The University of Alberta

ON THE BOUNDEDNESS OF RELATIONAL DATABASE SCHEMES WITH RESPECT TO FUNCTIONAL DEPENDENCIES

by

Héctor Juan Hernández-López

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Department of Computing Science

Edmonton, Alberta
Spring, 1987

ISBN 0-315-37633-3

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Héctor Juan Hernández-López

TITLE OF THESIS: On the Boundedness of Relational Database Schemes with respect to Functional Dependencies

DEGREE FOR WHICH THIS THESIS WAS PRESENTED: Doctor of Philosophy

YEAR THIS DEGREE GRANTED: 1987

(Signed) Héctor J. H.

Permanent Address:
Senda De la Creación 4252
Fraccionamiento Villa las Fuentes
Monterrey, Nuevo León
México

Date: November 25, 86

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **On the Boundedness of Relational Database Schemes with respect to Functional Dependencies** submitted by **Héctor Juan Hernández-Lópes** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Edward [illegible]

Supervisor

Joseph Culberson

Wm. W. Armstrong

[illegible]

Date: November 21, 86

To my parents: Hector and Irene

(A mes parents: Hector et Irene)

(A mis padres: Hector e Irene)

ABSTRACT

Under the *Weak Instance Model (WIM)*, the *representative instance* can be used as a query-answering device via its total projections. However, generating the representative instance can be very expensive. Under this approach, it is very desirable to be able to answer queries by simulating the representative instance via a relational algebra expression which uses only the projection, join, and union operators. This is possible exactly when a database scheme is bounded with respect to dependencies. However, testing boundedness of relational database schemes with respect to dependencies is an extremely difficult problem to solve; it is believed to be undecidable even for simple cases where only functional dependencies are given.

Also there are other very desirable properties of relational database schemes which seem to be a consequence of boundedness. In particular, the problem of incremental enforcement of satisfaction of functional dependencies has very efficient solutions for the classes of database schemes known to be bounded with respect to functional dependencies.

In this thesis, we study under the WIM the characterization of database schemes which are bounded with respect to functional dependencies. We prove that gamma-acyclic Boyce-Codd Normal Form database schemes are bounded with respect to functional dependencies embodied in their schemes, and show that their boundedness implies this class of schemes is highly desirable with respect to query processing and incremental enforcement of satisfaction of functional dependencies. Then we show how to design database schemes bounded with respect to functional dependencies using a new technique called extensibility. Finally we present a sufficient condition for unboundedness when functional dependencies are considered.

This thesis is then an effort in identifying classes of relational database schemes which are highly desirable with respect to query processing and updates.

Acknowledgements

I would like to thank my supervisor Ed Chan. His guidance and support made this thesis possible.

I also would like to thank the members of my thesis committee, Professors W. W. Armstrong, Joe Culberson, Alberto Mendelzon (of University of Toronto), and Y. S. Wong for their many contributions and suggestions for this thesis. I also wish to thank Prof. Bob Reckhow, a member of my candidacy examination committee, for his comments on my thesis proposal. I also want to thank Prof. Tamer Ozsu; the motivation behind the topic of this thesis comes from a problem that I tried to solve as part of my term project for one of the courses he teaches.

I also would like to thank my friend Pepe Blakeley, at the University of Waterloo, who read this thesis and made valuable suggestions which improved its presentation. To my friends Pedro and Laura Celis, also at the University of Waterloo, thanks for their moral support.

To Maria Elena, my wife, who contributed a lot by enduring for so long and so far from our beloved country, my most gratefully thanks and love. To my children, Hector Luis, Helena Catalina, Hugo, and Hema a big kiss for their continuous cheering for their daddy. To my mother, relatives, and friends at Monterrey in Mexico, I would like to thank for their moral support.

Financially, this thesis was made possible by the support received from the National Council of Science and Technology of Mexico (CONACYT), from the Department of Computing Science, University of Alberta, and from the Natural Sciences and Engineering Research Council of Canada.

Table of Contents

List of Figures

Chapter 1

Introduction

The relational database model was introduced by Codd [Cod1]. Under this data model a database is viewed as a set of tables. Due to its structural simplicity and mathematical elegance the relational model is a formal framework in which we can formulate, study, and solve problems related to the management of databases. Since the inception of this model, a vast body of formal results has been obtained by a multitude of researchers. This is known as the theory of relational databases; for a survey in this topic see for example the textbooks by Maier [Ma] and Ullman [Ul].

A central problem in relational database theory is the schema design problem. The problem of schema design may be loosely stated as follows: given a description of an application, construct a database scheme that is "good" or "desirable" for the application. The "desirability" or "goodness" of a database scheme depends on the criteria we use to evaluate it.

In this thesis, we study under the relational model the design of database schemes which are very desirable with respect to (w.r.t.) query processing and enforcement of constraints.

The first criterion of "goodness" ever proposed for relational database schemes was freedom from update anomalies. Codd [Cod1] observed that with the presence of functional relationships, certain anomalies may exist when a relation is updated. He introduced first, second, and third normal forms [Cod2], via a process known as *normalization*, as a way to avoid those anomalies. Since then, some other normal forms have been proposed. Among these, Boyce-Codd Normal Form (BCNF) is one of the most important normal forms. A survey of normal forms can be found in standard texts like [Da][Ma][Ul].

The main approach to normalization of relational schemes is by decomposition;

that is, a relation scheme is represented by several new relation schemes, which are supposed to be more desirable w.r.t. freedom from update anomalies than the original one. These new schemes are the ones used to store the information. Then the question arises whether or not the decomposition preserves the information stored in the original relation. The following example illustrates this.

Example 1.1: Let $\{AB,\ BC\}$ be a decomposition of ABC and assume $abc = \{a_1b_1c_1,\ a_2b_1c_2\}$ is a relation on ABC. Then $ab = \{a_1b_1,\ a_2b_1\}$ and $bc = \{b_1c_1,\ b_1c_2\}$ are the relations on AB and BC respectively used to stored the information on ABC. However, $\{AB,\ BC\}$ does not preserve the information on ABC, since $abc \neq ab \bowtie bc = \{a_1b_1c_1,\ a_2b_1c_2,\ a_1b_1c_2,\ a_2b_1c_1\}$. □

If the decomposition of a relation scheme is *lossless* [ABU], then we are guaranteed that the decomposition preserves the information content of the original relation. Thus when decomposition of a scheme is involved, for any reason, in the design of relational database schemes, losslessness is a very desirable property.

The semantics of the data to be stored in a relation imposes certain constraints on the actual values that a relation can take. These constraints are modeled in relational database design using the idea of *data dependencies*. The most important and fundamental class of data dependencies is the class of *functional dependencies (fd's)* [A][Cod1]. In this thesis, we deal only with database schemes where the constraints considered are exclusively fd's.

The decomposition of a scheme also introduces the problem of preservation of dependencies. More precisely, the union of the dependencies defined on each scheme in the decomposition may not be logically equivalent to the set of dependencies that the user wants to consider as meaningful on the original scheme.

Example 1.2: Let $\{AC,\ BC\}$ be a decomposition of ABC and assume the fd's $\{AB \to C,\ C \to B\}$ must be obeyed by relations on ABC. $\{AC,\ BC\}$ does not preserve the given fd's, since $AB \rightarrow C$ is not logically implied by the fd's defined on the decomposition. □

If the decomposition does not preserve the dependencies, then the semantics of the information stored on the decomposition may change. Thus preserving dependencies in this sense is another desirable property of relational database schemes. Beeri and Honeyman [BH] gave a polynomial-time algorithm to test fd-preservation. In [BMSU], Beeri et al. defined preservation of dependencies in another sense.

The formalization of the above mentioned properties requires the concept of a *universal relation scheme* [FMU]; the scheme that represents our universe of discourse and models the global knowledge we have about the enterprise we are modeling. Along with a universal relation scheme concept, we require a precise definition of the universal relation we have in mind when dealing with it. The first universal relation assumption ever proposed was the *pure universal relation assumption* [FMU]; this assumption states that the relations stored in the database are exactly the projections of a satisfying universal relation. This assumption is very controversial and led to several published attacks [AP][BG][K], since, for instance, testing if a set of relations has a common universal instance is in general *NP*-complete [HLY].

Since a universal relation assumption is required to formalize, among other things, consistency of a database or its information content, several researchers [GMV][H2][M][S1][Y1] worked on this issue and proposed a more appealing notion of a satisfying universal relation: the *Weak Instance Model (WIM)*. This is a weaker notion of the pure universal relation assumption. The WIM states that the relations stored in the database are included in the projections of a satisfying universal relation.

The WIM was first proposed as a means to define satisfaction of fd's by a database [H2][V]. This universal relation model also provides an adequate and correct representation of the information content of a database via its *representative instance*

[M][S1][S2][Y1]. Intuitively, the representative instance of a database contains all the information that can be logically inferred from the database using certain rules derived from the semantic constraints that the database must satisfy. In this thesis, we work under the WIM framework.

Under the WIM, the representative instance can be used as a query-answering device. This idea was initially proposed by Sagiv [S1][S2] and Yannakakis [Y1]. The most popular approach to answer queries via the representative instance is based on the so-called total projections of the representative instance. In this approach, a query on a set of attributes X is answered based on X-total tuples in the representative instance, that is, using tuples that do not contain missing information on X. The set of total tuples on X in the representative instance is called *the X-total projection*.

A straightforward method to obtain the X-total projection is to compute the representative instance and then extract the tuples whose X-components are total. However computing the representative instance of a database can be very expensive. In the presence of fd's, it takes polynomial time and space in the size of the database. This is not practical, since in general the database is large.

Then under this approach, it is very desirable for query processing to have a database scheme that would allow a "cost-effective" way to compute the X-total projections. Under the WIM, we consider an algorithm that computes the X-total projections as cost-effective if it does not require the generation of the representative instance. A highly desirable database scheme in this respect is one that would allow the simulation of the representative instance via a relational expression which only uses the projection, join, and union operators. This is the case exactly when the database scheme is bounded w.r.t. the dependencies given [GM][MUV]. The main objective of this thesis is to study the characterization of database schemes which are bounded w.r.t. fd's.

Unfortunately, the problem of testing boundedness of database schemes is an extremely difficult problem to solve; it is conjectured to be undecidable even for simple cases where only fd's are given [MUV]. Thus it is understandable why most research on this problem so far has concentrated on finding sufficient conditions for boundedness; we describe this work below.

The work on boundedness has mainly centered around independent database schemes. A database scheme is *independent* w.r.t. a set of dependencies if verifying that each relation in a database state satisfies its local dependencies is sufficient to ensure that the database state is globally consistent w.r.t. the given set of dependencies [GY][IIK][S1][S2]. Sagiv [S2] proved that independent BCNF database schemes are bounded w.r.t. fd's embodied in the database scheme. Later on, a more general class of independent database schemes was proven to be bounded w.r.t. fd's embedded in the database scheme. This was done independently by several researchers [AtC][C1][IIK][MRW][S3]. As a consequence of this work, the largest class of database schemes known to be bounded w.r.t. fd's is the class of independent database schemes.

Recently, Brossda and Vossen [BVo] used a modified version of Sagiv's uniqueness condition [S2] plus the foreign-key constraint [S1] to define a class of database schemes that is bounded w.r.t. fd's.

The importance of knowing whether a database scheme is bounded goes beyond cost-effective query processing. There are other very desirable properties of database schemes which seem to be its consequences. In particular, the problem of enforcement of constraints has efficient solutions for the classes of database schemes which are known to be bounded w.r.t. fd's. In fact, we are interested in boundedness of database schemes because one of the objectives of this thesis is in characterizing database schemes in which enforcement of fd's can be done efficiently. Efficient enforcement of constraints is concerned with determining very efficiently if an update to a database which satisfies a set of dependencies, produces a database which still satisfies the given dependencies. In general, however, efficient enforcement of constraints is a difficult problem. Some work has been done on this problem, see for example [BBC][St].

Under the WIM, the problem of enforcement of fd's can be solved in polynomial time in the number of tuples in the database state. Honeyman [H2] and independently Vassiliou [V] presented algorithms to test satisfaction of fd's by a database state. Their time complexities are $O(n \log_2 n)$ and $O(n^4 \log_2 n)$ respectively, where n is the number of tuples in the database state. These algorithms can be used as start-over algorithms for enforcing fd's after an update on a consistent database state. The fact that their algorithms do not exploit the consistency of the database state before an insertion, led Chan [C2] to investigate an incremental approach for enforcing fd's. He presented an algorithm whose time complexity is $O(n^2 \log_2 n)$. The main advantage of these algorithms is that they can be used for any database scheme where only fd's are considered. However, these algorithms are not good enough from the practical point of view, since checking an insertion via those algorithms may require accessing the whole database state.

Even with only fd's, it is not clear if they can be enforced very efficiently in real-life applications. One way to resolve this problem is to find a class of database schemes that would allow a "cost-effective" way to determine if an updated state satisfies the constraints. As in the case of query processing, under the WIM we regard an algorithm for incrementally testing fd's as cost-effective if it does not require the generation of the representative instance, but additionally we require that the verification process be done on some specific relations efficiently. The class of independent database schemes mentioned above is a class that allows cost-effective enforcement of fd's. For an independent scheme, ensuring that the constraints imposed on each relation are satisfied is sufficient to guarantee that the state globally satisfies the constraints. Therefore the problem of ensuring that a database satisfies a set of dependencies is reduced to the problem of verifying that each relation satisfies the constraints locally. Since checking that each relation satisfies the local constraints does not require the generation of the representative instance, this class of database schemes allows enforcement of constraints to be carried out cost-effectively.

Recently, Graham and Wang [GW] provided a generalization of independent database schemes via the concept of constant-time-maintainability. They defined and characterized the class of database schemes for which the *maintenance problem* [GW][GY] has solutions in time independent of the size of the database state. The maintenance problem is the problem of how to ensure that a consistent database state satisfies its constraints after an insertion. They defined a database scheme to be *constant-time-maintainable (ctm)* if there is an algorithm that solves the maintenance problem for any of its consistent database states in constant time. Independent database schemes are ctm by definition.

Previous to Graham and Wang's work, Brossda and Vossen [BVo] used a modification of Sagiv's uniqueness condition [S2] plus the foreign-key constraint [S1] to define a class of ctm database schemes. Because they used the foreign-key constraint, they have to check that a consistent database satisfies this constraint after a deletion, and to do this, their algorithm takes time proportional to the size of the database.

Since constant-time solutions to the maintenance problem are crucial and fundamental in real-life applications, ctm database schemes are highly desirable.

As can be observed from the previous summary of work in the maintenance and boundedness problems, there is a strong relationship between ctm database schemes and bounded database schemes. Prior to our work, database schemes have been proven to be bounded after defining conditions for them to be ctm database schemes. In fact, in [GW] it is conjectured that the class of ctm database schemes characterized by them is bounded. Although their conjecture seems to be supported by results in [GM], no proof has been given in this respect.

Unlike Graham and Wang [GW], in this thesis we approach the characterization of bounded and ctm database schemes from the opposite direction: First we prove boundedness for a class of database schemes, and then we show that constant-time-maintainability follows for that class. This approach to boundedness and constant-time-maintainability of database schemes is completely unexplored, because proving boundedness seems to be extremely difficult, if possible at all. Nevertheless, we conjecture that if we can effectively characterize boundedness w.r.t. fd's for a class of database schemes, then we can prove constant-time-maintainability for that specific class. This conjecture seems appropriate simply because a bounded database scheme is a "well-behaved" database scheme. However, by results in [GM] this is not a direct consequence; certain conditions are required apart from boundedness in order to have constant-time-maintainability of a database scheme if we are considering only fd's.

This thesis is then an effort in identifying classes of relational database schemes which, under the WIM, are highly desirable w.r.t. query processing and updates.

1.1. Overview of Thesis

In Chapter 2, we give the basic background and definitions of relational database theory required in this thesis. Other definitions will be given where required.

In this thesis, we study first boundedness of database schemes w.r.t. fd's. This is done in Chapters 3 and 4. Having done this, we study unboundedness of database schemes w.r.t. fd's in Chapter 5.

In Chapter 3, we prove that γ-acyclicity and BCNF is a sufficient condition for boundedness of database schemes w.r.t. fd's embodied in their relation schemes. Contrary to our initial hopes, but as should be expected from the undecidability conjecture in [MUV] for the boundedness problem, even for this restricted class of database schemes our proof of boundedness is long and complex. Also in Chapter 3, we show for the class of γ-acyclic BCNF database schemes, that its boundedness implies that this class of database schemes is highly desirable w.r.t. query processing and incremental enforcement of satisfaction of fd's. This last fact supports our conjecture that if we can prove boundedness w.r.t. fd's for a class of database schemes, then we can prove constant-time-maintainability for that class.

From our results in Chapter 3, it is apparent that determining whether or not a class of database schemes is bounded is fundamental to the analysis of the behavior of the database schemes w.r.t. query processing and updates. On the other hand, proving a class of database schemes is bounded seems to be very difficult, even in our restricted case of γ-acyclic BCNF database schemes. To resolve this problem, we need to develop techniques for characterizing bounded database schemes. In Chapter 4, we investigate an alternative approach. We show how to design database schemes bounded w.r.t. fd's using a new technique called *extensibility*. This technique can also be used to design ctm database schemes.

In Chapter 5, we investigate unboundedness w.r.t. fd's, the other side of the coin in the problem of boundedness. This problem heretofore remained unexplored. We show that the unboundedness problem is not as difficult as its counterpart, the boundedness problem, in the sense that we can prove unboundedness for a very general class of database schemes in a shorter and easier proof than the one for boundedness for the restricted class of database schemes in Chapter 3. We present a very general and sufficient condition for unboundedness when fd's are considered. The condition is very general in the sense that neither the fd's nor the schemes are restricted to be in some specific form.

Finally in Chapter 6, we present our conclusions and suggestions for future research.

Chapter 2

Relational Background

In this chapter, we give most of the notation required for the rest of this thesis. Additional definitions will be given when they are needed.

2.1. Basic Definitions

We fix a finite set of attributes $\mathbf{U} = \{A_1, A_2, \ldots, A_m\}$, which we call the *universe*. Following traditional relational database theory notation [Ma][U1], we omit brackets and commas when representing sets of attributes, and we represent the union of two sets of attributes X and Y as XY.

A *relation scheme* R is any nonempty subset of $\mathbf{U}$. With each attribute A_i, there is a set of associated values $dom(A_i)$, called the *domain* of A_i. A *tuple* t over R is a mapping t such that for all $A_i \in R$, $t[A_i] \in \text{dom}(A_i)$. A tuple t over R is denoted as $t[R]$. If t is a tuple over R and $X \subseteq R$, $t[X]$ is the restriction of t to the attributes in X. A *relation* r over R is a set of tuples over R. A relation defined over $\mathbf{U}$ is called an *universal relation.*

Functional dependencies (fd's) [A][Cod1] are statements of the form $X \rightarrow Y$, where X and Y are sets of attributes such that XY is included in a relation scheme R. Semantically, a relation r on R satisfies $X \rightarrow Y$ if whenever there exist two tuples t and u in r such that if $t[X] = u[X]$, then $t[Y] = u[Y]$. An fd $X \rightarrow Y$ is *trivial* if $X \supseteq Y$. A set of fd's F *logically implies* an fd d, written $F \models d$, if every relation that satisfies F also satisfies d. The set of fd's that is logically implied by a set of fd's F is called the *closure of* F; we denote it by F^+. There exists a complete and sound set of inference rules to derive F^+ [A]. A set of fd's G is a *cover* for a set of fd's F if $G^+ = F^+$. Given a set of attributes X, we can compute the *closure* of X w.r.t. a set of fd's F, which is $\{A \mid X \rightarrow A \in F^+\}$. This is denoted by X_F^+, or by X^+ if F is clearly understood. $X \rightarrow Y$ is *embedded* in a relation scheme R if $XY \subseteq R$. We denote the set of fd's $X \rightarrow Y \in F^+$

such that XY is embedded in a relation scheme R by $F^+|R$. This set is called *the projection* of F onto R, or *the set of projected* fd's.

Given a set of fd's F, a nonempty subset K of a relation scheme R is called a *key* of R if $K \to R \in F^+$ and no proper subset of K has this property. If K is a key of R, we say that R *embodies* the fd $K \to R - K$. If K is a key of R and the nontrivial fd $K \to A \in F^+$ is embedded in R, we say that K is a *key (of R) that determines* A.

A *database scheme* is a pair ($\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$, F) such that R_i, $1 \le i \le n$, is a relation scheme, F is a set of fd's defined on $\mathbf{U}$, and $\mathbf{U}$ is the union of the R_i's. A database scheme ($\mathbf{R}$, F) is *cover embedding (w.r.t. F)* if there exists a cover G of F such that each fd in G is embedded in some scheme in $\mathbf{R}$. A database scheme ($\mathbf{R}$, F) is a *Boyce-Codd Normal Form (BCNF)* database scheme if for all nontrivial $X \to Y \in F^+$ embedded in some R_i, $R_i \in \mathbf{R}$, X contains a key of R_i. If ($\mathbf{R}$, F) is a BCNF database scheme, we assume the keys of every relation scheme in $\mathbf{R}$ are explicitly given, and F is the set of fd's embodied in the relation schemes in $\mathbf{R}$. For the sake of brevity, only a cover of F is given in the examples involving BCNF database schemes. A *(database) state* for a database scheme ($\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$, F) is $\mathbf{r} = (r_1, r_2, \ldots, r_n)$ such that for $1 \le i \le n$, r_i is a relation over R_i.

The only relational operations that are required in this thesis are projection, join, and union. If r_i is a relation defined over R_i, the *projection* of r_i onto X, $X \subseteq R_i$, is $\pi_X(r_i) = \{t[X] \mid t \in r_i\}$. If r and s are relations over R and S respectively, the *join* of r and s, denoted by $r \bowtie s$, is a relation over RS such that $r \bowtie s = \{t \mid t[R] \in r$ and $t[S] \in s\}$. The *union* of r and s, denoted by $r \cup s$, where r and s are relations defined over the same set of attributes, is the set of tuples that are in r, or s, or both.

We shall consider relational expressions [Cod1] in which the only operators are projection, join, and union. The operands are relation schemes in a database scheme. Let E be a relational expression with operands in $\mathbf{R} = \{R_1, \ldots, R_n\}$. Then $E(\mathbf{r})$

denotes the value returned by E if a state $\mathbf{r} = (r_1, \ldots, r_n)$ of $(\mathbf{R}, F)$ is substituted into the corresponding relation variables and the expression is evaluated according to the above definitions for the relational operations.

Let $S = \{S_1, \ldots, S_k\}$ and $\bigcup_i S_i \subseteq \mathbf{U}$. An *embedded join dependency (ejd)* is a statement of the form $\bowtie S$. A relation I satisfies $\bowtie S$ if $I[\bigcup_{i=1}^{k} S_i] = \bowtie_{i=1}^{k}(\pi_{S_i}(I))$. A set of fd's F ***logically implies*** an ejd d, written $F \models d$, if every relation that satisfies F also satisfies d. A database scheme $(\mathbf{R}, F)$ is ***lossless w.r.t.*** F if $F \models \bowtie \mathbf{R}$. S is ***lossless w.r.t.*** F if $F \models \bowtie S$.

2.2. The Weak Instance Model

The universal relation model we are working with in this thesis is the ***weak instance model*** [GMV][H2][M][S1][V][Y1]. Given a state $\mathbf{r}$ for the database scheme $(\mathbf{R} = \{R_1, R_2, \ldots, R_n\}, F)$, we say that I_r a relation over $\mathbf{U}$, is a ***weak instance*** for $\mathbf{r}$ w.r.t. F if

- $\pi_{R_i}(I) \supseteq r_i$, for $1 \leq i \leq n$; and
- I satisfies F.

Under the weak instance model, a database state $\mathbf{r}$ of $(\mathbf{R}, F)$ is said to be ***consistent*** (w.r.t. F) if a weak instance exists for that state w.r.t. F [H2][GMV].

A ***tableau*** is a set of tuples defined on $\mathbf{U}$ [ASU]. We denote a tableau by either a single letter, usually T, or by listing its components explicitly, $(t_1, t_2, \ldots, t_m)/t$; the t_i's are the ***rows*** of the tableau. The ***domain of*** A_i ***in the tableau*** consists of the ***distinguished variable (dv)*** a_i, countably many ***nondistinguished variables (ndv's)*** $\{b_j\}$, and constants drawn from $\mathrm{dom}(A_i)$. An ndv which appears only once in the tableau is called an ***unique*** ndv. No variable can appear in more than one column in the tableau.

A *valuation function* v: $S_1 \rightarrow S_2$ is a function from symbols on S_1 to symbols on S_2, which is the identity on constants, maps dv's to dv's or constants, and ndv's to ndv's, dv's, or constants. Assume T_1 and T_2 are tableaux. A *containment mapping* h: $T_1 \rightarrow T_2$ is a valuation function from the set of symbols in the rows of T_1 ~~to~~ the symbols in the rows of T_2 such that if t_i is a row in T_1, then $h(t_i)$ is a row in T_2.

A tableau T_1 is said to *contain* T_2, written $T_1 \supseteq T_2$, if there is a containment mapping from T_1 to T_2. T_1 is *equivalent* to T_2, written $T_1 \equiv T_2$, if and only if $T_1 \supseteq T_2$ and $T_2 \supseteq T_1$.

Given a state $\mathbf{r}$ for the database scheme $(\mathbf{R}, F)$, the *tableau* for $\mathbf{r}$, written T_r, is defined as follows. For each relation $r_i \in \mathbf{r}$, and for each tuple $t \in r_i$, there is a row s in T_r such that $s[R_i] = t$, and, for all A in $\mathbf{U}-R_i$, $s[A]$ is δ_l, where δ_l appears once in the rows of T_r.

Example 2.1: Let $(\mathbf{R}, F) = (\{R_1(CT), R_2(HRC), R_3(HTR), R_4(CSG), R_5(HSR)\}, \{C \rightarrow T, HR \rightarrow C, HT \rightarrow R, CS \rightarrow G, HS \rightarrow R\})$ and let a state of $(\mathbf{R}, F)$ be $\mathbf{r} = (\ r_1 = \{<c_1, t_1>, <c_2, t_1>, <c_3, t_2>\}, r_2 = \{<h_1, r_1, c_1>, <h_2, r_1, c_2>\}, r_3 = \{<h_2, t_1, r_1>\}, r_4 = \{<c_1, s_1, g_1>, <c_2, s_2, g_2>\}, r_5 = \{<h_1, s_1, r_1>, <h_1, s_2, r_2>\})$. Figure 2.1 shows the tableau for $\mathbf{r}$. □

Given the tableau T_r for a database state $\mathbf{r}$ of $(\mathbf{R}, F)$, we associate with each fd $X \rightarrow Y$ in F the following *fd-rule for* $X \rightarrow Y$: If T_r has two rows t and u such that $t[X] = u[X]$, but they are not equal on some columns of Y, then for all columns A in Y such that $t[A] \neq u[A]$ do,

- if $t[A] = \delta_k$ and $u[A] = \delta_i$, then replace all occurrences of δ_i in T_r by δ_k,
- else if $t[A] = c$, c not an ndv, and $u[A] = \delta_i$, then replace all occurrences of δ_i in T_r by c;
- otherwise if none of the above two cases hold, we obtain the empty tableau.

C	T	H	R	S	G
c_1	t_1	δ_1	δ_2	δ_3	δ_4
c_2	t_1	δ_5	δ_6	δ_7	δ_8
c_1	δ_9	h_1	r_1	δ_{10}	δ_{11}
c_2	δ_{12}	h_2	r_1	δ_{13}	δ_{14}
δ_{15}	t_1	h_2	r_1	δ_{16}	δ_{17}
δ_{18}	δ_{19}	h_1	r_1	s_1	δ_{20}
δ_{21}	δ_{22}	h_1	r_2	s_2	δ_{23}
c_1	δ_{24}	δ_{25}	δ_{26}	s_1	g_1
c_2	δ_{27}	δ_{28}	δ_{29}	s_2	g_2
c_3	t_2	δ_{30}	δ_{31}	δ_{32}	δ_{33}

Figure 2.1 T_r for r in Example 2.1

The *chase* of T_r w.r.t. F is the process of repeatedly applying to T_r the rules for the fd's in F as long as a change can be made. The nonempty tableau obtained after no more fd-rules can be applied is called $CHASE_F(T_r)$ [ABU][BV][MMS]. It has been shown in [H2] that for any nonempty state r, $CHASE_F(T_r)$ is nonempty if and only if r is a consistent state. $CHASE_F(T_r)$ is called the *representative instance* for state r. For a given database, the representative instance is, in some sense, exactly the common information of all its weak instances [GMV][M][MUV].

Example 2.2: $CHASE_F$ for the tableau of Example 2.1 is shown below in Figure 2.2. □

Assume r is a state of (**R**, F) and let T_r be its tableau. Let t be a tuple in T_r and let $X \subseteq$ **U**. We say that t is *total* on X if for all $A \in X$, $t[A]$ is not an ndv. Also, we define $[X]$ as $\{t[X] \mid t \in CHASE_F(T_r)$ and t is total on $X\}$. We also say that $[X]$ is the *X-total projection* of the representative instance for r.

The set of all consistent states for a database scheme (**R**, F) is $WSAT$(**R**, F) = {r| r is a state of (**R**, F) and r is consistent w.r.t. F}. A relation r_i is *consistent* w.r.t.

C	T	H	R	S	G
c_1	t_1	δ_1	δ_2	δ_3	δ_4
c_2	t_1	δ_5	δ_6	δ_7	δ_8
c_1	t_1	h_1	r_1	δ_{10}	δ_{11}
c_2	t_1	h_2	r_1	δ_{13}	δ_{14}
c_2	t_1	h_2	r_1	δ_{16}	δ_{17}
c_1	t_1	h_1	r_1	s_1	g_1
δ_{21}	δ_{22}	h_1	r_2	s_2	δ_{23}
c_1	t_1	δ_{25}	δ_{26}	s_1	g_1
c_2	t_1	δ_{28}	δ_{29}	s_2	g_2
c_3	t_2	δ_{30}	δ_{31}	δ_{32}	δ_{33}

Figure 2.2 $CHASE_F$ of T_r in Example 2.1

$F^+ | R_i$ if there is a universal relation I satisfying F such that $\pi_{R_i}(I) \supseteq r_i$. The locally consistent states of (**R**, F) are elements of the set $LSAT$(**R**, F)= { **r**| r_i is consistent w.r.t. $F^+ | R_i$, for each $r_i \in$ **r**}.

A database scheme (**R**, F) is said to be *independent* (w.r.t. F) if and only if $LSAT$(**R**, F)= $WSAT$(**R**, F) [GY][HK][S1][S2]. It has been shown that (**R**, F) is independent if verifying that each relation in a state of (**R**, F) satisfies its projected fd's is sufficient to ensure that the state is consistent [GY].

There are several equivalent definitions of ~~boundedness of~~ a database scheme w.r.t. dependencies. Unless otherwise stated, the following is the one we assume.

Let $[X]_r$ denote the X-total projection of the representative instance for **r** and let $|\mathbf{r}|$ denote the number of tuples in **r**. Then we say that a database scheme (**R**, F) is *bounded* (w.r.t. F) if for all $X \subseteq \mathbf{U}$ there is a constant $k>0$ such that, for every consistent state **r** of (**R**, F), and for every $t \in [X]_r$, there exists a substate **r**' of **r** such that $t \in [X]_{r'}$ and $|\mathbf{r}'| \leq k$ [GM]. We say that a database scheme (**R**, F) is *unbounded* (w.r.t. F) if it is not bounded (w.r.t. F).

We now give the canonical example of unbounded database schemes.

Example 2.3: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(AC), R_3(CB)\}, \{A \to B, C \to B\})$. Let a state of $(\mathbf{R}, F)$ be $\mathbf{r} = (r_1 = \{\langle a_1, b_1 \rangle\}, r_2 = \{\langle a_1, c_1 \rangle, \langle a_2, c_1 \rangle, \langle a_2, c_2 \rangle, \ldots, \langle a_{n-1}, c_{n-1} \rangle, \langle a_n, c_{n-1} \rangle\}, r_3 = \varnothing)$. T_r for that state is shown in Figure 2.3 below; unique ndv's are represented by "-". r is consistent.

A	B	C
a_1	b_1	-
a_1	-	c_1
a_2	-	c_1
a_2	-	c_2
•	•	•
•	•	•
•	•	•
a_{n-1}	-	c_{n-1}
a_n	-	c_{n-1}

Figure 2.3 T_r for r in Example 2.3

Observe $t = \langle a_n, b_1, c_{n-1} \rangle$ is in the representative instance of r. However, since the representative instance of any proper substate of r does not contain $\{t\}$, $(\mathbf{R}, F)$ is unbounded. □

The largest class of database schemes known to be bounded w.r.t. fd's is the class of independent database schemes [AtC][C1][IIK][MRW][S3].

2.3. Hypergraphs for Database Schemes

A *hypergraph* is a pair $H = <V, E>$, where V is a set of *nodes* and E is a collection of nonempty subsets of V called *edges* [B].

Given a database scheme $(\mathbf{R}, F)$, its hypergraph, denoted by $H_{\mathbf{R}}$, has U as its set of nodes, and $\mathbf{R}$ as its set of edges. If $(\mathbf{R}, F)$ is a BCNF database scheme, we are also interested in the hypergraph of R_i^+, $R_i \in \mathbf{R}$, denoted by $H_{R_i^+}$. $H_{R_i^+}$ has R_i^+ as its set of nodes, and its set of edges is formed by the R_j's included in R_i^+. It is clear that $H_{R_i^+}$ is a subgraph of $H_{\mathbf{R}}$.

We now define the concept of cycle in a hypergraph. There are, however, several degrees of cyclicity for hypergraphs [DM][F]. Among these, the most interesting are *Berge*-, α-, β-, and γ-cyclicity [F]. Following [ADM][F], we give below the required terminology of hypergraphs used in this thesis.

Let $H = <V, E>$ be a hypergraph. A *path* from x_1 (E_1) to x_m (E_m) is a sequence $<E_1, E_2, \ldots, E_m>$ such that:

- $x_1 \in E_1$ and $x_m \in E_m$;
- $E_1, E_2, \ldots, E_m$ are edges in E, $m \geq 1$;
- $E_k \cap E_{k+1} \neq \emptyset$, for $k = 1, 2, \ldots, m-1$;
- no proper subsequence of it satisfies the above properties.

Two nodes (edges) are *connected* if there exists a path from one to the other. $H = <V, E>$ is *connected* if every pair of nodes (edges) in H are connected.

Example 2.4: Let $(\mathbf{R}, F) = (\{R_1(CT), R_2(HRC), R_3(HTR), R_4(CSG), R_5(HSR)\}, \{C \to T, HR \to C, HT \to R, CS \to G, HS \to R\})$ be a database scheme. In $H_{\mathbf{R}}$, we have that $<R_2, R_5>$ and $<R_4>$ are paths from C to S. □

A γ-*cycle* of length m is a sequence $<E_1, x_1, \cancel{E_2, x_2}, \ldots, E_m, x_m, E_{m+1}>$ such that:

- $x_1, x_2, \ldots, x_m$ are all distinct nodes of H;
- $E_1, E_2, \ldots, E_m$ are all distinct edges in E, and $E_1 = E_{m+1}$;
- $m \geq 3$;
- x_k is in E_k and E_{k+1}, for $k = 1, 2, \ldots, m$;
- if $1 \leq i < m$, then x_i is in no E_j except E_i and E_{i+1}.

A hypergraph $H = \langle V, E \rangle$ is *γ-acyclic* if H does not have a γ-cycle; otherwise it is γ-cyclic. Similarly, a database scheme $(\mathbf{R}, F)$ is *γ-acyclic* if $H_{\mathbf{R}}$ does not have a γ-cycle; otherwise it is γ-cyclic.

Example 2.5: The hypergraph of the BCNF database scheme $(\{R_1(ABC), R_2(AD), R_3(DE), R_4(ABG), R_5(BCEF), R_6(FG)\}, \{F \rightarrow G, BCE \rightarrow F, AB \rightarrow G, D \rightarrow E, A \rightarrow D\})$ has the following γ-cycle in $H_{R_1^+}$: $\langle R_4, B, R_5, F, R_6, G, R_4 \rangle$. See Figure 2.4 below. $\langle R_1, A, R_4, B, R_1 \rangle$ is not a γ-cycle since $m = 2$. Notice that $\langle R_1, A, R_2, D, R_3, E, R_5, F, R_6, G, R_4, B, R_1 \rangle$ is not a γ-cycle either since A is in R_1, R_2, and R_4. □

We are not going to define *Berge*-, β- or α-cyclicity; we refer the interested reader to [F].

We shall prove in Chapter 3 that γ-acyclic BCNF database schemes are bounded.

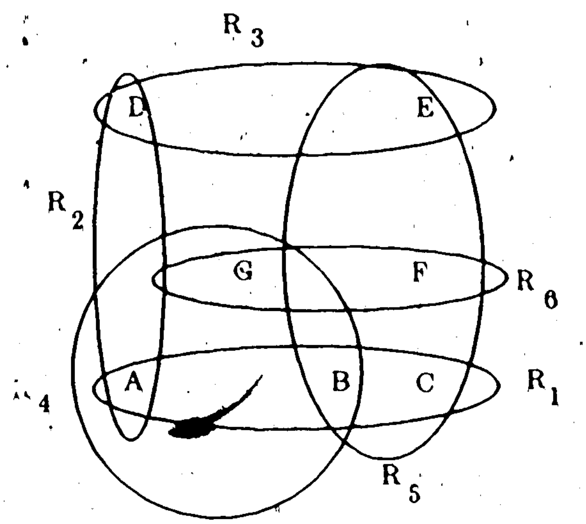


Figure 2.4 H_R for Example 2.5

Chapter 3

A Sufficient Condition for Boundedness w.r.t. Fd's

3.1. Introduction

In this chapter, we present a sufficient condition for boundedness of database schemes w.r.t. fd's and show that the class of database schemes characterized by this condition is highly desirable w.r.t. query processing and enforcement of fd's.

First we give the intuition behind the condition presented in this chapter. In order to do this, let us consider below the canonical example of unbounded database schemes that shows what seems to be a crucial factor involved in unboundedness w.r.t. fd's.

Example 3.1: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(AC), R_3(CB)\}, \{A \to B, C \to B\})$. From Example 2.3, $(\mathbf{R}, F)$ is unbounded. Figure 3.1 shows its hypergraph. Observe that $(\mathbf{R}, F)$ is cyclic. □

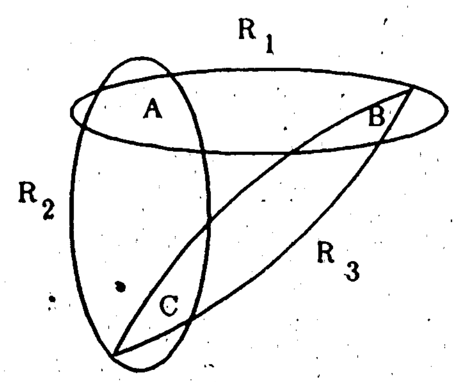


Figure 3.1 $H_{\mathbf{R}}$ for Example 3.1

We believe that some sort of cyclicity is responsible for the unboundedness of a database scheme when fd's are considered. In other words, we believe that by restricting the structure of the database scheme, in terms of hypergraphs, we may have boundedness w.r.t. fd's for some class of acyclic database schemes. Therefore let us restrict ourselves to the class of γ-acyclic database schemes, a subclass of acyclic database schemes with some crucial properties that we feel make tractable the problem of testing boundedness of database schemes w.r.t. fd's. We consider a simple γ-acyclic database scheme in the following example and see whether or not it is bounded.

Example 3.2: Let $(\mathbf{R}, F) = (\{R_1(AC), R_2(ABC)\}, \{A \rightarrow B, C \rightarrow B\})$. $(\mathbf{R}, F)$ is γ-acyclic. But it is unbounded; the state in the tableau in Figure 3.2 below can be used to prove it. □

A	B	C
a_1	b_1	c_0
a_1	-	c_1
a_2	-	c_1
a_2	-	c_2
●	●	●
●	●	●
●	●	●
a_{n-1}	-	c_{n-1}
a_n	-	c_{n-1}

Figure 3.2 T_r for Example 3.2

From Example 3.2, it is clear that γ-acyclicity by itself is not a sufficient condition for boundedness w.r.t. fd's. We conjecture that in this example we have unboundedness because there still exists some kind of cyclicity on a hypergraph whose edges are the sets of attributes on which the fd's are defined; for instance, the hypergraph for AB (the attributes in $A \rightarrow B$), BC (the attributes in $C \rightarrow B$), and AC is cyclic. See Figure 3.1 above. We have to add some other restriction to γ-acyclicity to make the above sort of cycle on the schemes of the fd's disappear.

One of the most important and desirable normal forms for database schemes when fd's are given is BCNF. Although there are problems with the construction of BCNF database schemes in general [BB][BG][LO][O], under certain reasonable assumptions, it has been shown that BCNF database schemes are free from anomaly problems [LeP]. In fact, LeDoux and Parker [LeP] suggested that BCNF is a useful design criterion and showed that the problems with BCNF database schemes do not exist in most real-life applications. With fd's as the constraints imposed on the database schemes, we believe BCNF is a good design goal toward which a database designer should strive, since this class of database schemes seems to capture the principle of separation stated in [BBG].

Returning to the database scheme in Example 3.2, observe that it is not BCNF. In the following example, we add to that database scheme the fd's that it is missing in order to be BCNF and consider its boundedness.

Example 3.3: Let $(\mathbf{R}, F) = (\{R_1(AC), R_2(ABC)\}, \{A \rightarrow B, C \rightarrow B\})$. $(\mathbf{R}, F)$ is γ-acyclic, but it is not BCNF. We need to add $A \rightarrow C$ and $C \rightarrow A$ to F for $(\mathbf{R}, F)$ to be a BCNF database scheme. Notice $F \cup \{A \rightarrow C, C \rightarrow A\} \equiv G = \{A \rightarrow C, C \rightarrow AB\}$. More important, also observe that the structure of a hypergraph whose edges are the sets of attributes on which the fd's in G are defined is the same as $H_{\mathbf{R}}$. That is, in a γ-acyclic BCNF database scheme the structure of the schemes for the fd's and relations match in the above sense. It is not difficult to see that $(\mathbf{R}, G)$ is bounded. □

We claim that γ-acyclic BCNF database schemes are bounded w.r.t. the fd's embodied in their relation schemes.

In this chapter, we first prove that the class of γ-acyclic BCNF schemes is bounded w.r.t. the fd's embodied in their relation schemes. We then show that this class of database schemes is simple in semantics by proving that there is a simple and efficient way to compute the X-total projection of the representative instance. Since the set of total tuples represents the information content of a database [GMV][M][MUV][NG][S1][S2], the user is able to understand the semantics of the application easily. Answers to many queries for this class of database schemes can also be generated efficiently. We then show that if a γ-acyclic BCNF database scheme is lossless, then it is connection-trap-free [CA]. This demonstrates that the class of γ-acyclic BCNF database schemes is highly desirable w.r.t. query processing.

Furthermore, supporting our conjecture that boundedness is a sufficient condition for solving the maintenance problem efficiently, we also show that the class of γ-acyclic BCNF database schemes allows enforcement of constraints to be performed cost-effectively. This demonstrates the desirability of this class of schemes w.r.t. updates.

The only other known class of database schemes with all these desirable properties is the class of independent and connection-trap-free database schemes [CA]. So the result in this chapter is another effort in identifying classes of highly desirable database schemes w.r.t. query processing and updates.

3.2. Overview of Chapter

The plan for the rest of this chapter is as follows. In Section 3.3, we give some definitions needed for this chapter. In Section 3.4, we present an algorithm to chase a consistent state of a γ-acyclic BCNF database scheme. In Section 3.5, we prove that γ-acyclic BCNF database schemes are bounded w.r.t. fd's embodied in their relation schemes. In Section 3.6, we show that there is a simple and efficient method for computing the X-total projection of the representative instance for a γ-acyclic BCNF database scheme. In Section 3.7, we prove that lossless γ-acyclic BCNF database schemes are connection-trap-free. In Section 3.8, we present an efficient algorithm to test incrementally the satisfaction of fd's for a γ-acyclic BCNF database scheme. After that, we give our conclusions in Section 3.9.

3.3. Some Definitions and Properties of γ-acyclic Hypergraphs

In this chapter we use the following definition of boundedness. A database scheme $(\mathbf{R}, F)$ is *bounded* (w.r.t. F) if for every tuple t in the representative instance of any consistent state $\mathbf{r}$ of $(\mathbf{R}, F)$, t's total part can be obtained in at most k fd-rule applications starting from T_r, for some constant $k \geq 0$ [GM][MUV].

In what follows we give some useful properties of γ-acyclic hypergraphs that we use later on in this chapter.

Given a family of sets $E = \{E_1, \ldots, E_n\}$, $Bachman(E)$ is defined as follows:

- if $E_i \in E$, then $E_i \in Bachman(E)$;
- if X and Y are in $Bachman(E)$, then $X \cap Y$ is in $Bachman(E)$.

A family of sets $\{W_1, \ldots, W_m\}$ is *connected* if the hypergraph $H = <\bigcup_{i=1}^{m} W_i, \bigcup_{i=1}^{m} \{W_i\}>$ is connected.

A connected set $V = \{V_1, \ldots, V_m\} \subseteq$ Bachman($\mathbf{R}$) is the *unique minimal connection (u.m.c.) (among)* $X \subseteq \mathbf{U}$, if

- $\bigcup_{i=1}^{m} V_i \supseteq X$, and
- for all connected subsets $\{W_1, \ldots, W_k\}$ of Bachman($\mathbf{R}$) such that $\bigcup_{i=1}^{k} W_i \supseteq X$, there exists $\{W_{i_1}, \ldots, W_{i_m}\} \subseteq \{W_1, \ldots, W_k\}$ such that $W_{i_j} \supseteq V_j$, for $1 \leq j \leq m$.

There are several efficient methods of finding the u.m.c. [BBSK][C3][Y2]. The following result concerning u.m.c.'s and γ-acyclic hypergraphs is stated in [F][Y2], and recently proven in [BBSK].

Theorem 3.1: Let $\mathbf{R}$ be connected. $\mathbf{R}$ is γ-acyclic if and only if $\mathbf{R}$ has the u.m.c. among any $X \subseteq \mathbf{U}$.

Another useful property of γ-acyclic hypergraphs is the following from [ADM].

Theorem 3.2: A hypergraph is γ-acyclic if and only if for every pair of its nodes n and m, all paths from n to m have the same length.

Example 3.4: $H_{\mathbf{R}}$ in Example 2.4 is γ-cyclic (and therefore (**R**, F)) since there are two paths from C to S of different length. □

In the rest of this chapter when we refer to (a)cyclicity we shall be referring to γ-(a)cyclicity.

3.4. Algorithms for Proving Boundedness

In this section, we present the algorithms used in this chapter to prove that acyclic BCNF database schemes are bounded. We present first Algorithm 1, the algorithm we use to compute $H_{R_i^+}$ for each $R_i \in \mathbf{R}$, and for any BCNF database scheme (**R**, F). Then we prove in Lemma 3.1 the key observation behind our results about computations of R_i^+ in the following section. Having done that, we introduce an algorithm that chases the tableau T_r for a consistent state $\mathbf{r}$ of an acyclic BCNF database scheme in a particular way.

Algorithm 1 is shown below. We associate with each computation of $H_{R_i^+}$ a sequence $S_i = <S_{i_0}, S_{i_1}, \ldots, S_{i_m}>$, which consists of the relation schemes in R_i^+ in the order in which they were added to $H_{R_i^+}$ by Algorithm 1; $S_{i_0} = R_i$.

Example 3.5: Let the input to Algorithm 1 be the BCNF database scheme (**R** $=\{R_1(ABCF), R_2(AD), R_3(DE), R_4(ABCG)\}$, $\{AB \rightarrow CG, A \rightarrow D, AB \rightarrow CF\}$). Figure 3.3, below, shows the hypergraph for R_1^+. □

Now, we prove in the following lemma that when we add R_j to $H_{R_i^+}$, the intersection of R_j with the attributes in $H_{R_i^+}$ is always included in some relation scheme already in the hypergraph.

Algorithm 1

Input: A BCNF Database scheme $(\mathbf{R}, F)$.

Output: Hypergraph of R_i^+, for each $R_i \in \mathbf{R}$.

(1) **for each** R_i **in R do begin**
(2) Let $H_{R_i^+} = \langle V = R_i, E = \{R_i\} \rangle$
(3) $Rest = \mathbf{R} - \{R_i\}$
(4) **while** there is an R_j in $Rest$ such that V contains a key of R_j **do begin**
(5) $E = E \bigcup \{R_j\}$; $V = V \bigcup R_j$;
(6) $Rest = Rest - \{R_j\}$;
(7) **end**
(8) **end**

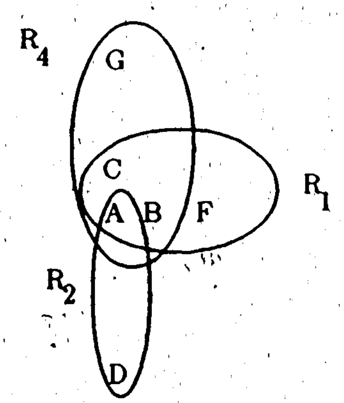


Figure 3.3 $H_{R_1^+}$ for Example 3.5

Lemma 3.1: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme, and let $R_i \in \mathbf{R}$. Let $H' = \langle V, E \rangle$ be a partial hypergraph of R_i^+ before an execution of the while-loop in Algorithm 1. Let $R_j \in \mathbf{R}$ be the edge chosen in line (4) in Algorithm 1. Let $CP_j = R_j \bigcap V$. Then there exists an edge E_l in H' such that $E_l \supseteq CP_j$.

Proof: Let $H'' = \langle V \bigcup R_j,\ E' = E \bigcup \{R_j\}\rangle$. Since H'' is connected and acyclic, by Theorem 3.1 the u.m.c. among CP_j exists in Bachman(E'). Let this be $\{W\}$. $\{W\}$ is a singleton since R_j contains CP_j. Now, let us consider any connected subset $\{R_{i_1}, \ldots, R_{i_q}\}$ of E such that $(\bigcup_{l=1}^{q} R_{i_l}) \supseteq CP_j$. By the u.m.c. among CP_j, there exists $R_{i_l} \in \{R_{i_1}, \ldots, R_{i_q}\}$ such that $R_{i_l} \supseteq CP_j$. □

In the rest of this chapter, we refer to a computation of $H_{R_i^+}$ by Algorithm 1 simply as a computation of R_i^+.

We now introduce Algorithm 2, shown below, an algorithm to chase a consistent state of an acyclic BCNF database scheme. To illustrate how tuples in T_r are extended using Algorithm 2, let us consider the following example.

Example 3.6: Let $(\{R_1(AB), R_2(BC), R_3(BCD)\}, \{B \rightarrow CD\})$ be an acyclic BCNF database scheme. The edges in $H_{R_1^+}$ are R_1, R_2, and R_3. Then in Algorithm 2, tuples originating from r_1 can be extended with tuples from r_3 or with those from r_2. Therefore if we want to compute the AC-total projection, we will show in Section 3.6 that the expression to compute this total projection is $\pi_{AC}(R_1 \bowtie R_2) \bigcup \pi_{AC}(R_1 \bowtie R_3)$. □

In the following section, we are going to prove that this algorithm obtains the total part of any tuple in $CHASE_F(T_r)$ in a fixed number of applications of fd-rules. We shall do this by proving that Step 2 of Algorithm 2 equates only ndv's. Unlike previous approaches [AtC][C1][IIK][MRW][S3], which assume independence, proving that Step 2 equates only ndv's is a difficult task in our case since we are not guaranteed that each attribute in a closure is "added" by a unique fd embedded in a unique relation scheme. In fact, this property makes our proof of boundedness much more difficult than in the independent case.

Our proof that Step 2 of Algorithm 2 equates only ndv's requires the proof of several facts about any computation of $CHASE_F(T_r)$. We illustrate only some of the

Algorithm 2.

Input: T_r for a consistent state **r** of an acyclic BCNF database scheme $(\mathbf{R}, F)$.
For each $R_i \in \mathbf{R}$, $H_{R_i^+} = <V_i, E_i>$ as computed by Algorithm 1.

Output: $CHASE_F(T_r)$.

```
Step 1:
for each R_i ∈ R do begin
  for each t in T_r originating from r_i do begin
      Rest = E_i - {R_i}
      while there exist R_j in Rest, t' in T_r from r_j, and
        K_j, a key of R_j, such that t'[K_j] = t[K_j] then do begin
          t[R_j] = t'[R_j]
          Rest = Rest - {R_j}
      end
  end
end

Step 2:
Let T_r' be the outcome from Step 1.
Obtain CHASE_F(T_r) from T_r'.
```

crucial facts in the following example.

Example 3.7: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF)\}, \{D \rightarrow B, D \rightarrow BC, D \rightarrow BCAF, F \rightarrow ABCD\})$ be an acyclic BCNF database scheme. Figure 3.4, below, shows $H_{\mathbf{R}}$.

Let us consider the following state of $(\mathbf{R}, F)$: $\mathbf{r} = (r_1 = \{<e, d>\}, r_2 = \{<d, b>\}, r_3 = \emptyset, r_4 = \{<d, b, c, a, f>\})$. Figure 3.5 below shows T_r; unique ndv's are denoted by "-."

Let T_r' be a tableau in a computation of $CHASE_F(T_r)$. Let t_1, t_2, and t_3 be the tuples in T_r' from r_1, r_2, and r_4 respectively. We want to show that when t_1 and t_2 are the same ndv on some attribute, then these tuples are also identical on a certain unique set of attributes that "can add" (to be defined in next section) that attribute to R_1^+ or R_2^+. Also we want to show that when t_1 is a constant on some attribute, then

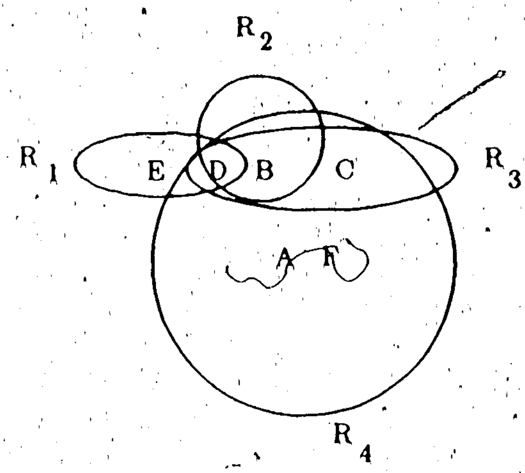


Figure 3.4 H_R for Example 3.7

t_1 also consists of constants on that same unique set of attributes. These and other crucial facts hold in general for the class of acyclic BCNF database schemes.

Let us apply (the fd-rule for) $D \to B$ to t_1 and t_2. Thus $t_1[B]$ gets equated to the constant b in $t_2[B]$. Observe that $t_1[D]$ is a constant, and in the following section we shall find out that $\{D\}$ is the unique set of attributes that "can add" B to R_1^+. Now let us apply $D \to ABCF$ to t_1 and t_2. Figure 3.5 shows T_r after the application of these two fd-rules.

Observe that $t_1[A] = t_2[A]$ and is an ndv. Notice that t_1 and t_2 are identical on DBC; in the following section we shall see that DBC is the maximal set of attributes that "can add" A to R_1^+.

Let us apply again $D \to ABCF$, but now we apply it to t_1 and t_3. Figure 3.5 shows T_r after this. Observe now that $t_1[A]$ is a constant and t_1 also consists of constants on DBC, which as mentioned before is the maximal set that "can add" A to R_1^+. □

A	B	C	D	E	F
-	-	-	d	e	-
-	b	-	d	-	-
a	b	c	d	-	f

T_r

A	B	C	D	E	F
δ_1	b	δ_2	d	e	δ_3
δ_1	b	δ_2	d	-	δ_3
a	b	c	d	-	f

T_r after $D \rightarrow B$ and $D \rightarrow ABCF$

A	B	C	D	E	F
a	b	c	d	e	f
a	b	c	d	-	f
a	b	c	d	-	f

T_r after $D \rightarrow ABCF$

Figure 3.5 Tableaux for Example 3.7

Most of the following section deals with proving that the observations and facts illustrated in the previous example hold for the kind of schemes considered in this chapter. These and other properties of acyclic BCNF schemes are the key to our proofs in the following section.

3.5. γ-acyclic BCNF Database Schemes are Bounded

In this section, we prove that acyclic BCNF database schemes are bounded. We do this by proving that if we chase the tableau T_r for a consistent state of an acyclic BCNF database scheme using Algorithm 2, then Step 2 of Algorithm 2 equates only ndv's. This implies that the total part of every tuple in $CHASE_F(T_r)$ is obtained in Step 1 of Algorithm 2 in a number of applications of fd-rule which depends only on the number of relation schemes in the database scheme.

3.5.1. Some Definitions

We give now most of the definitions needed in this section. Let $(\mathbf{R}, F)$ be a BCNF database scheme and let $R_i \in \mathbf{R}$. Let us consider a computation of R_i^+. Let $H_{R_i^+} = \langle V, E \rangle$ be the partial hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1. Let $R_j \in \mathbf{R}$ be such that it can be chosen at line (4) in Algorithm 1. We say that R_j *can be added* to $H_{R_i^+}$ (R_i^+). Assume R_j is chosen at line (4) in Algorithm 1. We say that R_j is a relation scheme that *is added* to $H_{R_i^+}$ (R_i^+). $R_j \cap V$ is called *a connection point* of R_j (*in* R_i^+) and it is denoted by CP_j. Now, let $K_{j_1}, \ldots, K_{j_q}$ be the keys of R_j in CP_j. Then, for $1 \le l \le q$, if $A \in R_j - K_{j_l}$, we say that K_{j_l} (CP_j, or R_j) *adds* A to R_i^+; if $A \notin V$, we say that K_{j_l} (CP_j, or R_j) *A-extends* R_i^+; we say that R_j *uses* K_{j_l} (CP_j) in R_i^+.

Let $K_{i_1}, \ldots, K_{i_m}$ be the keys of R_i. We regard K_{i_l}, for $1 \le l \le m$, as adding A to R_i^+, for any $A \in R_i$. Also, we make the conventions that $CP_i = \bigcup K_{i_l}$ and that R_i (K_{i_l}, or CP_i) A-extends R_i^+, for any $A \in R_i$. We say that R_j in R_i^+ *can A-extend* R_i^+ if there is a computation of R_i^+ in which R_j A-extends R_i^+. Notice that if $A \in R_i$, then R_i is the only relation scheme that A-extends R_i^+. Let $A, B \in R_i^+$. We say that AB is (or A and B are) *not split* in R_i^+ if for all computations of R_i^+, R_j A-extends R_i^+ if and only if R_j B-extends R_i^+. The following example illustrates some of these definitions.

Example 3.8: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF)\}, \{D \to B, D \to BC, D \to BCAF, F \to ABCD\})$ be a BCNF database scheme. $(\mathbf{R}, F)$ is an acyclic BCNF database scheme. Figure 3.4, above, shows $H_{\mathbf{R}}$. Let us consider $<R_1, R_4, R_3, R_2>$, a computation of R_1^+. In this computation of R_1^+, $CP_4 = \{D\}$. Since $\{D\}$ is the only key of R_4 in CP_4 and since $ABCF \subseteq ABCDF - \{D\}$ (i.e., $R_4 - K_4$), R_4 adds A, B, C, and F to R_1^+. In fact, R_4 A-, B-, C-, and F-extends R_1^+ since A, B, C, and F are not in V when R_4 is added to R_1^+. R_2 adds B to R_1^+, since $B \in DB - \{D\}$ (i.e., $B \in R_2 - K_2$). Another connection point of R_4 in R_1^+, which can C-extend R_1^+ is DB; this occurs in the following computation of R_1^+: $<R_1, R_2, R_4, R_3>$. Other computations of R_1^+ in which $CP_4 = DBC$ can A-extend R_1^+ are: $<R_1, R_3, R_4, R_2>$ and $<R_1, R_2, R_3, R_4>$. AF is not split in R_1^+, but B and C are split in R_1^+. It is worth noting that the connection points of R_4 in different computations of R_1^+ are totally ordered by set inclusion. □

3.5.2. Overview of Section

This section is organized as follows. In Section 3.5.3, we prove first that for every A in R_i^+ there is a unique maximal set of attributes (CP_{iA}) which A-extends R_i^+. Then using this fact we characterize when A and B are not split in R_i^+. With these results at hand, the rest of Section 3.5.3 is concerned with proving that if K_p, a key of some R_p in R_i^+, determines A and $K_p \not\subseteq CP_{iA}$, then there is a B in K_p such that either AB is not split in R_i^+ or $CP_{iB} \supseteq \{A\}$. This result is used in Section 3.5.6, in Lemma 3.5, to prove that when computing $CHASE_F(T_r)$ the A-component of a tuple in T_r originating from r_i can be equated only by keys in CP_{iA}. In Section 3.5.4, we study the structure of CP_{jA} in R_i^+. We prove that if $CP_{jA} \subseteq R_i^+$, then under some specific conditions CP_{jA}^{*} is not split in R_i^+. This result is used in Lemma 3.5 to prove that while computing $CHASE_F(T_r)$, if the A-component of a tuple t in T_r from r_j is not an ndv, then t consists of constants on the unique maximal set of attributes (CP_{jA}) which A-extends R_j^+. Section 3.5.5 contains some technical results about nonsplitness of attributes required in Section 3.5.6. In Section 3.5.6, as mentioned above, we prove some important facts about the computation of $CHASE_F(T_r)$ required in Sections 3.5.7 and 3.5.8 for proving that acyclic BCNF database schemes are embedded-complete and bounded respectively.

3.5.3. Some Properties of CP_j in R_i^+

We study first the connection points that can A-extend R_i^+ and prove that there exists a unique maximal connection point that can A-extend R_i^+.

Example 3.8 above shows that the connection points of R_4 in R_1^+ are ordered by set inclusion. The following proposition proves that in general this is the case, provided that the BCNF database scheme is acyclic.

Proposition 3.1: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_j in R_i^+ and let $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_q}$ be a sequence of connection points of R_j corresponding to q different computations of R_i^+. Then $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_q}$ are totally ordered by set inclusion.

Proof: By contradiction. Assume that $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_n}$, $1 < n \leq q$, is a sequence of connection points as defined above such that CP_{j_n} is the first connection point that violates the set inclusion total ordering. Observe this implies $R_j \neq R_i$; else there is only one connection point. Then, there exists CP_{j_l}, for some $1 \leq l < n$, such that CP_{j_l} and CP_{j_n} are not comparable. (Two sets are *not comparable* if neither one contains the other one.) Consequently there exist x_n in $CP_{j_n} - CP_{j_l}$, and x_l in $CP_{j_l} -$

CP_{j_n}. Notice that $R_j \supseteq \{x_n, x_l\}$.

Consider the paths from x_l to x_n in $H_{R_i^+}$. One of them is $<R_j>$ itself. Now, we are going to show that there is another path from x_l to x_n of length greater than or equal to 2; contradicting Theorem 3.2. Let us consider a computation of R_i^+ in which the connection point of R_j is CP_{j_l}. That is, let $H_l = <V_l, E_l>$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, be such that R_j can be added to H_l and $R_j \bigcap V_l = CP_{j_l}$; $x_n \notin V_l$; else $x_n \in CP_{j_l}$; therefore, there is no edge in E_l containing x_l and x_n. Let us consider a computation of R_i^+ in which the connection point of R_j is CP_{j_n}. That is, let $H_n = <V_n, E_n>$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, be such that R_j can be added to H_n and $R_j \bigcap V_n = CP_{j_n}$; $x_l \notin V_n$; else $x_l \in CP_{j_n}$; therefore, there is no edge in E_n containing x_l and x_n. Let $H' = <V_l \bigcup V_n, E_l \bigcup E_n>$. Since H' is a connected subset of **R**, x_l and x_n are connected in H'. Consider the paths in H' from x_l to x_n. Since there is no edge containing both of them, any of these paths is of length greater than or equal to 2. This contradicts Theorem 3.2.

Hence, our claim that $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_n}$ are totally ordered by set inclusion must hold. □

Corollary 3.1: Let (**R**, F) be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_j in R_i^+ be such that it can A-extend R_i^+. Let $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_q}$ be connection points of R_j corresponding to q different computations of R_i^+ in which R_j A-extends R_i^+. Then $CP_{j_1}, CP_{j_2}, \ldots, CP_{j_q}$ are totally ordered by set inclusion.

Proof: Since, by Proposition 3.1, all the connection points of R_j in R_i^+ are ordered by set inclusion, the same holds for a subset of connection points of R_j in R_i^+ which can A-extend R_i^+. □

Let R_j in R_i^+ be such that it can A-extend R_i^+. We shall denote the *maximal connection point* of R_j that can A-extend R_i^+ as CP_{jA_i}.

Proposition 3.2: Let (**R**, F) be an acyclic BCNF database scheme, and let $A \in R_i^+ - R_i$. Let R_j in R_i^+ be such that it can A-extend R_i^+. Then, there exists R_p in R_i^+, such that $R_p \supseteq CP_{jA_i}$, $A \notin R_p$, and $A \notin CP_{jA_i}$.

Proof: Let us consider a computation of R_i^+ such that CP_{jA_i} A-extends R_i^+. Let $H = \langle V, E \rangle$ be a partial hypergraph for R_i^+ before R_j A-extends R_i^+ using CP_{jA_i}. Notice $R_j \neq R_i$, since $A \notin R_i$. Then, by Lemma 3.1, there exists R_p in E such that $R_p \supseteq CP_{jA_i}$. By the definition of A-extend R_i^+, $A \notin CP_{jA_i}$. Clearly $A \notin R_p$. □

Let us now consider the connection points of two relation schemes in R_i^+ that can A-extend R_i^+. In the following proposition, we prove that if there is more than one R_j in R_i^+ which can A-extend R_i^+, then their maximal connection points that can A-extend R_i^+ are identical, provided that (**R**, F) is an acyclic BCNF database scheme.

Proposition 3.3: Let (**R**, F) be an acyclic BCNF database scheme and let $R_i \in$ **R**. Let $R_{j_1}, R_{j_2}, \ldots, R_{j_q}$ in R_i^+ be such that for all $1 \leq l \leq q$, R_{j_l} can A-extend R_i^+. Then $CP_{j_1A_i} = CP_{j_2A_i} = \ldots = CP_{j_qA_i}$.

Proof: By contradiction. Assume that $CP_{j_1A_i}, CP_{j_2A_i}, \ldots, CP_{j_kA_i}$, $1 < k \leq q$, is a sequence of connection points such that $CP_{j_kA_i}$ is the first connection point that violates the equality among them; notice that this implies that $A \notin R_i$. Then there are two cases to be considered depending on whether $CP_{j_kA_i}$ is comparable with any other connection point, say $CP_{j_1A_i}$.

Case 1: $CP_{j_kA_i}$ is not comparable with $CP_{j_1A_i}$. Then there exist $z_1 \in CP_{j_1A_i} - CP_{j_kA_i}$ and $z_k \in CP_{j_kA_i} - CP_{j_1A_i}$. Consider the paths from z_1 to A in $H_{R_i^+}$. One path is $\langle R_{j_1} \rangle$ itself. Now we prove there exists a path from z_1 to A in $H_{R_i^+}$ of length greater than or equal to 2. Let us consider a computation of R_i^+ in which the connection point of R_{j_k} is $CP_{j_kA_i}$. That is, let $H_k = < V_k, E_k >$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, be such that $A \notin V_k$, R_{j_k} can be added to H_k, and $R_{j_k} \cap V_k = CP_{j_kA_i}$; $x_k \in V_k$. Notice that since $A \notin V_k$, we have that, for all $1 \leq l \leq q$, $R_{j_l} \notin E_k$. Let us consider a computation of R_i^+ in which the connection point of R_{j_1} is $CP_{j_1A_i}$. That is, let $H_1 = < V_1, E_1 >$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, be such that $A \notin V_1$, R_{j_1} can be added to H_1, and $R_{j_1} \cap V_1 = CP_{j_1A_i}$; $x_1 \in V_1$. Notice that since $A \notin V_1$, we have that, for all $1 \leq l \leq q$, $R_{j_l} \notin E_1$. Let $H' = < V' = V_k \cup V_1, E_k \cup E_1 >$.

We claim $x_1 \notin R_{j_k}$. Assume otherwise. Then H' represents a partial hypergraph in a computation of R_i^+ and it is such that R_{j_k} can A-extend R_i^+. Since $x_1 \in R_{j_k}$ and x_1 is in H', $R_{j_k} \cap V'$ contains $\{x_1\}$ and $CP_{j_kA_i}$. This violates the maximality assumption of $CP_{j_kA_i}$.

Since H' is a connected subset of **R**, x_1 and x_k are connected in H'. Consider the paths in H' from x_1 to x_k; any of these paths neither contain $\{A\}$ nor any of its edges is R_{i_l}, for $1 \leq l \leq q$. Attach to any of these paths the edge R_{j_k}, and, since $x_1 \notin R_{j_k}$, we obtain a path of length greater than or equal to 2 from x_1 to A (going through x_k) in $H_{R_i^+}$. This contradicts Theorem 3.2. Thus this case is not possible.

Case 2: $CP_{j_kA_i}$ is comparable with $CP_{j_1A_i}$. Assume that $CP_{j_kA_i} \supset CP_{j_1A_i}$. (The case $CP_{j_1A_i} \supset CP_{j_kA_i}$ is analogous.) Since $A \notin R_i$, by Proposition 3.2 there exists R_p in R_i^+ such that $R_p \supseteq CP_{j_kA_i}$ and $A \notin R_p$. Since $CP_{j_kA_i} \supset CP_{j_1A_i}$, there exists x_k in $CP_{j_kA_i} - CP_{j_1A_i}$. Now $x_k \notin R_{j_1}$ since otherwise $R_{j_1} \cap R_p$, which is a connection point of R_{j_1} that can A-extend R_i^+, contains $\{x_k\}$ and $CP_{j_1A_i}$, and this violates the maximality assumption of $CP_{j_1A_i}$. Then, the following sequence is a cycle: $<R_{j_1}, A, R_{j_k}, x_k, R_p, x_{1pk}, R_{j_1}>$, where $x_{1pk} \in R_{j_1} \cap R_p \cap R_{j_k}$, and does exist since $R_p \supseteq CP_{j_kA_i} \supseteq CP_{j_1A_i}$ and none of them is empty. Hence, this case is not possible.

Therefore $CP_{j_1A_i} = CP_{j_2A_i} = \ldots = CP_{j_kA_i}$. □

We shall refer to the *(unique) maximal connection point* that can A-extend R_i^+ by CP_{iA}.

The following example illustrates that the above implication does not hold for cyclic BCNF database schemes.

Example 3.9: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCA), R_5(DA)\}, \{D \rightarrow A, D \rightarrow B, D \rightarrow BC, D \rightarrow BCA\})$ be a BCNF database scheme. Figure 3.6, below, shows $H_{\mathbf{R}}$. The maximal connection point of R_4 that can A-extend R_1^+ is DBC. And the maximal connection point of R_5 that can A-extend R_1^+ is $\{D\}$. They are different because $\mathbf{R}$ is cyclic. The cyclicity of $\mathbf{R}$ can be demonstrated by the cycle $<R_5, A, R_4, C, R_3, D, R_5>$. □

Before proceeding, we state the following facts; their proofs are omitted since they are trivial.

Fact 3.1: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Assume $A \in R_i^+ - R_i$. Let R_p in $\mathbf{R}$ be such that $A \in R_p$ and either $R_p \supseteq CP_{iA}$ or $K_p \subseteq CP_{iA}$, for some key K_p of R_p. Then R_p is in R_i^+ and it can A-extend R_i^+.

Fact 3.2: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_p in R_i^+ be such that it can A-extend R_i^+. Then $R_p \supseteq CP_{iA}A$.

Fact 3.3: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_p in R_i^+ be such that it can A-extend R_i^+. Assume $B \in R_p - CP_{iA}$. Then R_p (or CP_{iA}) can B-extend R_i^+ and, furthermore, for all computations of R_i^+, if R_p A-extends R_i^+,

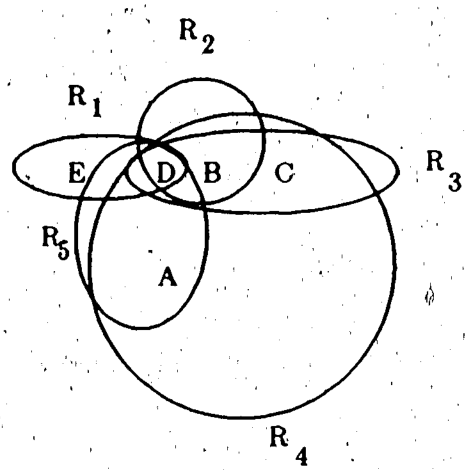


Figure 3.6 $H_{\mathbf{R}}$ for Example 3.9

then R_p B-extends R_i^+.

Fact 3.4: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Assume $A \in R_i^+ - R_i$, and $B \in R_i$. Let R_p be such that it can A-extend R_i^+ and $B \in R_p$. Then $CP_{iA} \supseteq \{B\}$.

With these results at hand, in the rest of this subsection we characterize a relationship between a particular pair of attributes A and B in R_i^+. We are interested in the relationship between A and B when B is in K_p, K_p is a key of R_p in R_i^+, that determines A, but $B \notin CP_{iA}$. We are interested in proving in Lemma 3.2 that for these B and A, it is the case that AB is not split in R_i^+ or $CP_{iB} \supseteq \{A\}$. The following example illustrates this situation.

Example 3.10: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF), R_5(AG), R_6(AGI), R_7(FAJ)\}, \{D \rightarrow B, D \rightarrow BC, D \rightarrow BCAF, F \rightarrow ABCD, A \rightarrow G, G \rightarrow AI, I \rightarrow GA, F \rightarrow AJ\})$ be a BCNF database scheme. $(\mathbf{R}, F)$ is an acyclic BCNF database scheme. Figure 3.7, below, shows $H_{\mathbf{R}}$. Let us examine the keys in R_1^+ that determine

A. $\{D\}$ is the only key that determines A in CP_{1A}, which is DBC. $\{F\}$ is in a relation scheme that can A-extend R_1^+, it is a key that determines A, and satisfies AF is not split in R_1^+. $\{G\}$, in R_5 and R_6, determines A, satisfies $CP_{1G} \supseteq \{A\}$, and it is in a relation scheme, R_6, that cannot A-extend R_1^+. For $\{I\}$, note that it is a key determining A that cannot even add A to R_1^+; it satisfies $CP_{1I} \supseteq \{A\}$. □

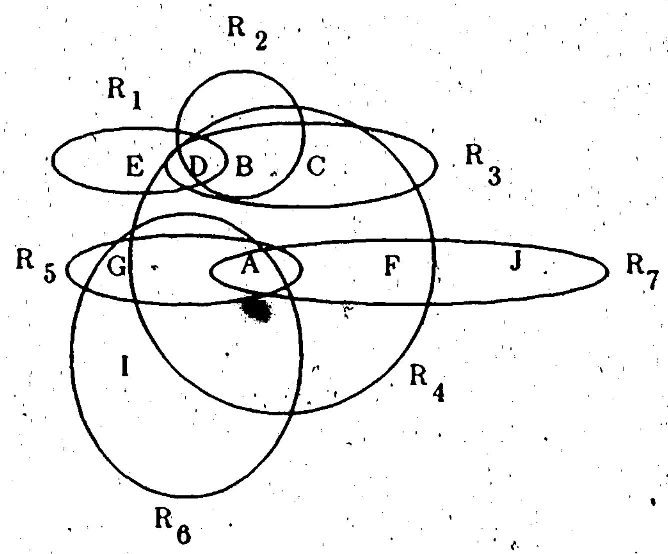


Figure 3.7 $H_{\mathbf{R}}$ for Example 3.10

We are going to tackle first the case when B is in a relation scheme which can A-extend R_i^+. After that, Proposition 3.6 shall consider the case when B is an element of a relation scheme containing A, but B is not in a relation scheme that can A-extend R_i^+. After that, Lemma 3.2 shall summarize these results proving our main claim in this subsection.

Proposition 3.4: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_j in R_i^+ be any relation scheme that can A-extend R_i^+. Let $B \in R_j - CP_{jA_i}$.

Then, either $CP_{jA_i} = CP_{jB_i}$ or $CP_{jB_i} \supseteq CP_{jA_i} \cup \{A\}$.

Proof: If $R_j = R_i$, then by convention CP_{jA_i} and CP_{jB_i} are equal. We assume $R_j \neq R_i$ in the rest of the proof.

Since $B \in R_j - CP_{jA_i}$, by Fact 3.3, CP_{jA_i} is a connection point of R_j that can B-extend R_i^+. However we do not know if it is the maximal connection point of R_j that can B-extend R_i^+. Let us compare CP_{jA_i} against CP_{jB_i}; they are ordered by set inclusion by Proposition 3.1; and they are in R_j. CP_{jA_i} does not include CP_{jB_i}; else the maximality assumption of CP_{jB_i} is violated. Then, either $CP_{jA_i} = CP_{jB_i}$ or $CP_{jA_i} \subset CP_{jB_i}$. If they are equal, then we finish with the proof. Hence assume $CP_{jA_i} \subset CP_{jB_i}$. But this implies $CP_{jB_i} \supseteq \{A\}$, else if $A \in R_j - CP_{jB_i}$, then CP_{jB_i} can A-extend R_i^+, violating the maximality assumption of CP_{jA_i}. □

In the following proposition, we prove that if $B \in R_l$, where R_l is a relation scheme in R_i^+ that can A-extend R_i^+, $B \notin CP_{lA_i}$, and $CP_{lA_i} = CP_{lB_i}$, then AB is not split in R_i^+.

Proposition 3.5: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_j in R_i^+ be such that it can A-extend R_i^+, $B \in R_j - CP_{jA_i}$, and $CP_{jA_i} = CP_{jB_i}$. Then AB is not split in R_i^+.

Proof: Assume R_q can A-extend R_i^+. We have to prove that for all computations of R_i^+, R_q A-extends R_i^+ if and only if R_q B-extends R_i^+. We prove the *only-if* part; the *if* part is symmetric. If $R_i = R_j$, then the proposition is trivially true. Hence assume $R_i \neq R_j$ in the rest of the proof. Observe that by Fact 3.3, R_j can B-extend R_i^+. Notice that $A \notin R_i$; else R_j cannot A-extend R_i^+. For the same reason, $B \notin R_i$.

We first prove B must be in R_q. Assume otherwise. Then $R_q \neq R_j$. By assumption, $CP_{jB_i} = CP_{jA_i}$. Since $A \notin R_i$, consider a computation of R_i^+ where $H' = < V,$

$E>$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, is such that $A \notin V$ and we can use R_q to A-extend R_i^+ and the connection point is the maximal one. $B \notin V$; else we can add R_j to H' obtaining $CP_{jA_i} \supseteq \{B\}$; this fact and $CP_{jB_i} = CP_{jA_i}$ imply $CP_{jB_i} \supseteq \{B\}$, which is a contradiction to Proposition 3.2. Now, add R_q to H'. After doing that we add R_j and we have that $CP_{jB_i} \supseteq \{A\}$, since R_j B-extends R_i^+. This fact along with $CP_{jA_i} = CP_{jB_i}$ imply $CP_{jA_i} \supseteq \{A\}$; a contradiction to Proposition 3.2. Hence B must be in R_q.

Next we want to show that if R_q A-extends R_i^+, then R_q B-extends R_i^+. If we show $B \in R_q - CP_{qA_i}$, then we finish with the proof, since by Fact 3.3, if R_q A-extends R_i^+, then R_q B-extends R_i^+. This is trivial. Since $B \in R_j - CP_{jA_i}$ and, by Proposition 3.3, $CP_{jA_i} = CP_{qA_i}$, and hence $B \notin CP_{qA_i}$. Then since $B \in R_q$, $B \in R_q - CP_{qA_i}$. □

Example 3.11: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF), R_5(DBCAFG)\}, \{D \to B, D \to BC, D \to BCAFG, F \to ABCD\})$ be a BCNF database scheme. $(\mathbf{R}, F)$ is an acyclic BCNF database scheme. Since $CP_{1A} = CP_{1F} = DBC$ and $F \in R_4 - CP_{1A}$, by Proposition 3.5 AF is not split in R_1^+. □

So far, we have considered only B in relation schemes which can A-extend R_i^+. In the following proposition, we consider attribute B in relation schemes which contain A, but B is not an element of any relation scheme that can A-extend R_i^+.

Proposition 3.6: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_l in R_i^+ be such that it contains AB. Assume B is not in any relation scheme that can A-extend R_i^+. Then $CP_{jB} \supseteq \{A\}$ and $B \notin R_i$.

Proof: First observe that the assumptions in the proposition imply that $A \neq B$. If R_l can B-extend R_i^+, then the proposition holds since A must be in V when R_l is added to the partial hypergraph to B-extend R_i^+ in Algorithm 1. Hence assume in the rest of the proof that R_l cannot B-extend R_i^+.

Since $B \in R_i^+$, let R_p in R_i^+ be such that R_p can B-extend R_i^+. By assumption, R_p cannot A-extend R_i^+. We prove by contradiction that $CP_{pB_i} \supseteq \{A\}$. Assume otherwise.

We first prove A cannot be in R_p. Assume $A \in R_p$. If $A \notin V$ when R_p is added to the partial hypergraph to B-extend R_i^+ in Algorithm 1, then R_p A-extends R_i^+; a contradiction since R_p is a relation scheme that cannot A-extend R_i^+. On the other hand, if $A \in V$, then $CP_{pB_i} \supseteq \{A\}$, which contradicts our assumption. Hence $A \notin R_p$.

Let us consider the paths from B to A in $H_{R_i^+}$. One of them is $<R_i>$. We want to prove there is another path of length greater than or equal to 2. If A and B cannot be extended at the same time, then there are two cases to be considered depending on whether R_i^+ is B- or A-extended first. Since there is no relation scheme that can A-extend R_i^+ and contains $\{B\}$, AB cannot be extended at the same time. Assume R_i^+ is B-extended before being A-extended. (The other case is symmetric with B and R_p in the roles of A and R_j respectively.) Let R_j in R_i^+ be such that it can A-extend R_i^+, and consider a computation of R_i^+. Let $H' = <V, E>$, the hypergraph for R_i^+ before an execution of the while-loop in Algorithm 1, be such that $B \in V$, $A \notin V$, and R_j can be added to H'. Since $B \notin R_j$, there is a path from A to B in $H'' = <V \cup R_j, E \cup \{R_j\}>$ of length greater than or equal to 2. This contradicts Theorem 3.2.

Therefore A must be in $CP_{pB_i} = CP_{iB}$. Observe that this implies $A \in R_p$. If $R_p = R_i$, then R_p can A-extend R_i^+ and contains $\{B\}$. This contradicts the assumption about R_p. Hence $R_p \neq R_i$. Also $B \notin R_i$; else R_p cannot B-extend R_i^+, contradicting our assumption about R_p. □

Finally, we are ready to prove the main claim in this subsection.

Lemma 3.2: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_p in R_i^+ be such that $A \in R_p - K_p$, for some key K_p of R_p. If $K_p \not\subseteq CP_{iA}$, then there is $B \in K_p - CP_{iA}$ such that either AB is not split in R_i^+ or $CP_{iB} \supseteq \{A\}$.

Proof: Assume $K_p \not\subseteq CP_{iA}$. Then there is a B such that $B \in K_p - CP_{iA}$. There are two cases to be considered depending on whether $B \in R_l$, for some R_l in R_i^+ which can A-extend R_i^+.

Case 1: $B \in R_l$ and R_l can A-extend R_i^+. By Proposition 3.4, $CP_{lA_1} = CP_{lB_1}$ or $CP_{lB_1} \supseteq \{A\}$. If $CP_{lA_1} = CP_{lB_1}$, then by Proposition 3.5, AB is not split in R_i^+; else $CP_{lB_1} \supseteq \{A\}$, which, by Proposition 3.3 and definition of CP_{iB}, implies $CP_{iB} \supseteq \{A\}$.

Case 2: B is not in any relation scheme that can A-extend R_i^+. In this case, R_p contains AB. Then by Proposition 3.6, $CP_{iB} \supseteq \{A\}$. □

3.5.4. Some Properties of CP_{jA} in R_i^+

In this subsection, we study the maximal connection point that A-extends R_j^+ (CP_{jA}) when $CP_{jA} \subseteq R_i^+$. First, we take a look at the case when CP_{iA} and CP_{jA} have a key in common. We prove that if this is the case, then $CP_{iA} = CP_{jA}$, provided A is neither in R_i nor in R_j. After that, we prove that CP_{jA} is not split in R_i^+ under some specific conditions determined by the induction part of the proof for Part B of Lemma 3.5 in Section 3.5.6. As mentioned earlier in the Overview of this section, these results are used to prove in Part B of Lemma 3.5, that in any computation of $CHASE_F(T_r)$, if the A-component of a tuple originating from r_j is a constant, then the tuple must be constants on CP_{jA}.

We need to prove first the following fact.

Proposition 3.7: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme. Let $R_i \in \mathbf{R}$ be such that $A \in R_i^+ - R_i$. Assume $B \in CP_{iA}$. If R_q is a relation scheme in $\mathbf{R}$ that contains AB, then $CP_{iA} \subseteq R_q$.

Proof: Since $A \notin R_i$, let $R_p \in \mathbf{R}$ be such that it can A-extend R_i^+. By Fact 3.2,

$R_p \supseteq CP_{iA}A$. Notice that this implies AB is in R_p.

Assume $R_q \in \mathbf{R}$ contains AB. We claim $R_q \supseteq CP_{iA}$. Assume otherwise. Hence $R_p \neq R_q$. We know B is in CP_{iA} and R_q. Since $A \notin R_i$, Proposition 3.2 implies that there exists R_t in R_i^+ such that $R_t \supseteq CP_{iA}$ and $A \notin R_t$. Let $x_i \in CP_{iA} - R_q$. Then $x_i \in R_p$ since $CP_{iA} \subseteq R_p$; by similar reason, $x_i \in R_t$. Then, the following cycle exists: $<R_t, x_i, R_p, A, R_q, B, R_t>$. A contradiction to the fact that $\mathbf{R}$ is acyclic. Hence $CP_{iA} \subseteq R_q$. □

The following proposition proves a more general claim than the first one we want to prove in this subsection.

Proposition 3.8: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme. Let R_i and R_j be elements of $\mathbf{R}$. Assume AB is in both R_i^+ and R_j^+, but neither A nor B is in R_iR_j. Let R_p be such that $R_p \supseteq CP_{iA}CP_{jB}AB$ and $AB \subseteq R_p - CP_{iA}CP_{jB}$. Then, if $CP_{iA} \cap CP_{jB} \neq \emptyset$, then $CP_{iA} = CP_{jB}$.

Proof: Assume $CP_{iA} \cap CP_{jB} \neq \emptyset$. By assumption about R_p and Fact 3.1, R_p is in R_j^+ and R_i^+ and it can A-extend R_i^+ and B-extend R_j^+.

We claim $CP_{iA} = CP_{jB}$, if $R_i = R_j$. Assume $R_i = R_j$. Then we have to prove $CP_{iA} = CP_{iB}$. Since R_p can A-extend R_i^+ and $B \in R_p - CP_{iA}$, by Proposition 3.4, either $CP_{iA} = CP_{iB}$ or $CP_{iB} \supseteq CP_{iA}A$. But we know $A \notin CP_{iB}$. Thus $CP_{iA} = CP_{iB}$. Hence for the rest of the proof we assume $R_i \neq R_j$.

If CP_{iA} and CP_{jB} are not comparable, then the u.m.c. among $CP_{iA}CP_{jB}$ is a singleton, since R_p contains both. But $\{CP_{iA}, CP_{jB}\}$ violates the u.m.c. among them. Hence CP_{iA} and CP_{jB} are comparable.

Now we prove one cannot be a superset of the other. Assume $CP_{iA} \supset CP_{jB}$. Let $x \in CP_{iA} - CP_{jB}$. Since $A \notin R_i$, by Proposition 3.2, there exists R_t in R_i^+ such that $R_t \supseteq CP_{iA}$ and $A \notin R_t$.

We claim $B \notin R_t$. Assume otherwise. Then let us consider a computation of R_i^+ such that CP_{iA} A-extends R_i^+. Let $H = <V, E>$ be a partial hypergraph for R_i^+ such that R_p A-extends R_i^+ using CP_{iA} and such that R_t is in E. By Proposition 3.2, it is always possible. Since $B \in R_p$, $CP_{pA_i} \supsetneq \{B\}$. Then by Proposition 3.3 and definition of CP_{iA}, $CP_{iA} \supsetneq \{B\}$, which is a contradiction to the assumption about B. Hence our claim that $B \notin R_t$ is proven.

Since CP_{jB} contains at least a key and R_p can B-extend R_j^+, let this be K_p a key of R_p. Since $K_p \rightarrow x \in F^+$ is a nontrivial fd embedded in R_t, K_p is a key of R_t and R_t must be in R_j^+.

We prove this implies CP_{iA} is a connection point in R_j^+ that can B-extend R_j^+. Since $B \notin R_j$, consider a computation of R_j^+ where $H' = <V, E>$, the hypergraph for R_j^+ before an execution of the while-loop in Algorithm 1, is such that $B \notin V$, that we can use R_p to B-extend R_j^+ and the connection point is CP_{jB}. Observe $A \notin V$, since $A \notin CP_{jB}$ and $A \in R_p$. Since K_p is in CP_{jB} and is a key of R_t, we can add R_t to H'. Suppose we add R_t to H' giving H''; notice B is not in H'' since $B \notin R_t$. Now we add R_p, which B-extends R_j^+. Hence CP_{iA} is a connection point in R_j^+ that can B-extend R_j^+. This violates the maximality assumption of CP_{jB}.

By a similar argument, the other proper inclusion does not hold. Therefore $CP_{iA} = CP_{jB}$. □

The following example illustrates the next corollary which proves the first of our claims in this subsection.

***Example* 3.12: Let $(\mathbf{R}, F) = (\{R_1(CD), R_2(IB), R_3(DEAFH), R_4(BDE)\}, \{B \rightarrow DE, D \rightarrow BEAFH\})$ be a BCNF database scheme. $(\mathbf{R}, F)$ is an acyclic BCNF database scheme. Its hypergraph is shown in Figure 3.8 below. Let us consider R_1^+ and R_2^+. A is in both closures, but A is neither in R_1 nor in R_2. Also $\{D\}$ is a key of R_3 and it is**

in both CP_{1A} and CP_{2A}. Then it must be the case that $CP_{1A} = CP_{2A}$. □

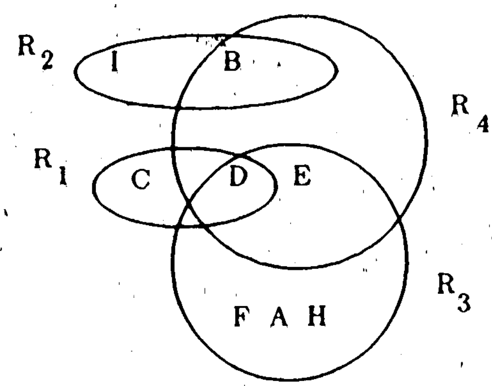


Figure 3.8 $H_{\mathbf{R}}$ for Example 3.12

Corollary 3.2: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme. Let R_i and R_j be elements of $\mathbf{R}$, and assume A is in both R_i^+ and R_j^+, but A is neither in R_i nor in R_j. Let K_p be a key of $R_p \in \mathbf{R}$ such that $A \in R_p - K_p$, and K_p is in both CP_{iA} and CP_{jA}. Then $CP_{iA} = CP_{jA}$.

Proof: By Facts 3.1 and 3.2, $R_p \supseteq CP_{iA}CP_{jA}$. Since A is neither in R_i nor in R_j, by Propositions 3.2 and 3.3 and definition of CP_{iA} and CP_{jA}, $A \notin CP_{iA}CP_{jA}$. Then since $A \in R_p$, $A \in R_p - CP_{iA}CP_{jA}$. Observe $CP_{iA} \cap CP_{jA} \neq \varnothing$, since they share K_p. Then, the Corollary follows from Proposition 3.8 with $A = B$. □

We want to prove that under certain conditions CP_{jA} is not split in R_i^+; that is, under certain conditions it holds that for every pair of attributes B and C in CP_{jA}, BC is not split in R_i^+. We first give an example to illustrate the first of the propositions proving this claim.

Example 3.13: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF)\}, \{D \rightarrow B,$

$D \rightarrow BC$, $D \rightarrow BCAF$, $F \rightarrow ABCD\}$) be a BCNF database scheme. (**R**, F) is an acyclic BCNF database scheme. Its hypergraph is shown in Figure 3.4 in Section 3.4. Notice that $C \in R_3 - CP_{1B}$. It is observed that when we C-extend R_1^+ using any $R_q \in \mathbf{R}$, R_q can B-extend R_1^+. But the converse of this observation does not hold. □

Proposition 3.9: Let (**R**, F) be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let R_j in R_i^+ be such that it can A-extend R_i^+, and $B \in R_j - CP_{jA_i}$. If R_q is in R_i^+ and it can B-extend R_i^+, then R_q can A-extend R_i^+.

Proof: First observe that R_j can A- and B-extend R_i^+. Assume there is R_q in R_i^+ such that it can B-extend R_i^+; $q \neq j$, else we finish with the proof. Notice that $B \notin R_i$; else $j = i$ and R_q cannot B-extend R_i^+. For the same reason, $A \notin R_i$. We want to show R_q can A-extend R_i^+.

We first prove A must be in R_q. Assume otherwise. By Proposition 3.4, either $CP_{jB_i} = CP_{jA_i}$ or $CP_{jB_i} \supseteq CP_{jA_i} \cup \{A\}$. But, by Proposition 3.3, $CP_{qB_i} = CP_{jB_i}$, and $A \notin CP_{qB_i}$ (since $A \notin R_q$), imply $CP_{jB_i} = CP_{jA_i}$. Then by Proposition 3.5, AB is not split in R_i^+. Hence R_q can A-extend R_i^+, and therefore $A \in R_q$. A contradiction to our assumption that $A \notin R_q$. Hence A must be in R_q.

Next we want to show that R_q can A-extend R_i^+. If we prove CP_{iA} is in R_q, then we finish since this fact and $A \notin R_i$ imply, by Fact 3.1, that R_q can A-extend R_i^+. But this is obvious. By Proposition 3.3, $CP_{qB_i} = CP_{jB_i}$, and, by Proposition 3.4, $CP_{jB_i} \supseteq CP_{jA_i}$. Thus $CP_{jA_i} \subseteq R_q$. □

Lemma 3.3: Let (**R**, F) be an acyclic BCNF database scheme and let R_i, $R_j \in \mathbf{R}$. Assume $A \in R_i^+ - R_i$ and $A \in R_j^+ - R_j$. Let $R_p \in \mathbf{R}$ be such that it can A-extend both R_j^+ and R_i^+ and assume $CP_{iA} \cap CP_{jA} = \varnothing$. Then CP_{jA} is not split in R_i^+.

Proof: By Fact 3.2, $R_p \supseteq CP_{jA}$. Hence $CP_{jA} \subseteq R_i^+$. If CP_{jA} is a singleton the proposition is trivially true. In the rest of the proof we assume $|CP_{jA}| \geq 2$. Hence let B

and C be two attributes in CP_{jA}. We have to prove that R_l in R_i^+ B-extends R_i^+ if and only if R_l C-extends R_i^+. We prove the *only-if* part, the other part is symmetric.

Let R_l be a relation scheme in R_i^+ such that it can B-extend R_i^+. By Fact 3.2, CP_{iA} is in R_p. Since $CP_{iA} \bigcap CP_{jA} = \emptyset$ and $CP_{jA} \subseteq R_p$, $CP_{jA} \subseteq R_p - CP_{iA}$. Hence $B \in R_p - CP_{iA}$. Then by Fact 3.3, R_p can B-extend R_i^+. Since R_p can A-extend R_i^+ and $B \in R_p - CP_{iA}$, by Proposition 3.9, R_l can A-extend R_i^+. Hence by Fact 3.2, $CP_{iA}A$ is in R_l. Now since AB is in R_l, by Proposition 3.7, $CP_{jA} \subseteq R_l$. Thus $C \in R_l$. Observe that C and B are in $R_l - CP_{iA}$, since $CP_{jA} \subseteq R_l - CP_{iA}$. By Proposition 3.4, either $CP_{lB_i} = CP_{lA_i}$ or $CP_{lB_i} \supseteq CP_{lA_i}A$.

If we prove $C \notin CP_{lB_i}$, then $C \in R_l - CP_{lB_i}$, and hence by Fact 3.3, if R_l B-extends R_i^+, then R_l C-extends R_i^+. If $CP_{lB_i} = CP_{lA_i}$, then $C \in R_l - CP_{lB_i}$, and the proof is complete. Hence assume $CP_{lB_i} \supseteq CP_{lA_i}A$ in the rest of the proof.

Assume $C \in CP_{lB_i}$. Observe $B \notin R_i$, else by Fact 3.4, $CP_{pA_i} \supseteq \{B\}$, which contradicts $B \in R_p - CP_{iA}$. By Proposition 3.2, there is R_m in R_i^+ such that $R_m \supseteq CP_{lB_i}$ and $B \notin R_m$. Since $AC \subseteq CP_{lB_i}$, $AC \subseteq R_m$. Since $R_m \supseteq AC$, by Proposition 3.7, $R_m \supseteq CP_{jA}$. Hence $B \in R_m$; a contradiction to the fact that $B \notin R_m$. Hence $C \notin CP_{lB_i}$ and the proof is complete. □

Lemma 3.4: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i, R_j \in \mathbf{R}$. Assume $A \in R_i^+$ and $A \in R_j^+ - R_j$. Let $R_p \in \mathbf{R}$ be in both R_i^+ and R_j^+ such that it can A-extend R_j^+ but cannot A-extend R_i^+. Then CP_{jA} is not split in R_i^+.

Proof: By Fact 3.2, $R_p \supseteq CP_{jA}A$. Hence $CP_{jA} \subseteq R_i^+$. If CP_{jA} is a singleton the proposition is trivially true. In the rest of the proof we assume $|CP_{jA}| \geq 2$. Hence let B and C be two attributes in CP_{jA}. There are two cases to be considered depending on whether $A \in R_i$. We examine first the case $A \notin R_i$.

Case 1: $A \notin R_i$. There are two subcases to be examined depending on whether B is in some relation scheme that can A-extend R_i^+.

Case 1.a: $B \in R_l$, for some R_l that can A-extend R_i^+. Since R_l can A-extend R_i^+, by Fact 3.2, $CP_{iA}A \subseteq R_l$. Since A and B are elements of R_l, by Proposition 3.7, $CP_{jA} \subseteq R_l$. Then, by Fact 3.2, R_l can A-extend R_j^+. We claim $CP_{iA} \cap CP_{jA} = \emptyset$. Assume otherwise. Since $R_p \supseteq CP_{jA}$ and $CP_{jA} \cap CP_{iA}$ is nonempty, R_p contains an element of CP_{iA} and A. Hence by Proposition 3.7, $CP_{iA} \subseteq R_p$. Now, since R_p contains both A and CP_{iA} and $A \notin R_i$, by Fact 3.1, R_p can A-extend R_i^+. A contradiction to our assumption about R_p. Therefore $CP_{iA} \cap CP_{jA} = \emptyset$. Since $CP_{iA} \cap CP_{jA} = \emptyset$ and R_l can A-extend both R_i^+ and R_j^+, by Lemma 3.3 (with R_l in the role of R_p), CP_{jA} is not split in R_i^+.

Case 1.b: B is not in any relation scheme that can A-extend R_i^+. Since $R_p \supseteq AB$, Proposition 3.6 implies $CP_{iB} \supseteq \{A\}$ and $B \notin R_i$. Let R_t in R_i^+ be an arbitrary relation scheme that can B-extend R_i^+. We want to prove that if R_t B-extends R_i^+, then R_t C-extends R_i^+.

Since $CP_{iB} \supseteq \{A\}$ and, by Fact 3.2, $R_t \supseteq CP_{iB}B$, $AB \subseteq R_t$. Since $R_t \supseteq AB$ and $B \in CP_{jA}$, by Proposition 3.7, $CP_{jA} \subseteq R_t$. Therefore, $C \in R_t$.

We claim $C \notin CP_{tB_i}$. Assume otherwise. Since $B \notin R_i$, by Proposition 3.2, there is R_m in R_i^+ such that R_m includes CP_{tB_i} and $B \notin R_m$. $AC \subseteq R_m$, since $AC \subseteq CP_{tB_i}$. Since AC is in R_m, by Proposition 3.7, $CP_{jA} \subseteq R_m$. Thus $B \in R_m$. A contradiction to the fact that $B \notin R_m$. Hence $C \in R_t - CP_{tB_i}$.

By Fact 3.3, R_t C-extends R_i^+ in any computation of R_i^+ in which B-extends R_i^+. By a similar argument we can prove that if a relation scheme C-extends R_i^+, then it B-extends R_i^+.

Case 2: $A \in R_i$. Notice that this implies R_i is the only relation that can A-extend R_i^+. If there is B in CP_{jA} such that $B \in R_i$, then, by Proposition 3.7, $CP_{jA} \subseteq R_i$. Hence CP_{jA} is not split in R_i^+.

On the other hand, if $CP_{jA} \cap R_i = \emptyset$, then for all B in CP_{jA} it holds that $B \notin R_i$. That is, B is not in a relation scheme that can A-extend R_i^+. Thus the proof for Case 1.b above applies here to prove CP_{jA} is not split in R_i^+. □

3.5.5. More Facts about Nonsplitness

In this subsection, we prove more technical results about nonsplitness of attributes required in the proof of Lemma 3.5 in the next subsection.

Proposition 3.10: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and let $R_i \in \mathbf{R}$. Let AB in R_i^+ be such that AB is not split in R_i^+. Then $CP_{iA} = CP_{iB}$.

Proof: Assume that A and B are in R_i^+ and they are not split in R_i^+. Let R_j in R_i^+ be such that it can A-extend R_i^+. Since AB is not split in R_i^+, R_j can B-extend R_i^+; by Fact 3.2, $R_j \supseteq AB$. We want to prove that $CP_{jA_i} = CP_{jB_i}$. Notice that if $R_j = R_i$, then the proposition trivially holds. Hence in the rest of the proof we assume $R_j \neq R_i$.

$A \notin R_i$, else R_j cannot A-extend R_i^+. The same holds for B. Since for all computations of R_i^+, R_j A-extends R_i^+ if and only if R_j B-extends R_i^+, it must be the case that $B \in R_j - CP_{jA_i}$ and $A \in R_j - CP_{jB_i}$. Hence by Proposition 3.4, either $CP_{jB_i} \supseteq CP_{jA_i}A$ or $CP_{jB_i} = CP_{jA_i}$, and either $CP_{jA_i} \supseteq CP_{jB_i}B$ or $CP_{jB_i} = CP_{jA_i}$. It is easy to see that $CP_{jA_i} = CP_{jB_i}$. By Proposition 3.3 and the definition of CP_{iA} and CP_{iB}, the proposition follows. □

The implication in the previous proposition does not hold in the other direction as is illustrated in the following example.

Example 3.14: Let $(\mathbf{R}, F) = (\{R_1(ED), R_2(DB), R_3(DBC), R_4(DBCAF), R_5(DBCH)\}, \{D \to BCH, D \to B, D \to BC, D \to BCAF, F \to ABCD\})$ be a BCNF database scheme. $(\mathbf{R}, F)$ is an acyclic BCNF database scheme. A and H are split in R_1^+, although $CP_{1H} = CP_{1A}$, which is equal to DBC. □

The following proposition establishes the fact that if the maximal connection points that A-extend two relation schemes are identical and there is another attribute B such that AB is not split in one of the closures, then the same is true for AB in the other closure.

Proposition 3.11: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme. Let R_i and R_j be elements of $\mathbf{R}$ such that A is in both R_i^+ and R_j^+, but A is neither in R_i nor in R_j. Let K_p be a key of $R_p \in \mathbf{R}$ such that $A \in R_p - K_p$, and K_p is in both CP_{iA} and CP_{jA}. Then AB is not split in R_i^+ if and only if AB is not split in R_j^+.

Proof: We prove the *only-if* part; the *if* part is symmetric. Assume AB is in R_i^+ such that it is not split in R_i^+. From Proposition 3.10, $CP_{iA} = CP_{iB}$. Also, from Corollary 3.2, $CP_{iA} = CP_{jA}$. Hence $CP_{jA} = CP_{iB}$.

By Fact 3.1, R_p can A-extend R_i^+ and by Fact 3.2, $R_p \supseteq CP_{iA}A$. Since AB is not split in R_i^+, R_p can B-extend R_i^+ and $B \in R_p - CP_{iA}$. Since $CP_{iA} = CP_{jA}$, $B \in R_p - CP_{jA}$, and by Fact 3.3, R_p can B-extend R_j^+. By Proposition 3.4, $CP_{jB} = CP_{jA}$ or $CP_{jB} \supseteq CP_{jA}A$. If $CP_{jA} = CP_{jB}$, then we finish with the proof since by Proposition 3.5, AB is not split in R_j^+.

We prove $CP_{jB} \supseteq CP_{jA}A$ is impossible. Assume $CP_{jB} \supseteq CP_{jA}A$. We have that $B \notin R_i$; else R_p cannot B-extend R_i^+. Also $B \notin R_j$; else R_p cannot B-extend R_j^+. Since $CP_{jA} = CP_{iB}$ and $CP_{jB} \supseteq CP_{jA}$, $CP_{jB} \supseteq CP_{iB}$. Also we have that $R_p \supseteq CP_{iB}CP_{jB}$, and CP_{iB} and CP_{jB} share a key, since K_p is in CP_{iB}. Then Corollary 3.2 implies $CP_{iB} = CP_{jB}$. But since $CP_{iB} = CP_{jA}$, $CP_{jB} = CP_{jA}$. This implies $CP_{jA} \supseteq$

$CP_{jA}A$, which is a contradiction to Proposition 3.2. □

Proposition 3.12: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme and assume R_1 and R_2 are in $\mathbf{R}$. Let R_p be in both R_1^+ and R_2^+ and such that it can A-extend R_1^+ using a key K_p; $A \notin R_1$. Assume $B \in K_p - CP_{2A}$ and $C \in R_2^+ - CP_{2B}$ is such that AC is not split in R_1^+. Then BC is not split in R_2^+.

Proof: There are two cases to be considered depending on whether B is an element of some relation scheme in R_2^+ that can A-extend R_2^+. Observe that B is in CP_{1A} since R_p can A-extend R_1^+ using K_p.

Case 1: $B \in R_t$, for some R_t that can A-extend R_2^+. By Fact 3.2, $R_t \supseteq CP_{2A}$. Since $B \notin CP_{2A}$, $B \in R_t - CP_{2A}$. Hence, by Fact 3.3, R_t can B-extend R_2^+.

We first prove $C \in R_t$. Since B and $A \in R_t$, by Proposition 3.7, $CP_{1A} \subseteq R_t$. Since both CP_{1A} and A are in R_t and $A \notin R_1$, by Fact 3.1, R_t is in R_1^+ and it can A-extend R_1^+. Since AC is not split in R_1^+, R_t can C-extend R_1^+. Hence $C \in R_t$.

Since $C \in R_t - CP_{2B}$, by Fact 3.3, if R_t B-extends R_2^+, then R_t C-extends R_2^+. If $R_t = R_2$, then we finish with the proof since R_2 is the only relation scheme that B- and C-extends R_2^+. In the rest of the proof we assume $R_t \neq R_2$.

We prove that neither A nor B nor C are elements of R_2. If $A \in R_2$, then R_t cannot A-extend R_2^+. Similarly for B and C.

Since $C \in R_t - CP_{2B}$, by Proposition 3.4, either $CP_{tC_2} = CP_{tB_2}$ or $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$. If they are equal, then, by Proposition 3.5, BC is not split in R_2^+ and we finish with the proof.

We prove that $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$ is impossible. Assume otherwise. By Proposition 3.2, there exists R_l in R_2^+ such that $R_l \supseteq CP_{tC_2}$ and $C \notin R_l$. We now prove $A \in R_l$ by proving $A \in CP_{tC_2}$. Assume $A \notin CP_{tC_2}$. Then $A \in R_t - CP_{tC_2}$ and Proposition 3.4 implies $CP_{tA_2} \supseteq CP_{tC_2}$. Since $B \in R_t - CP_{2A}$, Proposition 3.4 implies $CP_{tB_2} \supseteq CP_{tA_2}$. Hence $CP_{tA_2} \supseteq CP_{tC_2} \supset CP_{tB_2} \supseteq CP_{tA_2}$; a contradiction. Thus $A \in CP_{tC_2}$, and hence $A \in R_l$.

Notice that $B \in CP_{tC_2}$, and hence $B \in R_l$. Also $A \in R_l$. Since $B \in CP_{1A}$ and $A \notin R_1$, by Proposition 3.7, $CP_{1A} \subseteq R_l$. Since R_l also contains $\{A\}$ and $A \notin R_1$, by Fact 3.1, R_l is in R_1^+ and it can A-extend R_1^+, but cannot C-extend R_1^+, because $C \notin R_l$. This contradicts our assumption that A and C are not split in R_1^+. Hence $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$ is impossible.

Case 2: B is not an element of any relation scheme that can A-extend R_2^+. By assumption about R_p and Fact 3.2, $CP_{1A}A \subseteq R_p$. Let us consider R_t in R_2^+ which can B-extend R_2^+. Since $R_p \supseteq AB$ and by assumption of B, Proposition 3.6 implies that $CP_{tB_2} \supseteq \{A\}$ and $B \notin R_2$; therefore $A \in R_t$. Since A and B are in R_t, by Proposition 3.7, $CP_{1A} \subseteq R_t$. Since $A \notin R_1$, by Fact 3.1, R_t is in R_1^+ and it can A-extend R_1^+. Hence $C \in R_t$, since A and C are not split in R_1^+. Also $C \notin R_2$; otherwise Fact 3.4 implies $C \in CP_{2B}$, which contradicts our assumption about C.

Since $C \in R_t - CP_{2B}$, by Proposition 3.4, either $CP_{tC_2} = CP_{tB_2}$ or $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$. If they are equal, then, by Proposition 3.5, BC is not split in R_2^+ and we finish with the proof.

We claim $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$ is impossible. By Proposition 3.2, there exists R_l in R_2^+ such that $R_l \supseteq CP_{tC_2}$ and $C \notin R_l$. Notice that from above $A \in CP_{tB_2}$ and hence $AB \subseteq CP_{tC_2}$. Therefore $AB \subseteq R_l$. Since $B \in CP_{1A}$ and $A \notin R_1$, by Proposition 3.7, $CP_{1A} \subseteq R_l$. Since R_l also contains $\{A\}$ and $A \notin R_1$, by Fact 3.1, R_l is in R_1^+ and it can A-extend R_1^+, but cannot C-extend R_1^+, because $C \notin R_l$. This contradicts our assumption that A and C are not split in R_1^+. Hence $CP_{tC_2} \supseteq CP_{tB_2} \cup \{B\}$ is impossible. □

3.5.6. Some Properties of $CHASE_F(T_r)$

In this subsection, we prove the facts about any computation of $CHASE_F(T_r)$ which are required in Sections 3.5.7 and 3.5.8 to prove that acyclic BCNF database schemes are embedded-complete and bounded respectively.

Before deriving the proofs in this subsection, we need the following definitions. Let t_1 be a tuple in T_r which originates from r_i; $R_i \in \mathbf{R}$, and assume $A \in R_i^+$. We say that K_j (CP_{jA_i}, or CP_{iA}) *A-extends* t_1, if K_j (CP_{jA_i}, or CP_{iA}, respectively) A-extends R_i^+. We shall denote the maximal connection point that A-extends t_1 by CP_{t_1A}. We also say that AB *is not split* (or A and B *are not split*) in t_1, if AB is not split (A and B are not split) in R_i^+.

We remind the reader that for BCNF database schemes the fd's considered are the ones embodied in the relation schemes in the database scheme. Therefore the fd used in any fd-rule is of the form $K_p \rightarrow R_p - K_p$, where K_p is a nontrivial key of R_p. In the following we denote a sequence of fd-rules by $\tau_1 \ldots \tau_k$, and $\tau_1 \ldots \tau_k(T)$ denotes $\tau_k(\ldots(\tau_1(T))\ldots)$, where $\tau_i(T)$ is the tableau obtained from applying the fd-rule τ_i to the tableau T.

Lemma 3.5: Let $(\mathbf{R}, F)$ be an acyclic BCNF database scheme. Let $\mathbf{r}$ be a state of $(\mathbf{R}, F)$, and let T_r be the tableau of $\mathbf{r}$. Suppose $T_r' = \tau_1 \ldots \tau_k(T_r)$ is nonempty, t_1 and t_2 are in T_r', and $A \in \mathbf{U}$.

A: If $t_1[A] = t_2[A]$ and is an ndv, then

(1) $CP_{t_1A} = CP_{t_2A}$;

(2) $t_1[CP_{t_1A}] = t_2[CP_{t_1A}]$;

(3) if AB is not split in t_1, then $t_1[B] = t_2[B]$.

B: If $t_1[A]$ is a constant, then

(1) $t_1[CP_{t_1A}]$ are constants;

(2) if AB is not split in t_1, then $t_1[B]$ is a constant.

Proof: By induction on k, the number of applications of fd-rule that produce T_r' from T_r.

Basis: $k=0$. Hence $T_r' = T_r$. Part **A**: Trivially true since all ndv's are distinct in T_r. For Part **B**, let $t_1 \in T_r$ be a tuple originating from r_i, $R_i \in \mathbf{R}$. If $t_1[A]$ is a constant, then $A \in R_i$. Since $A \in R_i$, $CP_{iA} \subseteq R_i$ and $t_1[CP_{iA}]$ are constants since $t_1[R_i]$ are constants. Assume B is such that AB is not split in t_1. Then $B \in R_i$, and therefore $t_1[B]$ is a constant.

Induction: $k>0$. Assume T_r'' is nonempty and is obtained from T_r by $k-1 \geq 0$ fd-rule applications, and let us assume that T_r' is nonempty and is obtained from T_r'' by applying the fd-rule τ_k: $K_p \rightarrow R_p - K_p$, $R_p \in \mathbf{R}$, to equate v_1 and v_2 in T_r''. By the inductive hypothesis the proposition is true for T_r'', and we have to prove it for T_r'.

Let r_1 and r_2 be the relations from which v_1 and v_2 originate respectively, for some R_1 and $R_2 \in \mathbf{R}$. Since v_1 is equated with v_2 using $K_p \rightarrow R_p - K_p$, $R_p \subseteq R_1^+$ and $R_p \subseteq R_2^+$ [BDB].

Part A. We assume the transformation involved is equating ndv's; otherwise the induction is trivially true. We have to consider only t_1 and t_2 which are tuples in T_r'', and $A \in R_p - K_p$ such that (a) $t_1[A] = v_1[A]$ and is an ndv in T_r'', (b) $t_2[A] = v_2[A]$ and is an ndv in T_r'', and (c) $v_1[A] \neq v_2[A]$ in T_r''.

There are three cases to be considered depending on whether K_p is included in both CP_{1A} and CP_{2A}.

Case 1: K_p is included in both CP_{1A} and CP_{2A}. Since $v_1[A]$ and $v_2[A]$ are ndv's in T_r'', A is neither in R_1 nor in R_2. By (a) and the inductive hypothesis **A**, $CP_{t_1A} =$

CP_{1A}. By (b) and the inductive hypothesis **A**, $CP_{t_2A} = CP_{2A}$. But by Corollary 3.2, $CP_{1A} = CP_{2A}$. Hence $CP_{t_1A} = CP_{t_2A}$, and therefore (1) holds for t_1 and t_2.

Now we prove (2) and (3) holds for t_1 and t_2. For both parts, observe that $v_1[R_p] = v_2[R_p]$ in T_r'. First we prove (2). By Facts 3.1 and 3.2, $R_p \supseteq CP_{1A} = CP_{t_1A}$. Since CP_{t_1A} is in R_p, if we prove $t_1[CP_{t_1A}] = v_1[CP_{t_1A}]$ in T_r'' and $t_2[CP_{t_1A}] = v_2[CP_{t_1A}]$ in T_r'', then we finish with (2). By (a) and the inductive hypothesis **A**, $t_1[CP_{t_1A}] = v_1[CP_{t_1A}]$ in T_r''. Similarly $t_2[CP_{t_2A}] = v_2[CP_{t_2A}]$ in T_r''. But, since $CP_{t_1A} = CP_{t_2A}$, $t_2[CP_{t_1A}] = v_2[CP_{t_1A}]$ in T_r''. Hence (2) holds for t_1 and t_2.

Now we prove (3). Let B be such that AB is not split in t_1. Since R_p can A-extend t_1, $B \in R_p$. If we prove $v_1[B] = t_1[B]$ in T_r'' and $v_2[B] = t_2[B]$ in T_r'', then we finish with (3). By (a) and the inductive hypothesis **A**, $t_1[B] = v_1[B]$ in T_r'' for all B such that AB is not split in t_1. Similarly $t_2[B] = v_2[B]$ in T_r'' for all B such that AB is not split in t_2. But, by Proposition 3.11, AB is not split in t_1 if and only if AB is not split in t_2. These facts together imply $t_2[B] = v_2[B]$ in T_r'' for all B such that AB is not split in t_1. Hence (3) holds for t_1 and t_2.

Case 2: $K_p \not\subseteq CP_{1A}$. We prove this case is impossible. By Lemma 3.2, there is a B in K_p such that either AB is not split in R_1^+ (v_1) or $CP_{1B} \supseteq \{A\}$.

Case 2.a.i: AB is not split in v_1 and $v_1[B]$ is an ndv in T_r''. So $v_2[B] = v_1[B]$ and is an ndv in T_r'', since we can apply $K_p \to R_p - K_p$. By the inductive hypothesis **A**, $v_1[A] = v_2[A]$ in T_r''. This contradicts (c).

Case 2.a.ii: AB is not split in v_1 and $v_1[B]$ is a constant in T_r''. By the inductive hypothesis **B**, $v_1[A]$ is a constant in T_r''. This contradicts (a).

Case 2.b.i: $CP_{1B} \supseteq \{A\}$ and $v_1[B]$ is a constant in T_r''. By the inductive hypothesis **B**, $v_1[CP_{1B}]$ are constants in T_r'', and $v_1[A]$ is a constant in T_r'', since $A \in$

CP_{1B}. This contradicts (a).

Case 2.b.ii: $CP_{1B} \supseteq \{A\}$ and $v_1[B]$ is an ndv in T_r''. Since we can apply $K_p \to R_p - K_p$, $v_2[B] = v_1[B]$ and is an ndv in T_r''. By the inductive hypothesis **A**, $CP_{1B} = CP_{2B}$ and $v_1[CP_{1B}] = v_2[CP_{1B}]$ in T_r''. Thus, $v_1[A] = v_2[A]$ in T_r'', since $A \in CP_{1B}$. This contradicts (c).

Since all subcases lead to contradiction, this case is impossible.

Case 3: $K_p \not\subseteq CP_{2A}$. This case can be proven to be impossible by an argument similar to the one in Case 2 above.

This concludes the proof of **Part A**.

Part B. For this part, we only have to consider a transformation that equates an ndv to a constant. Let tuple t_1 in T_r'' and $A \in R_p - K_p$ be such that: (a) $v_2[A]$ is a constant in T_r'', (b) $v_1[A]$ is an ndv in T_r'', and (c) $t_1[A] = v_1[A]$ in T_r''.

By a similar argument as in the Case 2 of the induction in Part **A**, we can prove that it must be the case that $K_p \subseteq CP_{1A}$. Hence by Fact 3.1, R_p can A-extend R_1^+ using K_p.

We want to prove that $t_1[CP_{t_1A}]$ are constants in T_r', and for all C such that AC is not split in t_1, $t_1[C]$ is a constant in T_r'. Since $t_1[A] = v_1[A]$ and is an ndv in T_r'', the inductive hypothesis **A** implies, $CP_{1A} = CP_{t_1A}$, $t_1[CP_{1A}] = v_1[CP_{1A}]$ in T_r'', and for all C such that AC is not split in t_1, $t_1[C] = v_1[C]$ in T_r''. Since $CP_{1A} = CP_{t_1A}$, by Proposition 3.11, AC is not split in t_1 if and only if AC is not split in v_1. Since R_p can A-extend v_1, C is in R_p; also by Fact 3.2, $R_p \supseteq CP_{1A}$. Observe that $v_1[R_p] = v_2[R_p]$ in T_r'. Then it suffices to prove that $v_2[CP_{1A}]$ are constants in T_r'' and $v_2[C]$ is a constant in T_r'' for any C such that AC is not split in v_1.

There are two cases to be examined depending on whether R_p can A-extend R_2^+.

Case 1: R_p can A-extend R_2^+. First notice that if $A \in R_2$, then $R_p = R_2$ and $v_2[R_p]$ are constants in T_r'' and what we want to prove follows trivially. Hence, let us assume $A \notin R_2$ in the rest of the proof for this case.

Since R_p can A-extend both R_1^+ and R_2^+, by Fact 3.2, CP_{1A} and CP_{2A} are in R_p. Since A is neither in R_1 nor in R_2, by Propositions 3.2 and 3.3 and definition of CP_{1A} and CP_{2A}, $A \notin CP_{1A}CP_{2A}$. Then, since $A \in R_p$, $A \in R_p - CP_{1A}CP_{2A}$. Thus if $CP_{1A} \bigcap CP_{2A} \neq \emptyset$, by Proposition 3.8 with $A = B$, then $CP_{1A} = CP_{2A}$. Hence either $CP_{1A} = CP_{2A}$ or $CP_{1A} \bigcap CP_{2A} = \emptyset$. If $CP_{1A} = CP_{2A}$, then by the inductive hypothesis **B** and $v_2[A]$ is a constant in T_r'', $v_2[CP_{1A}]$ are constants in T_r''. Also by Proposition 3.11 AC is not split in v_1 if and only if AC is not split in v_2, which implies that $v_2[C]$ is a constant in T_r'', by the inductive hypothesis **B**. Hence for this case the inductive hypothesis is satisfied in T_r'.

On the other hand, if $CP_{1A} \bigcap CP_{2A} = \emptyset$, then $CP_{1A} \subseteq R_p - CP_{2A}$. Thus $K_p \not\subseteq CP_{2A}$ and by Lemma 3.2, there is $B \in K_p - CP_{2A}$ such that either AB is not split in R_2^+ or $CP_{2B} \supseteq \{A\}$. By similar arguments as in cases 2.a.i and 2.b.ii of the inductive proof in Part **A**, we can prove that $v_2[B]$ must be a constant in T_r''; else $v_1[A] = v_2[A]$ in T_r''. By Lemma 3.3, CP_{1A} is not split in R_2^+. Since $B \in CP_{1A}$ and $v_2[B]$ is a constant in T_r'', by the inductive hypothesis **B**, $v_2[CP_{1A}]$ are constants in T_r''.

Now, let us consider C such that AC is not split in v_1. This implies C is in R_p and therefore in R_2^+. If $C \in CP_{2B}$, then $v_2[C]$ is a constant in T_r'', since by the inductive hypothesis **B** and $v_2[B]$ is a constant in T_r'', $v_2[CP_{2B}]$ are constants in T_r''. Else, if $C \notin CP_{2B}$, then by Proposition 3.12, BC is not split in R_2^+, and by inductive hypothesis **B** and $v_2[B]$ is a constant in T_r'', $v_2[C]$ is a constant in T_r''.

Case 2: R_p cannot A-extend R_2^+. There are two subcases to be considered depending on whether $A \in R_2$.

Case 2.a: $A \in R_2$. Notice that this implies R_2 is the only relation scheme that can A-extend R_2^+. If there is B in CP_{1A} such that $B \in R_2$, then, by Proposition 3.7, $CP_{1A} \subseteq R_2$. Since $R_2 \supseteq \{A\}$ and $A \notin R_1$, by Fact 3.1, R_2 can A-extend R_1^+. Hence if AC is not split in R_1^+, then $C \in R_2$. Therefore $v_2[CP_{1A}]$ and $v_2[C]$ are constants in T_r''.

On the other hand, if for all $B \in CP_{1A}$, $B \notin R_2$, then for all $B \in CP_{1A}$, B is not in any relation scheme that can A-extend R_2^+. Hence there exists B such that $B \in K_p - CP_{2A}$ and $B \in CP_{1A}$. Since R_p contains AB and B is not in any relation that can A-extend R_2^+, by Proposition 3.6, $CP_{2B} \supseteq \{A\}$. Then by similar arguments as in cases 2.a.i and 2.b.ii of the inductive proof of Part **A**, $v_2[B]$ must be a constant in T_r'', else $v_1[A] = v_2[A]$ in T_r''. Since R_p can A-extend R_1^+ using K_p and by assumption in this case, Lemma 3.4 implies CP_{1A} is not split in R_2^+. Since $B \in CP_{1A}$ and $v_2[B]$ is a constant in T_r'', by the inductive hypothesis **B**, $v_2[CP_{1A}]$ are constants in T_r''.

Let C be such that AC is not split in R_1^+. By a similar argument as in Case 1 above, we can prove $v_2[C]$ is a constant in T_r''.

Case 2.b: $A \notin R_2$. Since R_p cannot A-extend R_2^+, R_p cannot A-extend R_2^+ using K_p. Hence $K_p \not\subseteq CP_{2A}$. By Lemma 3.2, there is B in $K_p - CP_{2A}$ such that AB is not split in R_2^+ or $CP_{2B} \supseteq \{A\}$. Then by similar arguments as in Case 2.a above, we can prove $v_2[B]$ must be a constant in T_r'', CP_{1A} is not split in R_2^+, and $v_2[CP_{1A}]$ are constants in T_r''.

Assume C is such that AC is not split in v_1. By a similar argument as in Case 1 above, we can prove $v_2[C]$ is a constant in T_r''.

This concludes the proof of Part **B** and the proof of the proposition. □

Corollary 3.3: Let (**R**, F) be an acyclic BCNF database scheme. Let **r** be a state of (**R**, F) and let T_r be the tableau of **r**. Let $\tau_1 \ldots \tau_k$ be a sequence of fd-rules applied to T_r. Suppose $T_r'' = \tau_1 \ldots \tau_{k-1}(T_r)$ is nonempty and t_1, t_2 are in T_r''. Assume we can apply τ_k: $K_p \to R_p - K_p$ to equate t_1 and t_2 such that the ndv $t_2[A]$ is equated to the constant $t_1[A]$, for some $A \in R_p - K_p$. Suppose $T_r' = \tau_k(T_r'')$ is nonempty. Then

A: $K_p \subseteq CP_{t_2A}$.

B: For any t in T_r'' such that $t[A] = t_2[A]$ in T_r'', $t_1[CP_{tA}]$ are constants in T_r'' and $t_1[CP_{tA}] = t[CP_{tA}]$ in T_r'.

Proof: Part A follows from the argument at the beginning of the inductive proof of Part **B** in Lemma 3.5.

For part (B), we have proven that in the inductive part of Lemma 3.5 Part B, with v_2, v_1, and t_1 in place of t_1, t_2, and t respectively. □

Lemma 3.6: Let ($\mathbf{R}$, F) be an acyclic BCNF database scheme. Let $\mathbf{r}$ be a state of ($\mathbf{R}$, F) and let T_r be the tableau of $\mathbf{r}$. Let $\tau_1 \ldots \tau_k$ be a sequence of fd-rules applied to T_r. Suppose $T_r'' = \tau_1 \ldots \tau_{k-1}(T_r)$ is nonempty and t_1, t_2 are in T_r''. Assume we can apply τ_k: $K_p \to R_p - K_p$ to equate t_1 and t_2 such that the ndv $t_2[A]$ is equated to the constant $t_1[A]$, for some $A \in R_p - K_p$. Suppose $T_r' = \tau_k(T_r'')$ is nonempty. Then for all $B \in R_p - K_p$, if $t_1[B]$ is an ndv in T_r'', then $t_2[B]$ is an ndv in T_r''.

Proof: By contradiction. Assume there is $B \in R_p - K_p$ such that $t_1[B]$ is an ndv in T_r'' and $t_2[B]$ is a constant in T_r''; it is clear that $A \neq B$. Assume t_1 and t_2 originate from r_1 and r_2 respectively, for some R_1 and R_2 in $\mathbf{R}$.

By assumption, τ_k equates the ndv $t_2[A]$ to the constant $t_1[A]$. But observe that τ_k also equates the ndv $t_1[B]$ to the constant $t_2[B]$. Since K_p is the key used by τ_k to equate t_1 and t_2, from Corollary 3.3.A, K_p is in CP_{1B} and K_p is in CP_{2A}. Since $B \notin R_1$ and $A \notin R_2$, by Fact 3.1, R_p can both B-extend R_1^+ and A-extend R_2^+. By Fact 3.2, R_p contains both CP_{1B} and CP_{2A} as well as A and B.

By Corollary 3.3.B, $t_1[CP_{2A}]$ are constants in T_r''. Hence since $t_1[B]$ is an ndv in T_r'', $B \notin CP_{2A}$. Similarly, by Corollary 3.3.B, $t_2[CP_{1B}]$ are constants in T_r''. Hence since $t_2[A]$ is an ndv in T_r'', $A \notin CP_{1B}$. Since $A \notin R_2$ and $B \notin R_1$, by Propositions 3.2 and 3.3 and definitions of CP_{1B} and CP_{2A}, $A \notin CP_{2A}$ and $B \notin CP_{1B}$. Therefore neither A nor B are in $CP_{1B}CP_{2A}$.

Since R_p can B-extend R_1^+ and $B \notin R_1$, $A \notin R_1$; else by Fact 3.4, $CP_{1B} \supseteq \{A\}$. Similarly, $B \notin R_2$. Observe $CP_{1B} \cap CP_{2A} \neq \emptyset$, since they share K_p. Since $R_p \supseteq CP_{1B}CP_{2A}AB$, and AB is in $R_p - CP_{1B}CP_{2A}$, and neither A nor B is in R_1R_2, by Proposition 3.8, $CP_{1B} = CP_{2A}$. We prove this leads to a contradiction.

Since $A \in R_p - CP_{1B}$, by Fact 3.3, CP_{1B} is a connection point that can A-extend R_1^+ and contains K_p. Hence CP_{1A} contains K_p and since CP_{2A} does too, by Corollary 3.2, $CP_{1A} = CP_{2A}$. Since $CP_{2A} = CP_{1B}$, $CP_{1A} = CP_{1B}$. Since $R_p \supseteq CP_{1A}A$, by Fact 3.1, R_p can A-extend R_1^+. This fact, $B \in R_p - CP_{1A}$, and Proposition 3.5, imply AB is not split in R_1^+. Since $t_1[A]$ is a constant in T_r'', by Lemma 3.5 part B, $t_1[B]$ must be a constant in T_r''. A contradiction. □

3.5.7. γ-acyclic BCNF Database Schemes are Embedded-complete

A class of schemes called embedded-complete database schemes was recently proposed to capture the intuition that every piece of information on some relation scheme is explicitly represented in a database. We define that class of schemes now. Let $W \subseteq \mathbf{U}$. We define $\downarrow W \downarrow = \bigcup \{\pi_W(r_j) \mid r_j$ is a relation on $R_j \in \mathbf{R}$ and $R_j \supseteq W\}$. A database scheme $(\mathbf{R}, F)$ is *embedded-complete* (w.r.t. F) if for any consistent state $\mathbf{r}$ of $(\mathbf{R}, F)$, $[X] = \downarrow X \downarrow$, for any $X \subseteq R_i$, for some $R_i \in \mathbf{R}$ [CM]. Before we can show that an acyclic BCNF database scheme $(\mathbf{R}, F)$ is bounded, we need to show that it is embedded complete.

Lemma 3.7: Let $\mathbf{r}$ be a state of an acyclic BCNF database scheme $(\mathbf{R}, F)$. Let T_r be the tableau for $\mathbf{r}$. Let $\tau_1 \ldots \tau_k$ be a sequence of fd-rules applied to T_r. If $T_r' = \tau_1 \ldots \tau_k(T_r)$ is nonempty, then $\downarrow X \downarrow = [X]$, for any $X \subseteq R_i$, for some $R_i \in \mathbf{R}$, where $[X]$ is the X-total projection in T_r'.

Proof: By induction on k.

Basis: If $k = 0$, then clearly $\downarrow X \downarrow = [X]$, for any $X \subseteq R_i$, for some $R_i \in \mathbf{R}$.

Induction: Assume the inductive hypothesis is true for $T_r'' = \tau_1 \cdots \tau_{k-1}(T_r)$. Let τ_k: $K_l \rightarrow R_l - K_l$, where K_l is a nontrivial key of R_l, be the fd-rule applied to T_r'' and assume it equates t_1 and t_2. We have to consider only $A \in R_l - K_l$ and t_1 and t_2 in T_r'' such that $t_1[A]$ is a constant and $t_2[A]$ is an ndv; since if t_1 and t_2 are distinct ndv's on A the induction holds trivially. So after the application of τ_k, all the entries with $t_2[A]$ are changed to the constant $t_1[A]$. Assume $T_r' = \tau_k(T_r'')$ is nonempty. Suppose there exists X in some R_i, $R_i \in \mathbf{R}$, such that $\downarrow X \downarrow \neq [X]$. We prove this case is impossible.

Let t be the tuple such that $t[X] \notin \downarrow X \downarrow$ in T_r'. Clearly $A \in X$ or else by the inductive hypothesis $[X] = \downarrow X \downarrow$. Also $t[A] = t_2[A]$ in T_r''. Assume t_1, t_2, and t come from R_1, R_2, and R_3 respectively. Let $A_1, \ldots, A_m$ be an ordering of the elements in X such that $A = A_m$.

We claim there exists A_r, for some $1 \leq r \leq m-1$, such that $t_1[A_r] \neq t[A_r]$ in T_r'' and $t[A_r]$ is a constant in T_r''. There are two cases to be considered depending on whether $t_1[X]$ are constants in T_r''.

Case 1: $t_1[X]$ are constants in T_r''. Hence $t_1[X]$ are constants in T_r'. Since $t[X]$ are constants in T_r', $t_1[X] \neq t[X]$ in T_r'; else our assumption about t is violated. Hence there exists A_r such that $t_1[A_r] \neq t[A_r]$ and $t[A_r]$ is a constant in T_r'. Thus $t[A_r]$ is not equated by τ_k. Therefore $t[A_r] \neq t_1[A_r]$ and $t[A_r]$ is a constant in T_r''.

Case 2: $t_1[A_r]$ is an ndv in T_r'', for some $1 \leq r \leq m-1$. If $A_r \notin R_l$, then $t[A_r]$ is not equated by τ_k and hence it is a constant in T_r'' and $t_1[A_r] \neq t[A_r]$ in T_r''. On the other hand, if $A_r \in R_l$, we claim that $A_r \notin K_l$. Assume otherwise. By Corollary 3.3.A, $K_l \subseteq$

CP_{2A} and by Corollary 3.3.B, $t_1[CP_{2A}]$ are constants in T_r''. Then $t_1[K_l]$ are constants in T_r''. A contradiction to $t_1[A_r]$ is an ndv in T_r''. Hence $A_r \in R_l - K_l$; therefore from Lemma 3.6, $t_2[A_r]$ is an ndv in T_r'' and we have two ndv's being equated on column A_r when applying τ_k. Hence $t_1[A_r]$ is an ndv in T_r'. Since $t[A_r]$ is a constant in T_r', $t[A_r]$ is also a constant in T_r''. Thus $t[A_r] \neq t_1[A_r]$ in T_r''.

From the above two cases, our claim holds. And notice that from the arguments in the above claim $t_1[A_r] \neq t[A_r]$ and $t[A_r]$ is a constant in T_r'.

It is obvious, then that AA_r is split in R_3^+, else from Lemma 3.5 part B and $t[A_r]$ is a constant in T_r'', $t[A]$ must be a constant in T_r''. We claim $A_r \notin CP_{3A}$. Assume otherwise. That is, assume $A_r \in CP_{3A}$. From Corollary 3.3.B, $t_1[CP_{3A}] = t[CP_{3A}]$ and are constants in T_r'. Then $t[A_r] = t_1[A_r]$ in T_r'; but we have already proven this is not true. Hence $A_r \notin CP_{3A}$.

It is clear that $A_rA \subseteq R_3^+$. Assume A_rA is not contained in any relation scheme in R_3^+. Then there is a path of length greater than or equal to 2 from A_r to A in $H_{R_3^+}$, hence in $\mathbf{R}$, and a path of length one in $\mathbf{R}$ from A_r to A formed by R_i, the relation scheme in $\mathbf{R}$ containing X. This contradicts Theorem 3.2. Hence there exists R_t in R_3^+ such that it contains A_rA. There are two cases to be examined depending on whether R_t can A-extend R_3^+.

Case i: Assume R_t can A-extend R_3^+. By Fact 3.2, $R_t \supseteq CP_{3A}$. Since $A_r \notin CP_{3A}$, $A_r \in R_t - CP_{3A}$. By Proposition 3.4, either $CP_{3A_r} \supseteq CP_{3A}A$ or $CP_{3A_r} = CP_{3A}$. If they are equal, then by Proposition 3.5, A_rA is not split in R_3^+; a contradiction to the fact that A_rA is split in R_3^+. Hence $CP_{3A_r} \supseteq \{A\}$. Then, Lemma 3.5 part B and $t[A_r]$ is a constant in T_r'' imply $t[A]$ is a constant in T_r''. A contradiction to the assumption that $t[A]$ is an ndv in T_r''.

Case ii: Assume R_t cannot A-extend R_3^+. If A_r is in some R_l such that it can A-extend R_3^+, then $A_r \in R_l - CP_{3A}$ and the proof in Case i above applies here. On the other hand, if A_r is not in any relation scheme that can A-extend R_3^+, then by Proposition 3.6, $CP_{3A_r}^* \supseteq \{A\}$, and the proof in Case i above for the case when $CP_{3A_r} \supseteq \{A\}$ applies here.

Hence it is impossible that $t[X] \notin \downarrow X \downarrow$ and the induction is complete. □

Corollary 3.4: Let (**R**, F) be an acyclic BCNF database scheme. Then (**R**, F) is embedded-complete.

Proof: It follows from Lemma 3.7. □

3.5.8. γ-acyclic BCNF Database Schemes are Bounded

The only thing left to do in order to show that acyclicity is a sufficient condition for BCNF database schemes to be bounded is to prove that Step 2 of Algorithm 2 does indeed equate only ndv's.

Lemma 3.8: Let (**R**, F) be an acyclic BCNF database scheme. Let **r** be a consistent state of (**R**, F). Let T_r be the tableau for **r**. Then Step 2 of Algorithm 2 equates only ndv symbols.

Proof: Assume otherwise. Then, there is a sequence of fd-rule transformations $\tau_1 \ldots \tau_l$ used in Step 2 of Algorithm 2 such that τ_l: $K_l \rightarrow R_l - K_l$, $R_l \in \mathbf{R}$, is the first fd-rule to equate an ndv with a constant. Assume τ_l equates the ndv in $t_2[A]$ with a constant from $t_1[A]$, $A \in R_l - K_l$. Also assume t_2 originates from r_2 and t_1 originates from r_1, for some R_1 and $R_2 \in \mathbf{R}$. R_l must be in both R_2^+ and R_1^+ [BDB].

There are two cases to be considered depending on whether $t_2[K_l]$ consists of constants.

Case 1: $t_2[K_l]$ are constants. Hence $t_1[K_lA]$ are constants. Since $K_lA \subseteq R_l$, by Lemma 3.7, $t_1[K_lA] \in \downarrow K_lA \downarrow$. Thus there exists t' from r_p in T_r such that $t'[K_lA] =$

$t_1[K_l A]$, for some $R_p \in \mathbf{R}$ such that $R_p \supseteq K_l A$. Observe K_l is a key of R_p. By Corollary 3.3.A, $K_l \subseteq CP_{2A}$. Since $A \in R_2^+ - R_2$ and $R_p \supseteq K_l A$, by Facts 3.1 and 3.2, R_p is in R_2^+. Also $t_2[K_l] = t_1[K_l]$, hence $t'[K_l] = t_2[K_l]$. Therefore $t_2[A]$ should have been equated to a constant in Step 1 of Algorithm 2, since t' and R_p satisfy the conditions in while-loop in Algorithm 2.

Case 2: $t_2[K_l]$ has some ndv's. Then $t_1[K_l]$ has some ndv's. Since, by Corollary 3.3.A, $K_l \subseteq CP_{2A}$, $t_1[CP_{2A}]$ has some ndv's before applying τ_l. But this is impossible since, by Corollary 3.3.B, $t_1[CP_{2A}]$ are constants before applying τ_l.

Since both cases lead to contradiction, the lemma is proven. □

Theorem 3.3: Let (**R**, F) be an acyclic BCNF database scheme. (**R**, F) is bounded.

Proof: Let **r** be a consistent state of (**R**, F). Let T_r be the tableau for **r**. We chase T_r using Algorithm 2.

By Lemma 3.8, the total part of any tuple in $CHASE_F(T_r)$ is obtained in Step 1 of Algorithm 2. This is obtained by at most $|\mathbf{R}| - 1$ applications of fd-rules. Then (**R**, F) is bounded. □

3.6. Efficient Computation of X-total Projection

The problem of how to generate correct answers for queries is important in most systems. Following other authors [GMV][M][MUV][NG][S1][S2], we define the information content of a database as the set of total tuples in the representative instance. The set of total tuples is the set of sentences that are logically implied by the state and constraints [GMV][MUV].

In this section we show that there is a simple and efficient way of computing $[X]$, the X-total projection of the representative instance, for an acyclic BCNF database scheme, for any $X \subseteq \mathbf{U}$. Hence answers to many queries can be generated efficiently.

Clearly if for all $R_i \in \mathbf{R}$, R_i^+ does not contain X, then $[X]$ is an empty set. We will show that if there exists $R_i \in \mathbf{R}$ such that $R_i^+ \supseteq X$, then $[X]$ can be computed by a simple and efficient method. This also demonstrates that the semantics of relationships among attributes is simple.

We first give some definitions required for the rest of this chapter. Let $S' = \langle S_0, \ldots, S_n \rangle$ be a sequence of distinct relation schemes from $\mathbf{R}$. Then $\pi_X(S_0 \bowtie \ldots \bowtie S_n)$ is an *extension join* of S_0 *covering* X [C1][IIK][H1] if $[S_i \cap (\bigcup_{j=0}^{i-1} S_j)] \to [S_i - (S_i \cap (\bigcup_{j=0}^{i-1} S_j))] \in F^+$, where $S_i - (S_i \cap (\bigcup_{j=0}^{i-1} S_j)) \neq \emptyset$, for all $0 < i \leq n$, and $\bigcup_i S_i \supseteq X$.

Lemma 3.9: Let ($\mathbf{R}$, F) be an acyclic BCNF database scheme, $X \subseteq \mathbf{U}$ and $V = \{V_1, \ldots, V_k\}$ the u.m.c. among X. If there exists $R_i \in \mathbf{R}$ such that $R_i^+ \supseteq X$, then V is lossless w.r.t. F.

Proof: Let us consider $H_{R_i^+} = \langle R_i^+, \mathbf{S}_i \rangle$. Let $H = \mathbf{S}_i \cup V$. Since $(\bigcup_{i=1}^{k} V_i) \subseteq R_i^+$, H is a database scheme defined on R_i^+. It is easy to show that H is acyclic since H is a connected subset of Bachman($\mathbf{R}$). Also H is a lossless decomposition of R_i^+ w.r.t. the embodied fd's in $\mathbf{S}_i$ since $\mathbf{S}_i$ is lossless [U1]. Since H is acyclic and lossless, by a result in [Y1], V is a connected subset of H implies V is lossless w.r.t. the embodied fd's in $\mathbf{S}_i$ and hence is lossless w.r.t. F. □

Lemma 3.10: Let $\mathbf{R}$ be acyclic. Suppose $\{V_1, \ldots, V_k\}$ is the u.m.c. among X and $\{S_1, \ldots, S_k\} \subseteq \mathbf{R}$ such that $S_i \supseteq V_i$, for all $1 \leq i \leq k$. Then $\pi_X(\bowtie_{i=1}^{k} S_i) = \pi_X(\bowtie_{i=1}^{k} \pi_{V_i}(S_i))$.

Proof: See [CA]. □

In Section 3.5, we proved that Algorithm 2 correctly computes the total tuples in the representative instance. It is not difficult to see that every X-total tuple is computed by an extension join of an R_p covering X, for some $R_p^+ \supseteq X$ [AtC][C1][IIK][H1].

Let us denote $[X]_{R_p^+}$ as the union of all the extension joins of R_p covering X. Let E_X $= \bigcup_{R_p^+ \supseteq X} [X]_{R_p^+}$.

Theorem 3.4: Let (**R**, F) be an acyclic BCNF scheme. Suppose $X \subseteq \mathbf{U}$ and $V = \{V_1, \ldots, V_k\}$ is the u.m.c. among X. If there exists $R_i \in \mathbf{R}$ such that $R_i^+ \supseteq X$, then $[X]$ $= E_X = \pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow)$.

Proof: $\pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow) \subseteq [X]$. By Lemma 3.9, V is lossless w.r.t. F. Hence $\pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow) \subseteq [X]$ follows [CA][MUV].

$[X] \subseteq E_X$. Since every X-total tuple is generated by some extension join of some R_p such that $R_p^+ \supseteq X$, $[X] \subseteq E_X$.

$E_X \subseteq \pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow)$. Since every extension join in $[X]_{R_p^+}$ is a connected subset of **R** that contains X, by the definition of u.m.c. among X and Lemma 3.10, $[X]_{R_p^+} \subseteq \pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow)$. Hence $E_X \subseteq \pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow)$. □

By Theorem 3.4 for any $X \subseteq \mathbf{U}$, the relationship among X in an acyclic BCNF database scheme is simple. That is, either there is no relationship among X or the relationship is represented by the u.m.c. among X. Furthermore, this relationship (i.e., the X-total projection) can be computed easily and efficiently. In fact, the expression $\pi_X(\bowtie_{j=1}^{k} \downarrow V_j \downarrow)$ that computes the X-total projection is in some sense optimal, since $\{V_1, \ldots, V_k\}$ is the u.m.c. among X; in other words, it requires, in some sense, the minimal number of joins to compute the X-total projection.

3.7. Lossless γ-acyclic BCNF Database Schemes are Connection-trap-free

Query answering is an important function in a database system. Before a user is able to retrieve information from a database, he has to understand the semantics of the application as well as the operations (i.e., data language) required to retrieve data. As pointed out in [CA][CM], the understanding process might be difficult. A class of database schemes known as the connection-trap-free schemes has recently been proposed to allow users to easily retrieve correct information from a database [CA]. These schemes have the properties that they are simple in semantics and hence users are able to understand the application easily. Moreover, the information retrieval process is simple and answers to many queries can be generated easily and efficiently. We now define the class of connection-trap-free schemes.

Let $X \subseteq \mathbf{U}$ and let $(\mathbf{R}, F)$ be a database scheme. A state $\mathbf{r}$ of $(\mathbf{R}, F)$ is said to have a *complete unique minimal connection among* X if $[X] = \pi_X(\downarrow V_1 \downarrow \bowtie \ldots \bowtie \downarrow V_k \downarrow)$, where $\{V_1, \ldots, V_k\}$ is the unique minimal connection among X. Then we say that $(\mathbf{R}, F)$ is *connection-trap-free (ctf)* (w.r.t. F) if for any consistent state $\mathbf{r}$ of $(\mathbf{R}, F)$, $\mathbf{r}$ has a complete unique minimal connection among X, for any $X \subseteq \mathbf{U}$.

In this section we prove that lossless acyclic BCNF database schemes are connection-trap-free w.r.t. the fd's embodied in the relations in the database scheme.

Theorem 3.5: Let $(\mathbf{R}, F)$ be a lossless acyclic BCNF database scheme. Then $(\mathbf{R}, F)$ is ctf.

Proof: Since $(\mathbf{R}, F)$ is lossless w.r.t. F, there is $R_i \in \mathbf{R}$ such that $R_i^+ = \mathbf{U}$ [ABU]. By Theorem 3.4, the theorem follows. □

Let us consider $H_{R_i^+} = \langle R_i^+, \mathbf{S}_i \rangle$. Let F_{S_i} be the set of nontrivial fd's embodied in the relation schemes in $\mathbf{S}_i$.

Corollary 3.5: $(\mathbf{S}_i, F_{S_i})$ is ctf.

Proof: Since $(\mathbf{S}_i, F_{S_i})$ is a lossless acyclic BCNF scheme, the corollary follows from Theorem 3.5. □

3.8. Incremental Testing of Satisfaction

Constraints are logical restrictions imposed on data so that the information in a database correctly represents the part of real world that an application is interested in. Hence ensuring that the data satisfies the constraints is important in data management. However enforcing the constraints cost-effectively is a difficult problem in general. Some work has been done on this problem; see for example [BBC][St]. In the context of the weak instance model, testing satisfaction of fd's might be as expensive as generating the representative instance from the tableau of the state [H2]; it requires polynomial time and space in the size of the database state. Since database size is large in general and existing systems do not provide facilities for chasing, testing satisfaction of fd's by generating the representative instance might not be practical. We regard an algorithm for incrementally testing fd's as cost-effective if it does not require the generation of the representative instance and the verification process is done on some specific relations and can be carried out in polynomial time.

Under the weak instance model, several authors proposed the class of independent schemes to solve this problem [GY][IIK][S1][S2]. For the class of independent schemes, ensuring each relation satisfies the constraints locally guarantees the state is consistent. Hence this restricts the scope of verification to a single relation and the representative instance is not generated in the verification process. Therefore, independence allows a cost-effective way to test satisfaction of constraints incrementally.

In this section, we show that if $(\mathbf{R}, F)$ is an acyclic BCNF database scheme, then there is also a cost-effective way to test satisfaction of F incrementally. Since deletion of tuples do not affect the consistency of a state, we consider only the insertion operation.

Let $\mathbf{r}$ be a consistent state of an acyclic BCNF database scheme $(\mathbf{R}, F)$. Let r_p be a relation being updated, where $r_p \in \mathbf{r}$. Let t be the tuple being inserted in r_p. Let $\{K_{p_1}, \ldots, K_{p_m}\}$ be the set of nontrivial keys of R_p. Consider Algorithm 3, shown below. We claim that if no contradiction is found, that is, if Algorithm 3 prints *yes*, the updated state is consistent w.r.t. F. Theorem 3.6, below, proves this claim.

Algorithm 3

Input: A consistent state $\mathbf{r}$ of an acyclic BCNF database scheme $(\mathbf{R}, F)$.
A tuple t to be inserted in $r_p \in \mathbf{r}$, $R_p \in \mathbf{R}$.

Output: *no*, if $\mathbf{r} \cup \{t\}$ is not consistent w.r.t. F; *yes* otherwise.

Notation: $\{K_{p_1}, \ldots, K_{p_m}\}$ is the set of nontrivial keys of R_p.

```
for each K_{p_i} do begin
   for each A ∈ R_p − K_{p_i} do begin
      for all R_q ∈ R such that R_q ⊇ K_{p_i}A do begin
         if π_{K_{p_i}A}(r_q) ∪ π_{K_{p_i}A}({t}) does not satisfy K_{p_i}→A then do begin
            print no; halt end
      end
   end
end
print yes
```

We need the following definitions for proving Theorem 3.6. Let $\mathbf{K} = \{K_i \to A \mid K_i \to A$ is a nontrivial fd embodied in some R in $\mathbf{R}\}$. Let us index the elements of $\mathbf{K}$ as $\{f_1{:}K_1 \to A_1, \ldots, f_q{:}K_q \to A_q\}$. We define $\downarrow f_i \downarrow = \downarrow K_i A_i \downarrow$, which, by the definition in Section 3.5.7, is $\bigcup_{R_p \supseteq K_i A_i} \pi_{K_i A_i}(r_p)$.

Theorem 3.6: Let $\mathbf{r}$ be a state of an acyclic BCNF database scheme $(\mathbf{R}, F)$. Let T_r be the tableau for $\mathbf{r}$. If for all $1 \le i \le q$, $\downarrow f_i \downarrow$ satisfies the fd f_i, then $\mathbf{r}$ is a consistent state w.r.t. F.

Proof: We prove the theorem by induction on number of fd-rules applying to T_r.

Basis: Zero applications of fd-rules to T_r. Trivial since all f_i's are satisfied in T_r.

Induction: Assume $T_r'' = \tau_1 \ldots \tau_{k-1}(T_r)$ is nonempty; $k \geq 1$. Let τ_k: $K_l' \to R_l - K_l$, where K_l is a nontrivial key of R_l, ~~be~~ the fd-rule applied to T_r'' and assume τ_k equates tuples t_1 and t_2 in T_r''. We only have to consider $A \in R_l - K_l$ such that $t_1[A]$ is a constant, $t_2[A]$ is a constant and $t_1[A] \neq t_2[A]$ in T_r''. We claim this case is impossible. Assume otherwise. Notice that $t_1[K_l] = t_2[K_l]$ in T_r''. There are two cases to be examined depending on whether $t_1[K_l]$ contains an ndv.

Case 1: If $t_1[K_l]$ are constants, then by Lemma 3.7, $t_1[K_lA]$ and $t_2[K_lA]$ are elements in $\iota f_j \iota$ in T_r'', for some $1 \leq j \leq q$. Since by the inductive hypothesis $\iota f_j \iota$ satisfies the fd f_j, this case is impossible.

Case 2: We claim the case $t_1[K_l]$ contains an ndv is impossible. Assume otherwise. Then there is $B \in K_l$ such that $t_1[B]$ is an ndv in T_r''. Let r_1 be the relation from which t_1 originates, for some $R_1 \in \mathbf{R}$. A and B are split in R_1^+; else $t_1[B]$ is a constant in T_r'' by Lemma 3.5 part B. There are two cases to be considered depending on whether B is in a relation scheme that can A-extend R_1^+.

Case 2.a: Assume B is in a relation scheme R_t which can A-extend R_1^+. Observe $B \notin CP_{1A}$; else since $t_1[A]$ is a constant in T_r'', $t_1[B]$ is a constant in T_r'' by Lemma 3.5 part B. Hence $B \in R_t - CP_{1A}$. Then by Proposition 3.4, either $CP_{1B} \supseteq CP_{1A}A$ or $CP_{1B} = CP_{1A}$. If they are equal, then by Proposition 3.5, AB is not split in R_1^+. A ~~contradiction to~~ the fact that AB is split in R_1^+. Thus $CP_{1B} \supseteq \{A\}$. Since $B \in K_l$, $t_1[B] = t_2[B]$. Lemma 3.5 part A, $t_1[B] = t_2[B]$ and is a ndv in T_r'', imply $t_1[CP_{1B}] = t_2[CP_{1B}]$ in T_r''. Since $CP_{1B} \supseteq \{A\}$, $t_1[A] = t_2[A]$ in T_r''. This is a contradiction to the fact that they are distinct in T_r''.

Case 2.b: If B is not in any relation scheme that can A-extend R_1^+ and since $AB \subseteq R_l$, then by Proposition 3.6, $CP_{1B} \supseteq \{A\}$, and the above proof applies, which again leads to a contradiction.

From Cases 2.a and 2.b, the case $t_1[K_l]$ contains an ndv is impossible.

This completes the inductive proof. □

We have proven our claim that Algorithm 3 is an algorithm to test incrementally the satisfaction of fd's for an acyclic BCNF database scheme.

Therefore for acyclic BCNF schemes, the satisfaction of fd's is basically enforced by creating indices. Since relational systems allow indices to be created for keys in a relation scheme, the enforcement of fd's can be done in polynomial time and without generating the representative instance; that is, cost-effectively.

3.9. Conclusions

We proved that γ-acyclicity and BCNF is a sufficient condition for boundedness of database schemes w.r.t. fd's. More importantly, we showed that the boundedness w.r.t. fd's of the class of γ-acyclic BCNF database schemes determines that this class of schemes is highly desirable w.r.t. query answering and updates. We proved this by first showing that it is bounded, that the set of total tuples can be computed efficiently and that it allows enforcement of constraints to be performed incrementally and cost-effectively.

With fd's, the only class of database schemes that is proven to be bounded is the class of independent schemes [AtC][C1][IIK][MRW][S3]. Since γ-acyclic BCNF database schemes are not independent in general, our result enlarges the class of known bounded database schemes.

The set of total tuples can be considered as the information content of a database [GMV][M][MUV][NG][S1]. We derived a simple and an efficient algorithm to generate the X-total projection for this class of database schemes. This demonstrates that relationships among attributes are simple and answers to many queries can be computed very efficiently. Moreover, if a γ-acyclic BCNF database scheme is lossless, then it is also ctf. Hence the class of γ-acyclic BCNF database schemes is highly desirable w.r.t. query answering.

The problem of how to enforce constraints efficiently is a major problem in data management. In the context of the weak instance model, if we can find a cost-effective way to determine whether an updated state is consistent with the constraints, then this problem can be solved adequately. So far, the only class of database schemes that allows such enforcement of constraints is the class of independent database schemes [GY][IIK][S1][S2]. In this chapter, we showed that for γ-acyclic BCNF database schemes, there is a simple and cost-effective way of determining if an updated state satisfies the set of fd's embodied in the relation schemes. Unlike the incremental approach in [C2], our approach only needs to create indices on keys and no other data structure is required. Since relational systems allow indices to be created on keys, enforcement of fd's can be carried out cost-effectively. γ-acyclic BCNF database schemes are not independent database schemes in general, hence our result extends the class of database schemes that allows efficient enforcement of constraints. This shows the desirability of this class of database schemes w.r.t. updates.

Apparently to determine if a class of database schemes is bounded or not is fundamental to the analysis of the behavior of the database schemes w.r.t. query processing and updates. On the other hand, proving a class of database schemes is bounded seems to be very difficult, even in our restricted case of γ-acyclic BCNF database schemes. To resolve this problem, we might need to develop other techniques for characterizing the database schemes bounded w.r.t. fd's. An alternative approach is investigated in the next chapter.

Chapter 4

On Designing Bounded or Ctm Database Schemes

4.1. Introduction

As shown in the previous chapter, under the weak instance model, determining whether a class of database schemes is bounded w.r.t. dependencies is fundamental for the analysis of the behavior of the class of database schemes w.r.t. query processing and updates. However, proving that a class of database schemes is bounded w.r.t. dependencies seems to be very difficult even for our restricted case in Chapter 3.

It is desirable then to develop techniques or to explore other ideas that could help us in characterizing boundedness of database schemes w.r.t. dependencies. In particular the idea of designing bounded database schemes has not been explored at all.

In this chapter, we give a formal methodology for designing database schemes bounded w.r.t. fd's using a new technique called extensibility. This technique can also be used to design ctm database schemes. We do this characterization using the following concept of extensibility from Mendelzon [M]: **S** extends **R** w.r.t. a set of fd's F if for each relation scheme R in ~~**R**, **S** contains a~~ lossless decomposition of R w.r.t. F.

4.2. Overview of Chapter

In Section 4.3, we give some definitions required for this chapter. Then assuming that F and G are sets of equivalent fd's, we prove in Section 4.4 that if **S** extends **R** w.r.t. F and (**S**, F) is bounded, then (**R**, G) is bounded. We also prove that if (**S**, F) is ctm, then (**R**, G) is also ctm. In Section 4.5, we give some sound rules for designing bounded database schemes based on our above mentioned result about boundedness and extensibility. In Section ~~4.6 we use~~ the sound rules from Section 4.5 to give a methodology for designing new classes of bounded or ctm database schemes. In particular we show how to design a large class of bounded and ctm database schemes which are extensible into independent database schemes. The database schemes designed using our methodology are neither restricted to be in some normal form nor to be cover embedding. After that, we give our conclusions in Section 4.7.

4.3. Some Definitions

We now give some definitions required for this chapter.

In this section, we are interested in tableaux and chase rules for ejd's. An ejd can be represented as a tableau T of the form $(t_1, t_2, \ldots, t_k)/t$. In this context, the ejd T is satisfied by an universal relation I if whenever h is a containment mapping from T to I, there is a tuple s in I such that $s[N] = h(t[N])$, where N is the set of nonunique symbols in t.

Let T_r be the tableau for a state of $(\mathbf{R}, F)$. We associate with each ejd $(t_1, t_2, \ldots, t_k)/t$ a rule that transforms T_r into another tableau as follows. Assume there is a containment mapping h from $(t_1, t_2, \ldots, t_k)$ to T_r. An *extension* g of h to t is an assignment to unique symbols in t such that $g(t)$ is well-defined. A *distinct extension* f of h to t is an extension f of h to t such that for all unique symbols d in t, $f(d)$ is a new distinct value that appears no where else in T_r. Now, suppose there is a containment mapping h from $(t_1, t_2, \ldots, t_k)$ to T_r, but for no extension g of h to t we have $g(t)$ is in T_r. Then, we say that the ejd $(t_1, t_2, \ldots, t_k)/t$ is *applicable* to T_r, and the result of *applying* the rule to T_r is a new tableau $T_r' = T_r \bigcup \{f(t)\}$, where f is a distinct extension of h to t.

Now following [GW], we define constant-time-maintainable database schemes. The *maintenance problem* (for database states) of $(\mathbf{R}, F)$ is the following decision problem: Let $\mathbf{r}$ be a consistent state of a database scheme $(\mathbf{R}, F)$ and assume we insert a tuple t in $r_p \in \mathbf{r}$. Is $\mathbf{r}' = \mathbf{r} \bigcup \{t\}$ a consistent state of $(\mathbf{R}, F)$?

We say that $<\mathbf{r}, t>$ above is an *instance* of the maintenance problem of $(\mathbf{R}, F)$.

We assume that r is stored on a device that responds to requests of the form $<R_i, \Phi>$, where $R_i \in \mathbf{R}$ and Φ is a boolean combination of formulas of the form $A = a$, where $A \in R_i$ and $a \in \mathrm{dom}(A)$. The storage device responds to a request by returning, if it exists, a tuple from $r_i \in \mathbf{r}$ that satisfies Φ. In this case the request *succeeds*; otherwise it is said to *fail*.

Suppose there is an algorithm **A** to solve the maintenance problem of (**R**, F). For any instance $<\mathbf{r}, t>$ of the maintenance problem of (**R**, F), we define $\#\mathbf{A}(<\mathbf{r}, t>)$ to be the number of requests, of the above form, made by **A** on the instance $<\mathbf{r}, t>$. We say that **A** solves the maintenance problem of (**R**, F) in *constant time* if there is a constant k, $k \geq 0$, such that $k \geq \#\mathbf{A}(<\mathbf{r}, t>)$, for all instances $<\mathbf{r}, t>$ of the maintenance problem of (**R**, F).

A database scheme (**R**, F) is said to be *constant-time-maintainable (ctm)* if there is an algorithm that solves the maintenance problem of (**R**, F) in constant time.

Independent database schemes are ctm by definition, and from our results in Chapter 3, γ-acyclic BCNF database schemes are ctm as well.

In this chapter we use the following definition of boundedness. A database scheme (**R**, F) is *bounded* (w.r.t. F) if for any $X \subseteq \mathbf{U}$ there exists a predetermined relational expression E_X that computes $[X]$, using only the join, union, and projection operators, for any consistent state **r** of (**R**, F) [GM][MUV].

4.4. Extensibility to Bounded and Ctm Database Schemes

In this section, we prove the results related to extensibility among database schemes on which our design methodology relies.

We first define the concept of extensibility between database schemes. Let (**R**, G) and (**S**, F) be two database schemes defined on $\mathbf{U}'$ and **U** respectively and assume $F^+ = G$. We say that **S** *extends* **R** *(w.r.t. F)*, if for each R in **R** there exists $S =$

$\{S_1, \ldots, S_k\} \subseteq \mathbf{S}$ such that $R = \bigcup_{i=1}^{k} S_i$ and $F \models \bowtie S$ [M]. In this case, we also say that S is an *extension* of R *(w.r.t. F)* or that R is *extensible into* S *(w.r.t. F)*. We denote S extends R as $\mathbf{R} \leq \mathbf{S}$.

Suppose $F \equiv G$ and let (R, G) and (S, F) be such that $\mathbf{R} \leq \mathbf{S}$. We want to prove that if (S, F) is bounded, then (R, G) is bounded, and that if (S, F) is ctm, then (R, G) is ctm. In the next two subsections, we consider first the case when both database schemes are defined on the same universe, i.e., $\bigcup \mathbf{R} = \bigcup \mathbf{S}$. Then we consider the general case when $\bigcup \mathbf{R} \subseteq \bigcup \mathbf{S}$. We conclude this section by showing how to use these results to compute X-total projections and how to enforce constraints in constant time.

4.4.1. Extensibility to Bounded Database Schemes

In this subsection we assume that (R, G) and (S, F) are two database schemes, such that $F \equiv G$, $\mathbf{R} \leq \mathbf{S}$, and $\bigcup \mathbf{R} = \bigcup \mathbf{S}$. We want to prove that if (S, F) is bounded, then (R, G) is bounded.

Given a state r of (R, G), $\mathbf{s_r}$ is the state of (S, F) obtained from r as follows [M]:

- For each $S_i \in \mathbf{S}$, $s_i = \bigcup \{ \pi_{S_i}(r_k) | S_i \subseteq R_k, r_k \in \mathbf{r}, R_k \in \mathbf{R}\}$. s_i is empty for every S_i not contained in any relation scheme of R.

Let $X \subseteq \mathbf{U}$. We shall denote in this subsection the X-total projection of the representative instance for a state r as $[X]_\mathbf{r}$. The following is a key result about total projections computed in r and $\mathbf{s_r}$.

Proposition 4.1: Let r be a consistent state of (R, G) and let $\mathbf{s_r}$ be the state of (S, F) defined above. Then $\mathbf{s_r}$ is consistent, and for any $X \subseteq \mathbf{U}$, $[X]_{\mathbf{s_r}} = [X]_\mathbf{r}$.

Proof: It follows from results in [M]. □

We shall prove in the following subsection that actually $\mathbf{s}_r$ is consistent exactly when $\mathbf{r}$ is consistent.

Theorem 4.1: If (**S**, F) is bounded, then (**R**, G) is bounded.

Proof: Let $\mathbf{r}$ be a consistent state of (**R**, G) and let us consider $\mathbf{s}_r$ the state of (**S**, F) defined above. By Proposition 4.1, $\mathbf{s}_r$ is a consistent state of (**S**, F). Let $X \subseteq \mathbf{U}$. Since (**S**, F) is bounded, there exists a relational expression E_X which computes $[X]_{\mathbf{s}_r}$. Observe that E_X is an expression with operands in **S**.

We now obtain from E_X a relational expression E_X' with operands in **R** which computes $[X]_{\mathbf{s}_r}$. By construction of $\mathbf{s}_r$, either $s_i = \bigcup_{R_j \supseteq S_i} \pi_{S_i}(r_j)$, where $R_j \in \mathbf{R}$, or $s_i = \emptyset$. Then for each operand S_i in E_X do the following: If there is some R_l in **R** such that $R_l \supseteq S_i$, then substitute S_i in E_X by the expression $\bigcup_{R_j \supseteq S_i} \pi_{S_i}(R_j)$; else substitute S_i by the expression $\emptyset$ which we define to evaluate to the empty relation. Let the new expression be E_X'.

But, since by Proposition 4.1, $[X]_{\mathbf{s}_r} = [X]_{\mathbf{r}}$, E_X' computes $[X]_{\mathbf{r}}$. Therefore (**R**, G) is bounded. □

4.4.2. Extensibility to Ctm Database Schemes

In this subsection we assume that (**R**, G) and (**S**, F) are two database schemes such that $F = G$, $\mathbf{R} \leq \mathbf{S}$, and $\bigcup\mathbf{R} = \bigcup\mathbf{S}$. We want to prove that if (**S**, F) is ctm, then (**R**, G) is ctm.

The plan for the proof is the following. Let $\mathbf{r}$ be a state of (**R**, G) and let $\mathbf{s}_r$ be the state of (**S**, F) defined in the previous subsection. Let T_r and T_s be the tableaux for $\mathbf{r}$ and $\mathbf{s}_r$ respectively. We show then that there is a tableau T_s' in a finite-chase of T_s w.r.t. some set of ejd's implied by F, which is equivalent to T_r. We minimize both T_r and T_s' obtaining tableaux T_r^* and T_s^* respectively, which are identical up to renaming of ndv's. Then we prove this implies that if $(\mathbf{S}, F)$ is ctm, then $(\mathbf{R}, G)$ is ctm.

Proposition 4.2: Let T_r and T_s be the tableaux for $\mathbf{r}$ and $\mathbf{s}_r$ respectively. Then there is a containment mapping v: $T_s \rightarrow T_r$.

Proof: Let t be a tuple in T_s originating from $s_i \in \mathbf{s}_r$. By construction of $\mathbf{s}_r$, there is a tuple t' in some $r_k \in \mathbf{r}$, where $S_i \subseteq R_k$, such that $t[S_i] = t'[S_i]$. By construction of T_r, for t' there is some tuple t'' in T_r from r_k such that $t''[R_k] = t'$. v maps t into t''. □

By definition of $\mathbf{R}$ and $\mathbf{S}$, for each $R \in \mathbf{R}$ there exists $S = \{S_1, \ldots, S_k\} \subseteq \mathbf{S}$ such that $R = \bigcup_{i=1}^{k} S_i$ and $F \models \bowtie S$. With $\bowtie S$, or equivalently with R, we associate the ejd-tableau $(s_1, \ldots, s_k)/r$ formed as follows. For $1 \leq j \leq k$, $s_j[A_i]$ is a_i for all A_i in S_j, and s_j consists of unique ndv's in $\mathbf{U} - S_j$. $r[A_i]$ is a_i for all A_i in R, and r consists of unique ndv's in $\mathbf{U} - R$.

Let $F' = \{ \bowtie S \mid F \models \bowtie S,\ S = \{S_1, \ldots, S_k\} \subseteq \mathbf{S}$ such that $R = \bigcup_{i=1}^{k} S_i$ and $R \in \mathbf{R}\}$. Observe that by definition of F', $F = F \bigcup F'$. We want to apply to T_s a finite number of ejd-rules, corresponding to some ejd's in F', to obtain T_s' such that $T_r = T_s'$. Notice that by the previous observation $CHASE_F(T_s) = CHASE_F(T_s')$. In order to prove the equivalence between T_r and T_s', we have to show two containment mappings, one from T_s' to T_r and one from T_r to T_s'. We produce these two mappings by chasing T_s w.r.t. F' in a particular way. While doing that chase, we build incrementally a containment mapping ψ from T_r to T_s' and extend v, the containment mapping from T_s to T_r given by Proposition 4.2 above, to a containment mapping from T_s' to T_r.

Before showing the containment mappings, we need the following definitions first. For each $R_i \in \mathbf{R}$, let $(s_{i_1}, \ldots, s_{i_k})/r_i$ be its associated ejd. Observe that by construction of s_r, for each $t \in T_r$ from r_i there exist t_{i_l} in T_s, $1 \le l \le k$, such that $t_{i_l}[S_{i_l}] = t[S_{i_l}]$ and $t_{i_l}[U - S_{i_l}]$ are ndv's. Then we define the containment mapping h_t from $(s_{i_1}, \ldots, s_{i_k})$ to T_s as follows. For $1 \le l \le k$, $h_t(s_{i_l}) = t_{i_l}$. Now we are ready to show how to build ψ and how to extend v via the following chase of T_s w.r.t. F'.

Let $T_s' = T_s$. For each $R_i \in \mathbf{R}$, for each $t \in T_r$ from r_i, check whether the ejd-rule associated with R_i is applicable to T_s' using h_t, where h_t is the containment mapping defined above. If the ejd-rule is not applicable, i.e., if there exists a tuple t' in T_s' such that $t'[R_i] = t[R_i]$, then $\psi(t) = t'$. On the other hand, if the ejd-rule is applicable, $\psi(t) = t'$, where t' is the tuple added to T_s' by the application of the ejd-rule ($t'[R_i] = t[R_i]$ and $t'[U - R_i]$ consists of unique ndv's); in this case v is extended to t' by defining $v(t') = t$ and $T_s' = T_s' \cup \{t'\}$.

Proposition 4.3: Let T_s' be the outcome from above chase of T_s w.r.t. F'. Then $T_r \equiv T_s'$.

Proof: It follows trivially that at the end of the chase of T_s w.r.t. F' above, we have containment mappings ψ from T_r to T_s', and (an extension of) v, the mapping given by Proposition 4.2, from T_s' to T_r. Therefore the proposition follows from a result in [ASU]. □

Proposition 4.4: Let T_r^* and T_s^* be the minimal tableaux equivalent to T_r and T_s' respectively. Then T_r^* and T_s^* are identical up to renaming of ndv's.

Proof: It follows from Proposition 4.3 and a result in [ASU]. □

We want to derive an algorithm to solve the maintenance problem for $(\mathbf{R}, G)$ in constant time from an algorithm that solves that problem for $(\mathbf{S}, F)$ in constant time. This is going to be a consequence of the following lemma.

Lemma 4.1: Let $\mathbf{r}$ be a state of $(\mathbf{R}, G)$ and let $\mathbf{s}_r$ be the state of $(\mathbf{S}, F)$ defined above. Then $\mathbf{r}$ is a consistent state if and only if $\mathbf{s}_r$ is a consistent state.

Proof: By Proposition 4.4, there exist T_r^* and T_s^* tableaux of **r** and **s**$_r$ respectively such that they are identical up to renaming of ndv's. Then it follows that **r** is consistent w.r.t. G if and only if **s**$_r$ is consistent w.r.t. F. □

The following corollary suggests how to solve the maintenance problem for (**R**, G) in constant time if we know how to solve it for (**S**, F) in constant time.

Corollary 4.1: Assume we insert a tuple t in $r_p \in$ **r**. Let $S_1, \ldots, S_k$ be the elements of **S** such that $S_i \subseteq R_p$, $1 \le i \le k$. Let $t_i = t[S_i]$, $1 \le i \le k$. Then $\mathbf{r}' = \mathbf{r} \bigcup \{t\}$ is a consistent state of (**R**, G) if and only if $\mathbf{s}' = \mathbf{s}_r \bigcup \{t_1, \ldots, t_k\}$ is a consistent state of (**S**, F).

Proof: It follows from Lemma 4.1 with **r**$'$ and **s**$'$ in the roles of **r** and **s**$_r$ respectively. □

Now, we are ready to prove the main claim in this subsection.

Theorem 4.2: If (**S**, F) is ctm, then (**R**, G) is ctm.

Proof: Assume (**S**, F) is ctm. Let **r** be a consistent state of (**R**, G) and let us consider **s**$_r$. Assume we insert a tuple t in $r_p \in$ **r**.

Let $S_1, \ldots, S_k$ be the elements of **S** such that $S_i \subseteq R_p$, $1 \le i \le k$. Let $t_i = t[S_i]$, $1 \le i \le k$. Let $\mathbf{s}_1 = \mathbf{s}_r$ and let us consider solving the following k maintenance problems for (**S**, F); For $1 \le i \le k$, is $\mathbf{s}_{i+1} = \mathbf{s}_i \bigcup \{t_i\}$ consistent w.r.t. F?

Since (**S**, F) is ctm, there exists an algorithm **A** that issues at most q requests, $q \ge 0$, of the form $<S, \Phi>$, where $S \in$ **S**, to solve any of the above k maintenance problems. We want to obtain from **A** an algorithm **A**$'$ that solves the maintenance problem of (**R**, G) issuing a constant number of requests of the form $<R, \Phi'>$, where $R \in$ **R**.

Assume we are solving via **A** the i-th, $1 \le i \le k$, maintenance problem: is $\mathbf{s}_{i+1} = \mathbf{s}_i \bigcup \{t_i\}$ consistent w.r.t. F? By construction of **s**$_r$, for any $s \in \mathbf{s}_r$, either $s = \bigcup_{R_j \supseteq S}$

$\pi_S(r_j)$, where $R_j \in \mathbf{R}$, or $s = \varnothing$. Then for each request $<S, \Phi>$ issued by **A** do the following.

If there is some R in **R** such that $R \supseteq S$, then we translate $<S, \Phi>$ into l requests, $1 \leq l \leq |\mathbf{R}|$, $<R_1, \Phi>, \ldots, <R_l, \Phi>$, where for $1 \leq j \leq l$, $S \subseteq R_j$. If one of these requests succeeds, say the j-th, $1 \leq j \leq l$, and assume the request returns t', a tuple on R_j, then the request $<S, \Phi>$ succeeds and we return $t'[S]$ to **A**. But if all these l requests fail, we cannot yet tell **A** that its request failed. We have to verify if there is some tuple u over S among the tuples $t_1, \ldots, t_{i-1}$. If such an u exists and satisfies Φ, the request $<S, \Phi>$ succeeds and we return u to **A**. Else we return fail to **A**.

On the other hand, if S is not contained in any scheme in **R**, we return fail immediately to **A**.

Let **A'** be the algorithm obtained from **A** by translating **A**'s requests as outlined above. Notice **A'** solves the maintenance problem of (**S**, F) via a constant number of requests of the form $<R, \Phi>$ where $R \in \mathbf{R}$. But by Corollary 4.1, **A'** solves the maintenance problem of (**R**, G) as well. Therefore we can solve the maintenance problem for (**R**, G) using algorithm **A'** in no more than a constant number of requests of the form $<R, \Phi>$, where $R \in \mathbf{R}$.

Hence if (**S**, F) is ctm, then (**R**, G) is ctm. □

4.4.3. The Main Result about Extensibility, Boundedness, and Ctm

We are ready to state and prove our main result in this section.

Corollary 4.2: Suppose $F = G$. Let (**R**, G) and (**S**, F) be database schemes such that $\mathbf{R} \leq \mathbf{S}$ and $\bigcup \mathbf{R} \subsetneq \bigcup \mathbf{S}$. Then

A: If (**S**, F) is bounded, then (**R**, G) is bounded.

B: If (**S**, F) is ctm, then (**R**, G) is ctm.

Proof: Assume $\bigcup \mathbf{R} = \mathbf{U}'$. Since $F \equiv G$ and G is defined on $\mathbf{U}'$, F is defined on $\mathbf{U}'$.

Now consider ($\mathbf{S}'$, F), where $\mathbf{S}' = \{S \in \mathbf{S} \mid S$ is contained in some relation scheme in **R**$\}$. Observe that $\bigcup \mathbf{S}' = \mathbf{U}'$, since $\mathbf{R} \leq \mathbf{S}$. Also notice that if (**S**, F) is bounded, then ($\mathbf{S}'$, F) is bounded, and if (**S**, F) is ctm, then ($\mathbf{S}'$, F) is ctm.

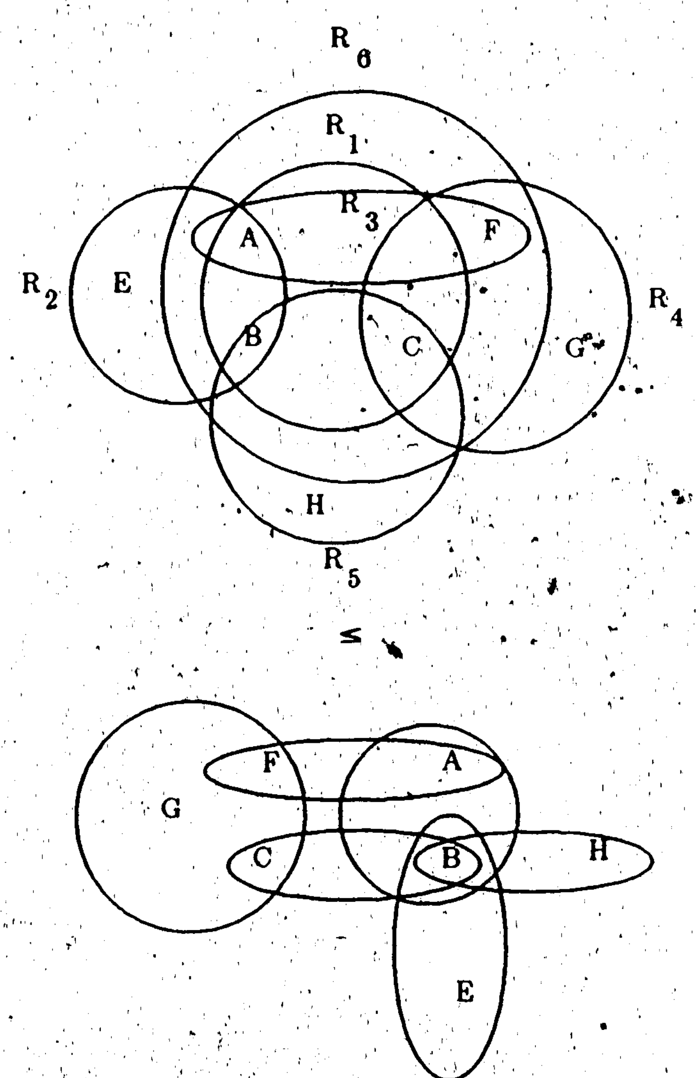
Then the corollary follows from Theorems 4.1 and 4.2, since $\mathbf{R} \leq \mathbf{S}'$. □

4.4.4. Computing Total Projections and Enforcing Fd's

The main purpose of studying boundedness or constant-time-maintainability of database schemes is to find relational expressions to compute total projections or to show how to enforce constraints in constant time. The idea behind our results in this chapter is to do this for a given database scheme by proving that it is extensible into a bounded or ctm database scheme. For if we know how to compute total projections or how to enforce fd's in constant time in the latter database scheme, we can use our above results to obtain relational expressions for computing total projections or a method to enforce fd's in constant time for the former database scheme. We illustrate our results via the following example.

Example 4.1: Let (**R**, F) $=$ ($\{R_1(ABC), R_2(ABE), R_3(AF), R_4(CFG), R_5(BCH), R_6(ABCF)\}$, $\{A \to BC, B \to AH, A \to F, A \to BE, B \to AE, CF \to G, B \to C\}$) be a database scheme. Figure 4.1 below shows it. Observe that (**R**, F) is neither independent nor γ-acyclic. Hence we cannot tell a priori whether (**R**, F) is bounded or ctm. However by Corollary 4.2, (**R**, F) is ctm and bounded since $\mathbf{R} \leq \mathbf{S}$, where (**S**, G) $=$ ($\{AB, BE, AF, BC, BH, CFG\}$, $\{A \to B, B \to A, B \to C, A \to F, B \to E, CF \to G, B \to H\}$) is an independent database scheme and $F \equiv G$. Figure 4.1 also shows **S**.

To compute the X-total projections for databases of (**R**, F), we follow the method



A → BC
B → AH
A → F
A → BE
B → AE
CF → G
B → C

Figure 4.1 **R** and **S** in Example 4.1

in the proof of Theorem 4.1. We now illustrate it. Suppose we want to compute $[EF]$. Using the method in [S2], we obtain the following expression to compute $[EF]$ for databases of (**S**, G): $[EF] = \pi_{EF}(AB \bowtie AF \bowtie BE)$. Since $\mathbf{R} \leq \mathbf{S}$, $AB = \pi_{AB}(R_1) \cup \pi_{AB}(R_2) \cup \pi_{AB}(R_6)$, $AF = R_3 \cup \pi_{AF}(R_6)$, $BE = \pi_{BE}(R_2)$. Then accordingly to Theorem 4.1, $[EF] = \pi_{EF}\,(\,(\pi_{AB}(R_1) \cup \pi_{AB}(R_2) \cup \pi_{AB}(R_6)) \bowtie (R_3 \cup \pi_{AF}(R_6)) \bowtie$

$\pi_{BE}(R_2))$.

To enforce fd's in constant time in databases of $(\mathbf{R}, F)$, we use the method in the proof of Theorem 4.2. We illustrate it now. Assume we have a consistent state $\mathbf{r}$ of $(\mathbf{R}, F)$ and we want to insert a tuple t into $r_6(ABCF)$. According to Theorem 4.2, we have to find first which are the relation schemes in $\mathbf{S}$ which are subsets of R_6. These are AF, AB, and BC. Then we obtain $t_1 = t[AF]$, $t_2 = t[AB]$, and $t_3 = t[BC]$.

Let $\mathbf{s}_1 = \mathbf{s}_r$ and let us consider solving the following maintenance problems for $(\mathbf{S}, G)$: For $1 \leq i \leq 3$, is $\mathbf{s}_{i+1} = \mathbf{s}_i \cup \{t_i\}$ consistent w.r.t. G?

Since $(\mathbf{S}, G)$ is independent, there exists an algorithm $\mathbf{A}$ that issues the following requests:

(a) $<AF, A = t_1[A]>$; to solve the first maintenance problem above. It is issued to check $A \rightarrow F$ in $s(AF) \in \mathbf{s}_r$.

(b) $<AB, A = t_2[A]>$ and $<AB, B = t_2[B]>$; to solve the second maintenance problem above. They are issued to check $A \rightarrow B$ and $B \rightarrow A$ in $s(AB) \in \mathbf{s}_r$.

(c) $<BC, B = t_3[B]>$; to solve the third maintenance problem above. It is issued to check $B \rightarrow C$ in $s(BC) \in \mathbf{s}_r$.

According to Theorem 4.2, these requests get translated respectively

(a') $<R_3, A = t_1[A]>$ and $<R_6, A = t_1[A]>$ to check $A \rightarrow F$ in $s(AF) = \{t_1\} \cup r_3 \cup \pi_{AF}(r_6)$;

(b') $<R_1, A = t_2[A]>$, $<R_2, A = t_2[A]>$, and $<R_6, A = t_2[A]>$ to check $A \rightarrow B$ in $s(AB) = \{t_2\} \cup \pi_{AB}(r_1) \cup \pi_{AB}(r_2) \cup \pi_{AB}(r_6)$; and $<R_1, B = t_2[B]>$, $<R_2, B = t_2[B]>$, and $<R_6, B = t_2[B]>$ to check $B \rightarrow A$ in $s(AB)$;

(c') $<R_1, B = t_3[B]>$, $<R_5, B = t_3[B]>$, and $<R_6, B = t_3[B]>$, to check $B \rightarrow C$ in $s(BC) = \{t_3\} \cup \pi_{BC}(r_1) \cup \pi_{BC}(r_5) \cup \pi_{BC}(r_6)$.

Observe that no more than a constant number of requests of the form $<R, \Phi>$, where $R \in \mathbf{R}$, are required to solve the maintenance problem of $(\mathbf{R}, F)$.

It should be clear that proving these two properties for a database scheme like $(\mathbf{R}, F)$ by some other known techniques is not an easy task. □

It is obvious that for a given $\mathbf{R}$ there may be more that one extension of it. For instance, Figures 4.3 and 4.5 in Section 4.6 below show two distinct extensions of $\{ABC, ABE, AF, ABCF\}$ w.r.t. $\{A \to BC, B \to AC, A \to F\}$.

In the following sections we are going to give a methodology for designing bounded or ctm database schemes. We develop the methodology in terms of bounded schemes, but from the results in this section the methodology applies to the design of ctm database schemes as well. In Example 4.10 in Section 4.6, we show how to design $(\mathbf{R}, F)$ in Example 4.1 above starting from $\mathbf{D}_1 = (\{R_1(ABC)\}, \{A \to BC, B \to AC\})$.

4.5. Sound Rules for Designing Bounded Database Schemes

In this section we define our framework for designing bounded database schemes. The idea is to start with a bounded database scheme and then incrementally enrich it by adding or deleting relation schemes or fd's from it. We can add or delete a relation scheme or an fd if the resultant database scheme is bounded, that is, if boundedness is preserved.

We now define what we mean by designing. Let $\mathbf{D}_i = (\mathbf{R}_i, F_i)$ and $\mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1})$ be two database schemes.

We say that $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is an *incremental (design) step* if either

- $\mathbf{R}_{i+1} = \mathbf{R}_i \bigcup \{R_j\}$, for some relation scheme $R_j \notin \mathbf{R}_i$, and $F_{i+1} = F_i$; or
- $\mathbf{R}_{i+1} = \mathbf{R}_i$ and $F_{i+1} = F_i \bigcup \{f_j\}$, where f_j is an fd defined on $\bigcup \mathbf{R}_i$, and $f_j \notin F_i$.

We say that $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is a *decremental (design) step* if either

- $\mathbf{R}_{i+1} = \mathbf{R}_i - \{R_j\}$, for some $R_j \in \mathbf{R}_i$, but $\mathbf{R}_{i+1}$ is such that $F_{i+1} = F_i$ is defined on $\bigcup \mathbf{R}_{i+1}$; or

- $\mathbf{R}_{i+1} = \mathbf{R}_i$ and $F_{i+1} = F_i - \{f_j\}$, for some $f_j \in F_i$.

Also we say that $<\mathbf{D}_1, \ldots, \mathbf{D}_m>$ is a *design sequence* (w.r.t. $\mathbf{D}_1$) if $<\mathbf{D}_i, \mathbf{D}_{i+1}>$, $1 \leq i \leq m-1$, is an incremental or decremental step.

Although in a design step we are not allowed to add or delete a relation scheme and an fd at the same time, this is not a restriction since this can be accomplished with two steps. Also notice that we are not allowed to delete a relation scheme if its deletion removes some attributes from the universe and some nontrivial fd's are defined on those attributes. The following example illustrates this design restriction.

Example 4.2: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(AC), R_3(BC), R_4(ACD)\}, \{A \rightarrow B, C \rightarrow B, A \rightarrow D, D \rightarrow C\})$. We are not allowed to delete $R_4(ACD)$ from $\mathbf{R}$, since then D is no longer an element of the new universe of attributes and $A \rightarrow D$ and $D \rightarrow C$ are nontrivial fd's defined on D. □

Following [DM], we define the following concepts. A *design rule* is a predicate that defines a set of valid steps in a design sequence w.r.t. an initial database scheme. Let $\mathbf{P}$ be a set of design rules. We denote by $Adm(\mathbf{P})$ the set of admissible database schemes that can be designed via design sequences whose steps are valid w.r.t. at least a rule in $\mathbf{P}$. Let $\mathbf{C}$ be a class of database schemes. $\mathbf{P}$ is *sound* w.r.t. $\mathbf{C}$ if Adm($\mathbf{P}$) $\subseteq$ $\mathbf{C}$.

Let $\mathbf{B}$ be the class of database schemes bounded w.r.t. fd's. We are interested in finding design rules that are sound w.r.t. $\mathbf{B}$. We shall start with $\mathbf{D}_1$, a database scheme bounded w.r.t. fd's. Typically $\mathbf{D}_1$ contains only one relation scheme, but not necessarily so as long as $\mathbf{D}_1$ can be proven to be bounded. Then we can add (or delete) a relation scheme or an fd to (or from) $\mathbf{D}_i$, $1 \leq i \leq m-1$, obtaining $\mathbf{D}_{i+1}$, if $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ satisfies a design rule that guarantees $\mathbf{D}_{i+1}$ to be bounded.

The plan for the rest of this section is as follows. In Section 4.5.1, we give some examples to show the design steps that do not preserve boundedness in general. In Section 4.5.2, we give some general design rules that are sound w.r.t. **B** which follow from Corollary 4.2, our result about extensibility and boundedness. We are going to use those rules in Section 4.6 to give a methodology for designing bounded or ctm database schemes. As mentioned in the Introduction of this chapter, it is worth to point out that the class of database schemes designed in this chapter is neither restricted to be in some normal form nor to be cover embedding.

4.5.1. Design Steps and Soundness w.r.t. B

We now show via some examples that, with the exception of the step that deletes a relation scheme from a bounded database scheme, the design steps do not preserve boundedness in general. This shall give us some intuition about the difficulty of finding general design rules that are sound w.r.t. **B**.

Let us consider first a step that adds a relation scheme to a bounded database scheme.

Example 4.3: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(BC)\}, \{A \rightarrow B, C \rightarrow B\})$ be a database scheme. Clearly, $(\mathbf{R}, F)$ is bounded. Let us add $R_3(AC)$ to $\mathbf{R}$. But from Example 2.3, we have that $(\{R_1(AB), R_2(BC), R_3(AC)\}, F)$ is unbounded. □

Now we consider a step that adds an fd to a bounded database scheme.

Example 4.4: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(BC), R_3(AC)\}, \{A \rightarrow B\})$ be a database scheme. It is easy to see that $(\mathbf{R}, F)$ is bounded. Let us add $C \rightarrow B$ to F. However from Example 2.3, $(\{R_1(AB), R_2(BC), R_3(AC)\}, \{C \rightarrow B, A \rightarrow B\})$ is unbounded. □

Now let us consider a step that deletes an fd from a bounded database scheme.

Example 4.5: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(BC), R_3(AC)\}, \{A \rightarrow B, C \rightarrow B, A \rightarrow C\})$. It is not difficult to prove $(\mathbf{R}, F)$ is bounded. Now let us delete $A \rightarrow C$ from F. However from Example 2.3, $(\{R_1(AB), R_2(BC), R_3(AC)\}, \{C \rightarrow B, A \rightarrow B\})$ is unbounded. □

We prove in the following subsection that a design step that deletes a relation scheme from a bounded database scheme is sound w.r.t. $\mathbf{B}$.

4.5.2. Sound Design Rules based on Extensibility

From the examples in the previous subsection, it should be clear that it is not a trivial task to find general design rules which are sound w.r.t. $\mathbf{B}$; simply because a bounded database scheme is too sensitive to any change to either its schemes or fd's. However Corollary 4.2 suggests some general rules for designing bounded database schemes. Corollary 4.2 says that in a step of a design sequence we are allowed to add or delete a relation scheme or an fd from a bounded database scheme as long as we can prove that there exists a bounded database scheme which extends the resultant database scheme and such that their fd's are equivalent. In this subsection, we list some rules suggested by this result.

We consider first a rule that allows us to add a relation scheme to a bounded database scheme.

$P_{add\_rel}(\mathbf{D}_i = (\mathbf{R}_i, F_i), \mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1}))$:

- $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is an incremental step such that $\mathbf{R}_{i+1} = \mathbf{R}_i \cup \{R_j\}$, for some $R_j \notin \mathbf{R}_i$, and $F_i = F_{i+1}$.
- There exists $S = \{S_1, \ldots, S_k\} \subseteq \mathbf{R}_i$ such that $R_j = \bigcup_{l=1}^{k} S_l$.
- $F_i \models \bowtie S$.

Lemma 4.2: Assume that $\mathbf{D}_1 = (\mathbf{R}_1, F_1) \in \mathbf{B}$. Then $\mathbf{P} = \{P_{add\_rel}\}$ is sound w.r.t. $\mathbf{B}$ for design sequences w.r.t. $\mathbf{D}_1$.

Proof: Let $<\mathbf{D}_1, \ldots, \mathbf{D}_m>$ be a design sequence whose steps are valid w.r.t. P_{add_rel}. By definition of P_{add_rel}, for $1 \le i \le m-1$, $\mathbf{R}_{i+1} \le \mathbf{R}_i$ and $F_i = F_{i+1}$. Hence, for $1 \le i \le m-1$, if $\mathbf{D}_i$ is bounded, then the fact that $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ satisfies P_{add_rel} and Corollary 4.2 imply $\mathbf{D}_{i+1}$ is bounded. □

Example 4.6: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(AC)\}, \{A \to B, A \to C\})$ be a database scheme. It is not difficult to see that $(\mathbf{R}, F)$ is bounded. P_{add_rel} allows us to add $R_3(ABC)$ to $\mathbf{R}$, and hence $(\{R_1(AB), R_2(AC), R_3(ABC)\}, F)$ is bounded. □

The following rule is concerned with subsets of bounded database schemes. It says that we are allowed to remove a relation scheme from a bounded scheme as long as the set of attributes on which the fd's are defined is not affected.

$P_{delete\_rel}(\mathbf{D}_i = (\mathbf{R}_i, F_i), \mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1}))$:

- $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is a decremental step such that $\mathbf{R}_{i+1} = \mathbf{R}_i - \{R_j\}$, for some $R_j \in \mathbf{R}_i$, but $\mathbf{R}_{i+1}$ is such that $F_{i+1} = F_i$ is defined on $\bigcup \mathbf{R}_{i+1}$.

Lemma 4.3: Assume that $\mathbf{D}_1 = (\mathbf{R}_1, F_1) \in \mathbf{B}$. Then $\mathbf{P} = \{P_{delete\_rel}\}$ is sound w.r.t. $\mathbf{B}$ for design sequences w.r.t. $\mathbf{D}_1$.

Proof: By a similar argument as the one in Lemma 4.2. □

Example 4.7: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(BC), R_3(CD)\}, \{A \to B, B \to C, C \to D\})$. It is clear that $(\mathbf{R}, F)$ is bounded. For this database scheme, P_{delete_rel} allows us to delete R_2 from $\mathbf{R}$. Hence, $(\{R_1, R_3\}, F)$ is bounded; observe it is not cover embedding. □

We want to allow, under certain conditions, to delete an fd from a bounded database scheme. To be able to give a sound rule w.r.t. $\mathbf{B}$ in this case, we have to think in terms of designing database schemes which are extensible into a known class of bounded database schemes. The idea is to be able to prove boundedness of the resultant database scheme in an easy way by making use of the property that the class of independent schemes and, under some condition, the class of γ-acyclic BCNF schemes preserve boundedness under fd-deletion. That is, if we delete an fd from an independent scheme, the resultant scheme is independent. And deleting fd's from a γ-acyclic BCNF scheme preserves boundedness, if BCNF is preserved. The following example illustrates this.

Example 4.8: Let $(\mathbf{R}, F)$ and $(\mathbf{S}, G)$ be the database schemes in Example 4.1. We know from Example 4.1 that $(\mathbf{R}, F)$ is bounded since $\mathbf{R} \leq \mathbf{S}$, $(\mathbf{S}, G)$ is an independent database scheme, and $F \equiv G$.

We claim that we can remove $B \to C$ from F and still have $(\mathbf{R}, F' = F - \{B \to C\})$ is bounded. This is true because $\mathbf{R} \leq \mathbf{S}$ holds w.r.t. F'. But more importantly, since removing an fd from an independent scheme preserves independence, and therefore boundedness, $(\mathbf{S}, G' = G - \{B \to C\})$ is bounded and $G' \equiv F'$. □

The following rule formalizes the main idea illustrated in the above discussion and example.

$P_{delete\_fd}(\mathbf{D}_i = (\mathbf{R}_i, F_i), \mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1}))$:

- $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is a decremental step such that $\mathbf{R}_i = \mathbf{R}_{i+1}$ and $F_{i+1} = F_i - \{f_j\}$, for some $f_j \in F_i$.
- There exists a database scheme $(\mathbf{S}_i, G_i) \in \mathbf{B}$ such that $\mathbf{R}_i \leq \mathbf{S}_i$ and $F_i \equiv G_i$.
- $(\mathbf{S}_{i+1} = \mathbf{S}_i, G_{i+1} = F_{i+1}) \in \mathbf{B}$ and $\mathbf{R}_{i+1} \leq \mathbf{S}_{i+1}$.

We now prove this rule is sound w.r.t. $\mathbf{B}$.

Lemma 4.4: Assume that $\mathbf{D}_1 = (\mathbf{R}_1, F_1')$ is a database scheme. Then $\mathbf{P} = \{P_{delete\_fd}\}$ is sound w.r.t. $\mathbf{B}$ for design sequences w.r.t. $\mathbf{D}_1$.

Proof: It suffices to observe that in a step the rule preserves, after removing the fd, extensibility between $\mathbf{R}_i$ and $\mathbf{S}_i$, boundedness of $\mathbf{S}_i$, and equivalence between the sets of fd's. □

Now let us consider the general case of adding a relation scheme or an fd to a database scheme which is extensible into a bounded database scheme. The following example illustrates when we want to allow an incremental step in this case.

Example 4.9: Let $(\mathbf{R}, F)$ and $(\mathbf{S}, G)$ be the database schemes in Example 4.1. We know from Example 4.1 that $(\mathbf{R}, F)$ is bounded since $\mathbf{R} \leq \mathbf{S}$, $(\mathbf{S}, G)$ is an independent database scheme, and $F = G$.

Assume we want to add $R_7(BCI)$ to $\mathbf{R}$. We want to allow such an incremental step if, as for instance, $(\mathbf{S}' = \mathbf{S} \bigcup \{R_7\}, G)$ is bounded. Observe that $(\mathbf{S}', G)$ is not independent, and therefore we do not know whether is bounded. However, $(\mathbf{S}'' = \mathbf{S} \bigcup \{BI\}, G)$ is independent and $\mathbf{S}''$ is an extension of $\mathbf{S}'$. Hence we can add R_7 to $\mathbf{R}$ and still have a bounded database scheme, since $\mathbf{R} \bigcup \{R_7\} \leq \mathbf{S}''$. □

The following rule, $P_{add\_gen(eral)}$, states in general the conditions under which we can add a relation scheme or an fd. It is a generalization of the above example.

$P_{add\_gen}(\mathbf{D}_i = (\mathbf{R}_i, F_i), \mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1}))$:

- $<\mathbf{D}_i, \mathbf{D}_{i+1}>$ is an incremental step.
- There exists a database scheme $(\mathbf{S}_i, G_i) \in \mathbf{B}$ such that $\mathbf{R}_i \leq \mathbf{S}_i$ and $F_i = G_i$.
- If some R_j is added to $\mathbf{R}_i$, then let $\mathbf{S}_i' = \mathbf{S}_i \bigcup \{R_j\}$ and $G_i' = G_i$; otherwise if some f_j is added to F_i, then let $\mathbf{S}_i' = \mathbf{S}_i$ and $G_i' = G_i \bigcup \{f_j\}$.
- There exists a database scheme $(\mathbf{S}_{i+1}, G_{i+1}) \in \mathbf{B}$ such that $\mathbf{S}_i' \leq \mathbf{S}_{i+1}$ and $G_{i+1} = G_i'$.

As shown in the above example, this rule allows us to reduce the problem of proving boundedness for $\mathbf{D}_{i+1}$ to the hopefully easier problem of telling whether $(\mathbf{S}_i, G_i)$ plus the increment has an extension belonging to a class of known bounded database schemes. (This rule subsumes P_{add_rel}, but we keep both rules since their motivations are different.) We now prove its soundness.

Lemma 4.5: Assume that $\mathbf{D}_1 = (\mathbf{R}_1, F_1)$ is a database scheme. Then $\mathbf{P} = \{P_{add\_gen}\}$ is sound w.r.t. **B** for design sequences w.r.t. $\mathbf{D}_1$.

Proof: Let $<\mathbf{D}_1, \ldots, \mathbf{D}_m>$ be a design sequence whose steps are valid w.r.t. P_{add_gen}. Since extensibility is a transitive relationship, it is not difficult to see that for $1 \leq i \leq m-1$, $\mathbf{R}_{i+1} \leq \mathbf{S}_{i+1}$ and $F_{i+1} \equiv G_{i+1}$, if $\mathbf{R}_i \leq \mathbf{S}_i$ and $F_i \equiv G_i$. □

In the following section we give a methodology for designing bounded or ctm database schemes using the rules presented in this section.

4.6. Designing Bounded or Ctm Database Schemes

Methodology 1, shown below, describes a methodology for designing bounded database schemes that uses the rules described above. Methodology 1 is also a methodology for designing ctm database schemes; simply take **B** as the class of ctm database schemes and then by Corollary 4.2 the methodology's output is a ctm database scheme. Its correctness follows from the proofs of soundness for the design rules.

When we use Methodology 1 for designing bounded schemes, the methodology requires that at each step we find a bounded database scheme $(\mathbf{S}_{i+1}, G_{i+1})$ which satisfies one of the design rules in the methodology. For the first three cases in Methodology 1 this is trivial. However, for Case 4 finding such a database scheme is as difficult as proving boundedness in general. Thus to make our methodology feasible, we have to choose from the classes of known bounded database schemes, the class to which $(\mathbf{S}_i, G_i)$ in Methodology 1 belongs. Also this should hopefully make easier the task of proving boundedness, as mentioned above in the motivation for the design rule of Case 4. To illustrate this point and the use of our methodology, we have chosen to give an example of designing a bounded database scheme which is extensible into independent database schemes. However, it should be pointed out that this methodology produces database schemes which may be non-independent.

Methodology 1

Input: $\mathbf{D}_1 = (\mathbf{R}_1, F_1)$ a database scheme such that there exists $(\mathbf{S}_1, G_1) \in \mathbf{B}$, $\mathbf{R}_1 \leq \mathbf{S}_1$, and $F_1 \equiv G_1$.

Output: $\mathbf{D}_m = (\mathbf{R}_m, F_m)$ and $(\mathbf{S}_m, G_m) \in \mathbf{B}$ such that $\mathbf{R}_m \leq \mathbf{S}_m, F_m \equiv G_m;\ m \geq 1$.

Comments: At each step, we compute $(\mathbf{S}_{i+1}, G_{i+1}) \in \mathbf{B}$ such that $\mathbf{R}_{i+1} \leq \mathbf{S}_{i+1}$ and $F_{i+1} \equiv G_{i+1}$.

Method: Use design rules described in text.

(1) Let $i = 1$

(2) **while** *desired* and *possible* **do**
begin

Obtain $\mathbf{D}_{i+1} = (\mathbf{R}_{i+1}, F_{i+1})$ and $(\mathbf{S}_{i+1}, G_{i+1})$ from $\mathbf{D}_i = (\mathbf{R}_i, F_i)$ and $(\mathbf{S}_i, G_i)$ by one of the following:

(3) *Case* 1: Apply R_{add_rel} to add some R_j to $\mathbf{R}_i$;
$\mathbf{S}_{i+1} = \mathbf{S}_i;\ G_{i+1} = G_i$.

(4) *Case* 2: Apply P_{delete_rel} to delete some R_j from $\mathbf{R}_i$;
$\mathbf{S}_{i+1} = \mathbf{S}_i;\ G_{i+1} = G_i$.

(5) *Case* 3: Apply P_{delete_fd} to delete some f_j from F_i;
$\mathbf{S}_{i+1} = \mathbf{S}_i;\ G_{i+1} = F_{i+1}$.

(6) *Case* 4: Apply P_{add_gen} to add R_j to $\mathbf{R}_i$ or add f_j to F_i;
Compute $\mathbf{S}_{i+1}$ and G_{i+1} according to rule.

(7) $i = i+1$

(8) **end**

(9) *Output* $\mathbf{D}_i$ and $(\mathbf{S}_i, G_i)$

As in that example, if the target of the extension of the database scheme being designed is also a ctm database scheme, then by Corollary 4.2 the designed database scheme is ctm as well. Since the idea is to design database schemes which are extensible into one of the known classes of database schemes, and these are ctm, then in this case the output from Methodology 1 is a ctm database scheme too. In this sense, Methodology 1 is a methodology for designing bounded and ctm database schemes.

Example 4.10: We show how to use Methodology 1 for designing bounded database schemes when for $1 \le i \le m$, $(\mathbf{S}_i, G_i)$ in Methodology 1 is an independent database scheme.

Assume we start with $\mathbf{D}_1 = (\{R_1(ABC)\}, \{A \to BC, B \to AC\})$; then $(\mathbf{S}_1, G_1)$ required by Methodology 1 is $\mathbf{D}_1$ itself. Now using P_{add_gen}, we can add $R_2(AF)$ to $\mathbf{D}_1$ to obtain $\mathbf{D}_2 = (\{R_1(ABC), R_2(AF)\}, \{A \to BC, B \to AC\})$; $(\mathbf{S}_2, G_2) = \mathbf{D}_2$, since $\mathbf{D}_2$ itself is independent. Again using P_{add_gen}, we can add $A \to F$ to F_2 to obtain $\mathbf{D}_3 = (\mathbf{R}_3, F_3) = (\{R_1(ABC), R_2(AF)\}, \{A \to BC, B \to AC, A \to F\})$; $(\mathbf{S}_3, G_3) = \mathbf{D}_3$. Now using P_{add_rel}, we can add $R_3(ABCF)$ to $\mathbf{D}_3$ to obtain $\mathbf{D}_4 = (\mathbf{R}_4, F_4) = (\{R_1(ABC), R_2(AF), R_3(ABCF)\}, \{A \to F, A \to BC, B \to AC\})$ and $(\mathbf{S}_4, G_4) = (\mathbf{S}_3, G_3)$. Figure 4.2 below shows $\mathbf{R}_4$ and $\mathbf{S}_4$.

Assume now we want to add to $\mathbf{D}_4$ the scheme $R_4(ABE)$. However $(\mathbf{S}_4 \cup \{R_4\}, G_4)$ is not independent; because $A \to B$ and $B \to A$ are embedded in both ABE and ABC. To use P_{add_gen}, we have to look for an independent, fd-preserving extension of $\mathbf{S}_4 \cup \{R_4\}$. We have to use $A \to B$ or $B \to A$ to look for such a lossless decomposition.

Assume we pick $A \to B$ to try to find such a decomposition. Then $\mathbf{S}_5 = \{AF, AE, AB, AC\}$ is a lossless decomposition of $\mathbf{S}_4 \cup \{R_4\}$ and this decomposition embeds a cover $G_5 = \{A \to F, A \to B, A \to C, B \to A\}$ of G_4, which we know is equivalent to F_4. Also $(\mathbf{S}_5, G_5)$ is independent. Then $\mathbf{D}_5 = (\mathbf{R}_5, F_5) = (\{R_1, \ldots, R_4\}, F_4)$ is a valid database scheme w.r.t. P_{add_gen}. Figure 4.3 below depicts the database schemes at this point.

Now using P_{add_gen} twice in a row, we add $R_5(CFG)$ and $CF \to G$ to $\mathbf{D}_5$ to produce $\mathbf{D}_7 = (\mathbf{R}_7, F_7) = (\mathbf{R}_5 \cup \{CFG\}, F_5 \cup \{CF \to G\})$; observe $\mathbf{S}_7 = \mathbf{S}_5 \cup \{CFG\}$ and $G_7 = G_5 \cup \{CF \to G\}$ satisfy $\mathbf{R}_7 \le \mathbf{S}_7$, $F_7 = G_7$, and $(\mathbf{S}_7, G_7)$ is independent. Figure 4.4 below shows this.

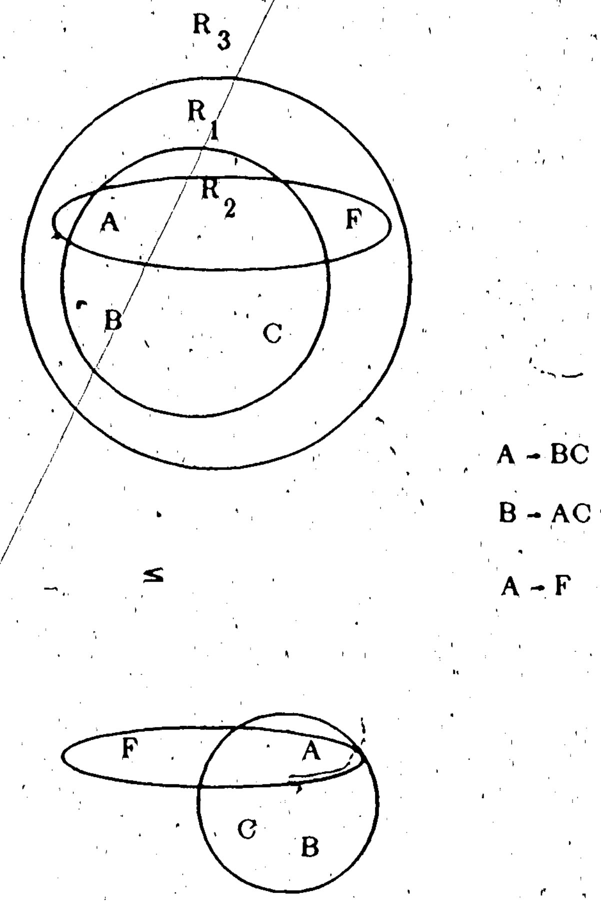


Figure 4.2 $\mathbf{R}_4$ and $\mathbf{S}_4$ in Example 4.10

Now assume we want to add $R_6(BCH)$ to $\mathbf{D}_7$. This is not possible since there is no lossless decomposition of $\mathbf{S}_7 \cup \{BCH\}$ which is independent; C can be derived in more than one way in AB's closure.

Let us see what happens if we choose $B \rightarrow A$ rather than $A \rightarrow B$ to find a lossless, fd-preserving, independent decomposition, that satisfies P_{add_gen}, when adding R_4 to $\mathbf{D}_4$. In this case, the following is such a decomposition: $(\mathbf{S}_5, G_5) = (\{AF, AB, BC, BE\}, \{A \rightarrow F, A \rightarrow B, B \rightarrow A, B \rightarrow C\})$; see Figure 4.5 below.

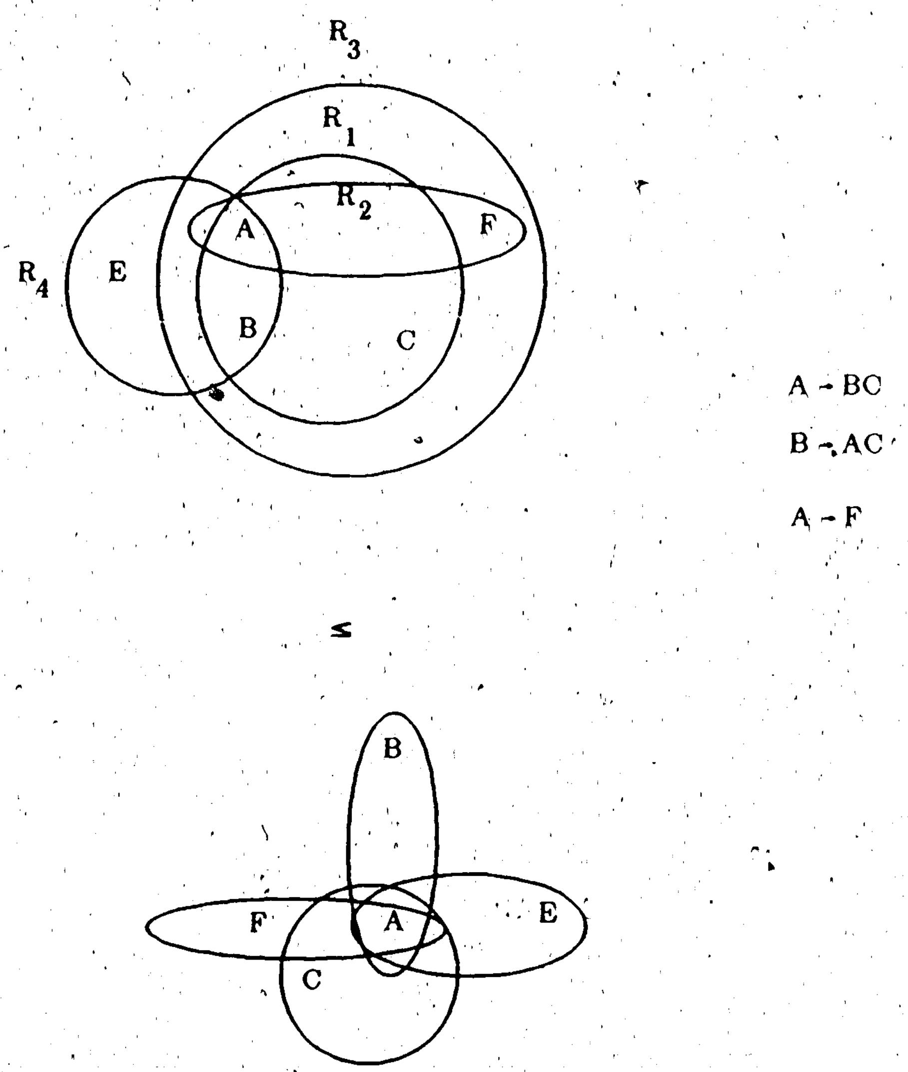


Figure 4.3 $\mathbf{R}_5$ and $\mathbf{S}_5$ in Example 4.10

Now using P_{add_gen} twice in a row, we can add $R_6(CFG)$ and the fd $CF \to G$ to obtain $\mathbf{D}_7 = (\mathbf{R}_7, F_7) = (\mathbf{R}_5 \cup \{R_6\}, F_5 \cup \{CF \to G\})$; $\mathbf{S}_7 = \mathbf{S}_5 \cup \{CFG\}$ and $G_7 = G_5 \cup \{CF \to G\}$. Observe that $(\mathbf{S}_7, G_7)$ is independent. Figure 4.6 below shows $\mathbf{D}_7$ and $\mathbf{S}_7$.

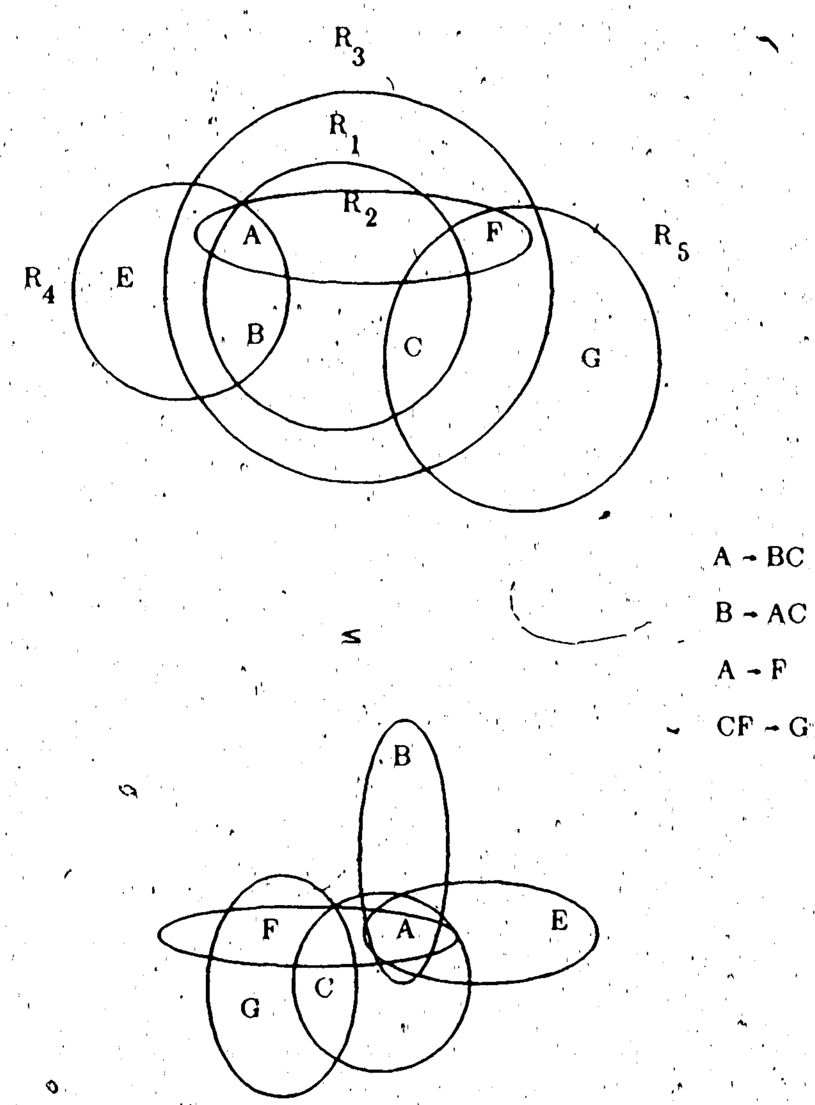


Figure 4.4 $\mathbf{R}_7$ and $\mathbf{S}_7$ in Example 4.10

Now let us consider adding $R_6(BCH)$ to $\mathbf{R}_7$. A decomposition of $\mathbf{S}_7 \cup \{BCH\}$ that satisfies P_{add_gen} is $(\mathbf{S}_8, G_8) = (\{AF, AB, BC, BE, BH, CFG\}, \{A \to F, A \to B, B \to A, B \to C, CF \to G\})$; it is independent and it extends $\mathbf{D}_8 = (\mathbf{R}_8, F_8) = (\mathbf{R}_7 \cup \{R_6\}, F_7)$ and $F_8 = G_8$; Figure 4.7 below shows this.

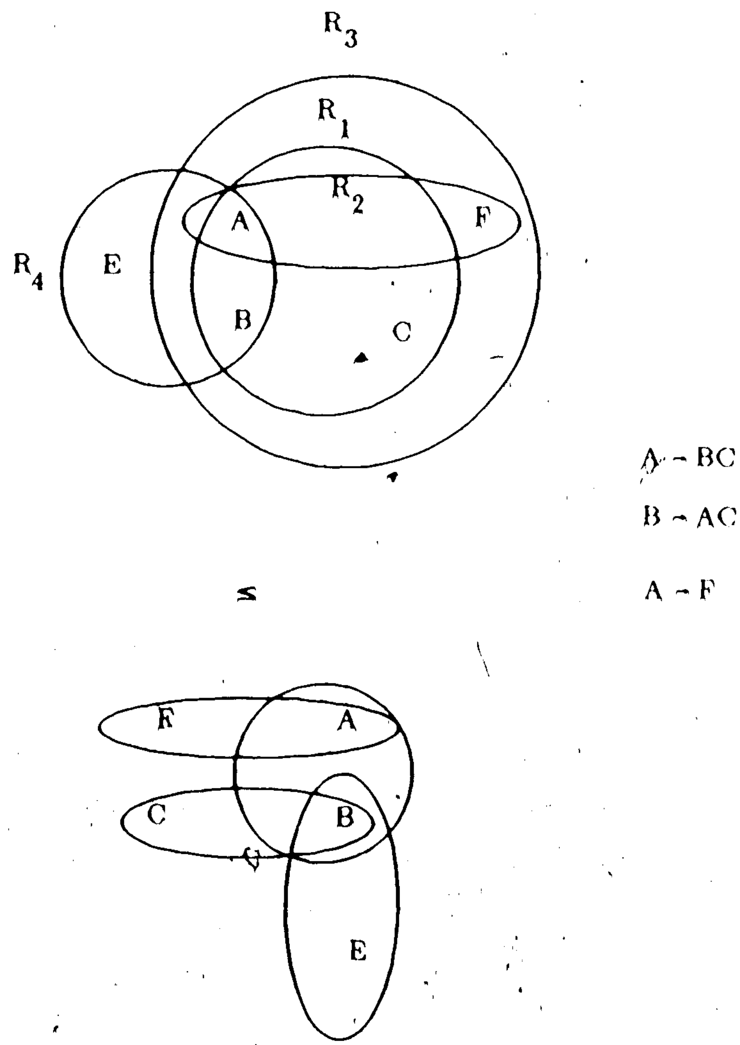


Figure 4.5 $\mathbf{R}_5$ and $\mathbf{S}_5$ in Example 4.10

Now using P_{add_gen} we can add $B \to EH$ to F_8 to obtain $\mathbf{D}_9 = (\mathbf{R}_8, F_8 \cup \{B \to EH\})$, which is (equivalent to) the database scheme in Example 4.1. □

Observe that the class of bounded database schemes we are designing are neither BCNF γ-acyclic nor independent nor cover embedding. It should be clear that proving boundedness or constant-time-maintainability for such a class by the techniques

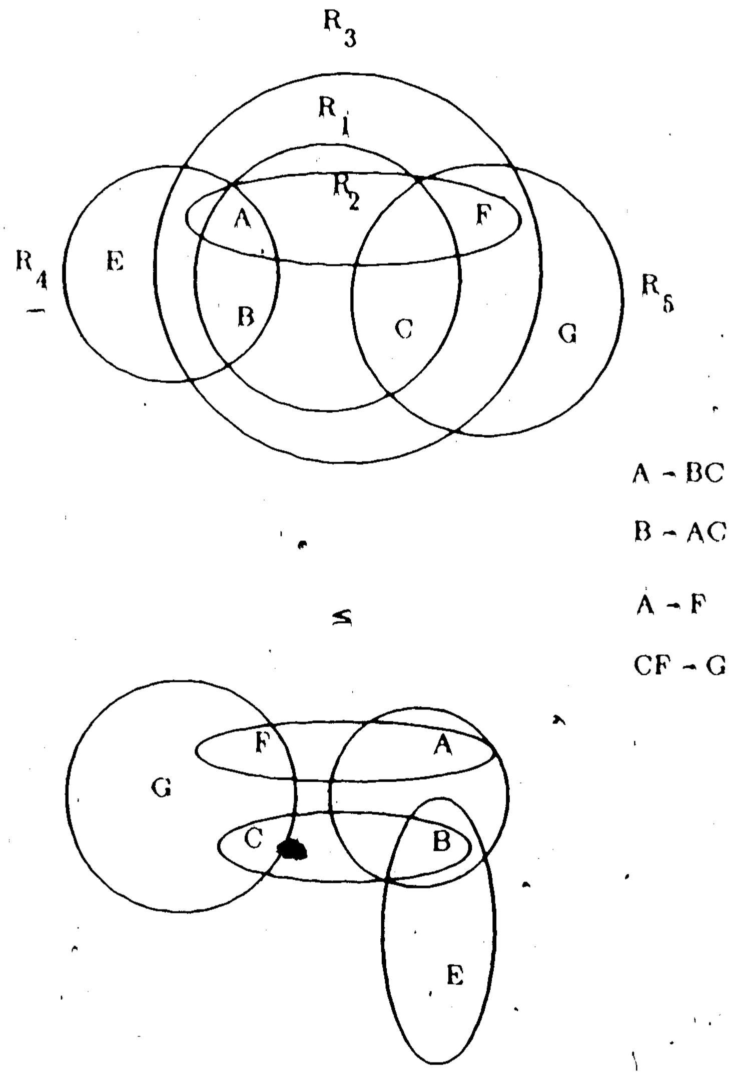


Figure 4.6 $\mathbf{R}_7$ and $\mathbf{S}_7$ in Example 4.10

known so far is a difficult problem.

It must be clear that for database schemes designed using Methodology 1, we can compute its X-total projections using the method in the proof of Theorem 4.1. Also, if the database scheme $(\mathbf{S}_m, G_m)$ output by Methodology 1 is ctm, we can use the method outlined in the proof of Theorem 4.2 to solve the maintenance problem of $(\mathbf{R}_m,$

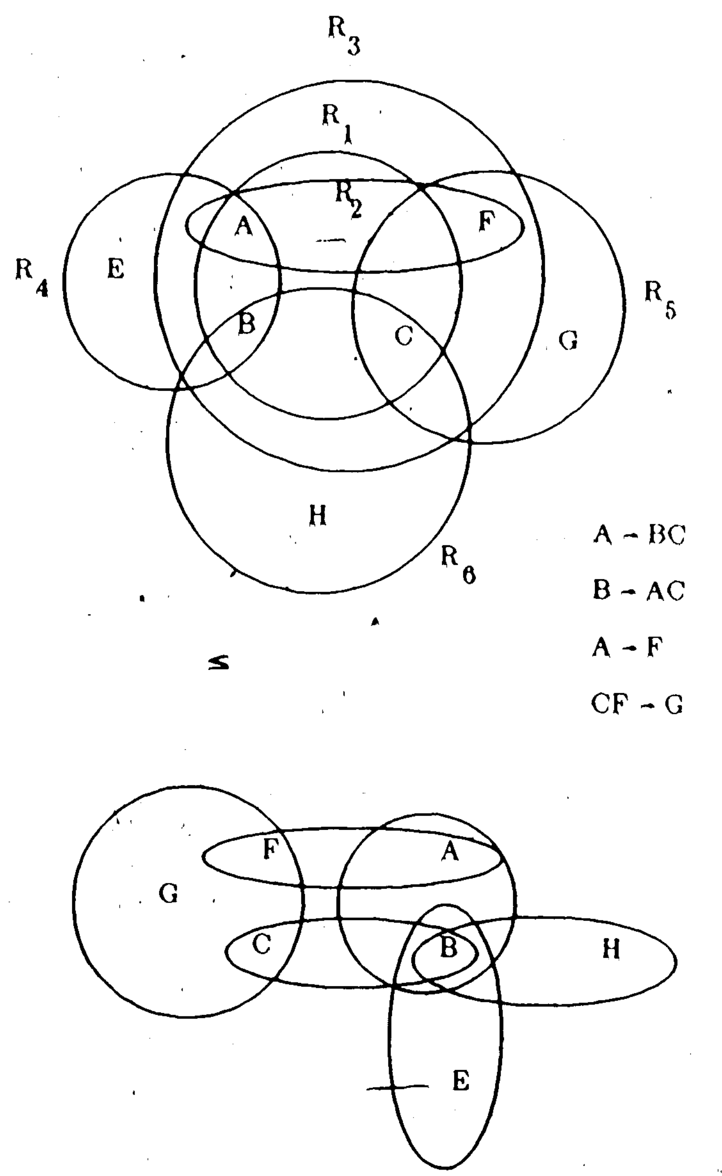


Figure 4.7 $\mathbf{R}_8$ and $\mathbf{S}_8$ in Example 4.10

F_m) in constant time. In Example 4.1, we have shown how to do that for the database scheme in Example 4.10.

There are of course some deficiencies in our methodology. Some of them come from the decomposition approach we have taken. The others come from the definition of the design process. For instance, among the former deficiencies, using this methodology we could not design the class of γ-acyclic BCNF database schemes starting from the class of independent database schemes. Among the latter deficiencies, we can see that the restriction of adding or deleting only an fd from a database scheme makes difficult to preserve a property like BCNF between schemes in a design step.

4.7. Conclusions

We showed that a database scheme is bounded w.r.t. fd's if it is extensible into a bounded one. Also we showed that if a database scheme is extensible into a ctm database scheme, then it is ctm. We showed how to apply these results to compute total projections or to enforce fd's in constant time for database schemes proven to be bounded or ctm using our results.

Then using these results we presented a formal methodology for designing database schemes bounded or ctm w.r.t. fd's using a new technique called extensibility. The major advantage of this technique is its iterative nature. We can apply this techni[illegible] a known class of bounded or ctm database schemes and generate other class of bounded or ctm database schemes. For instance, we showed how to design a new class of bounded and ctm database schemes which are neither acyclic schemes nor independent.

Our proposed methodology for designing bounded or ctm database schemes brings new insight in how to design database schemes under the weak instance model. This is clearly very helpful since cost-effective query processing or cost-effective constraint enforcement are highly-desirable properties of any database and these are possible only if a database scheme is bounded or ctm.

Chapter 5

Testing Unboundedness of Database Schemes and Fd's

5.1. Introduction

There are two variants of the notion of boundedness. One of these is boundedness of database schemes w.r.t. dependencies, which is the one we have been referring to in this thesis so far. And the other one is the concept of boundedness of a set of dependencies w.r.t. database schemes [GV]. Cost-effective query processing in a database is possible exactly when the database scheme is bounded w.r.t. its dependencies. On the other hand, boundedness of dependencies w.r.t. database schemes is a necessary condition for the maintenance problem to be solved in time independent of the database size [GW].

Both problems are very difficult to solve in general, if possible at all. As mentioned previously in this thesis, boundedness of database schemes w.r.t. dependencies is conjectured to be undecidable even in the case of fd's [MUV]. Testing boundedness of dependencies w.r.t. database schemes is also conjectured to be undecidable even in the case of total dependencies; it is believed to be decidable when only fd's are given [GV].

Since query answering and constraint enforcement are two important functions in any database system, characterizing both notions of boundedness is essential for real-life applications. However, research in this area so far indicates that finding such a characterization is extremely difficult, if it is possible at all [AtC][GW][IIK][MUV][S3]. Most of this research concentrates on finding sufficient conditions for boundedness. Our conditions in Chapters 3 and 4 extend the results in this line of research. Hence finding weak conditions for boundedness or unboundedness might be the best that we can do.

Unboundedness, the other side of the coin in the boundedness problem, is completely unexplored. We investigate this problem in this chapter and show that there exists a general, effective, and sufficient condition for both notions of unboundedness when fd's are considered. The condition is very general in the sense that no assumption is made w.r.t. the database scheme or the set of fd's.

5.2. Overview of Chapter

In Section 5.3, we give some definitions required in this chapter. In Section 5.4, we give a sufficient condition for both types of unboundedness when fd's are considered. In Section 5.5, we give our conclusions.

5.3. Some Definitions

In this chapter, we use the definition of boundedness of a database scheme w.r.t. fd's given in Chapter 2. We repeat it here for convenience.

Let $[X]_r$ denote the X-total projection of the representative instance for $\mathbf{r}$ and let $|\mathbf{r}|$ denote the number of tuples in $\mathbf{r}$. Then we say that a database scheme $(\mathbf{R}, F)$ is *bounded* (w.r.t. F) if for all $X \subseteq \mathbf{U}$ there is a constant $k>0$ such that, for every consistent state $\mathbf{r}$ of $(\mathbf{R}, F)$, and for every $t \in [X]_r$, there exists a substate $\mathbf{r}'$ of $\mathbf{r}$ such that $t \in [X]_{r'}$ and $|\mathbf{r}'| \leq k$ [GM]. We say that a database scheme $(\mathbf{R}, F)$ is *unbounded* (w.r.t. F) if it is not bounded (w.r.t. F).

To define the concept of boundedness of a set of fd's w.r.t. a database scheme, we need the following definitions. Let r be a relation on R. The set of all attribute values in r is denoted by $Val(r)$. Let $\mathbf{r} = (r_1, \ldots, r_n)$ be a state of $(\mathbf{R}, F)$. Then $Val(\mathbf{r}) = \bigcup_{j=1}^{n} \mathrm{Val}(r_j)$. Let $\mathbf{r}'$ be a state of $(\mathbf{R}, F)$. We say $\mathbf{r}' \subseteq \mathbf{r}$ if $r_j' \subseteq r_j$, for all $1 \leq j \leq n$.

A set of fd's F is *bounded* w.r.t. a database scheme $(\mathbf{R}, F)$, if there is some constant $k>0$ such that, for every state $\mathbf{r}$ of $(\mathbf{R}, F)$ we have that $\mathbf{r}$ is consistent w.r.t. F if and only if for all states $\mathbf{r}'$ of $(\mathbf{R}, F)$ such that $\mathbf{r}' \subsetneq \mathbf{r}$ and $|\text{Val}(\mathbf{r}')| \leq k$, we have $\mathbf{r}'$ is consistent w.r.t. F [GV]. We say that a set of fd's F is *unbounded* w.r.t. a database scheme $(\mathbf{R}, F)$ if it is not bounded w.r.t. $(\mathbf{R}, F)$.

Graham and Mendelzon [GM] have shown that boundedness of F w.r.t. $(\mathbf{R}, F)$ implies boundedness of $(\mathbf{R}, F)$ and that under certain conditions the reverse implication also holds.

Let $\mathbf{R}_i \subseteq \mathbf{R}$. The *tableau* for $\mathbf{R}_i$, written $T_{\mathbf{R}_i}$ is defined as follows. For each R_l in $\mathbf{R}_i$, $T_{\mathbf{R}_i}$ has a row t such that for all A_j in R_l, $t[A_j]$ is a dv a_j, and for all A in $\mathbf{U}-R_l$, $t[A]$ is a distinct ndv appearing nowhere else in $T_{\mathbf{R}_i}$. We can chase $T_{\mathbf{R}_i}$ w.r.t. F. $CHASE_F(T_{\mathbf{R}_i})$ is denoted by $T^*_{\mathbf{R}_i}$. Let $X \subseteq \mathbf{U}$. We say that $\pi_X(\bowtie\mathbf{R}_i)$ is *lossless (w.r.t. F)* if there is some row in $T^*_{\mathbf{R}_i}$ that has a dv on any A in X [MUV]. We say $\pi_X(\bowtie\mathbf{R}_i)$ is *lossy* if it is not lossless.

5.4. A Sufficient Condition for Unboundedness

In this section, we give a condition for database schemes and fd's to be unbounded. First we motivate the condition by giving the canonical example of unboundedness. This is the same as Example 2.3, but now we also show the unboundedness of the set of fd's.

Example 5.1: Let $(\mathbf{R}, F) = (\{R_1(AB), R_2(AC), R_3(CB)\}, \{A \rightarrow B, C \rightarrow B\})$. Let a state of $(\mathbf{R}, F)$ be $\mathbf{r} = (r_1 = \{<a_1, b_1>\}, r_2 = \{<a_1, c_1>, <a_2, c_1>, <a_2, c_2>, \ldots, <a_{n-1}, c_{n-1}>, <a_n, c_{n-1}>\}, r_3 = \varnothing)$. T_r for that state is shown in Figure 5.1 below; distinct ndv's are represented by "-". $\mathbf{r}$ is consistent w.r.t. F.

Observe $t = <a_n, b_1, c_{n-1}>$ is in the representative instance of $\mathbf{r}$. However, since the representative instance of any proper substate of $\mathbf{r}$ does not contain $\{t\}$, $(\mathbf{R}, F)$ is unbounded.

A	B	C
a_1	b_1	-
a_1	-	c_1
a_2	-	c_1
a_2	-	c_2
●	●	●
●	●	●
●	●	●
a_{n-1}	-	c_{n-1}
a_n	-	c_{n-1}

Figure 5.1 T_r for **r** in Example 5.1

Now let **r** = **r** $\bigcup$ $\{<a_n, b_2>\}$. **r** is not consistent w.r.t. F. However any proper substate of **r** is consistent w.r.t. F since its representative instance does not contain both t and $<a_n, b_2>$. Thus F is unbounded w.r.t. (**R**, F). □

From the canonical example, we can make the following observations:

- For unboundedness to happen, T_r after some number of applications of the fd-rules must contain a "ladder", for instance, $<a_1, -, c_1>$, $<a_2, -, c_1>$, . . . , $<a_n, -, c_{n-1}>$ in Example 5.1, where the only way to get one of the tuples, $<a_n, b_1, c_{n-1}>$ in this case, is by a number of fd-rule applications proportional to the number of tuples in the database state.
- The behavior of the fd's $A \rightarrow B$ and $C \rightarrow B$ determining the same attribute, B in this case, is crucial for having unboundedness.
- Also the fact that neither $A \rightarrow C$ nor $C \rightarrow A$ is in F^+ is crucial.

In the following subsection we give a sufficient condition for unboundedness that captures in general the observations made above.

5.4.1. The Condition and its Correctness

Let (**R**, F) be a database scheme that satisfies the following condition C:

C1: There exist nontrivial fd's $X \to B$ and $Y \to B$ in F^+.

C2: Neither $X \to Y$ nor $Y \to X$ is in F^+.

C3: There exists $\mathbf{R}_j \subseteq \mathbf{R}$ such that either $\pi_{XB}(\bowtie \mathbf{R}_j)$ or $\pi_{YB}(\bowtie \mathbf{R}_j)$ is lossless.

C4: There exists $\mathbf{R}_i \subseteq \mathbf{R}$ such that $\pi_{XY}(\bowtie \mathbf{R}_i)$ is lossless.

C5: Let $W = X^+ \cap Y^+ \cap (\bigcup \mathbf{R}_i)$. Let u_1 and u_2 be tuples defined on U as follows: For each $A_j \in$ U, $u_1[A_j] = a_j$, where a_j is a constant appearing nowhere else in u_1; for each $A_j \in$ U $- W$, $u_2[A_j] = a_l$, where a_l is a constant appearing nowhere else in u_1 or u_2, and $u_2[W] = u_1[W]$. Let T_{ij} and T_i be the tableaux for the states $\pi_{\mathbf{R}_i \bigcup \mathbf{R}_j}(\{u_1\})$ and $\pi_{\mathbf{R}_i}(\{u_2\})$ respectively; assume any ndv in T_{ij} is distinct from any ndv in T_i. Then if t is a tuple in $CHASE_F(T_{ij} \bigcup T_i)$ that originates from T_i and $t[XY]$ are constants, then $t[B]$ is an ndv.

Intuitively, conditions C3 to C5 state the following properties of some consistent state **r** of (**R**, F): C3 states that in $CHASE_F(T_r)$ we have a tuple like the first one shown in Figure 5.1; C4 states that in $CHASE_F(T_r)$ we have a "ladder" defined on XY like the one on AC shown in Figure 5.1; C5 states that we need the whole "ladder" on XY to obtain a constant on B in the "last" tuple in T_r. Notice C5 implies $\mathbf{R}_j \not\subseteq \mathbf{R}_i$.

Example 5.2: Let (**R**, F) = ($\{R_1(AGC), R_2(CEB), R_3(ED), R_4(ADB)\}$, $\{GC \to B, C \to E, E \to D, AD \to B\}$). (**R**, F) satisfies C above with $\mathbf{R}_i = \{R_1, R_2, R_3\}$, $\mathbf{R}_j = \{R_4\}$, $X = AD$, $Y = GC$, and $B = B$. □

We claim (**R**, F) is unbounded and F is unbounded w.r.t. (**R**, F). Let $S_i = \bigcup \mathbf{R}_i$ and $S_j = \bigcup \mathbf{R}_j$. Let $F' = \{V \to A \in F^+ |\ VA \subseteq S_i\}$. The plan for the proof of these claims is the following. We define a tableau T whose tuples are constants on S_i and whose projection on S_i satisfies F'. After that we prove that $CHASE_F(T)$ is nonempty. Then from T we construct a consistent state **r** of (**R**, F). The construction of **r** and **C** shall imply that in $CHASE_F(T_r)$ there is a "ladder" like the one in the canonical example and a tuple t in $CHASE_F(T_r)$ whose B-total part cannot be inferred from any state obtained from a proper subset of T. Because the state **r** can be arbitrarily large, this shall imply (**R**, F) is unbounded. Then we use a result in [GM] to show that the condition is sufficient for unboundedness of F w.r.t. (**R**, F).

We now start the construction of T. By definition of $\mathbf{R}_i$, we have $S_i \supseteq XY$. Let us write S_i as $X'Y'Z$, where $X' = X^+ \cap S_i$, $Y' = Y^+ \cap S_i$, and $Z = S_i - X'Y'$. Let $W = X' \cap Y'$. Then we can write $X'Y'$ as X^*WY^*, where $X^* = X' - W$ and $Y^* = Y' - W$. $X^* \neq \varnothing$ and $Y^* \neq \varnothing$; else either $W \supseteq Y'$ or $W \supseteq X'$ and since $X \to W$ and $Y \to W$ are in F^+, then either $X \to Y$ or $Y \to X \in F^+$ and C2 is violated. Hence we can write $S_i = X^*WY^*Z$, where X^*, W, Y^*, and Z are pairwise disjoint, and X^* and Y^* are both nonempty.

Some crucial facts about T depend on whether $\pi_{XB}(\bowtie\mathbf{R}_j)$ is lossless. We assume without loss of generality that $\pi_{XB}(\bowtie\mathbf{R}_j)$ is lossless. (If $\pi_{YB}(\bowtie\mathbf{R}_j)$ is lossless but $\pi_{XB}(\bowtie\mathbf{R}_j)$ is lossy then the arguments below about the construction of T are symmetric on Y.)

Let us consider the following tableau where distinct ndv's are denoted by "-": $T = \{\, t_0 = <x_1^*, w_0, y_0^*, z_0, v_0, ->, t_1 = <x_1^*, w_0, y_1^*, z_1, ->, t_2 = <x_2^*, w_0, y_1^*, z_2, ->, t_3 = <x_2^*, w_0, y_2^*, z_3, ->, t_4 = <x_3^*, w_0, y_2^*, z_4, ->, \ldots, t_{2n-2} = <x_n^*, w_0, y_{n-1}^*, z_{2n-2}, ->\}$, where x_q^*, y_l^*, z_m denote constant values on X^*, Y^*, and Z respectively; w_0 and v_0 denote constant values on W and on $S_j - S_i$ respectively. Two values x_p^* and x_q^* are identical if $p = q$. However, if $p \neq q$, then they do not have any common component. Similarly for y_l^*'s and z_m's. By definition of $\mathbf{R}_j$, $B \in S_j$. Hence $t_0[B]$ is a constant. In the rest of the proof we assume the constant on $t_0[B]$ is b_1.

Let $s_i = \pi_{S_i}(T)$, where, by previous definition, $S_i = \bigcup \mathbf{R}_i$. In the following lemma we prove s_i satisfies F'.

Lemma 5.1: s_i satisfies F'.

Proof: Let us consider a nontrivial fd $V \to A \in F'$ and two distinct tuples u and t in s_i which are equal on V. We prove a violation of $V \to A$ is impossible in s_i. By construction of T, V lies in X' or Y'.

Case 1: $V \subseteq X' = X^* W$. First observe A cannot be in Z by definition of X'. Also observe $A \notin Y^*$; otherwise $A \in Y'$ and, since A is in X', thus in W, which is a contradiction to the fact that W and Y^* are disjoint.

Now if $V \subseteq W$, then $A \in Y'$ and $A \in X'$. Hence $A \in W$. In this case a violation of $V \to A$ is impossible since all the tuples in T have equal values on A. On the other hand, if $V \not\subseteq W$, then, by construction of T, u and t are the same on any attribute in X'. Hence no violation of $V \to A$ can occur in this case either.

Case 2: $V \subseteq Y' = WY^*$. By a similar argument as in Case 1 above, we can prove a violation of the fd $V \to A$ is impossible.

From the arguments above our claim is proven. □

The following facts, crucial to our proof, hold for T.

Fact 5.1: There exist at most two distinct tuples in T which are equal on X^*.

Fact 5.2: There exist at most two distinct tuples in T which are equal on Y^*.

Fact 5.3: Let t and u be two distinct tuples in T. Then exactly one of the following holds:

- $u[X^*] = t[X^*]$;
- $u[Y^*] = t[Y^*]$;
- $u[X^*] \neq t[X^*]$ and $u[Y^*] \neq t[Y^*]$.

Now we want to prove $CHASE_F(T)$ is nonempty. We first require to prove the following facts about any chase of T w.r.t. F. For this let us consider a sequence of fd-rules $\tau_1 \ldots \tau_k$ to be applied to T; $\tau_1 \ldots \tau_k(T)$ is $\tau_k(\ldots(\tau_1(T)))$.

Lemma 5.2: Assume $T' = \tau_1 \ldots \tau_k(T)$ is nonempty and t_1 and t_2 are in T'; $t_1 \neq t_2$, $A \in \mathbf{U}$. If $t_1[A] = t_2[A]$, then

(a) If $t_1[X^*] = t_2[X^*]$, then $X \to A \in F^+$.

(b) If $t_1[Y^*] = t_2[Y^*]$, then $Y \to A \in F^+$.

(c) If $t_1[X^*] \neq t_2[X^*]$ and $t_1[Y^*] \neq t_2[Y^*]$, then $X \to A \in F^+$ and $Y \to A \in F^+$.

Proof: By induction on k.

Basis: $k = 0$. Let t_1 and t_2 be two distinct tuples in T such that they are equal on $A \in \mathbf{U}$. By construction of T, A must be in S_i.

Case (a): If $t_1[X^*] = t_2[X^*]$, then, by Fact 5.3 and construction of T, $A \in X'$. By definition of X', $X \to A \in F^+$.

Case (b): If $t_1[Y^*] = t_2[Y^*]$, then, by Fact 5.3 and construction of T, $A \in Y'$. By definition of Y', $Y \to A \in F^+$.

Case (c): If $t_1[X^*] \neq t_2[X^*]$ and $t_1[Y^*] \neq t_2[Y^*]$, then, by construction of T, $A \in W$. By definition of W, $X \to A$ and $Y \to A \in F^+$.

Induction: $k > 0$. Assume T'' is nonempty and is obtained from T by $k-1 \geq 0$ fd-rule applications. Let us also assume that T' is nonempty and is obtained from T'' by applying the fd-rule τ_k: $V \to A$, $V \to A$ in F, to equate v_1 and v_2 in T'', $v_1[A] \neq v_2[A]$. By the inductive hypothesis the proposition is true for T'' and we have to prove it for T'. Observe $v_2[V] = v_1[V]$ in T''.

We have to consider t_1 and t_2 in T'' such that $t_1[A] = v_1[A]$ and $t_2[A] = v_2[A]$. By Fact 5.3, there are three cases to be considered depending on the equality among v_1 and v_2 on X^* and Y^*. First we consider the easiest case.

Case 1: $v_1[X^*] \neq v_2[X^*]$ and $v_1[Y^*] \neq v_2[Y^*]$. Since $v_1[V] = v_2[V]$, by the inductive hypothesis, $X \to V$ and $Y \to V \in F^+$. Hence $X \to A$ and $Y \to A \in F^+$. Then what we have to prove for $t_1[A]$ and $t_2[A]$ follows trivially.

Case 2: $v_1[X^*] = v_2[X^*]$. Since $v_1[V] = v_2[V]$, by the inductive hypothesis, $X \to V \in F^+$. Hence $X \to A \in F^+$. There are several cases to be considered depending on whether $t_1 = v_1$ and $t_2 = v_2$.

Case 2.a: $t_1 = v_1$ and $t_2 = v_2$. Hence $t_1[X^*] = t_2[X^*]$. Since we already know $X \to A \in F^+$, this case is proven.

Case 2.b: $t_1 \neq v_1$. (The argument for $t_2 \neq v_2$ is symmetric.) Then Fact 5.1, $v_1[X^*] = v_2[X^*]$, and $v_1 \neq v_2$ imply $t_1[X^*] \neq v_1[X^*]$. By Fact 5.3, $t_1[Y^*] = v_1[Y^*]$ or $t_1[Y^*] \neq v_1[Y^*]$. In either case since $t_1[A] = v_1[A]$, by the inductive hypothesis we have $Y \to A \in F^+$. Hence both $X \to A$ and $Y \to A$ are in F^+ and what we have to prove for $t_1[A]$ and $t_2[A]$ now follows trivially.

Case 3: $v_1[Y^*] = v_2[Y^*]$. By an argument similar to that in Case 2 above we can prove this case.

This completes the inductive proof and the proof of the lemma. □

Now we prove that $CHASE_F(T)$ is nonempty.

Lemma 5.3: $CHASE_F(T) \neq \emptyset$.

Proof: We prove it is impossible to have $CHASE_F(T) = \emptyset$. Let $\tau_1 \ldots \tau_k$, $k \geq 1$, be a sequence of fd-rules applied to T such that τ_k is the first fd-rule that tries to equate two distinct constants.

Assume τ_k: $V \to A$, $V \to A$ in F, equates t_1 and t_2 which are tuples in $\tau_1 \ldots \tau_{k-1}(T)$ such that $t_1[A] \neq t_2[A]$ and both are constants. Notice $t_1[V] = t_2[V]$. By construction of T and since the constraints are fd's, A must be in S_i. By Fact 5.3, there are three cases to be considered.

Case 1: $t_1[X^*] = t_2[X^*]$. By Lemma 5.2 and $t_1[V] = t_2[V]$, $X \to V \in F^+$. Hence $X \to A \in F^+$. Since $XA \subseteq S_i$, t_1 and t_2 violate $X \to A \in F'$. This contradicts Lemma 5.1. Hence this case is impossible.

Case 2: $t_1[Y^*] = t_2[Y^*]$. By an argument similar to that in Case 1 above, we can prove that this case is impossible.

Case 3: $t_1[X^*] \neq t_2[X^*]$ and $t_1[Y^*] \neq t_2[Y^*]$. By Lemma 5.2 and $t_1[V] = t_2[V]$, $X \to V$ and $Y \to V \in F^+$. Hence $X \to A$ and $Y \to A \in F^+$. Then $A \in W$, since $A \in S_i$. But then $t_1[A] = t_2[A]$ by construction of T. This is a contradiction to the assumption that $t_1[A] \neq t_2[A]$. Thus this case is impossible.

We have proven our claim. □

Now we construct a consistent state **r** of (**R**, F) from T as follows:

- For all $R_l \in \mathbf{R}_i$, $r_l = \pi_{R_l}(T - \{t_0\})$;
- for all $R_l \in \mathbf{R}_j$, $r_l = \pi_{R_l}(\{t_0\})$;
- for all $R_l \in \mathbf{R} - (\mathbf{R}_i \bigcup \mathbf{R}_j)$, $r_l = \emptyset$.

Lemma 5.4: **r** is consistent w.r.t. F.

Proof: From the construction of **r** and by Lemma 5.3, $CHASE_F(T)$ is a weak instance of **r** w.r.t. F. □

We now show there is a "ladder" in $\pi_{XY}(T)$. First observe that $X^\# = X^* \bigcap X \neq \emptyset$; else $X \subseteq W$ and hence $Y \to X \in F^+$, which violates C2; by a similar reason $Y^\# = Y^* \bigcap Y \neq \emptyset$. Then $\pi_{XY}(T) = \{ <x_1, y_0>, <x_1, y_1>, <x_2, y_1>, <x_2, y_2>, <x_3, y_2>, \ldots, <x_n, y_{n-1}> \}$ where $<x_1, y_0> = t_0[XY]$, $<x_1, y_1> = t_1[XY]$, $<x_2, y_1> = t_2[XY]$, $<x_2, y_2> = t_3[XY]$, $<x_3, y_2> = t_4[XY]$, $\ldots$, $<x_n, y_{n-1}> = t_{2n-2}[XY]$; and by construction of T x_p and x_q (respectively, y_p and y_q) are identical if $p = q$, but if $p \neq q$, then they do not have any common component on $X^\#$ (respectively, on $Y^\#$). In other words, $x_q = x_p$ if and only if $p = q$; similarly for y_p and y_q. Let T_r be the tableau state of **r**.

Lemma 5.5: $<x_n, y_{n-1}, b_1>$ is in the XYB-total projection of $CHASE_F(T_r)$.

Proof: From construction of **r**, $\pi_{XB}(\bowtie \mathbf{R}_j)$ is lossless, and $\pi_{XY}(\bowtie \mathbf{R}_i)$ is lossless, there exist in $CHASE_F(T_r)$ tuples $t_0', t_1', t_2', \ldots, t_{2n-2}'$, corresponding respectively to $t_0, t_1, t_2, \ldots, t_{2n-2}$ in T, such that $t_0'[XB] = <x_1, b_1>$, $t_1'[XY] = <x_1, y_1>$, $t_2'[XY] = <x_2, y_1>, \ldots, t_{2n-2}'[XY] = <x_n, y_{n-1}>$. Since $X \rightarrow B$ and $Y \rightarrow B$ are in F^+, $t_{2n-2}'[XYB]$ must be $<x_n, y_{n-1}, b_1>$ in $CHASE_F(T_r)$. □

Let T_{ij} and T_i be as defined in C5. Let us assume $\mathbf{R}_i = \{R_{i_1}, \ldots, R_{i_p}\}$, $p \geq 1$, and let $s_{i_1}, \ldots, s_{i_p}$ be the rows of $CHASE_F(T_{ij} \cup T_i)$ from T_i; $s_{i_1}, \ldots, s_{i_p}$ originate from $R_{i_1}, \ldots, R_{i_p}$ respectively. Let $t_q \in T$ be such that $t_q \neq t_0$ and $t_q \neq t_{2n-2}$. Let $T_1 = \{t_l \in T \mid l < q\}$ and $T_2 = \{t_l \in T \mid l > q\}$. Let $\mathbf{r}_1$ and $\mathbf{r}_2$ be states constructed from T_1 and T_2 respectively such that $\mathbf{r}_1$ contains tuples in **r** originating from T_1 while $\mathbf{r}_2$ contains tuples in **r** originating from T_2. Let T_{r_1} and T_{r_2} be the tableaux for $\mathbf{r}_1$ and $\mathbf{r}_2$ respectively. Observe $\mathbf{r}_1$ is a state defined on $\mathbf{R}_i \cup \mathbf{R}_j$ and $\mathbf{r}_2$ is a state defined on $\mathbf{R}_i$ only. Intuitively, C5 states that in $CHASE_F(T_{r_1} \cup T_{r_2})$ there is no tuple from $\mathbf{r}_2$ such that $t[XYB]$ are constants. That is, we need the whole "ladder" to obtain $<x_n, y_{n-1}, b_1>$ in the representative instance of **r**. Now we prove this in the following two lemmas.

Lemma 5.6: Let $T_{12} = T_{r_1} \cup T_{r_2}$. Let τ be a sequence of fd-rules of fd's in F that can be applied to T_{12} and assume $T' = \tau(T_{12}) \neq \varnothing$. Let u_1 be a tuple in T' from R_{i_l} and from $\mathbf{r}_2$, and $A \in \mathbf{U}$. Then if $u_1[A]$ is a constant, then $s_{i_l}[A]$ is a constant, where s_{i_l} is a row in $CHASE_F(T_{ij} \cup T_i)$ from T_i as defined above.

Proof: Similar to the proof for Lemma 4 in [G]. □

Lemma 5.7: Let t be any tuple in $CHASE_F(T_{r_1} \cup T_{r_2})$ such that $t[XY]$ are constants and equal to $<x_n, y_{n-1}>$. Then $t[B]$ is an ndv.

Proof: Assume t is a tuple in $CHASE_F(T_{r_1} \bigcup T_{r_2})$ such that $t[XY]$ are constants and equal to $<x_n, y_{n-1}>$. Assume $t[B]$ is a constant.

By construction of T_{r_2} and $\pi_{XY}(\bowtie \mathbf{R}_i)$ is lossless, there exists some tuple t' from R_{i_j} and from $\mathbf{r}_2$ such that $t'[XY]$ are constants and equal to $<x_n, y_{n-1}>$; such a tuple t' can be t'_{2n-2} in Lemma 5.5. Hence since $X \rightarrow B$ and $Y \rightarrow B$ are in F^+, $t'[B]$ must be the constant $t[B]$. Then by Lemma 5.6, s_{i_j} in $CHASE_F(T_{i_j} \bigcup T_i)$ has constants on XYB. This is a contradiction to condition C5. □

We are now ready to prove $(\mathbf{R}, F)$ is unbounded.

Lemma 5.8: $(\mathbf{R}, F)$ is unbounded.

Proof: Assume $(\mathbf{R}, F)$ is bounded and assume $k>0$ is $(\mathbf{R}, F)$'s boundedness constant.

We can make the number of tuples in T arbitrarily larger than k such that any substate $\mathbf{r}'$ of $\mathbf{r}$ with $|\mathbf{r}'| \leq k$ misses at least one of the tuples in T. However, by Lemma 5.7 we cannot infer $<x_n, y_{n-1}, b_1>$ if we miss a tuple from T. This is a contradiction to our assumption that $(\mathbf{R}, F)$ is bounded. □

We just finish proving our first claim in this chapter.

Theorem 5.1: The condition **C** is sufficient for unboundedness of $(\mathbf{R}, F)$.

Now we prove **C** is a sufficient condition for unboundedness of F w.r.t. $(\mathbf{R}, F)$.

Corollary 5.1: Let $(\mathbf{R}, F)$ be such that satisfies **C**. Then F is unbounded w.r.t. $(\mathbf{R}, F)$.

Proof: It follows from Theorem 5.1 and Proposition 6 in [GM]. □

Example 5.3: Let $(\mathbf{R}, F) = (\{R_1(AGC), R_2(CEB), R_3(ED), R_4(ADB)\}, \{GC \rightarrow B, C \rightarrow E, E \rightarrow D, AD \rightarrow B\})$. We saw in Example 5.2 that $(\mathbf{R}, F)$ satisfies **C**. Then both $(\mathbf{R}, F)$ and F are unbounded. □

5.5. Conclusions

We have shown that there is a general and sufficient condition for unboundedness of database schemes and fd's. The problem of testing unboundedness using our condition is in *NP* [GJ], since by guessing X, Y, B, $\mathbf{R}_i$, and $\mathbf{R}_j$ we can test if they satisfy each of the components of our condition in polynomial time. That is, the condition can be tested effectively; but probably not in polynomial time.

We believe our condition is important in practice, given its generality and the difficulty of ~~testing~~ boundedness for database schemes and fd's. Whether a weaker condition can be found is an open problem which, if possible to solve, seems to be a very difficult and complex one.

As mentioned in the Introduction of this chapter, characterizing the bounded schemes or bounded constraints is fundamental to finding classes of database schemes that are desirable w.r.t. query processing and updates. However these are extremely difficult problems to solve, if they are solvable at all. So establishing weak sufficient or weak necessary conditions for these problems might be the best that we can do. In this chapter we have given a very general and sufficient condition for the unboundedness problems when fd's are considered.

Chapter 6

Conclusions and Future Research

6.1. Conclusions

We have studied the problem of boundedness of relational database schemes when fd's are the constraints imposed on the database. We showed that determining whether a class of database schemes is bounded w.r.t. fd's is fundamental for the analysis of the class of database schemes not only w.r.t. query processing, but also w.r.t. efficient enforcement of fd's. In what follows, we summarize the contributions of this thesis.

One of the main contributions of this thesis is the identification of γ-acyclic BCNF database schemes as a class of database schemes which is highly desirable w.r.t. query processing and enforcement of fd's. In Chapter 3, we first proved that this class of schemes is bounded w.r.t. the set of fd's embodied in the database scheme. This result enlarges the class of known bounded database schemes. We then showed that this class of schemes is simple in semantics by proving that there is a simple and efficient way to compute the X-total projection of the representative instance. As a consequence, answers to many queries for this class of schemes can be computed easily and efficiently. We also showed that if a γ-acyclic BCNF database scheme is lossless, then it is connection-trap-free. Finally, we derived a simple and efficient algorithm that determines if an updated state is consistent. This allows the system to incrementally enforce the satisfaction of fd's embodied in the database scheme in constant time.

As mentioned in Chapter 3, contrary to our initial expectations, our proof of boundedness of this class of schemes turned out to be long and complex. This gives strong evidence supporting Maier et al.'s conjecture about the undecidability of the problem of testing boundedness of database schemes w.r.t. dependencies, even when only fd's are considered [MUV].

The only other known class of database schemes with all the desirable properties of γ-acyclic BCNF database schemes is the class of independent and connection-trap-free database schemes characterized in [CA].

In Chapter 4, we investigated an alternative approach for characterizing database schemes bounded w.r.t. fd's. We showed that a database scheme is bounded w.r.t. fd's if it is extensible into a bounded one. Also we showed that if a database scheme is extensible into a ctm database scheme, then it is also ctm. We showed how to compute total projections or enforce fd's in constant time for database schemes proven to be bounded or ctm using these results.

Then we presented a formal methodology for designing database schemes bounded w.r.t. fd's using a new technique called extensibility. This methodology can also be used to design ctm database schemes. The major advantage of this technique is its iterative nature. We can apply this technique to a known class of bounded or ctm database schemes and generate other classes of bounded or ctm database schemes. For instance, we showed how to design a class of bounded and ctm database schemes which are neither acyclic schemes nor independent.

Our proposed methodology for designing bounded or ctm database schemes brings new insight into how to design database schemes under the weak instance model. This is clearly very helpful since cost-effective query processing or constant-time fd enforcement are highly desirable properties for any database and these are possible only if a database scheme is bounded or ctm.

As discussed previously in Chapter 5, characterizing the bounded schemes or bounded constraints is fundamental to finding classes of database schemes that are desirable w.r.t. query processing and updates. However these are extremely difficult problems to solve, if they are solvable at all. So establishing weak sufficient or weak necessary conditions for these problems might be the best that we can do. Prior to this thesis, the characterization of unbounded database schemes or fd's was completely unexplored.

In Chapter 5, we studied the unboundedness problem when fd's are considered. We showed therein that there exists a very general sufficient condition for unboundedness of database schemes and fd's. We believe our condition is important in practice, given its generality and the difficulty of testing boundedness for database schemes and fd's. Whether a weaker condition can be found is an open problem which, if possible to solve, seems to be difficult and complex.

6.2. Future Research

We can observe from the unboundedness condition in Chapter 5, that there exist two fd's $X \rightarrow B$ and $Y \rightarrow B$ that can add B to the closure of some relation scheme; and, which is more important, X and Y are logically independent of each other, in the sense that neither $X \rightarrow Y$ nor $Y \rightarrow X$ is in F^{+}. For independent database schemes and γ-acyclic BCNF database schemes, we can observe in that respect the following. In the former class, there is exactly one fd that adds an attribute to a closure. In the latter class of database schemes, if more than one fd can add an attribute to a given relation scheme's closure, then they determine each other, since they are key dependencies embedded in the same relation scheme. Also the class of ctm database schemes characterized in [GW] seems to have the property that if an attribute can be added to a relation scheme's closure by more than one fd, then the fd's determine each other. We believe that when fd's are considered, boundedness, constant-time-maintainability, and unboundedness of a database scheme are basically consequences of the freedom, in the above sense, among the fd's that can add an attribute to the closure of a relation scheme.

Within the two extremes represented by the unbounded database schemes at one end, and the ctm and bounded database schemes at the other end, lies a class of bounded database schemes to which a database like $(\mathbf{R}, F) = (\{AB, BC, AC\}, \{A \rightarrow B, B \rightarrow C, A \rightarrow C\})$ belongs. This database scheme is neither ctm nor independent, however it is bounded. In such a class of schemes, an attribute can be added to a relation scheme's closure by more than one fd; in $(\mathbf{R}, F)$ above, C can be added to AB's closure by either $A \rightarrow C$ or $B \rightarrow C$; but observe $A \rightarrow B$ holds. An interesting open problem is to prove boundedness for such a class. A good start is to try to find an algorithm to test boundedness w.r.t. fd's of BCNF database schemes where each relation scheme has exactly one key; the database scheme shown above belongs to this class.

As noticed above, $(\mathbf{R}, F) = (\{AB, BC, AC\}, \{A \rightarrow B, B \rightarrow C, A \rightarrow C\})$ is not ctm. This is because the enforcement of $A \rightarrow C$ cannot be done in time independent of the database size. However, we can enforce $A \rightarrow C$ cost-effectively, that is, without generating the representative instance, via the relational expression $\pi_{AC}(AB \bowtie BC) \cup AC$. An open problem is to characterize a general class of database schemes where cost-effective fd enforcement is possible via relational expressions.

In the course of our investigation of a sufficient condition for unboundedness, we came to the following question which we leave as an open problem: Are *(complex) chase-join-expressions (cje's)* [C1][AtC] "complete" for the class of database schemes bounded w.r.t. fd's? That is using cje's, can we compute the X-total projections of any consistent state of any database scheme which is bounded w.r.t. fd's? Or perhaps, less ambitious, for which class of database schemes bounded w.r.t. fd's are cje's complete?

We believe that β-acyclic BCNF database schemes are bounded w.r.t. fd's. This is another open problem; but probably too difficult a problem to be solved.

In Chapter 4, we extended Mendelzon's extensibility results in [M] in two very important respects: boundedness and constant-time-maintainability. An important question arises there: In which other database design problems can the extensibility idea be applied to reduce the difficulty involved in solving them?

In Chapter 4, we did not address the problem of identifying a class of bounded database schemes using some of the sound design rules found in that chapter. We believe investigating this issue may give us more insight into how to design database schemes which are bounded or ctm.

Finally, the question of whether the problem of testing boundedness of database schemes w.r.t. fd's is undecidable remains open.

References

[A] Armstrong, W.W. "Dependency Structures of Data Base Relationships." *Proc. IFIP 1974*, North-Holland, pp. 580-583.

[ABU] Aho, A.V., Beeri, C., Ullman, J.D. "The Theory of Joins in Relational Databases." *ACM TODS 4*, 3, September 1979, pp. 297-314.

[ADM] Ausiello, G., A. D'Atri, Moscarini, M. "Minimal Coverings for Acyclic Database Schemata." *Advances in Database Theory, Vol. 2*, (H. Gallaire, J. Minker, and J.M. Nicolas, eds.) North Holland, 1984, pp. 27-51.

[AP] Atzeni, P., Parker, D.S. Jr. "Assumptions in Relational Database Theory." *Proc. ACM PODS 1982*, pp. 1-9.

[ASU] Aho, A.V., Sagiv, Y., Ullman, J.D. "Equivalence of Relational Expressions." *SIAM J. of Computing 8*, 2, 1979, pp. 435-454.

[AtC] Atzeni, P., Chan E.P.F. "Efficient Query Answering in the Representative Instance Approach." *Proc. ACM PODS 1985*, pp. 181-188.

[B] Berge, C. *Graphs and Hypergraphs.* North-Holland, Amsterdam, The Netherlands, 1973.

[BB] Beeri, C., Bernstein, P.A. "Computational Problems Related to the Design of Normal Form Relational Schemas." *ACM TODS 4*, 1, March 1979, pp. 30-59.

[BBC] Bernstein, P.A., Blaustein, B.T., Clarke, E.M. "Fast Maintenance of Semantic Integrity Assertions using Redundant Aggregate Data." *Proc. VLDB 1980*, pp. 126-136.

[BBG] Beeri, C., Bernstein, P.A., Goodman, N. "A Sophisticate's Introduction to Database Normalization Theory." *Proc. VLDB 1978*, pp. 113-124.

[BBSK] Biskup, J., Bruggemann, H.H., Schnetgoke, L., Kramer, M. "One Flavor Assumption and γ-acyclicity for Universal Relation Views." *Proc. ACM PODS 1986*, pp. 148-159.

[BDB] Biskup, J., Dayal, U., Bernstein, P.A. "Synthesizing Independent Database Schemas." *Proc. ACM SIGMOD 1979*, pp. 143-151.

[BG] Bernstein, P.A., Goodman, N. "What Does Boyce-Codd Normal Form Do?" *Proc. VLDB 1980*, pp. 245-259.

[BH] Beeri, C., Honeyman, P. "Preserving Functional Dependencies." *SIAM J. of Computing 10*, 3, August 1981, pp. 647-656.

[BMSU] Beeri, C., Mendelzon, A.O., Sagiv, Y., Ullman, J.D. "Equivalence of Relational Database Schemes." *SIAM J. of Computing 10*, 2, 1981, pp. 352-370.

[BV] Beeri, C., Vardi, M.Y. "A Proof Procedure for Data Dependencies." *JACM 31*, 4, October 1984, pp. 718-741.

[BVo] Brosda, V., Vossen, G. "Updating a Relational Database Through a Universal Schema Interface." *Proc. ACM PODS 1985*, pp. 66-75.

[C1] Chan, E.P.F. "Optimal Computation of Total Projections with Unions of Simple Chase Join Expressions." *Proc. ACM SIGMOD 1984*, pp. 149-163.

[C2] Chan, E.P.F. "An Incremental Approach to Testing Satisfaction of Functional Dependencies." Unpublished manuscript, The University of Toronto, 1981.

[C3] Chan, E.P.F. "On Finding Unique Minimal Connections for γ-acyclic Schemes." *TR-86-1*, Department of Computing Science, The University of Alberta.

[CA] Chan, E.P.F., Atzeni, P. "On the Properties and Characterization of Connection-trap-free Schemes." *Proc. ACM PODS 1986*, pp. 140-147.

[CM] Chan, E.P.F., Mendelzon, A.O. "Answering Queries on Embedded-complete Database Schemes." To appear in *JACM*.

[Cod1] Codd, E.F. "A Relational Model for Large Shared Data Banks." *CACM 13*, 6, June 1970, pp. 377-387.

[Cod2] Codd, E.F. "Further Normalization of the Data Base Relational Model." *Database Systems*, (R. Rustin ed.) pp. 33-64, Prentice Hall 1972.

[Da] Date, C.J. *An Introduction to Database Systems*. 3rd edition, Reading, Ma., Addison-Wesley, 1981.

[DM] D'Atri, A., Moscarini, M. "Acyclic Hypergraphs: Their Recognition and Top-down vs. Bottom-up Generation." *IASI-CNR, R.29*, Rome, Italy, 1982.

[F] Fagin, R. "Hypergraphs and Relational Database Schemes." *JACM 30*, 3, July 1983, pp. 514-550.

[FMU] Fagin, R., Mendelzon, A.O., Ullman, J.D. "A Simplified Universal Relation Assumption and Its Properties." *ACM TODS 7*, 3, 1982, pp. 343-360.

[G] Graham, M. "Functions in Databases." *ACM TODS 8*, 1, March 1983, pp. 81-109.

[GJ] Garey, M.R., Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[GM] Graham, M.H., Mendelzon, A.O. "The Power of Canonical Queries." Unpublished manuscript, September 1983.

[GMV] Graham, M.H., Mendelzon, A.O., Vardi, M.Y. "Notions of Dependency Satisfaction." *JACM 33*, 1, January 1986, pp. 105-129.

[GV] Graham, M.H., Vardi, M.Y. "On The Complexity and Axiomatizability of Consistent Database States." *Proc. ACM PODS 1984*, pp. 281-289.

[GW] Graham, M.H., Wang, K. "Constant Time Maintenance or The Triumph of the fd." *Proc. ACM PODS 1986*, pp. 202-216.

[GY] Graham, M.H., Yannakakis, M. "Independent Database Schemas." *JCSS* 28, 1984, pp. 121-141.

[H1] Honeyman, P. "Extension Joins." *Proc. VLDB 1980*, pp. 239-244.

[H2] Honeyman, P. "Testing Satisfaction of Functional Dependencies." *JACM 29*, 3, July 1982, pp. 668-677.

[HLY] Honeyman, P., Ladner, R.E., Yannakakis, M. "Testing the Universal Instance Assumption." *IPL* 10, 1, February 1980, pp. 14-19.

[IIK] Ito, M., Iwasaki, M., Kasami, T. "Some Results on the Representative Instance in Relational Databases." *SIAM J. of Computing 14*, 2, 1985, pp. 334-354.

[K] Kent, W. "Consequences of Assuming a Universal Relation." *ACM TODS 6*, 4, December 1981, pp. 539-556.

[LeP] LeDoux, C.H., Parker, D.S. "Reflections on Boyce-Codd Normal Form." *Proc. VLDB 1982*, pp. 131-141.

[LO] Lucchesi, C.L., Osborn, S.L. "Candidate Keys for Relations." *JCSS 17*, 2, October 1978, pp. 270-279.

[M] Mendelzon, A.O. "Database States and their Tableaux." *ACM TODS 9*, 2, June 1984, pp. 264-282.

[Ma] Maier, D. *The Theory of Relational Databases*. Computer Science Press, 1983.

[MMS] Maier, D., Mendelzon, A.O., Sagiv, Y. "Testing Implications of Data Dependencies." *ACM TODS 4*, 4, December 1979, pp. 455-469.

[MRW] Maier, D., Rozenshtein, D., Warren, D.S. "Window Functions." Oregon Graduate Center, *CSIE 84-002* (Revised May 1985).

[MUV] Maier, D., Ullman, J.D., Vardi, M.Y. "On the Foundations of the Universal Relation Model." *ACM TODS 9*, 2, June 1984, pp. 283-308.

[NG] Nicholas, J-M, Gallaire, H. "Data Bases: Theory vs Interpretation." *Logic and Data Bases*, Plenum Press, pp. 33-54, 1978.

[O] Osborn, S.L. "Testing for Existence of a Covering Boyce-Codd Normal Form." *IPL 8*, 1, January 1979, pp. 11-14.

[S1] Sagiv, Y. "Can We Use the Universal Instance Assumption Without Using Nulls?" *Proc. ACM SIGMOD 1981*, pp. 108-120.

[S2] Sagiv, Y. "A Characterization of Globally Consistent Databases and their Correct Access Paths." *ACM TODS 8*, 2, June 1983, pp. 266-286.

[S3] Sagiv, Y. "Evaluation of Queries in Independent Database Schemes." Unpublished manuscript, 1984.

[S4] Sagiv, Y. "On Computing Restricted Projections of Representative Instances." *Proc. ACM PODS 1985*, pp. 171-180.

[St] Stonebraker, M.R. "Implementation of Integrity Constraints and Views by Query Modification." *Proc. ACM SIGMOD 1975*, pp. 65-78.

[U1] Ullman, J.D. *Principles of Database Systems*. 2nd edition, Computer Science Press, 1982.

[U2] Ullman, J.D. "The U. R. Strikes Back." *Proc. ACM PODS 1982*, pp. 10-22.

[V] Vassiliou, Y. "A Formal Treatment of Imperfect Information in Data Management." *CSRG TR-123*, The University of Toronto, Nov. 1980.

[Y1] Yannakakis, M. "Algorithms for Acyclic Database Schemes." *Proc. VLDB 1981*, pp. 82-94.

[Y2] Yannakakis, M. private communication, cited in [F].