Chapter 4

FUNCTIONAL DEPENDENCIES

Two primary purposes of databases are to attenuate data redundancy and enhance data reliability. Any *a priori* knowledge of restrictions or constraints on permissible sets of data has considerable usefulness in reaching these goals, as we shall see. Data dependencies are one way to formulate such advance knowledge. In this chapter we shall cover one type of data dependency, the functional dependency. In Chapter 7 we cover two other types of data dependencies, the multivalued and join dependencies. Other general classes of data dependencies are treated in Chapter 14.

4.1 DEFINITIONS

We discussed keys in Chapter 1. Functional dependencies are a generalization. Table 4.1 depicts the relation *assign*(PILOT FLIGHT DATE DEPARTS). *Assign* tells which pilot flies a given flight on a given day, and what time the flight leaves. Not every combination of pilots, flights, dates, and times is allowable in *assign*. The following restrictions apply, among others.

1. For each flight there is exactly one time.
2. For any given pilot, date, and time, there is only one flight.
3. For a given flight and date, there is only one pilot.

These restrictions are examples of *functional dependencies*. Informally, a functional dependency occurs when the values of a tuple on one set of attributes uniquely determine the values on another set of attributes. Our restrictions can be phrased as

1. TIME functionally depends on FLIGHT,
2. FLIGHT functionally depends on {PILOT, DATE, TIME}, and
3. PILOT functionally depends on {FLIGHT, DATE}.

Table 4.1 The relation *assign*(PILOT FLIGHT DATE DEPARTS).

assign(PILOT	FLIGHT	DATE	DEPARTS)
Cushing	83	9 Aug	10:15a
Cushing	116	10 Aug	1:25p
Clark	281	8 Aug	5:50a
Clark	301	12 Aug	6:35p
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Chin	116	12 Aug	1:25p
Copely	281	9 Aug	5:50a
Copely	281	13 Aug	5:50a
Copely	412	15 Aug	1:25p

We generally reverse the order of the two sets and write FLIGHT, DATE *functionally determines* PILOT, or {FLIGHT, DATE}→PILOT. (Recall that we let a single attribute A stand for $\{A\}$.)

We now state the notion formally using our relational operators. Let r be a relation on scheme R, with X and Y subsets of R. Relation r *satisfies* the *functional dependency* (FD) $X \rightarrow Y$ if for every X-value x, $\pi_Y(\sigma_{X=x}(r))$ has at most one tuple. One way to interpret this expression is to look at two tuples, t_1 and t_2, in r. If $t_1(X) = t_2(X)$, then $t_1(Y) = t_2(Y)$. In the FD $X \rightarrow Y$, X is called the *left side* and Y is called the *right side*.

This interpretation of functional dependency is the basis for the algorithm SATISFIES given below.

Algorithm 4.1 SATISFIES
Input: A relation r and an FD $X \rightarrow Y$.
Output: *true* if r satisfies $X \rightarrow Y$, *false* otherwise.
SATISFIES(r, $X \rightarrow Y$);

```
1. Sort the relation $r$ on its $X$ columns to bring tuples with equal $X$-values
   together.
2. If each set of tuples with equal $X$-values has equal $Y$-values, return
   true. Otherwise, return false.
```

SATISFIES tests if a relation r satisfies an FD $X \rightarrow Y$. Table 4.2 shows the result of running SATISFIES(*assign*, FLIGHT → DEPARTS) on the relation *assign* from Table 4.1. The dashed lines mark off sets of tuples with equal FLIGHT-values. The DEPARTS-values for each set are the same, so the FD is satisfied.

Table 4.2 The result of running the algorithm SATISFIES on the relation *assign* from Table 4.1.

assign(PILOT	FLIGHT	DATE	DEPARTS)
Cushing	83	9 Aug	10:15a
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Cushing	116	10 Aug	1:25p
Chin	116	12 Aug	1:25p
Clark	281	8 Aug	5:50a
Copely	281	9 Aug	5:50a
Copely	281	13 Aug	5:50a
Clark	301	12 Aug	6:35p
Copely	412	15 Aug	1:25p

Table 4.3 shows the result of running SATISFIES(*assign*, DEPARTS → FLIGHT). There is a set of tuples with equal DEPARTS-values that does not have equal FLIGHT-values, so the FD is not satisfied by *assign*.

There are two extreme cases to consider, namely $X \rightarrow \emptyset$ and $\emptyset \rightarrow Y$. The FD $X \rightarrow \emptyset$ is trivially satisfied by any relation. The FD $\emptyset \rightarrow Y$ is satisfied by those relations in which every tuple has the same Y-value. In the sequel, we shall usually ignore FDs of these forms.

4.2 INFERENCE AXIOMS

For a relation $r(R)$, at any given moment there is some family of FDs F that r satisfies. We encounter the same problem we had with keys. One state of a relation may satisfy a certain FD, while another state does not. We want the family of FDs F that all permissible states of r satisfy. Finding F requires

Table 4.3 The result of running SATISFIES(*assign*, DEPARTS → FLIGHT)

assign(PILOT	FLIGHT	DATE	DEPARTS)
Clark	281	8 Aug	5:50a
Copely	281	9 Aug	5:50a
Copely	281	13 Aug	5:50a
Cushing	83	9 Aug	10:15a
Clark	83	11 Aug	10:15a
Chin	83	13 Aug	10:15a
Cushing	116	10 Aug	1:25p
Chin	116	12 Aug	1:25p
Copely	412	15 Aug	1:25p
Clark	301	12 Aug	6:35p

semantic knowledge of the relation r. We can also consider a family of FDs F applying to the relation scheme R. In this case, any relation $r(R)$ must satisfy all the FDs of F. It is not always clear which begets the other, the set of permissible states of a relation or the FDs on the relation scheme.

The number of FDs that can apply to a relation $r(R)$ is finite, since there is only a finite number of subsets of R. Thus it is always possible to find all the FDs that r satisfies, by trying all possibilities using the algorithm SATISFIES. This approach is time-consuming. Knowing some members of F, it is often possible to infer other members of F. A set F of FDs *implies* the FD $X \rightarrow Y$, written $F \models X \rightarrow Y$, if every relation that satisfies all the FDs in F also satisfies $X \rightarrow Y$. An *inference axiom* is a rule that states if a relation satisfies certain FDs, it must satisfy certain other FDs.

We now introduce six inference axioms for FDs. In the statement of the rules, r is a relation on R and W, X, Y, and Z are subsets of R.

F1. Reflexivity
The relation $\pi_X(\sigma_{X=x}(r))$ always has at most one tuple, so $X \rightarrow X$ always holds in r.

F2. Augmentation

This axiom deals with augmenting the left side of an FD. If r satisfies $X \to Y$, then $\pi_Y(\sigma_{X=x}(r))$ has at most one tuple for any X-value x. If Z is any subset of R, then $\sigma_{XZ=xz}(r) \subseteq \sigma_{X=x}(r)$ and hence

$$\pi_Y(\sigma_{XZ=xz}(r)) \subseteq \pi_Y(\sigma_{X=x}(r)).$$

Thus $\pi_Y(\sigma_{XZ=xz}(r))$ has at most one tuple and r must satisfy $X\,Z \to Y$.

Example 4.1 Consider relation r below. Relation r satisfies the FD $A \to B$,

$r(A$	B	C	$D)$
a_1	b_1	c_1	d_1
a_2	b_2	c_1	d_1
a_1	b_1	c_1	d_2
a_3	b_3	c_2	d_3

and hence the FDs $A\,B \to B$, $A\,C \to B$, $A\,D \to B$, $A\,B\,C \to B$, $A\,B\,D \to B$, $A\,C\,D \to B$, and $A\,B\,C\,D \to B$, by axiom F2.

F3. Additivity

This axiom allows us to combine two FDs with the same left sides. If r satisfies $X \to Y$ and $X \to Z$ then $\pi_Y(\sigma_{X=x}(r))$ and $\pi_Z(\sigma_{X=x}(r))$ both have at most one tuple for any X-value x. If $\pi_{YZ}(\sigma_{X=x}(r))$ had more than one tuple, then at least one of $\pi_Y(\sigma_{X=x}(r))$ and $\pi_Z(\sigma_{X=x}(r))$ would have more than one tuple. Thus, r satisfies $X \to Y\,Z$.

Example 4.2 In the relation of Example 4.1, r satisfies $A \to B$ and $A \to C$. By axiom F3, r must also satisfy $A \to B\,C$.

F4. Projectivity

This axiom is more or less the reverse of additivity. If r satisfies $X \to Y\,Z$, then $\pi_{YZ}(\sigma_{X=x}(r))$ has at most one tuple for any X-value x. Since $\pi_Y(\pi_{YZ}(\sigma_{X=x}(r))) = \pi_Y(\sigma_{X=x}(r))$, $\pi_Y(\sigma_{X=x}(r))$ can have at most one tuple. Hence r satisfies $X \to Y$.

Example 4.3 In the relation of Example 4.1, r satisfies $A \to B\,C$. By axiom F4, r must also satisfy $A \to B$ and $A \to C$.

F5. Transitivity

This axiom and the next are the most powerful of the inference axioms. Let r satisfy $X \rightarrow Y$ and $Y \rightarrow Z$. Consider tuples t_1 and t_2 in r. We know that if $t_1(X) = t_2(X)$, then $t_1(Y) = t_2(Y)$ and also if $t_1(Y) = t_2(Y)$, then $t_1(Z) = t_2(Z)$. Therefore, if $t_1(X) = t_2(X)$, then $t_1(Z) = t_2(Z)$, so r satisfies $X \rightarrow Z$.

Example 4.4 Relation r shown below satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. By axiom F5, r satisfies $A \rightarrow C$.

$r(A$	B	C	$D)$
a_1	b_1	c_2	d_1
a_2	b_2	c_1	d_2
a_3	b_1	c_2	d_1
a_4	b_1	c_2	d_3

F6. Pseudotransitivity

Let r satisfy the FDs $X \rightarrow Y$ and $Y Z \rightarrow W$ and let t_1 and t_2 be tuples in r. We know if $t_1(X) = t_2(X)$, then $t_1(Y) = t_2(Y)$ and also if $t_1(Y Z) = t_2(Y Z)$, then $t_1(W) = t_2(W)$. From $t_1(X Z) = t_2(X Z)$ we can deduce that $t_1(X) = t_2(X)$ and so $t_1(Y) = t_2(Y)$ and further $t_1(Y Z) = t_2(Y Z)$, which implies $t_1(W) = t_2(W)$. Thus r satisfies $X Z \rightarrow W$.

To summarize, if W, X, Y, and Z are subsets of R, for any relation r on R:

F1. Reflexivity: $X \rightarrow X$.
F2. Augmentation: $X \rightarrow Y$ implies $X Z \rightarrow Y$.
F3. Additivity: $X \rightarrow Y$ and $X \rightarrow Z$ imply $X \rightarrow Y Z$.
F4. Projectivity: $X \rightarrow Y Z$ implies $X \rightarrow Y$.
F5. Transitivity: $X \rightarrow Y$ and $Y \rightarrow Z$ imply $X \rightarrow Z$.
F6. Pseudotransitivity: $X \rightarrow Y$ and $Y Z \rightarrow W$ imply $X Z \rightarrow W$.

4.3 APPLYING THE INFERENCE AXIOMS

Using the axioms F1 to F6 it is possible to derive other inference rules for FDs.

Example 4.5 Let r be a relation on R with X and Y subsets of R. Axiom F1 says that r satisfies $Y \rightarrow Y$. Applying axiom F2 we get r satisfies $X Y \rightarrow Y$. Another way to state this rule is that for $Y \subseteq X \subseteq R$, r satisfies $X \rightarrow Y$.

Example 4.6 Let r be a relation on R with X, Y, and Z subsets of R. Suppose r satisfies $X\,Y \rightarrow Z$ and $X \rightarrow Y$. By axiom F6 we get r satisfies $X\,X \rightarrow Z$, which simplifies to $X \rightarrow Z$.

To disprove a conjecture about FDs, all we need to do is exhibit a relation where the conjecture does not hold.

Example 4.7 We want to disprove the conjecture $X\,Y \rightarrow Z\,W$ implies $X \rightarrow Z$. The relation r below satisfies $A\,B \rightarrow C\,D$, but $A \not\rightarrow C$.

$r(A$	B	C	$D)$
a	b	c	d
a	b'	c'	d

Some of the inference axioms can be derived from the others. For example, F5, transitivity, is a special case of F6, pseudotransitivity, where $Z = \emptyset$. F6 follows from F1, F2, F3, and F5: if $X \rightarrow Y$ and $Y\,Z \rightarrow W$, then by F1, $Z \rightarrow Z$. By F2, $X\,Z \rightarrow Y$, and $X\,Z \rightarrow Z$. Using F3, we get $X\,Z \rightarrow Y\,Z$. Finally, applying F5 we get $X\,Z \rightarrow W$.

We shall see in the next section that axioms F1 to F6 are complete; that is, every FD that is implied by a set F of FDs can be derived from the FDs in F by one or more applications of these axioms. We have shown that each axiom is correct, so applying the axioms to FDs in a set F can only yield FDs that are implied by F.

Given axioms F1, F2, and F6, we can prove the rest. We have just seen that F5 is a special case of F6. Given $X \rightarrow Y$ and $X \rightarrow Z$, we use F1 to get $Y\,Z \rightarrow Y\,Z$ and apply F6 twice, first to get $X\,Z \rightarrow Y\,Z$ and then to get $X \rightarrow Y\,Z$. Therefore, F3 follows from F1, F2, and F6. To prove F4, suppose $X \rightarrow Y\,Z$. By F1, $Y \rightarrow Y$, and by F2, $Y\,Z \rightarrow Y$. Applying F6 yields $X \rightarrow Y$. Thus axioms F1, F2, and F6 are a complete subset of F1 to F6. Axioms F1, F2, and F6 are also *independent*: no one of the axioms can be proved from the other two (see Exercise 4.5). These three axioms are sometimes called *Armstrong's axioms*, although they are not very similar to Armstrong's original axioms (but the name has a nice ring to it).

Let F be a set of FDs for a relation $r(R)$. The *closure* of F, written F^+, is the smallest set containing F such that Armstrong's axioms cannot be applied to the set to yield an FD not in the set. Since F^+ must be finite, we can compute it by starting with F, applying F1, F2, and F6, and adding the derived FDs to F until no new FDs can be derived. The closure of F depends on the scheme R. If $R = A\,B$, then F^+ will always contain $B \rightarrow B$, but if $R = A\,C$, F^+ never contains $B \rightarrow B$. When R is not explicitly defined, it is assumed to be the set of all attribute symbols used in the FDs of F.

The set F *derives* an FD $X \rightarrow Y$ if $X \rightarrow Y$ is in F^+. Since our inference axioms are correct, if F derives $X \rightarrow Y$, then F implies $X \rightarrow Y$. In the next section we prove the converse. Note that $F^+ = (F^+)^+$ (see Exercise 4.6).

Example 4.8 Let $F = \{A\ B \rightarrow C,\ C \rightarrow B\}$ be a set of FDs on $r(A\ B\ C)$. $F^+ = \{A \rightarrow A,\ A\ B \rightarrow A,\ A\ C \rightarrow A,\ A\ B\ C \rightarrow A,\ B \rightarrow B,\ A\ B \rightarrow B,\ B\ C \rightarrow B,\ A\ B\ C \rightarrow B,\ C \rightarrow C,\ A\ C \rightarrow C,\ B\ C \rightarrow C,\ A\ B\ C \rightarrow C,\ A\ B \rightarrow A\ B,\ A\ B\ C \rightarrow A\ B,\ A\ C \rightarrow A\ C,\ A\ B\ C \rightarrow A\ C,\ B\ C \rightarrow B\ C,\ A\ B\ C \rightarrow B\ C,\ A\ B\ C \rightarrow A\ B\ C,\ A\ B \rightarrow C,\ A\ B \rightarrow A\ C,\ A\ B \rightarrow B\ C,\ A\ B \rightarrow A\ B\ C,\ C \rightarrow B,\ C \rightarrow B\ C,\ A\ C \rightarrow B,\ A\ C \rightarrow A\ B\}$.

In Chapter 5 we shall see more succinct ways to express F^+.

4.4 COMPLETENESS OF THE INFERENCE AXIOMS

We wish to show that axioms F1 to F6 allow us to infer all the FDs implied by a set F of FDs.* That is, if F implies $X \rightarrow Y$, then F derives $X \rightarrow Y$. To prove this result, we shall show how to construct, for any F, a relation r that satisfies every FD in F^+ but no others.

Definition 4.1 $X \rightarrow Y$ is an FD *over* scheme R if X and Y are both subsets of R. F is a set of FDs *over* R if every FD in F is an FD *over* R.

Definition 4.2 If F is a set of FDs over R and G is the set of all possible FDs over R, then $F^- = G - F^+$. F^- is the *exterior* of F.

Definition 4.3 An FD $X \rightarrow Y$ is *trivial* if $X \supseteq Y$. If $X \rightarrow Y$ is a trivial FD over R, then any relation $r(R)$ satisfies $X \rightarrow Y$.

If F is a set of FDs over R and X is a subset of R, then there is an FD $X \rightarrow Y$ in F^+ such that Y is *maximal*: for any other FD $X \rightarrow Z$ in F^+, $Y \supseteq Z$. This result follows from additivity. The right side Y is called the *closure* of X and is denoted by X^+. The closure of X always contains X, by reflexivity.

*For the results of this section we must assume all domains are infinite in order to avoid unwanted combinatorial effects.

Example 4.9 Let $F = \{A \rightarrow D, A\,B \rightarrow D\,E, C\,E \rightarrow G, E \rightarrow H\}$. Then $(A\,B)^+ = A\,B\,D\,E\,H$.

Theorem 4.1 Inference axioms F1 to F6 are complete.

Proof Given a set F of FDs over scheme R, for any FD $X \rightarrow Y$ in F^- we shall exhibit a relation $r(R)$ that satisfies F^+ but not $X \rightarrow Y$. Hence we will know that there are no FDs implied by F that are not derived by F. Relation r will satisfy most of the FDs in F^+ vacuously: for an FD $W \rightarrow Z$ in F^+, there will be no distinct tuples in r with equal W-values.

Let $R = A_1 A_2 \cdots A_n$ and let a_i and b_i be distinct elements of $dom(A_i)$, $1 \leq i \leq n$. There will be only two tuples in r, t, and t'. Tuple t will be $\langle a_1 a_2 \cdots a_n \rangle$. Tuple t' is defined as

$$t'(A_i) = \begin{cases} a_i & \text{if } A_i \in X^+, \\ b_i & \text{otherwise.} \end{cases}$$

First we show that r does not satisfy $X \rightarrow Y$. From the definition of r, $t(X) = t'(X)$. Suppose $t(Y) = t'(Y)$. Then $t'(Y)$ must be all a's, and hence $Y \subseteq$ X^+. But since $X \rightarrow X^+ \in F^+$, by projectivity, $X \rightarrow Y$ is in F^+, a contradiction to $X \rightarrow Y \in F^-$.

Now we show that r satisfies all the FDs in F^+. The only FDs we need worry about are those of the form $W \rightarrow Z$, where $W \subseteq X^+$. If $W \not\subseteq X^+$, then $t(W) \neq t'(W)$. Since $W \subseteq X^+$, by reflexivity and projectivity, $X^+ \rightarrow W$ is in F^+, and by two applications of transitivity, so is $X \rightarrow Z$. Hence $Z \subseteq X^+$ and $t(Z) = t'(Z)$. So r satisfies $W \rightarrow Z$.

Corollary For any set of FDs F over scheme R, there is a relation $r(R)$ satisfying F^+ and violating every FD in F^-. (Such an r is called an *Armstrong relation*.)

Proof For each FD $X \rightarrow Y$ in F^-, use Theorem 4.1 to construct a relation $r_{X,Y}(R)$ that satisfies F^+ but violates $X \rightarrow Y$. Rename the entries in each such relation so that no pair of relations share a common entry. Let

$$r = \bigcup_{X \rightarrow Y \in F^-} r_{X,Y}.$$

It is clear that r violates every FD in F^-. It is left to the reader to show that r satisfies F^+.

We see now that inference axioms F1 to F6 are consistent and complete. Thus $F \models X \rightarrow Y$ if and only if $X \rightarrow Y \in F^+$. From now on we use the terms implies and derives interchangeably when discussing FDs. We generally shall use only Armstrong's axioms or some other complete set of axioms for computing F^+.

4.5 DERIVATIONS AND DERIVATION DAGs

If $F \models X \rightarrow Y$, then either $X \rightarrow Y$ is in F, or a series of applications of the inference axioms to F will yield $X \rightarrow Y$. This sequence of axiom applications and resulting FDs is a *derivation* of $X \rightarrow Y$ from F. More formally, let F be a set of FDs over scheme R. A sequence P of FDs over R is a *derivation sequence on* F if every FD in P either

1. is a member of F, or
2. follows from previous FDs in P by an application of one of the inference axioms F1 to F6.

P is a derivation sequence for $X \rightarrow Y$ if $X \rightarrow Y$ is one of the FDs in P.

Example 4.10 Let $F = \{A\,B \rightarrow E,\ A\,G \rightarrow J,\ B\,E \rightarrow I,\ E \rightarrow G,\ G\,I \rightarrow H\}$. The following sequence is a derivation sequence for $A\,B \rightarrow G\,H$.

1.	$A\,B \rightarrow E$	(given)
2.	$A\,B \rightarrow A\,B$	(reflexivity)
3.	$A\,B \rightarrow B$	(projectivity from 2)
4.	$A\,B \rightarrow B\,E$	(additivity from 1 and 3)
5.	$B\,E \rightarrow I$	(given)
6.	$A\,B \rightarrow I$	(transitivity from 4 and 5)
7.	$E \rightarrow G$	(given)
8.	$A\,B \rightarrow G$	(transitivity from 1 and 7)
9.	$A\,B \rightarrow G\,I$	(additivity from 6 and 8)
10.	$G\,I \rightarrow H$	(given)
11.	$A\,B \rightarrow H$	(transitivity from 9 and 10)
12.	$G\,I \rightarrow G\,I$	(reflexivity)
13.	$G\,I \rightarrow I$	(projectivity from 12)
14.	$A\,B \rightarrow G\,H$	(additivity from 8 and 11)

This sequence contains unneeded FDs, such as 12 and 13, and is also a derivation sequence for other FDs, such as $A\,B \rightarrow G\,I$.

Definition 4.4 Let P be a derivation sequence on F. The *use set* of P is the set of all FDs in F that appear in P.

We have seen that some subsets of the axioms F1 to F6 are complete. Their completeness implies that if there is a derivation sequence P for $X \to Y$ using all the axioms F1 to F6, there is a derivation sequence P' for $X \to Y$ using only the axioms in the complete subset (see Exercise 4.10).

We shall be using a complete set of inference axioms that are not a subset of F1 to F6, called *B-axioms*. For a relation $r(R)$, with W, X, Y, and Z subsets of R, and C an attribute in R:

B1. Reflexivity: $X \to X$.
B2. Accumulation: $X \to Y Z$ and $Z \to C W$ imply $X \to Y Z C$.
B3. Projectivity: $X \to Y Z$ implies $X \to Y$.

The B-axioms are easily shown correct (see Exercise 4.11). We show the B-axioms derive Armstrong's axioms and hence are complete.

F1. Reflexivity: same as B1.

F2. Augmentation: if $X \to Y$, then by B1, $X Z \to X Z$ for any subset Z in R. By repeated application of B2, we get $X Z \to X Y Z$, and B3 gives $X Z \to Y$.

F6. Pseudotransitivity: Let r satisfy $X \to Y$ and $Y Z \to W$. By B1, $X Z \to X Z$. By repeated application of B2, $X Z \to X Y Z$ and $X Z \to W X Y Z$. One application of B3 yields $X Z \to W$.

Since the B-axioms are complete, we can always find a derivation sequence using the B-axioms if $F \models X \to Y$.

Example 4.11 Let F be the set of FDs in Example 4.10. Then

1.	$E I \to E I$	(reflexivity)
2.	$E \to G$	(given)
3.	$E I \to E G I$	(accumulation from 1 and 2)
4.	$E I \to G I$	(projectivity from 3)
5.	$G I \to H$	(given)
6.	$E I \to G H I$	(accumulation from 4 and 5)
7.	$E I \to G H$	(projectivity from 6)
8.	$A B \to A B$	(reflexivity)
9.	$A B \to E$	(given)
10.	$A B \to A B E$	(accumulation from 8 and 9)

11.	$B E \to I$	(given)
12.	$A B \to A B E I$	(accumulation from 10 and 11)
13.	$A B \to A B E G I$	(accumulation from 4 and 12)
14.	$A B \to A B E G H I$	(accumulation from 7 and 13)
15.	$A B \to G H$	(projectivity from 14)

is a derivation sequence for $A B \to G H$ using only the B-axioms.

4.5.1 RAP-Derivation Sequences

Consider derivation sequences for $X \to Y$ on a set of FDs F using the B-axioms that satisfy the following constraints:

1. The first FD is $X \to X$.
2. The last FD is $X \to Y$.
3. Every FD other than the first and last is either an FD in F or an FD of the form $X \to Z$ that was derived using axiom B2.

Such a derivation sequence is called a *RAP-derivation sequence*, for the order in which the axioms are used.

Example 4.12 Let F be the set of FDs in Example 4.10. Then

1.	$A B \to A B$	(B1)
2.	$A B \to E$	(given)
3.	$A B \to A B E$	(B2)
4.	$B E \to I$	(given)
5.	$A B \to A B E I$	(B2)
6.	$E \to G$	(given)
7.	$A B \to A B E G I$	(B2)
8.	$G I \to H$	(given)
9.	$A B \to A B E G H I$	(B2)
10.	$A B \to G H$	(B3)

is a RAP-derivation sequence on F for $A B \to G H$.

Theorem 4.2 Let F be a set of FDs. If there is a derivation sequence on F for $X \to Y$, then there is a RAP-derivation sequence on F for $X \to Y$.

Proof Let P be a derivation sequence on F for $X \rightarrow Y$ using the B-axioms, which must exist by our earlier remarks. Remove all the FDs in P past the first occurrence of $X \rightarrow Y$. P is still a derivation sequence for $X \rightarrow Y$. Insert $X \rightarrow X$ at the head of the sequence, if it is not already there.

We next show that we are able to get by without the FDs in P generated by B3, except for perhaps $X \rightarrow Y$. Let $Z \rightarrow W$ be an FD in P (other than the last) that was derived from $Z \rightarrow V\ W$ by B3. If $Z \rightarrow W$ is not used to derive any FD further along P, then simply remove $Z \rightarrow W$ from P.

If $Z \rightarrow W$ is used to derive an FD further on, it must be by an application of B2 or B3. If $Z \rightarrow W$ is used by B3, it must be to generate an FD $Z \rightarrow W'$ where $W' \subseteq W$. But $Z \rightarrow W'$ can be derived from $Z \rightarrow V\ W$ by B3, so $Z \rightarrow W$ can be removed from P. If $Z \rightarrow W$ is used by B2, it must be in one of two ways:

1. with an FD $W' \rightarrow C\ U$ to derive $Z \rightarrow C\ W$, where $W' \subseteq W$, or
2. with an FD $U \rightarrow Z'$ to derive $U \rightarrow B\ Z'$, where $Z' \supseteq Z$, and B is an attribute in W.

In case 1, use $Z \rightarrow V\ W$ in place of $Z \rightarrow W$ to derive $Z \rightarrow C\ V\ W$ instead of $Z \rightarrow C\ W$. In case 2, $Z \rightarrow V\ W$ can be used in place of $Z \rightarrow W$ to derive $U \rightarrow B\ Z'$. Remove $Z \rightarrow W$ from P in either case.

We have just shown that we can substitute an FD with a larger right side in a derivation using the B-axioms. The only effect is possibly to generate an FD with a larger right side than the FD originally generated, such as in case 1 above, where $Z \rightarrow C\ V\ W$ replaced $Z \rightarrow C\ W$. This change is just another substitution of an FD with a larger right side. Thus, the substitution of FDs with larger right sides can propagate down the derivation sequence.

The only problem that may arise from such substitutions is that $X \rightarrow Y'$, $Y' \supseteq Y$, might be generated as the last FD in P instead of $X \rightarrow Y$, if $X \rightarrow Y$ was derived using B2. In this case, add $X \rightarrow Y$ as the new last FD in P. $X \rightarrow Y$ can be derived from $X \rightarrow Y'$ using B3.

We now have P to the point where it starts with $X \rightarrow X$, ends with $X \rightarrow Y$, and has no FDs derived by B3 except possibly the last. The next step is to show that $X \rightarrow Y$ can be derived using only B2 (except for the first and last FDs in P) applied to FDs of the form $X \rightarrow Z\ W$ and $W \rightarrow C\ V$, where $W \rightarrow C\ V$ is in F. Thus any FDs in P derived by reflexivity are superfluous (except the first) and can be removed.

This portion of the proof is left to the reader and is illustrated only by example here. The gist of the proof is that if a new attribute is introduced into the right side of $X \rightarrow Z$ by B2, it can be introduced directly from some FD in F.

However, it may first be necessary to add other attributes to the right side. Consider the following piece of a derivation sequence, where A is introduced into the right side of $X \rightarrow V\ V'$

$\vdots$

10. $X \rightarrow V\ V'$
11. $Z \rightarrow A\ W$ (given)
12. $V \rightarrow U\ Z$ (given, by B1 or by B2)
13. $V \rightarrow A\ U\ Z$ (from 11 and 12 by B2)
14. $X \rightarrow A\ V\ V'$ (from 10 and 13 by B2)

$\vdots$

We want to get rid of $V \rightarrow A\ U\ Z$ and instead introduce A into the right side of some FD with left side X, using $Z \rightarrow A\ W$ directly. Let $Z = B_1\ B_2\ \ldots\ B_k$. We replace FDs 13 and 14 by

13.1 $X \rightarrow V\ V'\ B_1$ (from 10 and 12 by B2)
13.2 $X \rightarrow V\ V'\ B_1\ B_2$ (from 10 and 12 by B2)
13.3 $X \rightarrow V\ V'\ B_1\ B_2\ B_3$ (from 10 and 12 by B2)

$\vdots$

13.k $X \rightarrow V\ V'\ B_1\ B_2\ \cdots\ B_k$ (from 10 and 12 by B2)
$(= X \rightarrow V\ V'\ Z)$
14. $X \rightarrow A\ V\ V'\ Z$ (from 11 and 13.k by B2)

This change gives us $X \rightarrow A\ V\ V'\ Z$ instead of $X \rightarrow A\ V\ V'$, but we have already seen that substitution of FDs with larger right sides poses no problems.

The basic idea of this part of the proof is to work backwards through P removing applications of B2 that yield FDs where the left side is not X, as shown in the example. Once this transformation is made, all applications of B1 (except the first) become superfluous and can be removed (see Exercise 4.14).

In the next section we shall introduce a pictorial means—a labeled DAG—to depict RAP-derivation sequences. We shall also show that every such graph models a derivation sequence.

4.5.2 Derivation DAGs

A directed acyclic graph (DAG) is a directed graph with no directed paths from any node to itself. A labeled DAG is a DAG with an element from some labeling set L associated with each node.

Definition 4.5 Let F be a set of FDs over scheme R. An F*-based derivation DAG* is a DAG labeled with attribute symbols from R constructed according to the following rules.

R1. Any set of unconnected nodes with labels from R is an F-based derivation DAG.

R2. Let H be an F-based derivation DAG that includes nodes $v_1, v_2, \ldots, v_k$ with labels $A_1, A_2, \ldots, A_k$ and let $A_1 A_2 \cdots A_k \rightarrow CZ$ be an FD in F. Form H' by adding a node u labeled C and edges (v_1, u), (v_2, u), $\ldots, (v_k, u)$ to H. H' is an F-based derivation DAG.

R3. Nothing else is an F-based derivation DAG.

We abbreviate F-based derivation DAG to F-based DDAG.

Example 4.13 Let F be the set of FDs in Example 4.10, namely $\{AB \rightarrow E, AG \rightarrow J, BE \rightarrow I, E \rightarrow G, GI \rightarrow H\}$. Figure 4.1 shows various stages in the construction of an F-based DDAG.

Any F-based DDAG is built by one application of rule R1 and any number of applications of rule R2. R2 insures that the graph constructed is actually a DAG.

Definition 4.6 If H is an F-based DDAG, a node v in H is an *initial node* if v has no incoming edges. Any initial nodes must have been added to H by rule R1.

Definition 4.7 Let H be an F-based DDAG. H is a *DDAG for* X → Y if

D1. X is the set of labels of initial nodes.

D2. Every attribute in Y labels some node in H.

Definition 4.8 The *use set* of an F-based DDAG H, denoted $U(H)$, is the set of all FDs in F used in the application of rule R2 during the construction of the DDAG.*

*Use set is not quite well-defined, since for some sets F, there may be more than one way to construct H. We should really write *a* use set of H, but we won't.

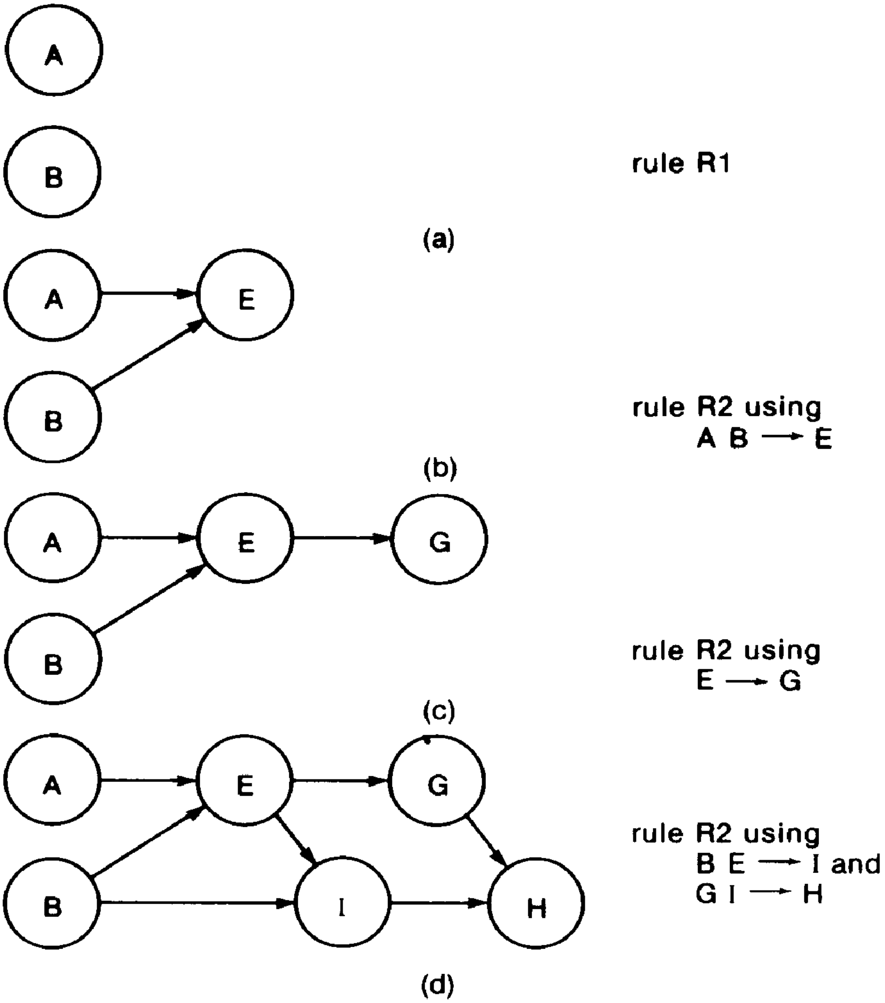


Figure 4.1

Example 4.14 The graph in Figure 4.1(d) is an $\mathcal{F}$-based DDAG for $A\ B \rightarrow G\ H$. Its use set is $\{A\ B \rightarrow E,\ E \rightarrow G,\ B\ E \rightarrow I,\ G\ I \rightarrow H\}$. The initial nodes are the ones labeled A and B.

Example 4.15 Figure 4.2 shows a DDAG for $A\ B\ C \rightarrow A\ B\ C$ for any set of FDs over a scheme R containing A, B, and C. Its use set is $\emptyset$.

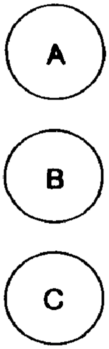


Figure 4.2

Observation Let H be an F-based DDAG with initial nodes labeled with exactly the attributes in some set X and all nodes in the graph labeled with exactly the attributes in some other set Y. If Y' is a subset of Y, then H is a DDAG for $X \rightarrow Y'$.

Theorem 4.3 Given a set of FDs F over R and an FD $X \rightarrow Y$, the following are equivalent.

1. $F \models X \rightarrow Y$.
2. There is a derivation sequence on F for $X \rightarrow Y$.
3. There is an F-based DDAG for $X \rightarrow Y$.

Proof We have already observed the equivalence of 1 and 2 from Theorem 4.1. Theorem 4.2 states that condition 2 is the same as there being a RAP-derivation sequence for $X \rightarrow Y$ on F. We shall show that we can construct an F-based DDAG for $X \rightarrow Y$ given a RAP-derivation sequence for $X \rightarrow Y$, and vice versa.

There is a natural correspondence between the B-axioms and the rules and conditions for an F-based DDAG for $X \rightarrow Y$. Axiom B1 corresponds to rule R1 for constructing DDAGs. Axiom B2 corresponds to rule R2. Axiom B3 is embodied in condition D2 of the definition of a DDAG for $X \rightarrow Y$.

Let P be a RAP-derivation sequence for $X \rightarrow Y$ on F. Let $X \rightarrow Z_1$, $X \rightarrow Z_2$, $\ldots$, $X \rightarrow Z_k$ be all the FDs in P, in order, that have X as the left side. We shall show inductively that we can construct a sequence of F-based DDAGs H_1, H_2, $\ldots$, H_k such that H_i is obtained from H_{i-1} by the rules for constructing DDAGs, and H_i is a DDAG for $X \rightarrow Z_i$.

We know that $X \rightarrow Z_1$ must be $X \rightarrow X$. We use rule R1 to construct DDAG H_1 that consists of unconnected nodes labeled with the attributes from X. Suppose $H_1, H_2, \ldots, H_{i-1}$ are DDAGs for $X \rightarrow Z_1, X \rightarrow Z_2, \ldots, X \rightarrow Z_{i-1}$. Consider $X \rightarrow Z_i$. This FD could have come from one of three places:

1. from F,
2. from FDs $X \rightarrow Z_j$ and $Z \rightarrow C\ W$ by axiom B2. In this case $j < i$, $Z \rightarrow C\ W$ is in F, Z_j contains Z, and $Z_i = C\ Z_j$.
3. From an FD $X \rightarrow Z_j$ by axiom B3. In this case $j < i$, Z_j contains Z_i, and $Z_i = Y$.

In case 1, let $Z_i = B_1 B_2 \cdots B_m$. DDAG H_{i-1} contains DDAG H_1. Apply rule 2 once for each attribute in Z_i (m times) to H_{i-1} to add nodes labeled B_1, B_2, $\ldots$, B_m, and edges to these nodes from nodes labeled with the attributes of X. The result is H_i. In case 2, we know H_{i-1} contains H_j and H_j contains nodes labeled with the attributes in Z_j. Use rule 2 to add a node labeled C to H_{i-1} to form H_i. In case 3, H_j is already a DDAG for $X \rightarrow Z_i$ and so is H_{i-1}, since it contains H_j. Let $H_i = H_{i-1}$.

When the process of constructing the H_i's is completed, H_k will be an F-based DDAG for $X \rightarrow Y$.

Now let H be an F-based DDAG for $X \rightarrow Y$. We construct a RAP-derivation sequence from H. Let $H_1, H_2, \ldots, H_k$ be a sequence of F-based DDAGs such that H_i is constructed from H_{i-1} by rule R2, $2 \le i \le k$, and $H_k = H$. Let Z_i be the set of node labels in H_i. We shall construct a RAP-derivation sequence P with $X \rightarrow Z_1, X \rightarrow Z_2, \ldots, X \rightarrow Z_k$ as a subsequence.

Z_1 must be X and H_1 must be the DDAG with unconnected nodes labeled with the attributes in X. Let P begin with $X \rightarrow X = X \rightarrow Z_1$. Now look at H_i, $i \ge 2$. H_i comes from H_{i-1} by rule 2, using an FD $Z \rightarrow C\ W$ in F, where C is the label of the node added to H_{i-1} and Z_{i-1} contains Z. Thus $Z_i = C\ Z_{i-1}$. If $Z \rightarrow C\ W$ is not in P, add it to the end of P. Then add $X \rightarrow Z_i$ to the end of P. $X \rightarrow Z_i$ can be obtained by axiom B2, using $X \rightarrow Z_{i-1}$ and $Z \rightarrow C\ W$.

When this process terminates, we have a RAP-derivation sequence for $X \rightarrow Z_k$, where Z_k contains Y. Add $X \rightarrow Y$ to the end of P using axiom B3. P is now a RAP-derivation sequence for $X \rightarrow Y$.

Corollary There is an F-based DDAG H for $X \rightarrow Y$ with $U(H) = G$ only if there is a RAP-derivation sequence on F for $X \rightarrow Y$ with use set G.

Proof Immediate from the proof of Theorem 4.3. (Why is this corollary not if and only if?)

Example 4.16 The $\mathcal{F}$-based DDAG in Figure 4.1(d) can be constructed from the RAP-derivation sequence in Example 4.12. The sequence of DDAGs in Figure 4.1(a)-(d) yields the RAP-derivation sequence

1. $A\,B \rightarrow A\,B$
2. $A\,B \rightarrow E$
3. $A\,B \rightarrow A\,B\,E$
4. $E \rightarrow G$
5. $A\,B \rightarrow A\,B\,E\,G$
6. $B\,E \rightarrow I$
7. $A\,B \rightarrow A\,B\,E\,G\,I$
8. $G\,I \rightarrow H$
9. $A\,B \rightarrow A\,B\,E\,G\,H\,I$
10. $A\,B \rightarrow G\,H.$

4.5.3 More about Derivation DAGs

Axiom B2 and rule R2 can both be strengthened in a similar manner. B2 can be strengthened to the following form, where V is also a subset of R.

B2′. $X \rightarrow Y\,Z$ and $Z \rightarrow V\,W$ imply $X \rightarrow V\,Y\,Z$.

The corresponding change in rule R2 is left to the reader (see Exercise 4.18).

Although the definition of DDAG allows multiple nodes with the same label, the freedom is not needed.

Lemma 4.1 Let H be an $\mathcal{F}$-based DDAG for $X \rightarrow Y$. There is an $\mathcal{F}$-based DDAG for $X \rightarrow Y$ wherein every node has a distinct label.

Proof Suppose H has two nodes with the same label, say v_1 and v_2 are both labeled C. In the construction of H, either v_1 and v_2 were added at the same time with rule R1, or one was added later than the other using rule R2. Assume v_2 was added to H at the same time as or later than v_1. There can be no directed path from v_2 to v_1 in H.

In the construction of H, any time rule R2 was applied using v_2, v_1 could have been used instead, as shown in Figure 4.3. Thus there is an $\mathcal{F}$-based DDAG H' for $X \rightarrow Y$ that has the same nodes and labels as H, as well as the same set of initial nodes, but v_2 has no outgoing edges in H'. H' is still a DDAG for $X \rightarrow Y$ when v_2 and its incoming edges are removed, since the set of attributes labeling nodes does not change, and if v_2 is an initial node, so is v_1.

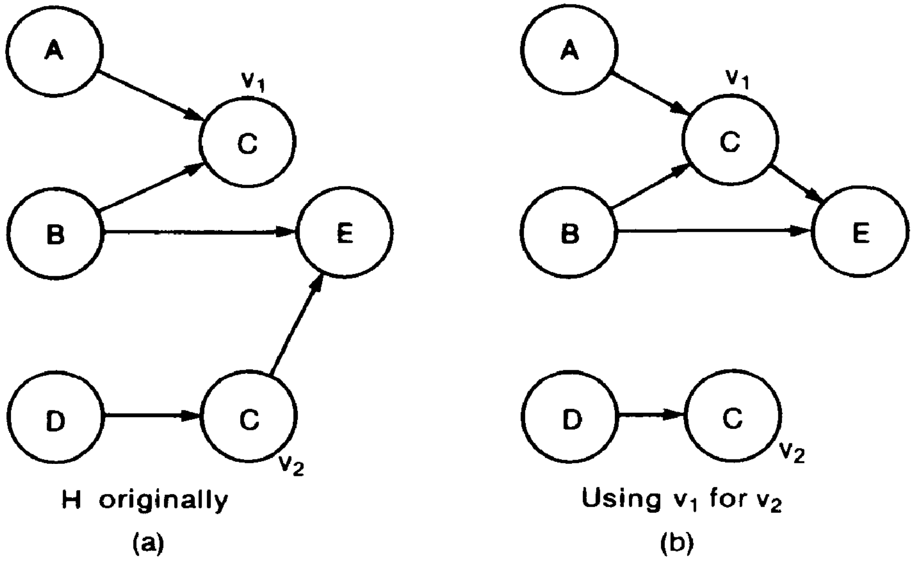


Figure 4.3

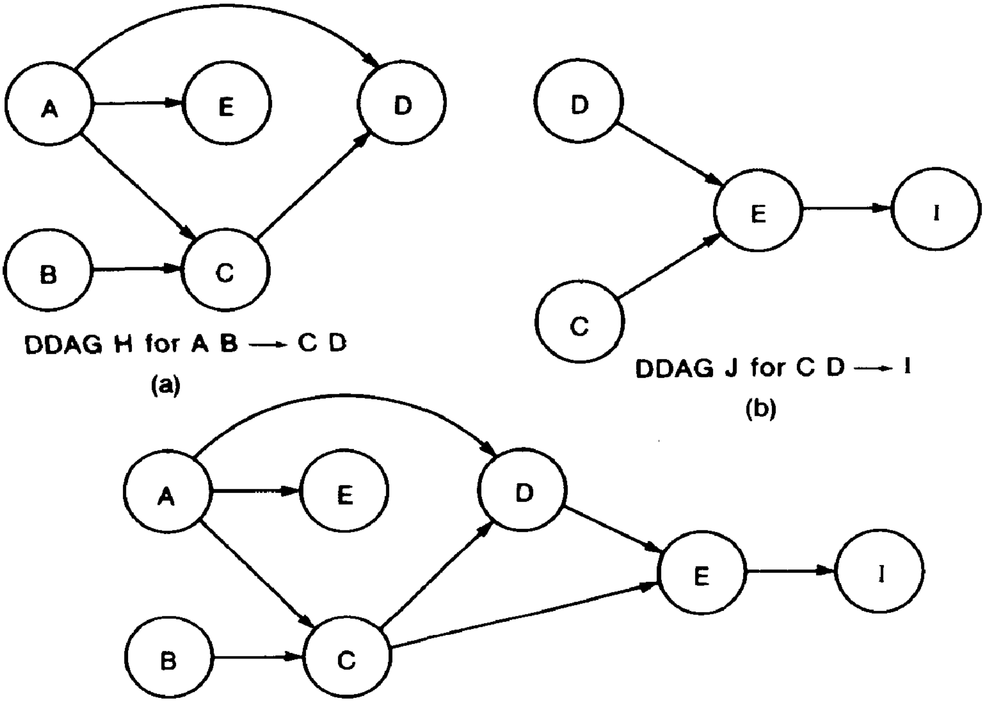
This transformation can be applied iteratively to all pairs of nodes with equal labels to remove all duplicate labels.

We have observed that if H is an F-based DDAG for $X \rightarrow Y$, then it is also an F-based DDAG for $X \rightarrow Y'$, where $Y \supseteq Y'$. Similarly, if $X \subseteq X'$, H is almost a DDAG for $X' \rightarrow Y$. The only problem is that not all the attributes in X' label some initial node in H. This problem can be solved by adding unconnected nodes to H with labels in $X' - X$.

Lemma 4.2 Let H and J be F-based DDAGs for $X \rightarrow Y$ and $Y \rightarrow Z$, respectively. There is an F-based DDAG K for $X \rightarrow Z$ with $U(K) \subseteq U(H) \cup U(J)$.

Proof We *splice* H and J together by overlapping the initial nodes of J with the same-labeled nodes of H. Figure 4.4 gives an example of the overlapping process where $F = \{A \rightarrow E, A\,B \rightarrow C, A\,C \rightarrow D, C\,D \rightarrow E, E \rightarrow I\}$. Notice that $U(H) = \{A \rightarrow E, A\,B \rightarrow C, A\,C \rightarrow D\}$, $U(J) = \{C\,D \rightarrow E, E \rightarrow I\}$, and $U(K) = F$.

Lemma 4.3 If H is an F-based DDAG for $X \rightarrow Y$, and $V \rightarrow W$ is in $U(H)$, then $F \models X \rightarrow V$.



DDAG H for A B → C D

(a)

DDAG J for C D → I

(b)

DDAG K = H overlapped with J

(c)

Figure 4.4

Proof For $V \rightarrow W$ to be used in constructing H, H must contain nodes with labels for every attribute in V. Hence H is an F-based DDAG for $X \rightarrow V$.

Corollary If H is an F-based DDAG for $X \rightarrow Y$ and $V \rightarrow W$ is in $U(H)$, there is an F-based DDAG for $X \rightarrow V$ that does not use $V \rightarrow W$.

Lemma 4.3 does not hold for derivation sequences, since $V \rightarrow W$ could be in the use set of a sequence without it being necessary to derive $X \rightarrow Y$.

4.6 TESTING MEMBERSHIP IN F^+

To determine if a set of FDs $F \models X \rightarrow Y$, we need only test if $X \rightarrow Y \in F^+$. However, as we saw in Example 4.8, F^+ can be considerably larger than F. We would like to find a means to test if $X \rightarrow Y$ is in F^+ without generating all of F^+. In this section we present such a membership algorithm. The core of the algorithm is a procedure that generates the closure of X under F. Once we have found X^+, we can test if F implies $X \rightarrow Y$.

We seek an algorithm for testing membership that is more efficient than generating all of F^+. One way to compare algorithms is to examine the maximum amount of time they consume for an input of a given size. The (worst-case) *time-complexity* of an algorithm is a function $T(n)$ that gives the maximum number of steps the algorithm will take on an input of size n. Naturally, $T(n)$ depends on what is counted as one step of computation. We shall use the RAM (random access machine) model as presented in Aho, Hopcroft, and Ullman as our model of computation. A RAM is basically a model of a simple digital computer with random access memory.

For a particular algorithm, $T(n)$ can be messy and complex, but often there is some "nice" function that approximates the behavior of $T(n)$. We write $T(n) = O(f(n))$ (read "$T(n)$ has order $f(n)$") if there are constants $c > 0$ and $n_1 \geq 0$ such that $T(n) \leq cf(n)$ for all $n \geq n_1$.

Example 4.17 $3n^2 + 2 \log_2 \log_2 n = O(n^2)$, since $3n^2 + 2 \log_2 \log_2 n \leq 4n^2$ for $n \geq 1$. Of course, $3n^2 + 2 \log_2 \log_2 n = O(n^3)$ as well, but we are more interested in the slower growing function, since it is a better approximation of $3n^2 + 2 \log_2 \log_2 n$.

For most algorithms, the time complexity $T(n)$ is at least $O(n)$, since most algorithms read all their input, which takes n steps. We first present a membership algorithm for FDs that is not $O(n)$, but is easy to understand. We then present a version of the algorithm that is more complex, but has $O(n)$ time complexity.

We start with the function CLOSURE given below. CLOSURE(X, F) returns X^+ under F, where X is a set of attributes and F is a set of FDs. *OLDDEP* and *NEWDEP* are variables for sets of attributes.

Algorithm 4.2 CLOSURE
Input: A set of attributes X and a set of FDs F.
Output: The closure of X under F.

```
CLOSURE($X$, $F$)
begin
OLDDEP := $\emptyset$; NEWDEP := $X$;
while NEWDEP $\neq$ OLDDEP do begin
      OLDDEP := NEWDEP;
      for every FD $W \to Z$ in $F$ do
         if NEWDEP $\supseteq W$ then
         NEWDEP := NEWDEP $\cup Z$
   end;
return(NEWDEP)
end.
```

Example 4.18 Let $F = \{A \to D, A\ B \to E, B\ I \to E, C\ D \to I, E \to C\}$. CLOSURE($AE$, F) begins with *NEWDEP* $= A\ E$. On the first pass through F, $A \to D$ is used to add D to *NEWDEP*, and $E \to C$ is used to add C to *NEWDEP*, so *NEWDEP* $= A\ C\ D\ E$ at the end of the **for** loop. The second time through F, $C\ D \to I$ is used to add I to *NEWDEP*, so *NEWDEP* $=$ $A\ C\ D\ E\ I$ at the end of the **for** loop. The next pass through F causes no changes in *NEWDEP*, so $A\ C\ D\ E\ I$ is returned as $(A\ E)^+$.

The algorithm essentially constructs an F-based DDAG for $X \to X^+$, using a modified version of rule R2 in the definition of DDAG where more than one node is added at a time (see Exercise 4.18). We start with initial nodes labeled X and keep adding nodes to the DDAG until no new labels can be added. It is not necessary to record the edges of the DDAG, however, since whether or not we can use an FD $W \to Z$ depends only on there being nodes in the DDAG with labels for all the attributes in W. The value of X^+ does not depend on the edges in the DDAG either, just on the final set of node labels. Thus, it suffices to keep track of only the set of node labels in the DDAG during its construction. We keep track of the labels in *OLDDEP* and *NEWDEP*.

Since we are constructing an F-based DDAG with initial nodes labeled X, it follows that at any point in the execution of CLOSURE, *NEWDEP* $\subseteq X^+$. For any attribute A in X^+, A will eventually be added to *NEWDEP*. Since A is in X^+, $F \models X \to A$, and there must be an F-based DDAG H for $X \to A$. Any FD $W \to Z$ used in constructing H can eventually be used in the construction of the DDAG for the algorithm, and the DDAG in the algorithm will contain labels for every attribute in Z. Therefore A will be added to *NEWDEP* and we conclude that CLOSURE correctly computes X^+.

Using CLOSURE, it is simple to devise an algorithm to test membership in F^+. Algorithm 4.3 MEMBER performs this test.

```
Algorithm 4.3 MEMBER
Input: A set of FDs $F$ and an FD $X \rightarrow Y$.
Output: true if $F \models X \rightarrow Y$, false otherwise.
MEMBER($F$, $X \rightarrow Y$)
  begin
  if $Y \subseteq$ CLOSURE($X$, $F$) then return(true) else return(false)
  end.
```

The time complexity for MEMBER is the same as the time complexity for CLOSURE, since CLOSURE makes up the body of MEMBER. The worst case for CLOSURE occurs when only one new right side of an FD is added to *NEWDEP* for each execution of the **for** loop. If $F = \{A_2 \rightarrow A_1, A_3 \rightarrow A_2, A_4 \rightarrow A_3, \ldots, A_m \rightarrow A_{m-1}\}$, in the computation for CLOSURE(A_m, F), only attribute A_{m-i}, is added to *NEWDEP* on the i^{th} execution of the **for** loop. If a is the number of different attribute symbols in F and p is the number of FDs in F, then each execution of the **for** loop takes $O(ap)$ time, since a steps are required to test containment of two sets over a elements. The **while** loop can be executed p times before no changes occur to *NEWDEP*. Therefore, the time complexity of CLOSURE, and hence of MEMBER, is $O(ap^2)$. Note that the length of the input, n, is $O(ap)$ (see Exercise 4.21).

To see how the time complexity of CLOSURE can be improved, observe that during the execution of CLOSURE, if for some FD $W \rightarrow Z$, W is contained in *NEWDEP*, then Z is added to *NEWDEP* and $W \rightarrow Z$ is of no further use. At this point we could exclude $W \rightarrow Z$ from F and still compute the correct closure. By excluding FDs from F after their right sides are added to *NEWDEP*, we can reduce the number of FDs scanned during each execution of the **for** loop. We can save even more time if we also know which FDs in F currently have their left sides contained in *NEWDEP*. If such information is available we can consider an FD $W \rightarrow Z$ in F only when its left side is contained in *NEWDEP* and then remove it from subsequent consideration. Thus every FD in F would be considered only once. If each FD in F can be processed in time proportional to its length in attribute symbols, we would have an $O(n)$ membership algorithm, where n is the number of symbols required to represent F and $X \rightarrow Y$.

We accomplish these ends as follows. For each FD $W \rightarrow Z$ in F we shall keep track of the number of attributes in W that are not in *NEWDEP*. When this count becomes zero, it will be time to consider $W \rightarrow Z$. To decrement the count properly for each FD when a new attribute A is added to *NEWDEP*, it is necessary to access all FDs with attribute A on their left sides. We therefore maintain a series of lists, one for each attribute, consisting of all FDs in F with that attribute on the left side. Whenever an attribute is added to *NEWDEP*, the list for that attribute is traversed, and all FDs on the list have their counts decremented by 1. If some FD $W \rightarrow Z$ on the list has its count decremented to zero, then W is a subset of *NEWDEP* and Z is added to *NEWDEP*.

We must be careful when adding Z to *NEWDEP*. Suppose there is an attribute A in Z that is already in *NEWDEP*. If we traverse the list for A a second time, we get erroneous values for the counts of FDs on the list. To prevent this problem, we keep a set of attributes called *UPDATE* that is the subset of *NEWDEP* consisting of attributes that have not yet had their lists traversed. When an attribute is added to *NEWDEP* for the first time, it is also added to *UPDATE* until its list can be traversed. *UPDATE* allows us to do away with *OLDDEP*, since when $UPDATE = \emptyset$, there are no more FDs that can be used to add new attributes to *NEWDEP*.

In the algorithm LINCLOSURE, below, there is an array *COUNT* of integers containing the counts for each FD in F, and an array *LIST* of lists of FDs for each attribute symbol in F. While an FD may seem to occur in lists for more than one attribute, we actually store only one copy of the FD and have the various lists point to this copy.

Algorithm 4.4 The function LINCLOSURE
Input and Output: identical to CLOSURE in Algorithm 4.2

```
LINCLOSURE($X$, $F$)
1. Initialization
     for each FD $W \rightarrow Z$ in $F$ do begin
       $COUNT[W \rightarrow Z] := |W|$;
       for each attribute $A$ in $W$ do add $W \rightarrow Z$ to $LIST[A]$
       end;
     $NEWDEP := X$; $UPDATE := X$.
2. Computation
     while $UPDATE \neq \emptyset$ do begin
       choose an $A$ in $UPDATE$;
       $UPDATE := UPDATE - A$;
       for each FD $W \rightarrow Z$ in $LIST[A]$ do begin
         $COUNT[W \rightarrow Z] := COUNT[W \rightarrow Z] - 1$;
         if $COUNT[W \rightarrow Z] = 0$ then begin
           $ADD := Z - NEWDEP$;
           $NEWDEP := NEWDEP \cup ADD$;
           $UPDATE := UPDATE \cup ADD$
           end
         end
       end.
3. return($NEWDEP$).
```

Example 4.19 Let F be as in Example 4.18. LINCLOSURE($A\ E$, F) initializes $NEWDEP$, $UPDATE$, $COUNT$, and $LIST$ as follows:

$NEWDEP = A\ E \qquad UPDATE = A\ E$
$LIST[A] = A \to D,\ A\ B \to E \qquad COUNT[A \to D] = 1$
$LIST[B] = B\ I \to E,\ A\ B \to E \qquad COUNT[A\ B \to E] = 2$
$LIST[C] = C\ D \to I \qquad COUNT[B\ I \to E] = 2$
$LIST[D] = C\ D \to I \qquad COUNT[C\ D \to I] = 2$
$LIST[E] = E \to C \qquad COUNT[E \to C] = 1$
$LIST[I] = B\ I \to E$

We select the A in $UPDATE$ and traverse $LIST[A]$. $COUNT[A \to D]$ goes to 0 and D is added to $NEWDEP$ and $UPDATE$. $COUNT[A\ B \to E]$ goes to 1. If we next select E from $UPDATE$ the result is

$NEWDEP = A\ C\ D\ E \qquad UPDATE = C\ D$
$COUNT[A \to D] = 0$
$COUNT[A\ B \to E] = 1$
$COUNT[B\ I \to E] = 2$
$COUNT[C\ D \to I] = 2$
$COUNT[E \to C] = 0.$

Traversing the lists for C and D leaves us with
$NEWDEP = A\ C\ D\ E\ I \qquad UPDATE = I$
$COUNT[A \to D] = 0$
$COUNT[A\ B \to E] = 1$
$COUNT[B\ I \to E] = 2$
$COUNT[C\ D \to I] = 0$
$COUNT[E \to C] = 0$

Traversing the list for I fails to reduce any counts to 0, so the algorithm returns $A\ C\ D\ E\ I$.

Let us review the workings of the computation step of LINCLOSURE. The **while** loop continues to execute while there are attributes in $NEWDEP$ whose lists have not been traversed. We choose one such attribute and traverse its list. For each FD $W \to Z$ in the list, we reduce $COUNT[W \to Z]$. If the count goes to 0, it is time to consider $W \to Z$. We compute the set of attributes in Z that are not already in $NEWDEP$ and add these attributes to both $NEWDEP$ and $UPDATE$. The **while** loop stops executing when there are no more FDs whose counts can be reduced.

Theorem 4.4 LINCLOSURE has time complexity $O(n)$ for input of length n.

Proof Computing $COUNT[W \rightarrow Z]$ in the initialization step takes time proportional to $|W|$ if W is represented as a list of attributes. Computing all the initial values for $COUNT$ therefore takes $O(n)$ time. Each FD $W \rightarrow Z$ in F gets inserted into $|W|$ lists in $LIST$. For an appropriate list representation, adding one FD to one list takes a constant amount of time. Thus, filling in $LIST$ takes $O(n)$ time. $NEWDEP$ and $UPDATE$ can also be initialized in $O(n)$ time.

In the computation step, each attribute is added to $UPDATE$ once, at most. For each attribute A added to $UPDATE$, one pass of the **while** loop is performed. For each pass of the **while** loop, an operation (decrement $COUNT$) is performed for each FD in $LIST[A]$. Since any FD $W \rightarrow Z$ appears in $|W|$ lists, the decrement operation is performed at most

$$\sum_{W \rightarrow Z \text{ in } F} |W|$$

times. Thus $O(n)$ time is spent decrementing COUNT.

For any FD $W \rightarrow Z$ in F, the predicate $COUNT[W \rightarrow Z] = 0$ evaluates to **true** at most once, since once $COUNT[W \rightarrow Z]$ reaches 0, all the attributes in W have been added to $NEWDEP$ and removed from $UPDATE$. Thus, $W \rightarrow Z$ does not appear in any attribute list remaining to be traversed. The computation involving ADD takes time proportional to $|Z|$ if $NEWDEP$ is represented as a bit vector. The total time spent with the statements involving ADD is proportional to

$$\sum_{W \rightarrow Z \text{ in } F} |Z|,$$

which is $O(n)$. Since no step of the algorithm takes more than $O(n)$ time, LINCLOSURE has time complexity $O(n)$.

Corollary Membership in F^+ can be tested in $O(n)$ time for inputs of length n.

Proof Substitute LINCLOSURE for CLOSURE in Algorithm 4.3. Henceforth we shall assume MEMBER uses LINCLOSURE.

4.7 EXERCISES

4.1 Consider the relation r below.

$r(A$	B	C	D	$E)$
a_1	b_1	c_1	d_1	e_1
a_1	b_2	c_2	d_2	d_1
a_2	b_1	c_3	d_3	e_1
a_2	b_1	c_4	d_3	e_1
a_3	b_2	c_5	d_1	e_1

Which of the following FDs does r satisfy?
$A \rightarrow D, A\ B \rightarrow D, C \rightarrow B\ D\ E, E \rightarrow A, A \rightarrow E$

4.2 Prove that r satisfies $X \rightarrow Y$ if and only if X is a key of $\pi_{XY}(r)$.

4.3 Let r be a relation on R, with X a subset of R. Show that if $\pi_X(r)$ has the same number of tuples as r, then r satisfies $X \rightarrow Y$ for any subset Y of R.

4.4 Prove or disprove the following inference rules for a relation $r(R)$ with W, X, Y, and Z subsets of R.
(a) $X \rightarrow Y$ and $Z \rightarrow W$ imply $X\ Z \rightarrow Y\ W$.
(b) $X\ Y \rightarrow Z$ and $Z \rightarrow X$ imply $Z \rightarrow Y$.
(c) $X \rightarrow Y$ and $Y \rightarrow Z$ imply $X \rightarrow Y\ Z$.
(d) $X \rightarrow Y$, $W \rightarrow Z$, and $Y \supseteq W$ imply $X \rightarrow Z$.

4.5 Prove that inference axioms F1, F2, and F6 are independent. That is, no one of them can be proved from the other two.

4.6 Show that for any set of FDs F, $F^+ = (F^+)^+$.

4.7 Suppose F is a set of FDs over scheme R. If $F = \emptyset$, what does F^+ look like?

4.8 For a set of FDs F, show that there is no relation satisfying all the FDs in F^- and no others.

4.9* Show inference axioms F1, F3, F4, and F5 are complete. Is this set of axioms independent?

4.10 Show that if there is a derivation sequence for $X \rightarrow Y$ using inference axioms F1 to F6, then there is a derivation sequence for $X \rightarrow Y$ using only Armstrong's axioms.

4.11 Prove the B-axioms are correct.

4.12 Find a set of two inference rules that is complete. The rules need not be a subset of axioms F1 to F6.

4.13 Let $F = \{A\ B \rightarrow C, B \rightarrow D, C\ D \rightarrow E, C\ E \rightarrow G\ H, G \rightarrow A\}$.
(a) Give a derivation sequence on F for $A\ B \rightarrow E$.
(b) Give a derivation sequence on F for $B\ G \rightarrow C$ using only Armstrong's axioms
(c) Give a RAP-derivation sequence on F for $A\ B \rightarrow G$.

4.14* Complete the proof of Theorem 4.2.

4.15 Let F be as in Exercise 4.13. Construct an F-based DDAG for $A\ B \rightarrow G$.

4.16 Prove that for sets F and G of FDs, if F contains G, then F^+ contains G^+.

4.17 Prove that an invalid inference rule can always be disproved with a two-tuple relation.

4.18 Modify rule R2 in the definition of DDAG to reflect the change from axiom B2 to B2′.

4.19 Let F and G be sets of FDs. Suppose for every FD $Z \rightarrow W$ in F there is a G-based DDAG for $Z \rightarrow W$. Prove that if $X \rightarrow Y$ has an F-based DDAG, it has a G-based DDAG.

4.20 Let F be a set of FDs over R. Find a bound on the size of F^+ in FDs, in terms of the number of attributes in R.

4.21 Let F be a set of FDs where a is the number of distinct attributes in F, p is the number of FDs in F, and n is the number of symbols required to write F. Compare ap^2 and n.

4.22 The algorithm MEMBER (Algorithm 4.3) computes more information than is necessary to ascertain if $F \models X \rightarrow Y$. Once Y is found to be in X^+, the rest of X^+ is immaterial. Modify MEMBER and LINCLOSURE to remove this unnecessary computation.

4.8 BIBLIOGRAPHY AND COMMENTS

FDs were present when Codd [1970] first introduced the relational model, in the form of keys. Codd [1972a] later introduced FDs that do not follow from keys, for the purpose of normalization (see Chapter 6). Delobel and Casey [1973] gave a set of inference axioms, which Armstrong [1974] showed were complete and correct. He also gave a method for constructing an Armstrong relation for a set of FDs. Beeri, Dowd, *et al.* [1980] explore the structure of Armstrong relations.

The LINCLOSURE algorithm is from Beeri and Bernstein [1979]. They used *derivation trees* in their proofs. Derivation trees were the precursor of DDAGs, introduced by Maier [1980b]. An exposition of the RAM model of computation is given by Aho, Hopcroft, and Ullman [1974].

Much work on the implication of FDs has focused on the discovery of keys and the structure of sets of keys; see the papers by Békéssy and Demetrovics [1979]; Békéssy, Demetrovics, *et al.* [1980]; Demetrovics [1978, 1979]; Forsyth and Fadous [1975]; and Lucchesi and Osborn [1978].