq \Delta^*$, say w is in $\Delta^* - L(G)$. Assume that T_{Gk} is a k'-bounded b-CSS for some k', $1 \leq k' < k$. Then there exists a k'-bounded $\mathscr{B}''_{Gk} \subseteq \text{SEQ}(\langle U \rangle)$ such that $\sigma_{Gk} = c(\mathscr{B}''_{Gk})$. Let $\bar{u} = (1, w) \cdots (k, w)$. Since $\bar{u}$ does not satisfy σ_{Gk}, at least one subsequence of $\bar{u}$ is in $\mathscr{B}''_{Gk}$. However, each subsequence $\bar{u}'$ of $\bar{u}$, $\bar{u}' \neq \bar{u}$, satisfies σ_{Gk}, and thus is not in $\mathscr{B}''_{Gk}$. Therefore, $\bar{u}$ must be in $\mathscr{B}''_{Gk}$. This is a contradiction, since the length of $\bar{u}$ is k and $\mathscr{B}''_{Gk}$ is k'-bounded for some $k' < k$. Hence, there is no k', $1 \leq k' < k$ such that T_{Gk} is k'-bounded, and (*) is established. ∎

The same decision problem for an arbitrary local k-bounded b-CSS can be shown to be recursively unsolvable by slightly modifying σ_{Gk} given above. We omit the details.

Analogous to applying the concept of a local CSS to define local representability, we now use the notion of a b-CSS to obtain b-representability.

DEFINITION. A CSS $T=(\mathscr{C}, \Sigma, \mathscr{I})$ is (k-bounded, bounded, resp.) *b-representable* if there exists a CSS $T'=(\mathscr{C}, \Sigma', \mathscr{I})$ such that T' is a (k-bounded, bounded, resp.) b-CSS and $\mathrm{VSEQ}(T')=\mathrm{VSEQ}(T)$.

The next theorem establishes the unsolvability of b-representability.

THEOREM 4.3. *It is recursively unsolvable to decide for an arbitrary (local) CSS T whether or not*

(a) *T is b-representable*;

(b) *T is k-bounded b-representable for given $k \geqslant 1$*;

(c) *T is bounded b-representable.*

Proof. Let $T_{G_1G_2}$ be as in the proof of Theorem 4.1. We will show that

(1) $T_{G_1G_2}$ is b-representable iff $L(G_1) \cap L(G_2)=\varnothing$;

(2) $T_{G_1G_2}$ is k-bounded b-representable for given $k \geqslant 1$ iff $L(G_1) \cap L(G_2)=\varnothing$; and

(3) $T_{G_1G_2}$ is bounded b-representable iff $L(G_1) \cap L(G_2)=\varnothing$.

Then by (c) of Theorem 2A and standard reduction arguments, (a), (b), and (c) follow.

In view of the proof of Theorem 4.1, it suffices to show the "only if" parts.

Suppose that $L(G_1) \cap L(G_2) \neq \varnothing$. Let w be a word in $L(G_1) \cap L(G_2)$, $\bar{u}=(1, w)(2, d)(1, c)(2, w)$ and $\bar{v}=(1, w)(2, w)$. Clearly,

(4) $\bar{u}$ is in $\mathrm{VSEQ}(T_{G_1G_2})$,

(5) $\bar{v}$ is in $\mathrm{VSEQ}(\mathscr{C}) \cap \mathrm{VSEQ}(\mathscr{I})$, and

(6) $\bar{v}$ is not in $\mathrm{VSEQ}(T_{G_1G_2})$.

Suppose $T_{G_1G_2}$ is b-representable. Then there exists $T'_{G_1G_2}=(\mathscr{C}, \{\sigma'_{G_1G_2}\}, \mathscr{I})$ such that $\sigma'_{G_1G_2}=c(\mathscr{B}'_{G_1G_2})$ for some $\mathscr{B}'_{G_1G_2} \subseteq \mathrm{SEQ}(\langle U \rangle)$ and

(7) $\mathrm{VSEQ}(T'_{G_1G_2})=\mathrm{VSEQ}(T_{G_1G_2})$.

By (6) and (7),

(8) $\bar{v}$ is not in $\mathrm{VSEQ}(T'_{G_1G_2})$.

By (5) and (8),

(9) $\bar{v}$ is not in $\mathrm{VSEQ}(\sigma'_{G_1G_2})$.

Thus, at least one subsequence of $\bar{v}$ is in $\mathscr{B}'_{G_1G_2}$. Since each subsequence of $\bar{v}$ is also a subsequence of $\bar{u}$, it follows that $\bar{u}$ does not satisfy $c(\mathscr{B}'_{G_1G_2}) = \sigma'_{G_1G_2}$. Thus, $\bar{u}$ is not in $\mathrm{VSEQ}(T'_{G_1G_2}) = \mathrm{VSEQ}(T_{G_1G_2})$, which contradicts (4). Therefore, $T_{G_1G_2}$ is not b-representable and (1) holds.

Consider (2). By (1), it follows that

(10) If $L(G_1) \cap L(G_2) \neq \varnothing$, then there is no $k \geqslant 1$ such that $T_{G_1G_2}$ is k-bounded b-representable.

By (10), (2) holds.

Finally, (3) obviously holds by (10). ∎

We note in passing that for each $k \geqslant 2$, it is recursively unsolvable to decide for an arbitrary k-bounded b-representable CSS T whether or not there exists some k', $1 \leqslant k' < k$, such that T is k'-bounded b-representable. Indeed, given k let T_{Gk} be as in the proof of Theorem 4.2. It can readily be shown that T_{Gk} is k'-bounded b-representable for some k', $1 \leqslant k' < k$, iff $L(G) = \Delta^*$. The asserted result then follows by the usual reduction argument employing (b) of Theorem 2A.

5. Conclusions

We have shown that a number of decision problems about object histories are recursively unsolvable. Furthermore, the computational complexity of each of the components in the established CSS is very simple, namely, polynomial time bounded. (This follows from the fact that the recognition problem for context-free languages is polynomial time bounded (Harrison, 1978; Hopcroft and Ullman, 1979).)

We conclude by stating two open decidability questions of a philosophic nature:

(1) The CSS constructed in our proof tend toward the artificial. Can one find "natural" examples (in the sense of arising in real-life situations) of CSS for which the decidability questions are still unsolvable?

(2) Can one find a "natural" class of CSS for which (some of) the decision problems are solvable?

Received May 19, 1987; accepted February 1, 1989

References

Beeri, C., and Vardi, M. Y. (1984), Formal systems for tuple and equality generating dependencies, *SIAM J. Comput.* **13**, 76–98.

DONG, G., AND GINSBURG, S. (1986), "Localizable Constraints for Object Histories," Technical Report TR-86-217, Computer Science Dept., University of Southern California.

GINSBURG, S. (1966), "Mathematical Theory of Context-Free Languages," McGraw–Hill, New York.

GINSBURG, S., AND GYSSENS, M. (1987), Object histories which avoid certain subsequences, *Inform. and Comput.* **73**, 174–206.

GINSBURG, S., AND HULL, R. (1983), Order dependency in the relational model, *Theoret. Comput. Sci.* **26**, 149–195.

GINSBURG, S., AND TANAKA, K. (1986), Computation-tuple sequences and object histories, *ACM Trans. Database Systems* **11**, 186–212.

GINSBURG, S., AND TANAKA, K. (1984), Interval queries on object histories, *in* "Proceedings, of the 10th International Conference on Very Large Data Bases," pp. 208–217.

GINSBURG, S., AND TANG, C. (1986), Projection of object histories, *Theoret. Comput. Sci.* **48**, 297–328.

GINSBURG, S., AND TANG, C. (1989), Cohesion of object histories, *Theoret. Comput. Sci.* **63**, 63–90.

HARRISON, M. (1978), "Introduction to Formal Language Theory," Addison–Wesley, Reading, MA.

HOPCROFT, J., AND ULLMAN, J. D. (1979), "Introduction to Automata Theory, Languages, and Computation," Addison–Wesley, Reading, MA.

MACHTEY, M., AND YOUNG, P. (1978), "An Introduction to the General Theory of Algorithms," North-Holland, Amsterdam.

Printed in Belgium