JOURNAL OF COMPUTER AND SYSTEM SCIENCES **17**, 270–279 (1978)

Candidate Keys for Relations*

CLÁUDIO L. LUCCHESI

Universidade Estadual de Campinas, 13100 Campinas, SP, Brazil

AND

SYLVIA L. OSBORN

University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1

Received August 4, 1976; revised February 21, 1978

An algorithm is presented that finds **K**, the set of all keys for a given set A of attribute names and a given set $D[0]$ of functional dependencies, in time polynomial in $|A|$, $|D[0]|$ and $|\mathbf{K}|$. It is shown that the problem of deciding whether or not there is a key having cardinality not greater than a specified integer is NP-complete. It is also shown that the problem of deciding whether or not a specified attribute name is prime is NP-complete.

1. INTRODUCTION

The relational model for data was first introduced by Codd [4]. In this model, data is represented by tables containing indivisible data values. The rows of the table correspond to entities being described, and the columns contain the attributes by which they are described. Associated with each column is an attribute name which uniquely identifies the column.

Consider, for example, a student record file in which a table could have the attribute names: name, ID-number, course, professor and time. In such a situation, knowledge of a student's name and a course may imply knowledge of the professor teaching the course and the time it is given. This dependency can be denoted by {name, course} → {professor, time}. A more trivial dependency would be {name, course} → {name}.

The following nontrivial dependencies could be specified for this set of attribute names:

1. {name} → {ID-number},
2. {ID-number} → {name},
3. {name, course} → {professor, time},
4. {professor, time} → {course}.

* This research was supported in part by the National Research Council of Canada.

0022-0000/78/0172-0270$02.00/0

Given this set of dependencies, observe that knowing a name and a course implies knowing all the attributes for that entity. In such a case, {name, course} is called a key relative to these dependencies.

To formalize these ideas, let A be a finite set of elements called *attribute names* and let 2^A denote the power set of A. Let $D[0]$ be a binary relation on 2^A, each member of which is called a *functional dependency*. Define D, *the projective, transitive and additive closure of* $D[0]$, to be

$$\bigcup_{i \geqslant 0} D[i],$$

where each $D[i]$, $i \geqslant 1$ is inductively defined as follows:

1. Projectivity: for all subsets E and F of A, if E includes F then $E \text{—}(D[i])\text{→} F$.[1]
2. Transitivity: for all subsets E, F and G of A, if $E \text{—}(D[i-1])\text{→} F$ and $F \text{—}(D[i-1])\text{→} G$ then $E \text{—}(D[i])\text{→} G$.
3. Additivity: for all subsets E, F and G of A, if $E \text{—}(D[i-1])\text{→} F$ and $E \text{—}(D[i-1])\text{→} G$ then $E \text{—}(D[i])\text{→} F \cup G$.
4. No ordered pairs other than those given by 1–3 lie in $D[i]$.

Note that for all $i \geqslant 1$, $D[i-1]$ is a subset of $D[i]$. Also note that, because of additivity, we can assume a *canonical form* for functional dependencies in $D[0]$, in which, for each (L, R) in $D[0]$, $| R | = 1$.

A subset K of A is a *key* for system $\langle A, D[0]\rangle$ if $K \text{—}(D)\text{→} A$. An attribute name is *prime* relative to $\langle A, D[0]\rangle$ if it lies in some minimal key for $\langle A, D[0]\rangle$.[2] Minimal keys are also called *candidate keys* [5] in the literature.

For binary relation D' on 2^A and subset B of A, if D' contains an ordered pair (L, R) such that B includes L but not R, then B is *D'-expansible* and $B \cup R$ is a *D'-expansion* of B.

It is shown in [10] that the number of minimal keys for a relational system $\langle A, D[0]\rangle$ can be factorial in $| D[0] |$ or exponential in $| A |$, and that both of these upper bounds are attainable. However, in many practical cases, e.g. payroll applications, there are only one or two minimal keys. A family of examples is given in Appendix I where each $\langle A, D[0]\rangle$ in that family has exactly one minimal key but each of the algorithms presented previously [1, 3, 7, 8] requires time exponential either in $| A |$ or in $| D[0] |$ to find this one key. The algorithm presented in this paper finds $\mathbf{K}$, the set of all minimal keys for a given $\langle A, D[0]\rangle$, in time polynomial in $| A |$, $| D[0] |$ and $| \mathbf{K} |$. Thus, when there are only a few keys, the algorithm given here would behave much better than previous algorithms.

We show that the problem of deciding whether or not there is a key of cardinality less than or equal to a specified integer is NP-complete [6, 9]. We refer to this problem

[1] Throughout the paper, $E \text{—}(D)\text{→} F$ (or simply $E \rightarrow F$ if D is understood), is used as an abbreviation of "ordered pair (E,F) lies in binary relation D".

[2] If a set, but none of its proper subsets, satisfies a property, then the set is *minimal* with that property.

as the *key of cardinality m problem.* We also show that the problem of deciding whether or not a specified attribute name is prime is NP-complete. We refer to this problem as the *prime attribute name problem.*

Finding minimal keys is related to finding third normal forms for relational data bases [5]. For $\langle A, D[0]\rangle$ to be in third normal form, the nonprime attribute names must obey certain properties. In those cases where the number of minimal keys approaches the upper bounds mentioned above, one might wish to determine the union of all minimal keys, in other words the set of prime attribute names. The NP-completeness of the prime attribute name problem seems to suggest that even this is intractable.

2. Preliminary Results

The following lemma follows directly from the definition of D:

Lemma 1. *For subset B of A and $D[0]$-expansion B' of B, $B \xrightarrow{(D)} B'$ and $B' \xrightarrow{(D)} B$.*

Corollary. *For subset B of A and $D[0]$-expansion B' of B, B is a key if and only if B' is a key.*

Lemma 2. *For each i, a subset of A is $D[i]$-expansible if and only if it is $D[0]$-expansible. Consequently, a subset of A is D-expansible if and only if it is $D[0]$-expansible.*

Proof. The proof of sufficiency is by induction. Clearly for $i = 0$ the assertion holds. Assume it is true for all $i < k$. To prove that it follows for $i = k$, assume there is a subset B of A that is $D[k]$-expansible. That is, there is some (L, R) in $D[k]$ such that B includes L but not R. Assume also that (L, R) is not in $D[k-1]$. (L, R) cannot be derived by projectivity since this would not lead to an expansion. If it is derived by transitivity, then there are two functional dependencies in $D[k-1]$, (L, X) and (X, R) from which it is derived. If X is a subset of B, by the existence of (X, R), B is $D[k-1]$-expansible and thus, by the induction hypothesis, $D[0]$-expansible. If X is not contained in B then (L, X) makes B, $D[k-1]$-expansible and thus $D[0]$-expansible. If (L, R) is derived by additivity then there are two functional dependencies (L, X) and (L, Y) in $D[k-1]$ such that $R = X \cup Y$. Since R is not contained in B, at least one of X and Y is not contained in B. If it is X that is not contained in B, then the existence of (L, X) in $D[k-1]$ makes B, $D[k-1]$-expansible and, by the induction hypothesis, $D[0]$-expansible.

The converse follows from the fact that, for all i, $D[i-1]$ is contained in $D[i]$.

Corollary. *If a proper subset of A is a key for $\langle A, D[0]\rangle$, then it is $D[0]$-expansible.*

In the following lemma an algorithm is presented and its time complexity is given. For this algorithm and those that follow, we assume that the elementary step being counted is the comparison of two attribute names. Thus if we assume that subsets of A are represented as sorted lists of attribute names, then a Boolean operation on two subsets of A requires at most $|A|$ elementary steps.

The algorithm uses as a subroutine (called Key here) Bernstein and Beeri's algorithm to determine whether a given functional dependency is in the closure of a given set of functional dependencies [2]. Beginning with a given left-hand side, B, their algorithm constructs a set called DEPEND containing all attribute names in any right-hand side of a functional dependency in the closure with B as its left-hand side. Their algorithm ends by testing whether or not DEPEND contains a specific attribute name. To use this algorithm to determine if a given set is a key, we simply need to change this test to:

```
if DEPEND = $A$
    then return "true"
    else return "false".
```

Bernstein and Beeri state that this algorithm runs in time linear in the length of the input, which in their case is linear in the length of the description of the functional dependencies (in canonical form). We prefer to analyze algorithms in terms of $|A|$ and $|D[0]|$. Note that, for functional dependencies in canonical form, the length of the functional dependencies is bounded by $|A| * |D[0]|$, so there is a direct correspondence between their analysis and ours.

For key K of $\langle A, D[0]\rangle$, attribute name b of K is *essential* to K if $K - \{b\}$ is not a key for $\langle A, D[0]\rangle$. We then have

Lemma 3. *For key K of $\langle A, D[0]\rangle$, subset K' of K and attribute name b of K, if b is essential to K and K' is a key then b lies in K' and is essential to K'.*

Corollary. *The following algorithm determines a minimal key for $\langle A, D[0]\rangle$ that is a subset of a specified key K. Moreover, it requires $O(|D[0]|\,|A|^2)$ elementary operations.*

Algorithm. Minimal Key $(A, D[0], K)$:

```
$K' \leftarrow K$;
for each attribute name $b$ in $K$ do
    Comment if $b$ is nonessential to $K'$ then delete it from $K'$;
    if Key $(A, D[0], K' - \{b\})$
        then $K' \leftarrow K' - \{b\}$;
return $K'$.
```

3. An Algorithm for Determining the Set of Minimal Keys

The following lemma gives the condition on which the algorithm is based.

Lemma 4. *For nonnull set $\mathbf{K}$ of minimal keys for $\langle A, D[0]\rangle$, $2^A - \mathbf{K}$ contains a minimal key if and only if $\mathbf{K}$ contains a key K and $D[0]$ contains an ordered pair (L, R) such that $L \cup (K - R)$ does not include any key in $\mathbf{K}$.*

Proof. To prove that the asserted condition is sufficient for $2^A - \mathbf{K}$ to contain a minimal key, assume that the condition holds for K in $\mathbf{K}$ and (L, R) in $D[0]$. Since $L \cup K \cup R$ includes K, $L \cup K \cup R$ is a key. Moreover, $L \cup (K - R) \rightarrow L \cup K \cup R$. Thus $L \cup (K - R)$ is a key; therefore it includes a minimal key, say K'. Since $L \cup (K - R)$ does not include any key in $\mathbf{K}$, K' cannot already be in the set $\mathbf{K}$. That is, minimal key K' lies in $2^A - \mathbf{K}$.

To prove that the asserted condition is necessary, assume that $2^A - \mathbf{K}$ contains a minimal key K'. Define K'' to be a maximal subset of A that includes K' but does not include any key in $\mathbf{K}$.[3] Since $\mathbf{K}$ is nonnull, K'' is a proper subset of A. Moreover, since K' is a key, so is K''. By the corollaries of Lemmas 1 and 2, $D[0]$ contains an element, say (L, R), such that $K'' \cup R$ is a key and K'' includes L but does not include R. By the choice of K'', $K'' \cup R$ includes a key in $\mathbf{K}$, say K. That is, K'' includes both L and $K - R$; therefore it includes $L \cup (K - R)$. Since K'' does not include any key in $\mathbf{K}$, neither does $L \cup (K - R)$. This completes the proof of Lemma 4.

The lemma requires a nonnull set of keys before the test can be applied. Thus the algorithm starts by isolating one minimal key from A using the Minimal Key algorithm. It then proceeds to find the other keys by testing each key found with each of the pairs in $D[0]$, using Minimal Key to isolate a new Key whenever the test is satisfied.

COROLLARY. *The following algorithm determines the set of minimal keys for* $\langle A, D[0] \rangle$.

```
Algorithm. Set of Minimal Keys ($A$, $D[0]$);
Comment $\mathbf{K}$ is the set of minimal keys being accumulated. It is assumed that these
are accumulated in a sequence which can be scanned in the order in which the keys
are entered;
$\mathbf{K} \leftarrow \{\text{Minimal Key}(A, D[0], A)\}$;
for each $K$ in $\mathbf{K}$ do
  for each pair $(L, R)$ in $D[0]$ do
    $S \leftarrow L \cup (K - R)$;
    test ← true;
    for each $J$ in $\mathbf{K}$ do
      if $S$ includes $J$ then test ← false
    if test then $\mathbf{K} \leftarrow \mathbf{K} \cup \{\text{Minimal Key}(A, D[0], S)\}$
  end
end;
return $\mathbf{K}$.
```

Observe that the algorithm takes $O(|D[0]|\,|\mathbf{K}|\,(|A| + |\mathbf{K}|\,|A|)) + O(|\mathbf{K}|m)$ elementary operations, where m is the complexity of Minimal Key. That is, the algorithm has time complexity $O(|D[0]|\,|\mathbf{K}|\,|A|\,(|\mathbf{K}| + |A|))$.

[3] By *maximal* we mean there is no subset J of A which properly includes K'' that also has this property.

Referring to the example in section 1, $K = \{ID\text{-number, professor, time}\}$ would be the minimal key isolated from A by the first call of Minimal Key. Then this minimal key is tested against each member of $D[0]$. With the first member $\{\text{name}\} \to \{ID\text{-number}\}$, S is assigned {name, professor, time}. By Lemma 4, this is a key, and after checking it with Minimal Key, we find it is also minimal. For this value of K, one other minimal key is subsequently found, namely $\{ID\text{-number, course}\}$. Then K is assigned {name, professor, time}, the second entry in **K**. When it is tested with the second member of $D[0]$, $\{ID\text{-number}\} \to \{\text{name}\}$, S is assigned $\{ID\text{-number, professor, time}\}$. This includes a minimal key already in **K**, and thus the condition in Lemma 4 is not satisfied.

When the algorithm terminates, **K** contains the following minimal keys:

$\{ID\text{-number, professor, time}\}$,

{name, professor, time},

$\{ID\text{-number, course}\}$,

{name, course}.

4. The Key of Cardinality *m* Problem

In the final two sections we are going to show that two problems related to candidate keys belong to the class of *NP*-complete problems [6, 9]. Problems in this class can all be solved, or more precisely the corresponding languages can be recognized, in polynomial time on a nondeterministic Turing machine. It is an open problem whether or not they can be recognized in polynomial time on a deterministic Turing machine; the current feeling is that they cannot. In fact, all known algorithms for these problems are exponential.

There are two aspects to proving that a language L is *NP*-complete. The first is to exhibit a nondeterministic polynomial time algorithm for recognizing L. This shows that L lies in *NP*. The second part of such a proof involves transforming a known *NP*-complete problem into L. This transformation must itself be polynomial time so that a polynomial algorithm for recognizing L, if ever found, would yield a polynomial algorithm for all *NP*-complete languages.

The key of cardinality m problem can be stated as follows: given a set A of attribute names, binary relation $D[0]$ on 2^A and integer m, decide whether or not there exists a key for $\langle A, D[0]\rangle$ having cardinality less than or equal to m.

Theorem 1. *The key of cardinality m problem is NP-complete.*

Proof. The problem lies in *NP*. Nondeterministically generate a subset of A, say K, and then verify whether K is a key containing no more than m attributes. Since algorithm Key is polynomial, this algorithm is nondeterministic polynomial.

To complete the proof of the theorem, it now suffices to prove that the vertex cover problem, an *NP*-complete problem stated below, is polynomially transformable into the key of cardinality m problem.

The vertex cover problem: given integer m and graph G having vertex set $V(G)$ and edge set $E(G)$, decide whether or not G has a vertex cover having cardinality not greater than m. A *graph* G consists of elements called *edges*, together with a relation of *incidence* that associates two vertices of $V(G)$ with each edge α of $E(G)$, called the *ends* of α. Each edge α in $E(G)$ is *incident* upon its ends. A *vertex cover* K of $E(G)$ is a subset of $V(G)$ such that each edge of $E(G)$ is incident upon some vertex in K. Vertex b of $V(G)$ is *adjacent* to vertex c of $V(G)$ if $E(G)$ contains an edge having b and c as its ends [11]. The vertex cover problem is *NP*-complete [9].

To transform this into the corresponding key of cardinality m problem, define A to be $V(G)$ and $D[0]$ to be $\{Nv \rightarrow \{v\}$: v is in $V(G)\}$ where Nv denotes the set of vertices in $V(G)$ that are adjacent to v.

Observe that $\langle A, D[0]\rangle$ can be determined in time polynomial in $|V(G)|$ and $|E(G)|$. Note also that $|A| = |V(G)| = |D[0]|$. Moreover, by Lemma 5, asserted below, the vertex cover problem is polynomially transformable into the key of cardinality m problem.

LEMMA 5. *Subset K of A is a key for $\langle A, D[0]\rangle$ if and only if it is a vertex cover of $E(G)$ in G.*

Proof. Assume as inductive hypothesis that the assertion holds for each subset K' of A that properly includes K. If K is equal to A then the assertion holds trivially. Assume therefore that K is a proper subset of A. By Lemmas 1 and 2, K is a key for $\langle A, D[0]\rangle$ if and only if it has a $D[0]$-expansion that is a key for $\langle A, D[0]\rangle$. By construction of $D[0]$, K has a $D[0]$-expansion if and only if $V(G) - K$ contains a vertex, say v, such that K includes Nv. By the induction hypothesis, $K \cup \{v\}$ is a vertex cover of $E(G)$ in G. That is, proper subset K of A is a key for $\langle A, D[0]\rangle$ if and only if it is a vertex cover of $E(G)$ in G, as asserted. The proof of Lemma 5 completes the proof of Theorem 1.

5. THE PRIME ATTRIBUTE NAME PROBLEM

Given a set A of attribute names, binary relation $D[0]$ on 2^A and attribute name b in A, decide whether or not b is prime relative to $\langle A, D[0]\rangle$.

THEOREM 2. *The prime attribute name problem is NP-complete.*

Proof. The problem lies in *NP*. Nondeterministically generate a subset of A and then verify whether it is a minimal key that contains b. To complete the proof, it suffices to prove that the key of cardinality m problem is polynomially transformable into the prime attribute name problem. For this, assume that set A', binary relation $D[0]'$ and integer m have been given as input to the key of cardinality m problem. Define sets A, $D[0]$ and $\{b\}$ for the corresponding prime attribute name problem as follows: Define A to be $A' \cup$

$[A'' \times A'] \cup \{b\}$, where A'' is a set whose cardinality is the smaller of $|A'|$ and m, and b is a "new" attribute name not in $A' \cup [A'' \times A']$. Define binary relation $D[0]$ on 2^A as follows:

(i) for each pair E, F of subsets of A', if $E \text{—}(D[0]')\!\rightarrow F$ then $\{b\} \cup E \rightarrow F$,

(ii) $\{b\} \cup A' \rightarrow A'' \times A'$,

(iii) for each element i of A'' and each element e of A', $\{b\} \cup \{(i, e)\} \rightarrow \{e\}$,

(iv) for each element i of A'' and for each pair e, f of distinct elements of A', $\{(i, e), (i, f)\} \rightarrow \{b\}$,

(v) for each element e of A', $\{e\} \rightarrow \{b\}$,

(vi) no ordered pairs other than those given by (i)–(v) lie in $D[0]$.

Observe that $|A''| \leqslant |A'|$, $|A| \leqslant |A'|^2 + |A'| + 1$ and $|D[0]| \leqslant |D[0]'| + 1 + |A'|^2 + |A'|^3 + |A'|$. Thus, by Lemma 6 stated below, the key of cardinality m problem is polynomially transformable into the prime attribute name problem.

LEMMA 6. *Attribute name b is prime relative to $\langle A, D[0]\rangle$ if and only if $\langle A', D[0]'\rangle$ has a key of cardinality not greater than m.*

Proof. Consider first the case where $\langle A, D[0]\rangle$ has a minimal key, K, that contains b. By the minimality of K, and in view of (iv) and (v) above, $K = \{b\} \cup \{(i[1], c[1]),\dots, (i[n], c[n])\}$, where $i[1],\dots, i[n]$ are n distinct elements of A'', $c[1],\dots, c[n]$ are elements of A' and $n \geqslant 0$. We assert that $\{c[1],\dots, c[n]\}$, denoted by K', is a key for $\langle A', D[0]'\rangle$. To prove this, let L' denote a maximal subset of A' such that (K', L') lies in the closure of $D[0]'$. If L' is equal to A', then K' is indeed a key for $\langle A', D[0]'\rangle$. Assume thus that L' is a proper subset of A'. By the choice of L' and by Lemmas 1 and 2, L' is a proper subset of A' that includes K' but is not $D[0]'$-expansible. Thus, $K \cup L'$ is a proper subset of A that includes K but is not $D[0]$-expansible. By Lemma 2, K is not a key for $\langle A, D[0]\rangle$, a contradiction. As asserted, K' is a key for $\langle A', D[0]'\rangle$. Indeed, $|K'| \leqslant n \leqslant |A''| \leqslant m$. As asserted, if b is prime relative to $\langle A, D[0]\rangle$, then $\langle A', D[0]'\rangle$ has a key having cardinality not greater than m.

To prove the converse, assume that $\langle A', D[0]'\rangle$ has a key, denoted K', having cardinality n not greater than m. Denote the elements of K' by $c[1],\dots, c[n]$. Since $n \leqslant |A'|$ and $|A''| = \min(|A'|, m)$, A'' contains n distinct elements: denote them $i[1],\dots, i[n]$. We assert that $\{b\} \cup \{(i[1], c[1]),\dots, (i[n], c[n])\}$, denoted K, is a key for $\langle A, D[0]\rangle$. By (iii) above, $K \rightarrow \{b\} \cup K'$ is in the closure of $D[0]$. Since $K' \rightarrow A'$ is in the closure of $D[0]'$, then, by (i) above, $\{b\} \cup K' \rightarrow \{b\} \cup A'$ is in the closure of $D[0]$. By (ii) above, $\{b\} \cup A' \rightarrow A'' \times A'$ is in $D[0]$. Thus, by transitivity and additivity, $K \rightarrow \{b\} \cup A' \cup A'' \times A'$, or K is a key for $\langle A, D[0]\rangle$.

Finally, $K - \{b\}$ is a proper subset of A that is not $D[0]$-expansible; therefore b is essential to K. By Lemma 3, K includes a minimal key for $\langle A, D[0]\rangle$ that includes b. That is, b is prime relative to $\langle A, D[0]\rangle$. The proof of Lemma 6 completes the proof of Theorem 2.

6. Conclusions

We have discussed the computational complexity of finding minimal keys for a single relation given its attribute names and a set of functional dependencies. An algorithm has been given which finds all minimal keys of a relation in time polynomial in the number of keys, even when the number of keys is exponential in $|A|$ and $|D[0]|$. We have shown that two problems related to finding minimal keys are NP-complete.

Finding the minimal keys of a system $\langle A, D[0]\rangle$ is related to finding a third normal form for such a system. A third normal form algorithm would break A into subsets $A[i]$ such that each $A[i]$ together with all relevant functional dependencies is in third normal form. Finding keys for these sub-relations is complicated by the fact that those functional dependencies containing attribute names in $A[i]$ may also contain attribute names not in $A[i]$; therefore the algorithm presented here cannot be applied directly. Further discussion on finding keys for sub-relations, and then using them to find third normal forms, can be found in [10].

Appendix I

The following example has 8 attribute names and 4 functional dependencies. It can be generalized to larger examples where $|A| = 2\,|D[0]|$. Let $A = \{a,..., h\}$ and let $D[0]$ contain:

$$\{a\} \to \{b\},$$
$$\{c\} \to \{d\},$$
$$\{e\} \to \{f\},$$
$$\{g\} \to \{h\}.$$

This system has exactly one minimal key, namely $\{a, c, e, g\}$.

Bernstein's approach is to test, for all subsets A' of A, whether A' is a key and whether it is minimal [1, 2]. This algorithm checks $O(2^{A})$ subsets of A, regardless of the number of keys found.

For this example, Delobel and Casey's algorithm involves finding the prime implicants for the following Boolean function:

$$F = abcdefgh + a\bar{b} + c\bar{d} + e\bar{f} + g\bar{h}.$$

They show that the prime implicants with no complemented variables correspond to minimal keys. Known algorithms for generating prime implicants run in time exponential in the number of variables, or in this case, in time exponential in $|A|$.

Fadous and Forsyth's algorithm is not analyzed nor is it easy to deduce its worst-case behaviour. For examples in the family described above, one of the temporary sets used by this algorithm, $T2$, gives some idea of how it works. The first time $T2$ is constructed, it contains $\binom{|D[0]|}{2}$ elements after elements which are supersets of others have been discarded. The next time $T2$ is constructed it contains $\binom{|D[0]|}{3}$ elements after supersets are discarded. The pattern continues, and in general, the number of operations required to maintain $T2$ alone is more than

$$\sum_{i=2}^{|D[0]|} \binom{|D[0]|}{i}$$

or more than $2^{|D[0]|}$ steps.

Acknowledgments

We would like to thank Frank Tompa, David Taylor and Robert Moenck for so carefully reading earlier versions of the paper. We would also like to thank one of the referees for suggesting improvements to the proof of Lemma 6.

References

1. P. A. Bernstein, "Normalization and Functional Dependencies in the Relational Data Base Model," Ph.D. Dissertation, University of Toronto, 1975.
2. P. A. Bernstein and C. Beeri, "An Algorithmic Approach to Normalization of Relational Data Base Systems," CSRG 73, University of Toronto, 1976.
3. P. A. Bernstein, J. R. Swenson, and D. C. Tsichritzis, A unified approach to functional dependencies and relations, *in* "Proc. ACM-SIGMOD International Conference on Management of Data, 1975," pp. 237–245.
4. E. F. Codd, A relational model of data for large shared data banks, *Comm. ACM* **13** No. 6 (1970), 377–387.
5. E. F. Codd, Further normalization of the data base relational model, *in* "Courant Computer Science Symposium 6, Data Base Systems," (R. Rustin, Ed.), pp. 33–64, Prentice–Hall, 1971.
6. S. A. Cook, The complexity of theorem proving procedures, *in* "Proc. 3rd Annual ACM Symposium on Theory on Computing, 1971," pp. 151–158.
7. C. Delobel and R. G. Casey, Decomposition of a data base and the theory of Boolean switching functions, *IBM J. Res. Develop.* **17** No. 5 (1973), 374–386.
8. R. Fadous and J. Forsyth, Finding candidate keys for relational data bases, *in* "Proc. ACM-SIGMOD International Conference on Management of Data, 1975," pp. 203–210.
9. R. M. Karp, Reducibility among combinatorial problems, *in* "Complexity of Computer Computations," (R. E. Miller and J. W. Thatcher, Eds.), pp. 85–103, Plenum, New York, 1972.
10. S. L. Osborn, "Normal Forms for Relational Data Bases," Ph.D. Dissertation, University of Waterloo, 1977.
11. W. T. Tutte, "Connectivity in Graphs," University of Toronto Press, 1966.

Printed in Belgium