Reflections on Boyce-Codd Normal Form

Carol Helfgott LeDoux [*]
The Aerospace Corporation
El Segundo, California

D. Stott Parker, Jr. [*]
Computer Science Department
University of California, Los Angeles

ABSTRACT

The usefulness of Boyce-Codd Normal Form (BCNF) has been questioned by various researchers. A recent study showed that under common assumptions, BCNF no longer guarantees freedom from various 'anomalies', one of its purported virtues. Second, a BCNF covering is sometimes unattainable, i.e., some sets of FDs have no corresponding BCNF schema. Third, it is difficult to determine whether a given schema violates BCNF (doing so is known to be NP-complete).

This paper reviews the intent of the normal form, and suggests that these arguments may be discounted. We show that

- BCNF still provides a useful design criterion when such assumptions as the Universal Instance Assumption are dropped (although BCNF can be improved upon by taking into consideration the dynamic use of the database);
- even in schemata where BCNF is 'unattainable', BCNF can be attained by unconventional means, typically by renaming or adding attributes to better capture the semantic content of the data;
- testing violation of BCNF seems to require exponential time only in the worst case where the set of 'interesting' dependencies (nontrivial dependencies in minimal form) in the dependency closure grows exponentially large. Such situations do not seem to typify the 'real world': We investigate a model of FD schemata, called the *FD hierarchy model*, similar to many other data models proposed in recent years. For this model the FD closure is always small, and testing violation of BCNF is not only not NP-complete, it is linear in the size of the input. We also point out a relationship between keys, FDs which violate BCNF, and FD closures.

[*] Research supported by NSF grant IST 80-12418.

1. Introduction

An important database design problem is the development of a set of relations for the database which capture the underlying data semantics as accurately as possible. To the extent it can be achieved, the designer should attempt to 'normalize' the database into semantically independent components. One particular stopping point in the normalization process is known widely as 'Boyce-Codd Normal Form' (BCNF).

Unfortunately, BCNF has certain undesirable properties as a normal form: not all relation schemata have a BCNF representation, although they always may be decomposed into the less stringent 'Third Normal Form' (3NF). Also, several results have been presented which question the practical usefulness of BCNF as a logical database design goal. First, a number of questions involving both BCNF and keys have been shown to be NP-complete [Beeri & Bernstein, 1979]. Second, in [Bernstein & Goodman, 1980] it was demonstrated that BCNF does not prevent 'anomalous' behavior if the database is required to satisfy what is now termed 'the Universal Instance Assumption' [the condition whereby all tables (relation instances) in the database are required to be projections of a single universal table (universal instance), which is defined over all attributes]. This paper tries to put these criticisms in perspective.

There are certainly drawbacks in using simple normal forms. No existing normal form incorporates, in any way, the intended dynamic behavior of the database, although this behavior is important for the design. Moreover, all normal forms are defined in terms of dependencies. Not all semantics fall into this mold, and information can be lost when semantics are encoded into dependencies.[**] Normal forms therefore

[**] It is pointed out in [Atzeni & Parker, 1982] that dependency inferences may not be meaningful, first of all. For example, consider the dependencies

COMPANY → *ADDRESS* (companies have a plant address)
ADDRESS → *RESIDENT* (residential addresses have an owner)

suggesting by transitivity that each company has a unique resident! Second, dependencies usually require an artificial 'relationship uniqueness' assumption: Since functional dependencies specify only the attribute sets on provide no guarantee that a design is correct or appropriate for the application at hand. At best, normal forms provide guidelines in the logical design phase, and should be complemented with some deeper understanding of the application semantics.

However, BCNF does provide a useful design benchmark. In essence it expresses the design maxim, 'one fact in one place'***, assuming that all facts are many-one relationships. We present three arguments supporting its usefulness. First, we argue that the problems with BCNF noted by [Bernstein & Goodman, 1980] do not apply if we drop the Universal Instance Assumption, and that BCNF does then eliminate certain kinds of anomalous behavior. Second, we show that BCNF *can* be attained in many situations where it appears unreachable, by renaming schema attributes in such a way as to re-express semantics properly. Finally, we present simple algorithms for determining BCNF properties. The algorithms have exponential running time only in the pathological case where the given set of dependencies has a set of 'interesting' FDs (nontrivial dependencies in minimal form) in its closure which is exponentially large. Thus, even the NP-completeness results here may not be insurmountable.

2. Boyce-Codd Normal Form

BCNF and 3NF correspond to restrictions on the *functional dependencies* (FDs) which hold in a schema. A functional dependency is an integrity constraint among data attributes in a database. For example, the functional dependency $E \to C$ may represent the constraint that each employee, E, works for only one company, C. A relation schema with functional dependencies is said to be in Boyce-Codd Normal Form if all the dependencies are either *trivial* dependencies (such as $E \to E$ or $EC \to E$), or those in which a key functionally determines one or more attributes. Henceforth given any FD $f: X \to Y$, we will use $LHS(f)$ (Left-Hand-Side of f) and X interchangeably, and $RHS(f)$ and Y interchangeably. Typically $\|RHS(f)\| = 1$, i.e., $RHS(f)$ will consist of a single attribute. ($\|S\|$ denotes the cardinality of a set S).

We can formally define a *relation schema* $R = R\langle U,F\rangle$ as a set of attributes U and a set of functional dependencies F defined on U. Throughout this paper we will assume that F contains only functional dependencies.

A *database schema* is a collection of one or more relation schemata. A *key* for $R\langle U,F\rangle$ is a set of attributes K such that the FD $K \to U$ may be inferred from F. A *candidate key* is a key K such that no proper subset of K is also a key. A relation schema R with dependencies F is in *Boyce-Codd Normal Form* if whenever the non-trivial FD $X \to Y$ holds in R, then X includes a key for R. That is, a set of attributes is either a key or it does not determine any other attribute. For more discussion, see [Date, 1981] or [Ullman, 1980]. Any FD $X \to Y$ such that X does not contain a key is called a *non-Boyce-Codd functional dependency* (abbr. *non-BCFD).*

The relation schema $R\langle\{A,B,C\},\{A \to B, A \to C\}\rangle$ is in BCNF because A is a key, and neither B nor C determines any other attributes in R. On the other hand, the relation schema $R\langle\{A,B,C\},\{A \to B, B \to C\}\rangle$ is not in BCNF because B is not a key, yet it determines C. $B \to C$ is a non-BCFD. If R is decomposed into a pair of relation schemata $R_1\langle\{A,B\},\{A \to B\}\rangle$ and $R_2\langle\{B,C\},\{B \to C\}\rangle$, each of the schemata R_1, R_2 is in BCNF.

The *closure* F^+ of a set of dependencies F is the set of all dependencies which may be inferred from those in F via one or more applications of inference rules. A complete set of rules is Armstrong's first three axioms:

FD1	If $Y \subseteq X \subseteq U$, then $X \to Y$
FD2	If $X \to Y$ and $Z \subseteq U$, then $XZ \to YZ$
FD3	If $X \to Y$ and $Y \to Z$, then $X \to Z$

A good introduction to the subject may be found, for example, in [Ullman, 1980].

A *cover* F_0 of a set of dependencies F is a subset of F from which all of F may be inferred. It is easily verified that $F_0^+ = F^+$. Such a set F_0 is a *minimal cover* if no proper subset of F_0 is also a cover.

A *BCNF covering of a schema* $R\langle U,F\rangle$ is a database schema $\{R_i\langle U_i,F_i\rangle \mid i=1,\dots,n\}$ such that

(1) Every attribute in U appears in at least one of the sets U_i

(2) F_i^+ includes all dependencies of F^+ whose attributes are drawn from U_i

(3) The union of the F_i sets comprises a cover for F

(4) Each R_i is in BCNF.

If the union of the F_i sets is additionally a minimal cover for F, the database schema $\{R_i\langle U_i,F_i\rangle \mid i=1,\dots,n\}$ is called a *minimal BCNF covering*.

Clearly, whenever a minimal BCNF covering exists, a BCNF covering exists also. However, it is not always possible to find a BCNF covering. An example which is typically given to illustrate this is the schema $R\langle\{C,S,Z\},\{CS \to Z, Z \to C\}\rangle$ relating zip codes, Z, street addresses, S, and cities, C. R is not in BCNF and there is no BCNF covering for R.

which they hold, any two dependencies relating X to Y should be equivalent. Thus, if we have three functional dependencies

$f: EMP \to MGR$, $g: MGR \to SAL$, and $h: EMP \to SAL$,

we must assume that h is equivalent to the functional composition of g and f, although this might not be what was intended. Cautious choice of attribute names can avoid such problems.

*** Words of J.N. Gray.

3. BCNF and Update Anomalies

In this section we review the intent of database normalization. It is widely held that normalization eliminates various 'storage anomalies': those problems arising when one tries to update (insert, delete, or replace) a relationship which is stored on a subset of the columns in a relation. Perhaps one of the best ways to view the problem is through the notion of 'syntactic predictability' of [Bernstein & Goodman, 1980]. They feel that these anomalies do not exist when the effects of an update can be determined by examining the schema alone, and not the content of the database.

It is interesting that Codd also foresaw other desirable aspects of normalization. In [Codd, 1972] it is pointed out that reduction of databases into normal forms is desirable for

(1) reducing the amount of restructuring done to the database, thereby enhancing software lifetime;
(2) making relations more informative to naive users, when they become cluttered with too many attributes;
(3) simplifying the maintenance of usage statistics; and
(4) making authorization requirements simpler to enforce.

Each of these aspects restates the 'one fact in one place' credo. Bernstein and Goodman formalized the benefits of normalization in terms of improving update behavior, and proved that BCNF attains this goal for the single relation case.

For the multirelation case, Bernstein and Goodman went on to observe that anomalous behavior is not avoided when the Universal Instance Assumption (UIA) is required to hold. The UIA is an artificial constraint, requiring that each of the relation instances in the database be projections of some universal relation instance. Problems arise because the UIA forces the introduction of null values whenever updates are made on subsets of the set of all attributes U.

When the UIA is dropped, and we make other simple assumptions, we can show that BCNF has desirable normalization properties. We begin by formalizing the notion of update anomalies. We propose an operational definition of anomalies which takes into account the intended dynamic behavior of the database. In particular, we formalize the notion of 'update sets' over databases and define anomalous behavior in terms of these sets.

Intuitively, an update set (elsewhere referred to as an object, association, and by other terms) is a collection of attributes which is updated as a unit, and may be viewed as a primitive 'fact' in the database.

Definition An ***update set*** Z over a database schema $D = \{R_i < U_i, F_i > \mid i=1,...,n\}$ is any subset of $U = \bigcup_{i=1}^{n} U_i$, such that a database defined on D may be modified atomically on the columns in Z.

Each database schema has a fixed number of update sets. Appropriate update sets are determined by the database semantics. We further specify update sets according to their intended function:

(1) insertion sets
(2) deletion sets
(3) replacement sets

Update anomalies are defined in terms of the update sets selected for a database.

Definition A database schema $D = \{R_i < U_i, F_i > \mid i=1,...,n\}$ ***is free of update anomalies*** if every update set Z

(1) is in some scheme U_i of D,
(2) contains a key for U_i, and
(3) contains the entire scheme U_i, if Z is a deletion set.

This is an intuitively correct definition for the following reasons: First, we desire update sets to be contained within a single scheme so that we are not confronted with the problems of updating views. Some of these problems were addressed in [Bernstein & Goodman, 1980]. Second, we desire update sets to contain a key so that nulls need not be inserted in a key field, and so that replacements of large numbers of records are avoided. (We assume, as does Codd, that in every tuple at least the primary key must be stored without null values.) Third, deletion sets cover entire schemes to avoid difficulties in deleting part of a tuple, which again is part of the view update problem.

For example, the database schema $D = \{ R_1 < \{E,M\},\{E \rightarrow M\} > , R_2 < \{M,D\},\{M \rightarrow D\} > \}$, may have the following update sets:

(1) insertion sets:	$\{E\},\{E,M\},\{M,D\}$
(2) deletion sets:	$\{E,M\},\{M,D\}$
(3) replacement sets:	$\{E\},\{E,M\},\{M,D\}$

The replacement set $\{E,D\}$ is not permitted. With this definition we arrive at the following result:

Prop A database schema $D = \{R_i < U_i, F_i > \mid i=1,...,n\}$ is free of update anomalies if
(1) each R_i is in BCNF,
(2) every update set Z corresponds to a nontrivial FD, i.e., we can write $Z = X \cup Y$, where $X \rightarrow Y$ is an FD in some F_i,
(3) if Z is a deletion set, then in addition $Z = U_i$ for some i.

Proof Since all update sets are defined by FDs, every update set is in some scheme U_i of D. Then, since each R_i is in BCNF, the left hand side of every FD in F_i contains a key. Because all update sets are defined on FDs, every update set contains a key for some F_i. Condition (3) covers the special requirement that deletion sets cover an entire scheme. ▪

Now, consider the Zipcode database for which there is no BCNF covering:

$$D = \{ R < \{C,S,Z\},\{CS \rightarrow Z, Z \rightarrow C\} > \}.$$

Suppose $\{C,S,Z\}$ is the only update set (a natural assumption). By not using update sets which correspond to non-BCFDs, we avoid update anomalies. In situations where the natural update sets avoid these FDs (such as the situation here) we have no anomalies *even when BCNF cannot be achieved.*

The simple observations above show that BCNF does avoid anomalies under the right assumptions; however, BCNF may be too strong a condition under these assumptions -- we may be able to avoid anomalies even without having each scheme in BCNF. More research is necessary to define a 'normal form' which properly takes dynamic behavior of the database into account.

4. 'Unattainability' of BCNF

It is now largely taken for granted that BCNF is not attainable for some databases. The previous section has shown that this may not be serious, since we may not use any non-BCFDs as update sets: having BCNF may not be necessary to avoid anomalies. Regardless, we feel that believing BCNF to be unattainable is incorrect, and feel that those situations where BCNF is not immediate are results of improper naming of entities.

Consider the City, Street, Zip example. Note first that 'Street', 'Zip', and 'City' are increasingly larger abstractions for 'location'. The problem is that street names do not uniquely determine locations on their own -- one street name is associated with many physical streets. If street names were unique, we would have $S \rightarrow Z$, $Z \rightarrow C$ and could find BCNF easily.

One solution is to make street addresses unique somehow. We can achieve this by creating a new attribute A (address) to replace S, where the domain of A covers Streets and Cities. Our schema then becomes

$$< AZ, A \rightarrow Z > , \quad < ZC, Z \rightarrow C > .$$

Note that this solution could introduce update anomalies if we tried to update a city name. This is probably not a serious problem, but it may be avoided by picking a unique 'surrogate' identifier* for each city and letting A range over Streets and the surrogate domain. This solution may also be preferable in that storing a surrogate will probably be less wasteful of space.

The problem here seems to run deeply into the identification of entities and into naming. Kent points out that there are important problems in naming [Kent, 1978]:

(1) Names and entities are not necessarily in one-one correspondence. One name may encompass many entities (generically); one entity may be described by many names (e.g., by virtue of being an aggregate); and there are often multiple ways to view objects as entities, hence multiple naming conventions (or names) for the same phenomenon.

(2) Having a poor connection between names and entities leads to update problems. Kent focuses particularly on the problems encountered by aggregate entities.

Our feeling is that in many cases, 'unattainable' BCNF schemata can be reached by suitable renaming of certain attributes and by using surrogate attributes. We have a number of examples in which the rephrasing of FDs using these new attributes led to better schemata. Sciore has shown formally [Sciore, 1982] that many schemata can be improved simply by adding attributes. It is not clear *a priori* how these schemata should be modified, but intuitively we should be able to store most sets of facts in such a way that anomalies do not arise. This is an interesting area, calling for more research.

5. BCNF testing

This section is concerned with various aspects of testing for BCNF. There are three question to consider:

(i) Does a given schema $R < U,F >$ violate BCNF?

(ii) If so, does there exist a BCNF covering for R?

(iii) If a BCNF covering does exist, how do we find one? How do we find a minimal BCNF covering?

The first two questions have been shown to be NP-complete [Beeri & Bernstein, 1979]. This result led Osborn to develop a BCNF tester which always requires exponential time to execute. We will show below, however, that the problem is NP-complete only for pathological sets of FDs, and that for 'real-world' sets of FDs exponential amounts of time appears unnecessary.

* A *surrogate* is an attribute whose domain is a set of identifiers which are guaranteed to determine an entity uniquely. Surrogate values are immutable, they may bear no relationship to real-world data (i.e., they are internally defined), and may be invisible, in fact, to the database user. They are typically created when an entity has multiple names (keys), when entity names can change, or when it is desirable to provide aggregate entities with concise identifiers. Good discussions of many useful aspects of surrogates may be found in [Kent, 1978].

We should emphasize that we are *not* saying that BCNF is the way to go, or that we are encouraging the use of our algorithm for real database design. Frankly, we have doubts even about the usefulness of incorporating Armstrong's rules into the design phase. Rather, our intent in this section is to investigate the 'insurmountability' of the NP-completeness results, and to experiment with reasonable models for FDs.

5.1. Osborn's Algorithm

A BCNF testing algorithm was developed by Osborn [Osborn, 1979]. She notes that the following economies can be made when one is interested in BCNF. First, one can always restrict one's attention to FDs in 'canonical form', in which all FD right-hand sides have only one attribute. Moreover, one can consider only *full FDs*, functional dependencies whose left-hand sides are minimal. If the dependencies $X \rightarrow A$ and $Y \rightarrow A$ both hold in R, and X is a subset of Y, then $Y \rightarrow A$ is not a full FD and need not be considered as far as BCNF is concerned. (If X includes a key for R, then Y includes a key for R since $X \subseteq Y$.)

Second, in generating a BCNF covering for R it is sufficient to consider only those schemata whose attributes are obtained from canonical full FDs, determine which of them are in BCNF, and then group together the BCNF schemata. There is a BCNF covering if and only if these BCNF schemata comprise a covering.

The first step in Osborn's algorithm is to compute a *canonical closure*, F_c^+, of the set of dependencies F using Beeri and Bernstein's Membership algorithm. The canonical closure is just that part of the closure F^+ where all right-hand sides have only one attribute. In the second step, for each full FD $X_i \rightarrow A_i$ in F_c^+ a relation schema $R_i < U_i, F_i >$ is derived, where $U_i = X_i \cup \{A_i\}$, and F_i is that subset of the dependencies in F_c^+ which holds on U_i. These schemata are tested for compliance with BCNF, and the FDs $X_i \rightarrow A_i$ for which the test is successful are added to a set G. If the final set G forms a covering for F, then a BCNF covering for F exists. Furthermore, if a minimal BCNF covering exists, it can be found by eliminating redundant dependencies in G.

A pessimistic analysis of the performance of this algorithm for testing the existence of a BCNF covering shows that it runs in time $O(\|U\|^3 \|F\| (2^{\|U\|})^2)$. One can additionally find a minimal covering, if one exists, within a supplemental amount of time bounded by $O(\|U\| \, \|G\|^2)$, by constructing a minimal cover for G after one has completed the main algorithm.

The examples which follow illustrate the use of Osborn's BCNF tester. The algorithm does not capitalize on the situation where the input schema is initially in BCNF.

Example 1. Zip Code (not in BCNF)

$U = \{C, S, Z\}$
$F = \{CS \rightarrow Z, Z \rightarrow C\}$
$F_c^+ = \{C \rightarrow C,\ S \rightarrow S,\ Z \rightarrow C,\ Z \rightarrow Z,\ CS \rightarrow C,\ CS \rightarrow S,\ CS \rightarrow Z,\ CZ \rightarrow C,\ CZ \rightarrow Z,\ SZ \rightarrow C,\ SZ \rightarrow S,\ SZ \rightarrow Z,\ CSZ \rightarrow S,\ CSZ \rightarrow C,\ CSZ \rightarrow Z\}$

The following dependencies in F_c^+ are nontrivial full FDs:
$\{Z \rightarrow C, CS \rightarrow Z\}$.

$U_1 = \{ZC\}$, $F_1 = \{Z \rightarrow C\}$

$U_2 = \{CSZ\}$, $F_2 = F_c^+$

Since $Z \rightarrow C$ is in F_2, but $Z \rightarrow CSZ$ is not in F_c^+, the dependency $CS \rightarrow Z$ is not included in G.

$G = \{Z \rightarrow C\}$
$G_c^+ = \{C \rightarrow C,\ Z \rightarrow Z,\ Z \rightarrow C,\ CZ \rightarrow C,\ CZ \rightarrow Z\}$.
F_c^+ does not equal G_c^+

A BCNF covering does not exist, as was expected.

Example 2. from [Beeri and Bernstein, 1979]

$U = \{A, B, C, D, E\}$
$F = \{A \rightarrow BC, BC \rightarrow A, BCD \rightarrow E, E \rightarrow C\}$

The following dependencies in F_c^+ are nontrivial full FDs:
$\{A \rightarrow B,\ A \rightarrow C,\ E \rightarrow C,\ AD \rightarrow E,\ BC \rightarrow A,\ BCD \rightarrow E,\ BE \rightarrow A\}$

$G = \{A \rightarrow B,\ A \rightarrow C,\ E \rightarrow C,\ AD \rightarrow E,\ BC \rightarrow A\}$
$G_c^+ = F_c^+$

Therefore a BCNF covering exists.
A relational schema representing $< U, F >$ is:

$R_1 < \{A, B, C\}, \{A \rightarrow BC, BC \rightarrow A\} >$

$R_2 < \{A, D, E\}, \{AD \rightarrow E\} >$

$R_3 < \{E, C\}, \{E \rightarrow C\} >$

5.2. A 'Fast' BCNF Testing Algorithm

In this section we develop a simple algorithm for testing whether a schema is in BCNF. Although the presentation here appears (almost!) natural, in fact the development derives from earlier work on the equivalence between FD and MVD systems and certain fragments of propositional logic [Delobel & Casey, 1973; Fagin, 1977; Parker & Delobel, 1979; Sagiv, Delobel, Parker & Fagin, 1981].

Definition A ***key dependency*** is an FD of the form $X \rightarrow \phi$, where ϕ is the empty set.

This may seem strange at first, but should be interpreted simply as saying that 'X is a key'. Thus if U is the set of attributes in the schema where X is a key,

$$X \to \phi \quad \text{implies} \quad X \to U. ^{*}$$

Definition $X \to Y$ is a *canonical FD* if $\|Y\| = 0$ or 1.

Definition $X \to Y$ is a *full FD* if there is no proper subset X' of X such that $X' \to Y$ holds, and in addition either (1) $Y = \phi$, or (2) the key dependency $X \to \phi$ does not hold (since if it did, it would imply $X \to Y$).

Definition $X \to Y$ is an *interesting FD* if it is nontrivial, canonical, and full.

It is useful to observe that, when dealing with FDs, we need only concern ourselves with interesting FDs. All others are somehow redundant. We argue below that the number of interesting FDs in the closure F^+ of F is limited when F consists of 'real world' dependencies. Of course, F^+ is always exponentially large, since there are exponentially many trivial dependencies.

Henceforth, let us permit dependencies with null right-hand-sides in our dependency inferences. Thus we extend Armstrong's axiom FD3 (transitivity)

If $X \to Y$ and $Y \to Z$, then $X \to Z$

so that Z can be null. Below, when we speak of 'closures', it will be relative to this extended defintion.

We now have a direct relationship between keys, non-BCFDs, and interesting closures.

Theorem 1 Consider the relation schema $R\langle U,F\rangle$. Let $X \to Y$ be an interesting FD in the closure of

$$F \cup \{ U \to \phi \}.$$

Then

(1) $Y = \phi$ if and only if X is a *candidate key* for $R\langle U,F\rangle$;

(2) $Y \neq \phi$ if and only if $X \to Y$ is a *non-Boyce-Codd FD* for $R\langle U,F\rangle$ (i.e., an FD which violates BCNF).

* **This statement does *not* imply that $\phi \to \phi$, and hence for example the 'decomposition rule'**

$$A \to BC \text{ implies } A \to B \text{ and } A \to C$$

requires $B \neq \phi$ and $C \neq \phi$. Also clearly $X \to A$ does not imply $X \to \phi$, for otherwise every left hand side of a dependency will be a key. This is all consistent with the equivalence between dependencies and propositional logic [Delobel & Casey 1973; Fagin 1977], since the formula $(\mathbf{x} => \phi) \equiv (\neg \mathbf{x})$ logically implies $(\mathbf{x} => \mathbf{y})$, for any term $\mathbf{y}$.

Proof (1) It is clear that if $Y = \phi$, then X must be a candidate key: $X \to U$ must be a consequence of F, since F contains no key dependencies other than $U \to \phi$. Moreover since this is a full FD, X must be a candidate key (not just a key).

Conversely, if X is a candidate key, then the only full FD we can find is $X \to \phi$.

(2) Assume then that $Y \neq \phi$. Suppose that $X \to Y$ is *not* a non-Boyce-Codd FD. Then X must contain some key K. But this implies $K \to Y$, contradicting that $X \to Y$ is full. So $X \to Y$ must be a non-BCFD.

Conversely, let $X \to Y$ be a non-BCFD. Then X does not contain a key. But this implies that Y cannot be null, since if it were, we would have a key dependency and reach a contradiction. ▪

We can exploit this theorem to obtain an algorithm for BCNF testing. We need first to produce the set of all interesting FDs in the closure of F. This is easily accomplished, for example, using the algorithm for closure in [Parker & Delobel, 1979]. For reference we construct a version of the algorithm here which does not make use of propositional logic.

Algorithm INTERESTING_CLOSURE,
generating $F^+_{interesting}$, the nontrivial canonical full FDs in F^+

```
Input:  A set of FDs $F$, possibly containing key
        dependencies
Output: $F^+_{interesting}$
Step 1: Set $T1 \leftarrow F$.
Step 2: Set $T2 \leftarrow T1$. For each pair of FDs
        $\sigma_1, \sigma_2$ in $T1$, apply Armstrong's axioms
        if possible to produce a new result FD $\sigma$.
        Delete any FD in $T2$ implied by $\sigma$. Then,
        if $\sigma$ is not implied by any element in $T2$,
        add it to $T2$.
Step 3: If $T1$ and $T2$ are not equal, set $T1 \leftarrow T2$
        and proceed to Step 2. Otherwise set
        $F^+_{interesting}$ to $T2$ and halt. ▪
```

This algorithm leads directly to the following algorithm to test whether a given relation schema is in BCNF.

Algorithm BCNF_TEST,
checking compliance with BCNF

```
Input:  A relation schema $R\langle U,F\rangle$
Output: 'YES', if $R$ is in BCNF; alternatively a set
        of dependencies in $F^+$ which are non-
        BCFDs.
```

Step 1: Add the key dependency $U \rightarrow \phi$ to the set F.

Step 2: Compute the closure $F^+_{interesting}$ using algorithm INTERESTING_CLOSURE above.

Step 3: Every set of attributes X such that the FD $X \rightarrow \phi$ is in this closure is a key. All remaining dependencies in the closure are non-BCFDs. R is in BCNF if, and only if, there are no such dependencies. ▪

We can also use Theorem 1 to find a decomposition which forms a BCNF covering, if we add one more result. Suppose for the moment that we have an algorithm which determines, for a schema $<F,U>$ and a set of attributes X, the FDs in F^+ which hold on the set of attributes $U-X$. Then we can search for BCNF coverings in the following way:

Algorithm BCNF_COVER,
determining a BCNF covering if one exists.

Input: $R<U,F>$, a relation schema

Output: A BCNF covering $\{R_i<U_i,F_i> \mid i=1,...,n\}$ for R if one exists; otherwise 'NONE'

Step 1: Generate the set $F^+_{interesting}$ of all non-trivial canonical full FDs of the set F, using algorithm INTERESTING_CLOSURE above.

Step 2: For each interesting FD $X_j \rightarrow A_j$ in $F^+_{interesting}$ find the subset of F^+ which holds on the attribute set U_j, where $U_j = X_j \cup \{A_j\}$. Use the BCNF-testing algorithm above to find all non-BCFDs for this set. If no non-BCFDs exist, add $X_j \rightarrow A_j$ to a set G.

Step 3: Test whether $F \subseteq G^+$, by testing membership of every dependency of F in G. This may easily be effected using the Membership algorithm in [Beeri & Bernstein, 1979]. If $F \subseteq G^+$, then the database schema $\{<U_i,F_i> \mid i=1,...,n\}$ composed of schemata for which $X_i \rightarrow A_i$ is in G forms a BCNF covering for R. Otherwise, no BCNF covering exists, and the algorithm should output 'NONE'. ▪

Step 2 of this algorithm appears expensive to perform, but in fact this need not be so. The **mod** function [Parker & Delobel, 1979] * may be used to determine the set of dependencies which apply in a projection of a schema on a smaller set of attributes. An alternative method is to first evaluate the interesting closure and then discard dependencies involving attributes not in the projection. Below we will argue that the interesting closure is usually not large, so this alternative approach will be adequate. However we will refer to the dependencies in $F^+_{interesting}$ which hold on attributes $U-Z$ as F **mod** Z.

The algorithm above may be modified to find a minimal BCNF covering by removing redundant schemata after Step 3 (as in Osborn's algorithm). It should also be mentioned that, in some cases, the set of schemata generated in Step 3 will not contain all of the attributes in U. (Consider, for example, the initial schema $R<\{A,B,C\},\{A \rightarrow B\}>$.) This is a minor problem not handled by Osborn. If this happens one should include an additional schema $<K,\phi>$ to the final database schema, where K is an arbitrary candidate key for the entire initial set of attributes. This approach has been taken in [Delobel & Casey, 1973; Biskup, Dayal & Bernstein, 1979].

Below we consider several examples of schemata on which the above algorithms are applied.

Example 1. Zip code (not in BCNF)
$U=\{C,S,Z\}$
$F=\{CS \rightarrow Z, Z \rightarrow C\}$

First, we can test if $<U,F>$ is already in BCNF.

$(F \cup \{U \rightarrow \phi\})^+_{interesting} = \{CS \rightarrow \phi,\ SZ \rightarrow \phi,\ Z \rightarrow C\}$.

$Z \rightarrow C$ is a non-BCFD; therefore $<U,F>$ is not in BCNF.

Now, we search for a BCNF covering of $<U,F>$.

There are two interesting FDs in F; namely $CS \rightarrow Z$ and $Z \rightarrow C$.

$F_1 = F \textbf{ mod } (U-CSZ) = F \textbf{ mod } \phi = F$
$F_1 \cup \{U \rightarrow \phi\}$ has the non-BCFD $Z \rightarrow C$, so we do not add $CS \rightarrow Z$ to the set G.

$F_2 = F \textbf{ mod } (U-ZC) = F \textbf{ mod } S = \{Z \rightarrow C\}$.
$F_2 \cup \{ZC \rightarrow \phi\}$ has no non-BCFDs, so we add $Z \rightarrow C$ to G.
At this point we have examined all interesting FDs, and have $G = \{Z \rightarrow C\}$. The algorithm then detects that F is not a subset of G^+. Therefore no BCNF covering exists, as was expected.

* We give only a brief description of **mod** here in the interest of simplicity:

Suppose $F = \{X_i \rightarrow Y_i \mid i=1,...,m\}$ is a set of FDs. Then let

$$\mathbf{F} = \bigwedge_{i=1}^{m} (\mathbf{x}_i => \mathbf{y}_i)$$

be the propositional logic equivalent. Define

$$\mathbf{F} \textbf{ mod } \mathbf{z} = (\mathbf{F}_{|\mathbf{z}=0}) \vee (\mathbf{F}_{|\mathbf{z}=1})$$

and

$$\mathbf{F} \textbf{ mod } \mathbf{z}_1 \cdots \mathbf{z}_n = (...(\mathbf{F} \textbf{ mod } \mathbf{z}_1) ...) \textbf{mod } \mathbf{z}_n$$

We then have

Prop **F mod Z** corresponds precisely to the set of dependencies in the closure F^+ which hold in the domain $U-Z$. ▪

This result can be quickly used to develop the set of FDs required by Step 2.

Example 2. (already in BCNF)

$U = \{A,B,C\}$
$F=\{A\rightarrow B, A\rightarrow C\}$

Test if $< U,F>$ is in BCNF.

$(F \cup \{U\rightarrow\phi\})^{+}_{interesting} = \{A\rightarrow\phi\}$

Since there are no non-BCFDs in this set, $< U,F>$ is already in BCNF.

Example 3. from [Beeri and Bernstein, 1979]

$U=\{A,B,C,D,E\}$
$F=\{A\rightarrow BC, BC\rightarrow A, BCD\rightarrow E, E\rightarrow C\}$

$(F \cup \{U\rightarrow\phi\})^{+}_{interesting} = \{A\rightarrow B, A\rightarrow C, BC\rightarrow A, BCD\rightarrow E, E\rightarrow C, AD\rightarrow E, BE\rightarrow A\}$

We now construct F **mod** $(U-U_i)$ for each of the seven interesting FDs $X_i\rightarrow A_i$ in this set, and come up with the following corresponding sets of dependencies:

$F_1 = \{A\rightarrow B\}$
$F_2 = \{A\rightarrow C\}$
$F_3 = \{BC\rightarrow A, A\rightarrow B, A\rightarrow C\}$
$F_4 = \{BCD\rightarrow E, E\rightarrow C\}$
$F_5 = \{E\rightarrow C\}$
$F_6 = \{AD\rightarrow E\}$
$F_7 = \{BE\rightarrow A, A\rightarrow B\}$

After this, we find $G = \{A\rightarrow B, A\rightarrow C, BC\rightarrow A, E\rightarrow C, AD\rightarrow E\}$. Since F is a subset of G^{+}, G provides a BCNF covering for $< U,F>$.
Since the first two schemata obtained from G on AB and AC are redundant, we can generate the covering
$R_1 < \{A,B,C\},\{A\rightarrow BC, BC\rightarrow A\}>$.
$R_2 < \{A,D,E\},\{AD\rightarrow E\}>$
$R_3 < \{E,C\},\{E\rightarrow C\}>$
This is the same covering that was generated by Osborn's algorithm.

5.3. Complexity Analysis

The timing of the algorithms above is not simple to ascertain precisely, but in this section we provide bounds. The bounds are related to the number of interesting FDs in the closure, not the cardinality of the closure. We also provide a model of those schemata we feel typify the real world, called the *FD hierarchy model*. This model is an adaptation of Lien's hierarchical schemata [Lien, 1980] incorporating features of many other models. With this model, we find that time bounds are not exponential in $||F||$, but *linear* in $||F||$. Throughout this section, we assume w.l.o.g. that F contains only interesting FDs.

We begin by establishing a relatively tight upper bound on the number of interesting FDs in F's closure.

Definition Let F be a set of interesting FDs. The ***transitivity graph*** $T(F)$ is a labelled, directed graph (V,E) with vertices $V = F$ and edges

$$E = \{(f_1,f_2) \mid RHS(f_1) \subseteq LHS(f_2)\}$$

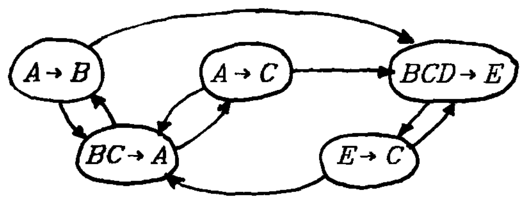
Note that here, $RHS(f)$ and $LHS(f)$ are the right- and left-hand sides of the FD f. If $RHS(f) = \phi$, then the vertex f is connected to all other vertices.

Definition A subset W of vertices of a directed graph G is called a *consensus set of G* if the restriction of G to W is a rooted DAG (directed acyclic graph) in which all edges point toward the root (a vertex of outdegree zero).

Example

$$F = \{A\rightarrow B, A\rightarrow C, BC\rightarrow A, BCD\rightarrow E, E\rightarrow C\}$$

Transitivity graph $T(F)$:



There are many consensus sets: $\{A\rightarrow B\}$, $\{BC\rightarrow A\}$, ... , $\{A\rightarrow B, BCD\rightarrow E\}$, $\{BCD\rightarrow E, E\rightarrow C\}$, $\{A\rightarrow B, A\rightarrow C, BCD\rightarrow E\}$, etc.

Theorem 2
$||F^{+}_{interesting}|| \leq$ ||consensus sets of $T(F)$||

Proof We show that there is a one-many relationship between $F^{+}_{interesting}$ and the consensus sets of $T(F)$.

Let $X\rightarrow A$ be a FD in $F^{+}_{interesting}$. Then there is an irredundant set S of interesting FDs in F which together imply $X\rightarrow A$. In addition, the FDs in S irredundantly imply no other interesting FD. We claim that S is a consensus set of $T(F)$, and that

$$X = \left(\bigcup_{v\epsilon S} LHS(v)\right) - \left(\bigcup_{v\epsilon S} RHS(v)\right)$$
$$A = RHS(v_0)$$

where v_0 is the root vertex of S.

If $||S|| = 1$, this statement is true since S is a single vertex in $T(F)$, and must equal $\{X\rightarrow A\}$. Assume then that $||S|| > 1$. It is clear that $T(F)$ restricted to S must have a single root vertex, for otherwise S is redundant. In addition, every vertex in this restricted graph must have a path to the root, for otherwise transitivity cannot be applied, and either S must be redundant or $X\rightarrow A$ must not be interesting.

Now, every attribute of X must be in $LHS(v)$ for some $v \varepsilon S$. However, notice that X cannot contain any attribute in $RHS(v)$ for any $v \varepsilon S$, since then $S-\{v\}$ would still imply $X \rightarrow A$, contradicting the irredundancy of S. Thus

$$X \subseteq \left[\bigcup_{v \varepsilon S} LHS(v)\right] - \left[\bigcup_{v \varepsilon S} RHS(v)\right].$$

Suppose that equality does not hold, i.e., there is some vertex $v \varepsilon S$ such that $LHS(v)$ contains an attribute B which is not in X, yet B is not in the RHS of any vertex in S. Then $S-\{v\}$ still implies $X \rightarrow A$, again contradicting the irredundancy of S.

A similar argument shows that the restricted graph must be acyclic. The attribute A cannot be in $LHS(v)$ for any $v \varepsilon S$, since otherwise we could remove v. We can now proceed inductively, showing no direct ancestor of the root vertex can have an edge to one of its ancestors v, etc. ▪

Corollary $\|F^+_{interesting}\| < 2^{\|F\|}$

Proof There are $2^{\|F\|}-1$ nonempty subsets of vertices in $T(F)$. ▪

Theorem 2 is interesting in that it provides us a method for computing $F^+_{interesting}$: we construct $T(F)$ rapidly, then enumerate all of its consensus sets. For each consensus set S we then generate the FD

$$\left[\bigcup_{v \varepsilon S} LHS(v)\right] - \left[\bigcup_{v \varepsilon S} RHS(v)\right] \quad \rightarrow \quad RHS(v_0)$$

where v_0 is the root vertex of S, and then eliminate subsumed FDs. The algorithm INTERESTING_CLOSURE essentially does this, but is inefficient in its searches for FDs to which Armstrong's rules may be applied. The algorithm executes in time at most quadratic in the number of consensus sets in $T(F)$, and this can be improved to near-linear time if we use data structures which reduce the time required for subsumption checking. An open problem is to find an optimal algorithm here.

The time to generate closures is thus at most quadratic in the number of rooted DAGs (consensus sets) in the graph T(F). Exponential time is required only when there are an exponential number of such DAGs. Furthermore, the running time will be a direct function of $\|F^+_{interesting}\|$ in the case where no FDs are subsumed or generated multiple times.

We feel that for 'real world' collections of FDs, the number of consensus sets will be subexponential, and that few FDs will be subsumed, or generated multiple times. To support this statement we consider the following adaptation of Lien's hierarchical schemata [Lien, 1980], called the *FD hierarchy model*. We feel this model is flexible enough to describe real databases, yet restrictive enough that useful conclusions regarding the complexity of logical design can be derived from it.

Definition A *FD hierarchy* over U (a fixed set of attributes) is a directed acyclic graph $H = (V,E)$ and a pair of maps KEY: $V \rightarrow 2^U$, $ATTR$: $V \rightarrow 2^U$, such that

(i) for every $v \varepsilon V$, $KEY(v) \subseteq ATTR(v)$

(ii) for every $v \varepsilon V$, $KEY(v) \rightarrow ATTR(v)$ is a valid FD

(iii) if the edges leaving any vertex v are $(v,w_1), \ldots, (v,w_n)$ then

$$KEY(v) \subseteq \bigcup_{i=1}^{n} KEY(w_i) \subseteq ATTR(v)$$

(iv) if $A \varepsilon ATTR(v)-KEY(v)$, and $A \varepsilon ATTR(w)$ with $v \neq w$, then necessarily $A \varepsilon KEY(w)$, and there must be a path from v to w.

An example of an FD hierarchy is shown in Figure 1. It is similar to a schema of [Lien, 1980] pp.66-67, but has been modified to make certain points.

Remarks

(1) The FD hierarchy model is similar to not only Lien's hierarchical schemata, but also many others -- Bachman diagrams, Entity-relationship schemata, etc.

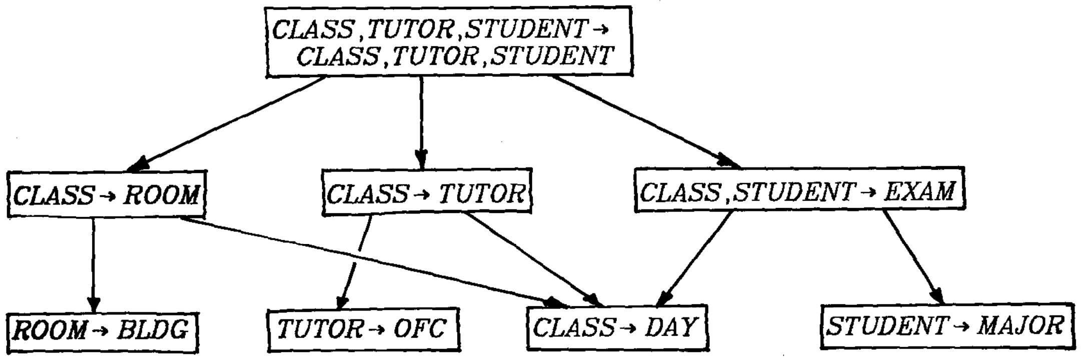


Figure 1. An FD hierarchy

(2) Note it is possible to have $KEY(v) = ATTR(v)$, i.e. a trivial FD. In fact, a consequence of the definition above is that every FD in H is either interesting or trivial. To see this, suppose $X \rightarrow A$ is a nontrivial FD in H. Then by condition (iv), A cannot appear in the *RHS* of any other FD.

(3) The FD hierarchy H is closely related to the transitivity graph $T(F)$. A vertex in H with associated FD $X \rightarrow A_1 A_2 \cdots A_n$ will have corresponding vertices $X \rightarrow A_1, \ldots, X \rightarrow A_n$ in $T(F)$ (assuming of course that these FDs are nontrivial).

Therefore, for schemata fitting this model $T(F)$ contains no cycles, and because of the condition (iv), it is impossible to derive a single FD in multiple ways with this model. In particular, duplicate keys $(A \rightarrow B, B \rightarrow A)$ cannot be represented directly.

(4) In the example, the only dependencies on which transitivity may be used are *CLASS*→ *TUTOR* & *TUTOR*→ *OFFICE*, and *CLASS*→ *ROOM* & *ROOM*→ *BLDG*. In Lien's scheme, in fact, no transitivity existed. This is not atypical of the 'real world', in our experience. For example, an E-R schema in which all relationships are many-many will have $F^+_{interesting} = F$.

Suppose we consider the largest indegree d, and the 'height' (length of the longest path) h, of an FD hierarchy H. The example above has a height of 2, and the vertex with FD *CLASS*→ *DAY* has a maximal indegree of 3. In our experience, d and h rarely exceed 4 in real schemata. If we consider only FD hierarchies with bounded indegrees and heights, we arrive at the following result.

Theorem 3 Suppose that in a FD hierarchy H, no vertex has more than d ancestors, and no path is longer than h, and that F is the set of interesting FDs given by H. Then

$$||F^+_{interesting}|| \leq ||\text{consensus sets of } T(F)|| < k\ ||F||$$

where $k = 2^{d^h}$.

Proof Since H is restricted by the constants d and h, $T(F)$ is also. (This follows since each vertex in $T(F)$ with FD $X \rightarrow A$ will have indegree less than or equal to that of the vertex in H with FD $X \rightarrow A$.) There are therefore only $||F||$ possible root vertices in $T(F)$, and each one is the root of a DAG, for which each vertex has indegree at most d, and in which no path is longer than h. For such a DAG there are fewer than $k = 2^{d^h}$ subsets of vertices which determine a sub-DAG connected to the root. So, altogether there are at most $k||F||$ consensus sets of $T(F)$. Combining this with Theorem 2 gives us the stated bound. ▪

This theorem states that the number of interesting dependencies in F's closure is not only not exponential, but is *linear*, in $||F||$. Thus the NP-completness results in this area are not necessarily obstacles to designers. Further exploration of the FD hierarchy model is necessary, but we are confident of the validity of results like Theorem 3.

6. Conclusions

We have reviewed various criticisms of BCNF, discussing its virtues and limitations. Some limitations to the effectiveness of BCNF disappear if we drop the Universal Instance Assumption. In fact, under suitable assumptions BCNF is more restrictive than necessary for avoiding update anomalies. (Essentially, BCNF strives to place 'one fact in one place', assuming that all facts are representable as many-one relationships. However not all facts may be updated by users of the database, and hence it may not be necessary to normalize all of these facts.) We also showed that BCNF can be often attained by renaming or adding attributes, even when it is formally unachievable.

We then pointed out a relationship between keys, FDs violating BCNF, and FD closures. As an application of this relationship, we presented algorithms for both BCNF testing and for the development of a BCNF covering for a relation schema. These algorithms have exponential running time only in the pathological case where the set of 'interesting' dependencies in the dependency closure grows exponentially large. For the FD hierarchy model with restrictions on the height and indegrees of the hierarchy, we showed these closures are not exponential, but are linear, in the size of the input set of FDs.

It is difficult to evaluate normal forms in any precise way, given the ambiguity of semantical constructs and design objectives which surround the logical design process. Our effort has been to reflect on the limitations of BCNF as a logical design benchmark, reviewing the various negative results against it put forth over the past years. It is hoped that this effort will spark work on new measures of logical design effectiveness, as well as on new areas such as attribute renaming, consideration of dynamic behavior of the database, and formal models.

Acknowledgement
The authors are grateful to the referees for a number of suggestions which have greatly improved the presentation here. D.S.P. would like also to acknowledge several insightful comments of T. Wheeler, and some recommendations by S. Osborn during this paper's first incarnation.

References

Atzeni, P. and D.S. Parker (March 1982). 'Assumptions in Relational Database Theory,' *Proc. First ACM SIGACT-SIGMOD Conf. on Principles of Database Systems*, Los Angeles, CA.

Beeri, C. and P.A. Bernstein (1979). 'Computational problems related to the design of normal form relation schemes,' *ACM Trans. on Database Systems* 4:1, pp.30-59.

Bernstein, P.A. and N. Goodman (Oct. 1980). 'What does Boyce-Codd Normal Form do?,' *Proc. of the 6th Intnl. Conf. on VLDB*, Montreal, pp. 245-259.

Biskup, J., U. Dayal, and P.A. Bernstein (May 1979). 'Synthesizing Independent Database Schemas,' Proc. ACM-SIGMOD Intnl. Conf. on Mgmt. of Data, Boston, pp. 143-151.

Codd, E.F. (1972). 'Further Normalizaiton of the Data Base Relational Model,' in *Data Base Systems* (R. Rustin, ed.), Prentice-Hall, Englewood Cliffs, NJ, pp. 33-64.

Date, C.J. (1981). *An Introduction to Database Systems* , Addison-Wesley, Reading, Mass.

Delobel, C. and R.C. Casey (1973). 'Decomposition of a Database and the Theory of Boolean Switching Functions,' *IBM J. Res. and Develop.* 17, p.374ff.

Fagin, R. (1977). 'Functional Dependencies in a Relational Database and Propositional Logic,' *IBM J. Res. Develop.* 21:6, pp. 543-544.

Kent, W. (1978). *Data and Reality*, North-Holland.

Lien, Y.E. (1981). 'Hierarchical Schemata for Relational Databases,' *ACM Transactions on Database Systems* 6:1, pp. 48-69.

Osborn, S.L. (Jan. 1979). 'Testing for existence of a covering Boyce-Codd Normal Form,' *Information Processing Letters* 8:1, pp. 11-14

Parker,D.S. and C. Delobel (October 1979). 'Algorithmic applications for a new result on multivalued dependencies', *Proceedings of the 5th Intnl. Conf. on VLDB*, Rio de Janeiro, pp. 67-74.

Sagiv, Y., Delobel, C., Parker, D.S., Fagin, R. (1981). 'An equivalence between relational database dependencies and a fragment of propositional logic,' *JACM* 28:3, pp. 435-453.

Sciore, E., (1982). 'Improving Database Schemes by Adding Attributes,' Tech. Rept. #82/035, Dept. of Computer Science, SUNY Stony Brook.

Ullman, J.D. (1980). *Principles of Database Systems*, Computer Science Press, Potomac, Maryland.