Chapter 9

REPRESENTATION THEORY

In this chapter we shall discuss several notions of when a relation is adequately represented by its decomposition onto a database scheme. We also introduce a new type of equivalence for database schemes, data equivalence. We first examine both topics for an arbitrary set of relations **P**, and then state further results for the case where $\mathbf{P} = SAT(\mathbf{C})$.

9.1 NOTIONS OF ADEQUATE REPRESENTATION

We state here some notations that will be used throughout the chapter. We want to represent members of a set **P** of relations. All relations in **P** are over the scheme **U**. **Q** denotes the set of all relations with scheme **U**. We shall refer to relations in **Q** as *instances*, to avoid confusion with relations that are components of databases. $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ will be a database scheme such that $\mathbf{U} = R_1R_2 \cdots R_p$. Let **M** be the set of all databases over **R**. We want to represent instances in **P** as databases in **M**, so we shall examine the restrictions on **R** necessary for an adequate representation.

In the chapters on normal forms, we were looking for database schemes that eliminated redundancy and gave lossless decompositions. In this chapter, we shall be concerned with enforcing constraints and unique representations. Lossless decomposition frequently will enter into the discussions of the second condition.

We have already seen the project-join mapping defined by **R**, $m_{\mathbf{R}}$. We shall find it useful to separate the projection and join functions.

Definition 9.1 The *project mapping for* **R**, $\pi_{\mathbf{R}}$, maps instances in **Q** to databases in **M**. For $r \in \mathbf{Q}$, we define

$$\pi_{\mathbf{R}}(r) = d$$

where $d = \{r_1, r_2, \ldots, r_p\}$ is the database in **M** such that

$$\pi_{R_i}(r) = r_i, \qquad 1 \leq i \leq p.$$

When it is understood we are projecting instances onto databases, we use π for $\pi_{\mathbf{R}}$.

Definition 9.2 The *join mapping for* **R**, $\bowtie$, maps databases in **M** to instances in **Q**. For database

$$\begin{aligned} d &= \{r_1, r_2, \ldots, r_p\} \text{ in } \mathbf{M}, \\ \bowtie(d) &= r_1 \bowtie r_2 \bowtie \cdots \bowtie r_p. \end{aligned}$$

Note that the description of **R** is implicit in the structure of d.

Example 9.1 Let $\mathbf{U} = ABC$ and $\mathbf{R} = \{AB, BC\}$. If r is the instance in Figure 9.1, then $\pi_{\mathbf{R}}(r)$ is the database $d = \{r_1, r_2\}$, where r_1 and r_2 are given in Figure 9.2. The result of $\bowtie(d)$ is the instance r' in Figure 9.3. Clearly $\pi_{\mathbf{R}}(r') = d$. If $d' = \{r_1, r_2'\}$, where r_2' is given in Figure 9.4, then $\bowtie(d')$ is the empty instance over **U**.

$r(A\ B\ C)$

1	3	5
1	4	6
2	4	5
2	4	6

Figure 9.1

$r_1(A\ B)$

1	3
1	4
2	4

$r_2(B\ C)$

3	5
4	5
4	6

Figure 9.2

$r'(A\ B\ C)$

1	3	5
1	4	5
1	4	6
2	4	5
2	4	6

Figure 9.3

$r_2'(B$	$C)$
7	5
8	6

Figure 9.4

Definition 9.3 For any subset **P** of **Q**, the *direct image of* **P** *under* **R**, written $\pi_{\mathbf{R}}\mathbf{P}$ (or $\pi\mathbf{P}$ when **R** is understood), is defined as

$$\pi_{\mathbf{R}}\mathbf{P} = \{d \in \mathbf{M} \mid d = \pi_{\mathbf{R}}(r) \quad \text{for some } r \in \mathbf{P}\}.$$

If R is a subset of **U**, the *image of* **P** *under* R, written $\pi_R\mathbf{P}$, is the set

$$\{s(R) \mid s = \pi_R(r) \quad \text{for some } r \in \mathbf{P}\}.$$

We see that $\pi\mathbf{P} \subseteq \pi\mathbf{Q} \subsetneq \mathbf{M}$. The last containment is proper because not every database d is $\pi_{\mathbf{R}}(r)$ for some instance $r \in \mathbf{Q}$.

Example 9.2 The database $d' = \{r_1, r_2'\}$, where r_1 is given in Figure 9.2 and r_2' is given in Figure 9.4, is not the projection of any instance on scheme ABC.

We shall only consider databases in $\pi\mathbf{Q}$ as candidates to represent instances in **P**, since we want π to be the mapping from **P** to **M**. The set $\pi\mathbf{Q}$ is exactly all those databases with scheme **R** that join completely (see Exercise 9.2). There is some controversy involved with this assumption, for three reasons.

1. It is an NP-complete problem to decide if a set of relations joins completely.
2. Databases where the relations do not join completely can still be meaningful.

It is sometimes desirable to store partial information.

Example 9.3 The database consisting of the relations *assigned* and *canfly* in Table 9.1 does not join completely, because Bentley is not assigned to any flight. We still might want to record the information that Bentley can fly a 727, even if he is not currently piloting any flight. However, we cannot represent that piece of information in an instance with scheme {PILOT, FLIGHT#, PLANE-TYPE} without using some special value in the FLIGHT# column.

3. The constraint that a database d is in $\pi\mathbf{Q}$ can only be checked by looking at the database as a whole. It is not sufficient to consider the relations in the database individually, or even in pairs.

Table 9.1 The relations *assigned* and *canfly*, constituting a database.

assigned (PILOT	FLIGHT#)	*canfly* (PILOT	PLANE-TYPE)
Bottom	62	Bentley	727
Brown	113	Bottom	727
Brown	114	Bottom	DC8
		Brown	727
		Brown	737

Example 9.4 The relations r_1, r_2 and r_3 in Figure 9.5 join completely when taken in pairs, but the three relations together do not join completely.

$r_1(A$	$B)$	$r_2(B$	$C)$	$r_3(A$	$C)$
a	b	b	c	a	c'
a'	b'	b'	c'	a'	c

Figure 9.5

We shall cover these three objections in further detail in Chapter 12.

Apart from the joinability condition, we want to enforce constraints on the databases in $\pi\mathbf{Q}$ by enforcing constraints on individual relations in the database. We do not want to construct the instance in $\mathbf{Q}$ we are representing every time we need to test a constraint.

Definition 9.4 For any subset $\mathbf{P}$ of $\mathbf{Q}$, the *component-wise image of* $\mathbf{P}$ *under* $\mathbf{R}$, written $CW_{\mathbf{R}}(\mathbf{P})$, is the set

$$\{d \in \mathbf{M} \mid d = \{r_1, r_2, \ldots, r_p\}, r_i \in \pi_{R_i}(\mathbf{P})\}.$$

That is, $CW_{\mathbf{R}}(\mathbf{P})$ consists of those databases over $\mathbf{R}$ where each relation is the projection of some instance in $\mathbf{P}$, but not necessarily the same instance.

Example 9.5 Clearly, $CW_{\mathbf{R}}(\mathbf{P}) \supseteq \pi\mathbf{P}$. The containment is generally proper. Suppose $\mathbf{U} = ABC$, $\mathbf{R} = \{AB, BC\}$ and $\mathbf{P} = SAT(A \rightarrow C)$. Then the database $d = \{r_1, r_2\}$, where r_1 and r_2 are given in Figure 9.6, is in $CW_{\mathbf{R}}(\mathbf{P})$ but not $\pi\mathbf{P}$.

$r_1(A$	$B)$	$r_2(B$	$C)$
1	2	2	4
1	3	3	5

Figure 9.6

To represent **P**, we are interested in using those databases that are projections of instances, and, component-wise, look like projections of instances in **P**. We call this set of interest **L**, which we define

$$\mathbf{L} = CW_{\mathbf{R}}(\mathbf{P}) \cap \pi\mathbf{Q}.$$

Since $\pi\mathbf{P} \subseteq \pi\mathbf{Q}$, $\pi\mathbf{P} \subseteq \mathbf{L}$. Figure 9.7 diagrams the relationship among sets introduced so far.

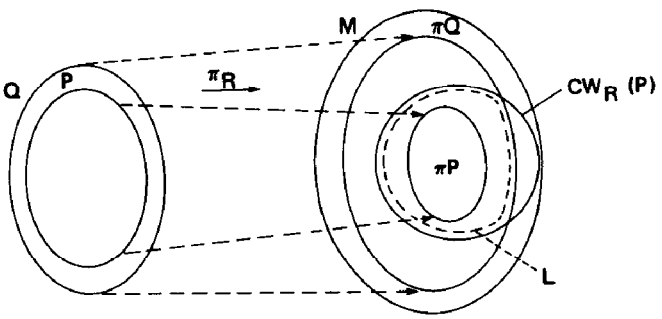

Figure 9.7

Definition 9.5 If **N** is a subset of **M**, we define $\bowtie\mathbf{N}$ to be the set of instances

$$\{\bowtie(d) \mid d \in \mathbf{N}\}.$$

We are ready to define our first notion of adequate representation, which is due to Rissanen.

Definition 9.6 Database scheme **R** *decomposes* **P** *into independent components* if the following two properties hold.

IC1. If $d \in \mathbf{L}$, then there is at most one instance $r \in \mathbf{P}$ with $\pi(r) = d$.
IC2. If $d \in \mathbf{L}$, then there is at least one instance $r \in \mathbf{P}$ with $\pi(r) = d$.

The first property is sometimes called *unique representation*. Properties IC1 and IC2 together are called the *independent component condition*.

The independent component condition requires π to be a one-to-one correspondence from **P** to **L**. For every database $d \in \mathbf{L}$, there is exactly one instance $r \in \mathbf{P}$ such that $\pi(r) = d$. Also, we can represent uniquely each instance in **P** by a database in **L**. We could write IC1 and IC2 as a single prop-

erty, but we separate them because we use IC1 in our other notions of adequacy. Note that IC2 holds if and only if $\pi\mathbf{P} = \mathbf{L}$ (see Exercise 9.5).

The problem with the independent component condition is that it may be hard to compute the inverse of π if we ever want to go from databases to instances. IC1 and IC2 do not require $\bowtie$ to be the inverse of π. The situation shown in Figure 9.8 could hold.

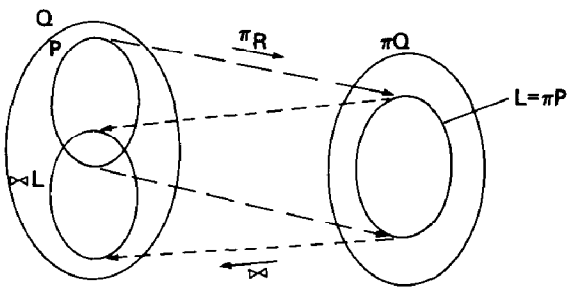

Figure 9.8

Example 9.6 This example is admittedly contrived. Let $\mathbf{U} = ABC$, $\mathbf{R} = \{AB, BC\}$ and $\mathbf{P}$ consists of all instances that satisfy the constraint:

> For every pair of tuples t_1 and t_2 in r,
> $t_1(A) \geq t_2(A)$ implies $t_1(C) \geq t_2(C)$.

We are assuming ordered domains for attributes A and C. $\mathbf{R}$ decomposes $\mathbf{P}$ into independent components (see Exercise 9.6), but the join mapping for $\mathbf{R}$ is not the inverse of the project mapping on $\mathbf{P}$. The instance r in Figure 9.9 decomposes over R into the database $d = \{r_1, r_2\}$ shown in Figure 9.10, but $\bowtie(d) \neq r$.

$r(A$	B	$C)$
1	3	4
2	3	5

Figure 9.9

$r_1(A$	$B)$
1	3
2	3

$r_2(B$	$C)$
3	4
3	5

Figure 9.10

The next notion of adequacy is due to Arora and Carlson.

Definition 9.7 Database scheme **R** describes an *information preserving decomposition of* **P** if the following two properties hold.

AC1. Same as IC1.
AC2. For any instance r in **Q**, $\pi(r) \in \mathbf{L}$ implies $r \in \mathbf{P}$.

The second property is called *constraint containment*. AC1 and AC2 together are called the *information preservation condition*.

We now introduce a new set that will allow us to restate property AC2.

Definition 9.8 For any subset **N** of $\pi\mathbf{Q}$, the *preimage* of **N**, denoted $\mathbf{N}^{\leftarrow}$, is the set of instances

$$\{r \in \mathbf{Q} \mid \pi(r) \in \mathbf{N}\}.$$

The set we are interested in is $\mathbf{L}^{\leftarrow}$. The relationship of $\mathbf{L}^{\leftarrow}$ to other sets is shown in Figure 9.11.

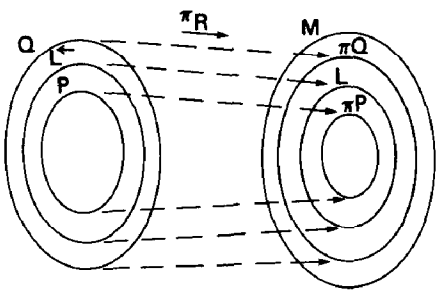

Figure 9.11

Lemma 9.1 AC2 is equivalent to the property $\mathbf{L}^{\leftarrow} = \mathbf{P}$.

Proof Suppose AC2 holds. For any instance r in $\mathbf{L}^{\leftarrow}$, $\pi(r) \in \mathbf{L}$, so $r \in \mathbf{P}$. Therefore $\mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$, so $\mathbf{L}^{\leftarrow} = \mathbf{P}$.

Suppose now $\mathbf{L}^{\leftarrow} = \mathbf{P}$. For any instance $r \in \mathbf{Q}$ where $\pi(r) \in \mathbf{L}$, $r \in \mathbf{L}^{\leftarrow}$. Hence, $r \in \mathbf{P}$ and AC2 is satisfied.

Clearly, $\mathbf{L}^{\leftarrow} = \mathbf{P}$ implies $\mathbf{L} = \pi\mathbf{P}$, hence we have the following result relating the independent component condition and the information preservation condition.

Corollary AC2 implies IC2, hence AC1 and AC2 imply IC1 and IC2.

The implication does not go the other way (see Exercise 9.8). Properties AC1 and AC2 require not only that we can represent every instance r in $\mathbf{P}$ by a database d in $\mathbf{L}$, but also that d could not possibly represent some other instance s in $\mathbf{Q} - \mathbf{P}$. It turns out that the information preservation condition actually specifies what the inverse of π must be.

Definition 9.9 The set of instances $\mathbf{P}$ satisfies the *lossless join property* for database scheme $\mathbf{R}$ if

$$\text{LJ. } FIX(\mathbf{R}) \supseteq \mathbf{P}.$$

That is, $\bowtie(\pi(r)) = r$ for all $r \in \mathbf{P}$. Note that $\bowtie(\pi(r))$ is the same as $m_{\mathbf{R}}(r)$, always.

Lemma 9.2 The LJ property implies the IC1 property.

Proof Suppose the LJ property holds. Let r and r' be instances in $\mathbf{P}$. If $\pi(r) = \pi(r')$, then surely $\bowtie(\pi(r)) = \bowtie(\pi(r'))$. From the LJ property, $r = \bowtie(\pi(r))$, hence $r = r'$ and IC1 is satisfied.

Corollary LJ and AC2 imply AC1 and AC2.

Lemma 9.3 Properties AC1 and AC2 imply LJ.

Proof Assume the information preservation condition holds. Let r be an instance $\mathbf{P}$ and let $d = \pi(r)$. Now $\bowtie(d)$ is also an instance such that $\pi(\bowtie(d)) = d$. Since $d \in \mathbf{L}$, by AC2, $\bowtie d$ is in P. AC1 requires $\bowtie(d) = r$, since $\pi(r) = \pi(\bowtie(d))$. Hence we have $r = \bowtie(d) = \bowtie(\pi(r))$, which is property LJ.

To summarize the previous results:

Theorem 9.1 Given constraint containment (AC2), AC1 is equivalent to LJ.

It should now be clear that $\bowtie$ is the inverse of π on $\mathbf{P}$ under the information preservation condition.

Example 9.7 Let $\mathbf{U} = ABC$, $\mathbf{R} = \{AB, BC\}$ and $\mathbf{P} = SAT(B \rightarrow C)$. The FD $B \rightarrow C$ ensures that the LJ property holds. If instance r obeys $B \rightarrow C$, then $\pi_{BC}(r)$ obeys $B \rightarrow C$ and if $\pi_{BC}(r)$ obeys $B \rightarrow C$, r must obey $B \rightarrow C$ (see Exercise 9.10). Therefore, for any instance r such that $\pi(r) \in \mathbf{L}$, $r \in \mathbf{P}$. $\mathbf{P}$ satisfies AC2 and hence the information preservation condition.

We have noted that the independent component condition has the problem that the inverse mapping from databases to instances may not always be easily computable. Testing membership in **P** can be of arbitrary complexity, and sometimes the complexity carries over to testing which instance in **P** a given database represents. We have seen that AC1 and AC2 require join to be the inverse mapping, but they also require that for no instance r in $\mathbf{Q} - \mathbf{P}$ is $\pi(r)$ in **L**. That is, not only can we represent every instance r in **P** by a unique database d in **L**, but there is no possibility that d also represents some instance r' not in **P**. This property is overly restrictive. Surely if a database d in **L** could represent an instance r in **P** and an instance r' not in **P**, we can ignore r', since it is outside the realm of interest. We present a condition intermediate between independent component and information preservation.

Definition 9.10 Database scheme R satisfies the *join condition* for **P** if the following properties hold.

J1. Same as IC1.
J2. For every database $d \in \mathbf{L}$, $\bowtie(d) \in \mathbf{P}$.

It is not hard to see that J1 and J2 require $\bowtie$ to be the inverse of π on **P**. If r is an instance in **P**, then $\pi(r) \in \mathbf{L}$ and, by J2, $m_{\mathbf{R}}(r) \in \mathbf{P}$. But $\pi(m_{\mathbf{R}}(r)) = \pi(r)$, so, by J1, $m_{\mathbf{R}}(r) = r$. We have also shown that the LJ property holds on **P**. The join condition also says that we let any database $d \in \mathbf{L}$ represent the maximal instance r in **P** such that $\bowtie(r) = d$ (see Exercise 9.11). We now give an alternative characterization of property J2.

Lemma 9.4 Property J2 holds if and only if $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$.

Proof (only if) Property J2 can be stated as $\bowtie\mathbf{L} \subseteq \mathbf{P}$. Let r be an instance in $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow}$. Since r is in $\mathbf{L}^{\leftarrow}$, $\pi(r) \in \mathbf{L}$. Since $r \in FIX(\mathbf{R})$, $r = \bowtie(\pi(r))$, so $r \in \bowtie\mathbf{L}$, hence $r \in \mathbf{P}$. We see $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$.

(if) Suppose $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$, and let r be an instance in $\bowtie\mathbf{L}$. Since $r \in \bowtie\mathbf{L}$, $r = \bowtie(d)$ for some database $d \in \mathbf{L}$. Clearly $\pi(r) = d$, so $m_{\mathbf{R}}(r) = r$ and hence $r \in FIX(\mathbf{R})$. Since $d \in \mathbf{L}$, $r \in \mathbf{R}^{\leftarrow}$. By our supposition, $r \in \mathbf{P}$. We have that $\bowtie\mathbf{L} \subseteq \mathbf{P}$, hence property J2 is satisfied.

We now compare the join condition with the independent component and information preservation conditions.

Lemma 9.5 Condition J2 implies condition IC2.

Proof Left to the reader (see Exercise 9.13).

Theorem 9.2 AC1 and AC2 imply J1 and J2, which in turn imply IC1 and IC2.

Proof Suppose AC1 and AC2 hold. J1 clearly holds. To see that J2 holds, pick any instance $r \in \mathbf{L}$, and let $r = \bowtie(d)$ for $d \in \mathbf{L}$. Since $\pi(r) = d \in \mathbf{L}$, AC2 requires r to be in **P**. Hence $\bowtie\mathbf{L} \subseteq \mathbf{P}$ and J2 is satisfied. The other implication follows from Lemma 9.5.

To compare AC1 and AC2 to J1 and J2, we note that both require $\mathbf{P} \subseteq FIX(\mathbf{R})$ (LJ condition), but AC2 restricts $\mathbf{L}^{\leftarrow} = \mathbf{P}$, where J1 and J2 allow **P** to be properly contained in $\mathbf{L}^{\leftarrow}$. In Figure 9.12, the pair of instances s, s' could not exist under either pair of properties, but J1 and J2 allow the pair of instances r, r', while AC1 and AC2 do not.

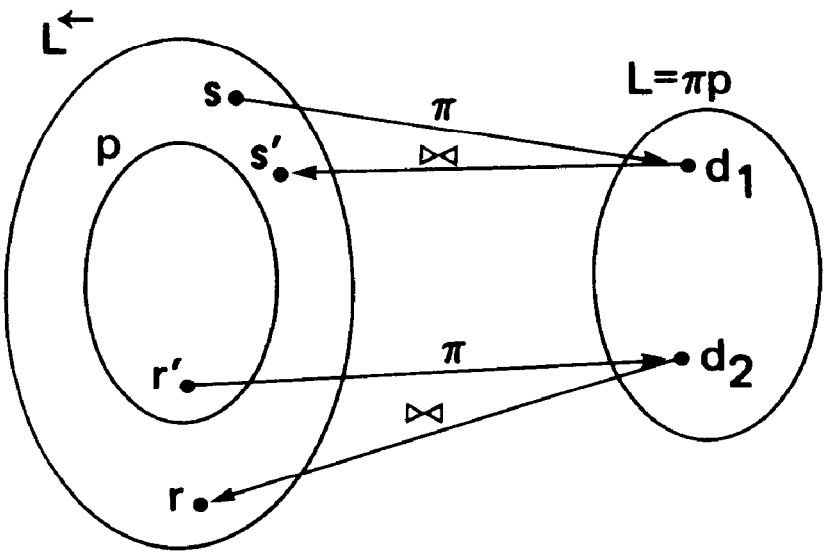

Figure 9.12

Example 9.6 shows that the independent components condition does not imply the join condition. The next example gives a database scheme and set of instances that satisfy the join condition, but not the information preservation condition.

Example 9.8 Let $\mathbf{U} = ABC$, let **R** be the database scheme $\{AB, BC\}$ and let $\mathbf{P} = SAT(B \twoheadrightarrow A)$. **L** consists of all pairs of completely joinable relations over AB and BC. $B \twoheadrightarrow A$ is another statement of $*[AB, BC]$, so **R** satisfies property LJ, hence property J1. The join of two relations over AB and BC

always satisfies $B \twoheadrightarrow A$, so property J2 holds. However, property A2 is violated, since for any instance $r \in \mathbf{Q}$, $\pi(r) \in \mathbf{L}$, and $\mathbf{P} \neq \mathbf{Q}$.

We now look at what property we can add to the independent components condition to get equivalence to the join condition, and also what property we can add to the join condition to get equivalence to the information preservation condition.

How can property IC2 hold while property J2 fails? This situation can only happen if there is some instance r in $\mathbf{P}$ such that $m_{\mathbf{R}}(r) \notin \mathbf{P}$. We formalize this property.

Definition 9.11 Database scheme $\mathbf{R}$ *preserves* $\mathbf{P}$ if the following property holds:

PR. For every instance r in $\mathbf{P}$, $m_{\mathbf{R}}(r) \in \mathbf{P}$. That is, $m_{\mathbf{R}}\mathbf{P} \subseteq \mathbf{P}$, where $m_{\mathbf{R}}\mathbf{P} = \bowtie \pi\mathbf{P}$.

Lemma 9.6 Property J2 implies property PR.

Proof Left to the reader (see Exercise 9.16).

Property PR does not imply property J2, however.

Example 9.9 Let $\mathbf{U} = ABC$, let database scheme $\mathbf{R} = \{AB, BC\}$, and let $\mathbf{P} = SAT(\{A \twoheadrightarrow B, B \twoheadrightarrow A\})$. Since $\mathbf{P} \subseteq SAT(B \twoheadrightarrow A)$, $m_{\mathbf{R}}\mathbf{P} = \mathbf{P}$, so PR is satisfied. Consider the database $d = \{r_1, r_2\}$, where relations r_1 and r_2 are given in Figure 9.13. Database d is in $\mathbf{L}$, since $r_1 = \pi_{AB}(s_1)$ and $r_2 = \pi_{BC}(s_2)$, where s_1 and s_2 are the instances from $\mathbf{P}$ shown in Figure 9.14. We see that $\bowtie(d)$, shown in Figure 9.15, is not in $\mathbf{P}$ (why?), so property J2 is violated.

$r_1(A$	$B)$
1	3
1	4
2	4

$r_2(B$	$C)$
3	5
3	6
4	5
4	6
4	7

Figure 9.13

$s_1(A$	B	$C)$
1	3	5
1	4	5
2	4	5

$s_2(A$	B	$C)$
1	3	5
1	3	6
2	4	5
2	4	6
2	4	7

Figure 9.14

$\bowtie(d)(A$	B	$C)$
1	3	5
1	3	6
1	4	5
1	4	6
1	4	7
2	4	5
2	4	6
2	4	7

Figure 9.15

Lemma 9.7 Properties PR and IC2 imply property J2.

Proof Let r be any instance in $\bowtie\mathbf{L}$ and let d be the database in $\mathbf{L}$ such that $r = \bowtie(d)$. By IC2, $\mathbf{P} = \mathbf{L}$, so $d \in \pi\mathbf{P}$. Let r' be an instance in $\mathbf{P}$ such that $\pi(r') = d$. By PR, $m_{\mathbf{R}}(r') \in \mathbf{P}$, but $m_{\mathbf{R}}(r') = r$, so, $r \in \mathbf{P}$. Hence $\bowtie\mathbf{L} \subseteq \mathbf{P}$ and J2 is satisfied.

Corollary Properties IC1, IC2, and PR hold if and only if J1 and J2 hold.

We can replace IC1 and PR in the corollary above by property LJ.

Lemma 9.8 Property LJ implies property PR.

Proof $FIX(\mathbf{R}) \supseteq \mathbf{P}$ implies $m_{\mathbf{R}}(\mathbf{P}) = \mathbf{P}$, which in turn implies $m_{\mathbf{R}}(\mathbf{P}) \subseteq \mathbf{P}$.

Theorem 9.3 Properties LJ and IC2 hold if and only if properties J1 and J2 hold.

Proof By Lemma 9.2, LJ implies IC1; by Lemma 9.8, LJ implies PR. Therefore, by the corollary to Lemma 9.7, LJ and IC2 imply the join condi-

tion. From Theorem 9.2, we know that J1 and J2 imply IC2. J1 and J2 also imply LJ (see Exercise 9.17), so they imply LJ and IC2.

We now look at what need be added to make the join condition equivalent to the information preservation condition.

Definition 9.12 Database scheme **R** satisfies *property S* if

S. $FIX(\mathbf{R}) \supseteq \mathbf{L}^{\leftarrow}$.

Theorem 9.4 Properties J1, J2, and S hold if and only if properties AC1 and AC2 hold.

Proof From Theorem 9.2, we know AC1 and AC2 imply J1 and J2. By Lemma 9.1, AC2 requires $\mathbf{L}^{\leftarrow} = \mathbf{P}$. By Lemma 9.3, AC1 and AC2 require property LJ, so $\mathbf{P} \subseteq FIX(\mathbf{R})$. Thus we have $\mathbf{L}^{\leftarrow} = \mathbf{P} \subseteq FIX(\mathbf{R})$, which implies property S.

We now show J2 and S imply AC2. Let r be an instance in $\mathbf{L}^{\leftarrow}$, where $\pi(r) = d$, for $d \in \mathbf{L}$. By J2, $\bowtie(d) \in \mathbf{P}$. But S says that $m_{\mathbf{R}}(r) = r$, so $r = \bowtie(d) \in \mathbf{P}$. Therefore, $\mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$. Since we always have $\mathbf{L}^{\leftarrow} \supseteq \mathbf{P}$, $\mathbf{L}^{\leftarrow} = \mathbf{P}$ and AC2 is satisfied.

The difference between the join condition and the information preservation condition is whether we require only $FIX(\mathbf{R}) \supseteq \mathbf{P}$ or the stronger condition $FIX(\mathbf{R}) \supseteq \mathbf{L}^{\leftarrow}$ (see Exercise 9.18). We shall be most interested in cases where **P** is described by some set of constraints **C**. In that case, $FIX(\mathbf{R}) \supseteq \mathbf{P}$ is the same as $\mathbf{C} \models *[\mathbf{R}]$, which we have means to test if **C** is FDs and JDs. The condition $FIX(\mathbf{R}) \supseteq \mathbf{L}^{\leftarrow}$ is the same as $\pi\mathbf{C} \models *[\mathbf{R}]$. We use $\pi\mathbf{C}$ to denote the "projected constraints" of **C** on **R**: the constraints that necessarily hold in the various projections of an instance r onto the schemes in **R**, where r is in $SAT(\mathbf{C})$. That is, $\pi\mathbf{C}$ is the set of constraints such that $\mathbf{L}^{\leftarrow} = SAT(\pi\mathbf{C})$. $SAT(\pi\mathbf{C})$ is exactly those instances r such that $\pi(r) \in \mathbf{L}$. One problem, which we examine later, is that the constraints in $\pi\mathbf{C}$ are not necessarily of the same type as those in **C**, or even embedded versions thereof. There can be problems with testing $\pi\mathbf{C} \models *[\mathbf{R}]$, therefore. The other part of testing the join and information preservation conditions is determining whether $\bowtie\mathbf{L} \subseteq \mathbf{P}$ (see Exercise 9.18). In terms of projected constraints, we want to test $\pi\mathbf{C} \cup \{*[\mathbf{R}]\} \models \mathbf{C}$. For information preservation, the test reduces to $\pi\mathbf{C} \models \mathbf{C}$, by Theorem 9.4. We shall return to testing the various representation properties when **P** is defined by constraints in Section 9.3.

9.2 DATA-EQUIVALENCE OF DATABASE SCHEMES

In Chapter 8 we saw one notion of equivalence of database schemes **R** and **S** relative to a set **P** of instances, namely $T_{\mathbf{R}} \equiv_{\mathbf{P}} T_{\mathbf{S}}$. That is, **R** and **S** are equivalent on **P** if $T_{\mathbf{R}}(r) = T_{\mathbf{S}}(r)$ for all instances r in **P**. In Chapter 8 we were mainly concerned with testing whether $T_{\mathbf{R}} \equiv_{\mathbf{P}} T_I$, where T_I is a tableau for the identity mapping. This notion of equivalence says schemes **R** and **S** are equivalent on **P** if their respective project-join mappings behave identically on **P**. That is, for any instance $r \in \mathbf{P}$, decomposing r onto **R** involves the same loss of information as decomposing r onto **S**.

Suppose, however, we are only interested in those instances where no information is lost through decomposition. We may only care that **R** and **S** can faithfully represent the same set of instances in **P**, but not that **R** and **S** may mangle instances differently that are not represented faithfully by decomposition. We present this weaker notion of equivalence in this section.

Definition 9.13 Given database scheme **R** and a set of instances **P**, the *fixed points of* **P** *under* **R**, written $FIX_{\mathbf{P}}(\mathbf{R})$, is $FIX(\mathbf{R}) \cap \mathbf{P}$.

Definition 9.14 Database scheme **R** and **S** are *data-equivalent on* **P**, written $\mathbf{R} \approx_{\mathbf{P}} \mathbf{S}$, if $FIX_{\mathbf{P}}(\mathbf{R}) = FIX_{\mathbf{P}}(\mathbf{S})$. That is, **R** and **S** can faithfully represent the same subset of instances in **P**. When proving data-equivalence, we shall generally show two containments. Note that the containment $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX_{\mathbf{P}}(\mathbf{S})$ holds exactly when the containment $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$ holds.

Lemma 9.9 $T_{\mathbf{R}} \equiv_{\mathbf{P}} T_{\mathbf{S}}$ implies $\mathbf{R} \approx_{\mathbf{P}} \mathbf{S}$.

Proof Left to the reader (see Exercise 9.19).

The converse of Lemma 9.9 does not hold, as we expect.

Example 9.10 Let $\mathbf{U} = ABCD$, let **R** be the database scheme $\{ABC, CD\}$ and let **S** be the scheme $\{AB, BCD\}$. If **P** consists of just the two instances r and s in Figure 9.16, then **R** and **S** are data-equivalent on **P**. Instance r decomposes losslessly onto both **R** and **S**, while s has a lossy decomposition onto both schemes. However, $T_{\mathbf{R}}$ and $T_{\mathbf{S}}$ are not equivalent on **P**, because $s' = T_{\mathbf{R}}(s)$ is not the same as $s'' = T_{\mathbf{S}}(s)$, as shown in Figure 9.17.

Definition 9.15 Given database scheme **R**, the *preserved set of* **P** *under* **R**, denoted $PRES_{\mathbf{P}}(\mathbf{R})$, is $\{r \in \mathbf{P} \mid m_{\mathbf{R}}(r) \in \mathbf{P}\}$.

$r(A$	B	C	$D)$
1	3	5	7
1	3	6	8
2	4	6	8

$s(A$	B	C	$D)$
1	3	5	7
2	3	6	7
2	4	6	8

Figure 9.16

$s'(A$	B	C	$D)$
1	3	5	7
2	3	6	7
2	4	6	7
2	3	6	8
2	4	6	8

$s''(A$	B	C	$D)$
1	3	5	7
1	3	6	7
2	3	5	7
2	3	6	7
2	4	6	8

Figure 9.17

Using this definition, **R** preserves **P** can be written $PRES_{\mathbf{P}}(\mathbf{R}) = \mathbf{P}$. We also note $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq PRES_{\mathbf{P}}(\mathbf{R})$ (see Exercise 9.21).

Theorem 9.5 Let **R** and **S** be database schemes and let **P** be a set of instances. Assume $\mathbf{P}'$ is any set of instances such that

$$FIX_{\mathbf{P}}(\mathbf{R}) \subseteq \mathbf{P}' \subseteq PRES_{\mathbf{P}}(\mathbf{R}).$$

Then $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$ if and only if $m_{\mathbf{S}}(r) \subseteq m_{\mathbf{R}}(r)$ for all r in $\mathbf{P}'$.

Proof (only if) Let r be an arbitrary instance in $\mathbf{P}'$. Since $r \in PRES_{\mathbf{P}}(\mathbf{R})$, instance $s = m_{\mathbf{R}}(r)$ is in **P**. Project-join mappings are idempotent, so s is in $FIX(\mathbf{R})$ and hence in $FIX_{\mathbf{P}}(\mathbf{R})$. By assumption, s is then in $FIX(\mathbf{S})$. Now $r \subseteq s$, so $m_{\mathbf{S}}(r) \subseteq m_{\mathbf{S}}(s)$. Since s is in $FIX(\mathbf{S})$, $m_{\mathbf{S}}(s) = s$. Combining equalities and containment we have $m_{\mathbf{S}}(r) \subseteq m_{\mathbf{R}}(r)$, as desired.

(if) Let r be an instance in $FIX_{\mathbf{P}}(\mathbf{R})$, which implies $r \in \mathbf{P}'$. By assumption, $m_{\mathbf{S}}(r) \subseteq m_{\mathbf{R}}(r)$. Now $m_{\mathbf{R}}(r) = r$ and $m_{\mathbf{S}}(r) \supseteq r$, so $m_{\mathbf{S}}(r) = r$ and $r \in FIX(\mathbf{S})$. Hence $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$.

Corollary The following are equivalent:

1. $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$.
2. $T_{\mathbf{S}} \sqsubseteq_K T_{\mathbf{R}}$ for $K = FIX_{\mathbf{P}}(\mathbf{R})$.
3. $T_{\mathbf{S}} \sqsubseteq_K T_{\mathbf{R}}$ for $K = PRES_{\mathbf{P}}(\mathbf{R})$.

Proof Left to the reader (see Exercise 9.22).

Corollary If **R** and **S** are database schemes where **R** preserves **P**, then $FIX_{\mathbf{P}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$ if and only if $T_{\mathbf{S}} \sqsubseteq_{\mathbf{P}} T_{\mathbf{R}}$.

Proof Immediate from the equivalence of 1 and 3 in the last corollary.

We have stumbled onto the result that data-equivalence and regular equivalence are the same when **P** is preserved.

Theorem 9.6 If **R** and **S** are database schemes that preserve **P**, then $\mathbf{R} \approx_{\mathbf{P}} \mathbf{S}$ is equivalent to $T_{\mathbf{R}} \equiv_{\mathbf{P}} T_{\mathbf{S}}$.

We know that both the join and information preservation conditions require property PR. Thus, if we choose either condition as a notion of adequate representation, data-equivalence and regular equivalence are the same for adequate database schemes.

9.3 TESTING ADEQUATE REPRESENTATION AND EQUIVALENCE UNDER CONSTRAINTS

We have seen that when the mapping from databases in **L** to relations in **Q** is the join, then the independent component and join conditions are equivalent. The difference between the join and information preservation conditions is whether we require $\mathbf{P} \subseteq FIX(\mathbf{R})$ or the stronger condition $\mathbf{L}^{\leftarrow} \subseteq FIX(\mathbf{R})$. If **P** is defined as $SAT(\mathbf{C})$ for a set of constraints **C**, the difference is whether $\mathbf{C} \models$ *[**R**] or $\pi\mathbf{C} \models$ *[**R**]. Recall that $\pi\mathbf{C}$ is our informal notation for the *projected constraints of* **C** *on* **R**: the constraints that define the set $\mathbf{L}^{\leftarrow}$. That is, $\pi\mathbf{C}$ is the set of restrictions that necessarily must apply to $\pi_R(r)$ for $r \in SAT(\mathbf{C})$ and for all relation schemes $R \in \mathbf{R}$.

We are defining $\pi\mathbf{C}$ in a backwards manner. We want $\pi\mathbf{C}$ to be a set of constraints such that $\mathbf{L}^{\leftarrow} = SAT(\pi\mathbf{C})$. One problem with providing a formal definition for $\pi\mathbf{C}$ is that $\pi\mathbf{C}$ cannot necessarily be expressed by the same types of dependencies as those in **C**. There often are dependencies in $\pi\mathbf{C}$ of the same type as those in **C** (see Exercises 9.10 and 9.24), but there can be dependencies of other types.

Example 9.11 Let $\mathbf{U} = ABCDE$, let $R = ABDE$ and let $\mathbf{P} = SAT\{A \rightarrow E, B \rightarrow E, CE \rightarrow D\}$. For any relation $r \in \mathbf{P}$, the set of FDs $\pi_{\mathbf{R}}(r)$ must satisfy is $F = \{AC \rightarrow D, BC \rightarrow D\}$. Consider the relation s in Figure 9.18. Relation $s \in SAT(F)$, but s is not the projection of any instance in **P**. (Add an E-column

to s and try chasing under the FDs of **P**.) It turns out that any relation $\pi_{\mathbf{R}}(r)$ for $r \in \mathbf{P}$ must satisfy the curious dependency:

If t_1, t_2, and t_3 are tuples in $\pi_{\mathbf{R}}(r)$ such that

1. $t_1(A) = t_3(A)$
2. $t_1(C) = t_2(C)$
3. $t_2(B) = t_3(B)$

then

4. $t_1(D) = t_2(D)$.

This dependency is not equivalent to any set of FDs. Note that s does not satisfy the dependency.

$s(A$	B	C	$D)$
1	3	5	7
2	4	5	8
1	4	6	8

Figure 9.18

Example 9.12 Let $\mathbf{U} = ABCD$, let $R = ABC$ and let $\mathbf{P} = SAT(\{A \twoheadrightarrow BC, B \twoheadrightarrow AC, CD \twoheadrightarrow A\})$. Any relation $\pi_{\mathbf{R}}(r)$ for $r \in \mathbf{P}$ need only satisfy trivial MVDs. However, relation s in Figure 9.19 is not the projection of any instance in **P**. (Add a D-column and chase under the MVDs of **P**.)

$s(A$	B	$C)$
1	3	5
1	4	6
2	4	5

Figure 9.19

9.3.1 P Specified by Functional Dependencies Only

We see there are problems with specifying $\pi\mathbf{C}$, and hence with testing $FIX(\mathbf{R}) \supseteq \mathbf{L}^{\leftarrow}$. In Chapter 14 we shall examine classes of dependencies that can be used to characterize $\pi\mathbf{C}$ when **C** consists of FDs and JDs. However no decision procedure for implication exists for those more general dependency classes. When **C** is just FDs, even though we cannot express $\pi\mathbf{C}$ with just FDs, the set of FDs in $\pi\mathbf{C}$ suffices for our representation conditions.

Theorem 9.7 If $\mathbf{P} = SAT(F)$ for a set F of FDs, then the independent components, join and information preservation conditions are equivalent.

Proof By Theorem 9.2, it suffices to show that if AC1 or AC2 fails, then IC1 or IC2 fails. Clearly, if AC1 fails, then IC1 fails.

Suppose AC2 fails: $\mathbf{L}^{\leftarrow} \neq \mathbf{P}$. Let r be an instance in $\mathbf{L}^{\leftarrow} - \mathbf{P}$. We may assume r has only two tuples (see Exercise 9.27). Instance r must violate some FD implied by F, say $X \rightarrow A$. Since r has but two tuples, they must agree on X and disagree on A. Thus r has one value in each of the X-columns and two values in the A-column.

If IC2 fails, we are done. Suppose it holds. Let $d = \pi(r)$; d must be in $\mathbf{L}$. By IC2, there is an instance r' in $\mathbf{P}$ such that $\pi(r') = d$. Instance r' can only have a single value in each of its X-columns. (Why?) However, r' must have two values in its A-column, and therefore violates $X \rightarrow A$. Hence $r' \in \mathbf{P}$ and IC2 must fail.

We now consider testing the join condition (and hence the other two representation conditions) when $\mathbf{P} = SAT(F)$ for a set F of FDs. We shall test properties LJ and J2 (see Exercise 9.28). LJ translates to $F \models *[\mathbf{R}]$, which we know how to test using the chase. We consider testing J2, which is $\bowtie \mathbf{L} \subseteq \mathbf{P}$.

Definition 9.16 For a database scheme $\mathbf{R}$ and a set F of FDs, *F restricted to* $\mathbf{R}$, denoted $F_{\mathbf{R}}$, is the set of FDs in F^+ that apply to any relation scheme R in $\mathbf{R}$.

Example 9.13 Let $\mathbf{U} = ABCD$, $\mathbf{R} = \{ABC, CD\}$ and $F = \{AD \rightarrow C, CD \rightarrow A, B \rightarrow D\}$. The only nontrivial FDs in $F_{\mathbf{R}}$ are $BC \rightarrow A$, $BC \rightarrow AB$, $BC \rightarrow AC$, $BC \rightarrow ABC$, $AB \rightarrow C$, $AB \rightarrow AC$, $AB \rightarrow BC$ and $AB \rightarrow ABC$.

Recall that F is *enforceable* on $\mathbf{R}$ if $F_{\mathbf{R}} \equiv F$.

Lemma 9.10 Let $\mathbf{P} = SAT(F)$ for a set F of FDs. Property J2 holds if and only if F is enforceable on $\mathbf{R}$.

Proof We leave the if direction as Exercise 9.30 (only if). We shall show that if F is not enforceable on $\mathbf{R}$, then there is a database d in $\mathbf{L}$ such that $\bowtie d$ is not in $\mathbf{P}$. Let $X \rightarrow A\,Y$ be an FD in F such that $X \rightarrow A$ is not in $G = F_{\mathbf{R}}^+$. Let Z be the closure of X under G. Clearly $A \notin Z$. We construct an instance r_X in $\mathbf{Q}$ as follows. Instance r_X has two tuples: t_0, which is 0 everywhere, t_Z, which is 0 on Z and 1 elsewhere.

Obviously, r_X is not in $\mathbf{P}$, since it violates F. Let $d = \pi(r_X)$. The instance $\bowtie(d)$ is not in $\mathbf{P}$ either, since $\bowtie(d) \supseteq r_X$ (see Exercise 9.26). We now show that each relation in d is the projection of some instance in $\mathbf{P}$. Let R be any relation scheme in $\mathbf{R}$. We define an instance r_R in $\mathbf{P}$ where $\pi_R(r_X) = \pi_R(r_R)$.

Let r_R contain two tuples: t_0 as in r_X, and t_1, which is all 1's except for 0's in $(Z \cap R)^+$ under F. Instance r_R cannot violate any FD in F, or else $(Z \cap R)^+$ would be incorrectly defined. To show that $\pi_R(r_X) = \pi_R(r_R)$, we must show $(Z \cap R)^+ \cap R = Z \cap R$. (Those sets are the columns where $\pi_R(r_R)$ and $\pi_R(r_X)$, respectively, contain two symbols.) Suppose B is an attribute that is in the first set but not the second. B is clearly in R. Since $B \in (Z \cap R)^+$, $F \models Z \cap R \rightarrow B$. But then $F \models Z \rightarrow B$, so $B \in Z$ and hence $B \in Z \cap R$, a contradiction. We have shown $d \in \mathbf{L}$. Since $\bowtie(d) \in \mathbf{P}$, property J2 is violated, as we predicted.

We saw in Chapter 8 that for a set of FDs F, $F \models$ *[**R**] can be tested in time polynomial in the space required to write F and **R**. We now present a polynomial-time test for F being enforceable on **R**. This test will give us a polynomial-time test for the join condition when $\mathbf{P} = SAT(F)$.

The Algorithm 9.1 computes X^+ under $F_{\mathbf{R}}$. The closure taken in line 5 is the closure under F.

Algorithm 9.1 PCLOSURE

Input: A set of FDs F over **U**, a database scheme **R** over **U** and a subset X of **U**.

Output: The closure of X under $F_{\mathbf{R}}$.

PCLOSURE(F, **R**, X)

1. **begin**
2. $Y := X$;
3. **while** there are changes to Y **do**
4. **for** each $R \in \mathbf{R}$ **do**
5. $Y := ((Y \cap R)^+ \cap R) \cup Y$;
6. **return** (Y);
7. **end.**

Example 9.14 Let $\mathbf{U} = ABCDE$, $R = \{AB, BC, CDE\}$ and $F = \{A \rightarrow D, D \rightarrow B, B \rightarrow CE\}$. PCLOSURE($F$, **R**, A) is ABC. Note that A^+ under F is $ABCDE$.

Theorem 9.8 PCLOSURE(F, **R**, X) returns X^+ under $F_{\mathbf{R}}$.

Proof Let Y be the set in the algorithm. Y is initially a subset of X^+, and remains so, since the FD being implicitly used in line 5 is $(Y \cap R) \rightarrow (Y \cap R)^+ \cap R$, which is in $F_{\mathbf{R}}$ (the closure $(Y \cap R)^+$ being taken under F). We want to show now that every attribute of X^+ is eventually added to Y.

Let H be an $F_{\mathbf{R}}$-based DDAG for $X \rightarrow X^+$. Assume that the successive sets

of node labels during the construction of H are $Z_0, Z_1, Z_2, \ldots, Z_n$, where $X = Z_0$ and $X^+ = Z_n$. We claim $Z_i \subseteq Y$ at the end of the i^{th} iteration of the **while**-loop. Clearly $Z_0 \subseteq Y$. Suppose to get from Z_i to Z_{i+1} the FD $V \rightarrow W$ is applied. Since $V \rightarrow W$ is in $F_{\mathbf{R}}$, there is some relation scheme $R \in \mathbf{R}$ such that $VW \subseteq R$. Moreover, $V \subseteq Z_i$, so $V \subseteq Y$ at the start of the $i + 1^{st}$ iteration. When R is reached in the **for**-loop, $V \subseteq Y \cap R$, so $W \subseteq (Y \cap R)^+ \cap R$, since $F \models F_{\mathbf{R}}$. Therefore, all of W is added to Y if it is not there already, and whatever attribute $A \in W$ was added to Z_i to get Z_{i+1} is also added to Y. Hence, when the algorithm terminates $Z_n \subseteq Y$, and so $Y = X^+$.

Lemma 9.11 PCLOSURE has time complexity $O(|\mathbf{U}| \cdot |\mathbf{R}| \cdot ||F||)$ where $||F||$ is the space required to write all the FDs in F.

Proof The **while**-loop in line 3 can execute no more than $|\mathbf{U}| - 1$ times, since Y can grow no larger than $\mathbf{U}$. The **for**-loop at line 4 executes $|\mathbf{R}|$ times for each iteration of the **while**-loop. The dominating factor in the computation of line 5 is determining $(Y \cap R)^+$, which we can do in time linear in $||F||$. Hence we have a total time complexity of $O(|\mathbf{U}| \cdot |\mathbf{R}| \cdot ||F||)$.

Theorem 9.9 The join condition can be tested in time polynomial in $|\mathbf{U}|$, $|\mathbf{R}|$, and $||F||$ when $\mathbf{P} = SAT(F)$ and F is all FDs.

Proof By our previous discussion, property LJ is testable in polynomial time. From Lemma 9.10 we know property J2 holds if $F \equiv F_{\mathbf{R}}$. We need only test $F_{\mathbf{R}} \models F$, which we can do using PCLOSURE for each FD $X \rightarrow Y$ in F to see if X^+ under $F_{\mathbf{R}}$ contains Y. This test involves $|F|$ calls to PCLOSURE, which is certainly of polynomial time-complexity.

When $F \equiv F_{\mathbf{R}}$, we can find a cover G for $F_{\mathbf{R}}$ that applies to R in time polynomial in the input (see Exercise 9.32). However, when $F \not\equiv F_{\mathbf{R}}$, it is an NP-complete problem to determine for a set G of FDs that applies to R whether $F_{\mathbf{R}} \equiv G$, given $\mathbf{R}$, F and G.

The next two examples show that the properties LJ and J2 are indeed independent when $\mathbf{P}$ is described by FDs.

Example 9.15 Let $\mathbf{U} = ABC$, $\mathbf{R} = \{AC, AB\}$ and $\mathbf{P} = SAT(\{A \rightarrow C, B \rightarrow C\})$. LJ is satisfied, because the FD $A \rightarrow C$ holds. J2 fails, because $F_{\mathbf{R}} \not\models B \rightarrow C$, so F is not enforceable on $\mathbf{R}$.

Example 9.16 Let $\mathbf{U}$ and $\mathbf{P}$ be the same as in Example 9.15, but now let $\mathbf{R} = \{AC, BC\}$. F applies to $\mathbf{R}$, so it is enforceable, and J2 is satisfied. However, now LJ fails, since $\{A \rightarrow C, B \rightarrow C\} \not\models *[AC, BC]$.

9.3.2 P Specified by Functional and Multivalued Dependencies

In the last subsection we saw how to test the join condition when **P** is determined by a set of FDs. We now work on a test for that condition when $\mathbf{P} = SAT(\mathbf{C})$ for a set **C** of FDs and MVDs and **R** is a 4NF database scheme. We need to test the LJ and J2 properties. Testing LJ is done by determining whether $\mathbf{C} \models *[\mathbf{R}]$, using the chase. For J2, we want to determine whether $\bowtie\mathbf{L} \subseteq \mathbf{P}$, which is $\pi\mathbf{C} \cup \{*[\mathbf{R}]\} \models \mathbf{C}$ in terms of dependencies. We have seen there are problems in testing implications by $\pi\mathbf{C}$. We shall show that when **R** is in 4NF, it suffices to test $F_{\mathbf{R}} \cup \{*[R]\} \models \mathbf{C}$, where F is the set of FDs implied by **C**.

Lemma 9.12 Let **R** be a 4NF database scheme and let $\mathbf{P} = SAT(\mathbf{C})$ for **C** a set of FDs and MVDs. Let F be the set of FDs implied by **C**, and let $X \rightarrow A$ be an FD in F that is not implied by $F_{\mathbf{R}}$ and $*[\mathbf{R}]$. Then $X \rightarrow A$ is not implied by $\pi\mathbf{C}$ and $*[\mathbf{R}]$. (That is, some instance in $\bowtie\mathbf{L}$ violates $X \rightarrow A$.)

Proof In this proof, for any set of attributes $V \subseteq \mathbf{U}$, r_V is the instance with one tuple of all 0's and a second tuple that is all 0's on V and 1's elsewhere.

We may assume that $X = X^+$ under $F_{\mathbf{R}}$ and $*[\mathbf{R}]$, since $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ do not imply $X^+ \rightarrow A$ and if instance r in $\bowtie$L violates $X^+ \rightarrow A$, it also violates $X \rightarrow A$. Let r_X be defined as above, and let d be the database $\pi(r_X)$. Since $d \supseteq r_X$ and r_X violates $X \rightarrow A$, so does d. We need to show that $\bowtie(d)$ is in $\bowtie\mathbf{L}$, so we show that d is in **L**. To do so, we show that every relation in d is the projection of an instance in **P**.

Let R be any relation scheme in **R**. Let $Y = R \cap X$ and let $r^* = chase_{\mathbf{C}}(r_Y)$. (Treat the 0's as distinguished variables and the 1's as nondistinguished variables.) The set of columns in r^* that are all 0's will be Y^+ under **C**. Now $Y^+ \cap R = X \cap R$, for if B is an attribute of B in Y^+ then $Y \rightarrow B$ is in F and hence in $F_{\mathbf{R}}$, so $B \in X$.

Suppose $\pi_R(r_X) \neq \pi_R(r^*)$. There must be tuple t in R^* that is 0 exactly on W, where $W \cap R \neq X \cap R = Y$. By Theorem 8.12, $\mathbf{C} \models Y \twoheadrightarrow W$, so $Y \twoheadrightarrow R \cap W$ holds on R (see Exercise 9.23). Now $\mathbf{C} \not\models Y \rightarrow W \cap R$, or else $W \cap R \subseteq X$. Therefore $\mathbf{C} \not\models Y \rightarrow R$. We see that R is not in 4NF, a contradiction. The projections above must be equal, and hence $d \in \bowtie\mathbf{L}$.

Lemma 9.13 Let **R** be a 4NF database scheme, let $\mathbf{P} = SAT(\mathbf{C})$ for **C** a set of FDs and MVDs, and assume property LJ holds. Let F be the set of FDs implied by **C**, and let $X \twoheadrightarrow Y$ be an MVD implied by **C** that is not implied by $F_{\mathbf{R}}$ and $*[\mathbf{R}]$. Then $X \twoheadrightarrow Y$ is not implied by $\pi\mathbf{C}$ and $*[\mathbf{R}]$.

Proof We shall assume Y is minimal with respect to the hypotheses given. That is, for no proper subset Y' of Y does $\mathbf{C} \models X \twoheadrightarrow Y'$ and $F_{\mathbf{R}} \cup \{*[\mathbf{R}]\} \not\models X \twoheadrightarrow Y'$. We also assume that X is maximal with respect to the hypotheses. First, $X = X^+$ under $F_{\mathbf{R}}$ and $*[\mathbf{R}]$, as in the proof of Lemma 9.12. Second, for no $Z \subsetneq \mathbf{U} - XY$ does $\mathbf{C} \models XZ \twoheadrightarrow Y$ while $F_{\mathbf{R}} \cup \{*[\mathbf{R}]\} \not\models XZ \twoheadrightarrow Y$.

Let r_X be as defined in the proof of Lemma 9.12. Instance r_X clearly satisfies F_R; we claim it also satisfies $*[\mathbf{R}]$. Suppose not. Chasing r_X under $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ must yield new tuples. Let t be one of these new tuples; say that XW is the set of attributes where t is 0 and X is disjoint from W. By Theorem 8.12, $*[\mathbf{R}] \models X \twoheadrightarrow W$. By property LJ, $\mathbf{C} \models *[\mathbf{R}]$, so $\mathbf{C} \models X \twoheadrightarrow W$. Surely $Y \neq W$. If $Y \subseteq W$, then $\mathbf{C} \models XZ \twoheadrightarrow Y$, where $Z = \mathbf{U} - WXY$. Z is nonempty because $XW \neq \mathbf{U}$. $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ cannot imply $XZ \twoheadrightarrow Y$, since $XZ \twoheadrightarrow Y$ and $X \twoheadrightarrow W$ imply $X \twoheadrightarrow Y$. (This implication may easily be tested using the chase.) So $Y \subseteq W$ leads to a contradiction to the maximality of X. If W and Y are disjoint, then $W' = \mathbf{U} - WX$ contains Y and $*[\mathbf{R}] \models X \twoheadrightarrow W'$, leading to the same contradiction.

The only possibility left is that W partially intersects Y. Let $Y' = Y - W$ and $Y'' = Y \cap W$. By projectivity, $\mathbf{C} \models \{X \twoheadrightarrow Y', X \twoheadrightarrow Y''\}$. $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ cannot imply both $X \twoheadrightarrow Y'$ and $X \twoheadrightarrow Y''$, since these two MVDs imply $X \twoheadrightarrow Y$ by additivity. This situation contradicts the minimality of Y. We see that chasing r_X under $*[\mathbf{R}]$ adds no new tuples, hence r_X satisfies $F_{\mathbf{R}}$ and $*[\mathbf{R}]$.

We now only need to show that the database $d = \pi(r_X)$ is in $\mathbf{L}$, since then $\bowtie(d) = r_X$ will be in $\bowtie\mathbf{L}$. Since $X = X^+$ under $F_{\mathbf{R}}$ and $*[\mathbf{R}]$, the proof of Lemma 9.12 suffices to show d is in $\mathbf{L}$.

We now have the tools for testing the join condition for a 4NF database scheme R and a set of instances $\mathbf{P}$ defined by a set $\mathbf{C}$ of FDs and MVDs. First we test property LJ, $\mathbf{C} \models *[\mathbf{R}]$, using the chase. Then we must find the set F of FDs implied by $\mathbf{C}$, or some cover for F. Unfortunately, no methods are known for finding F other than straightforward enumeration and testing. We next need to find $F_{\mathbf{R}}$. By Lemmas 9.12 and 9.13, we can then check property J2 by testing $F_{\mathbf{R}} \cup \{*[\mathbf{R}]\} \models \mathbf{C}$ using the chase. This process is nowhere near as efficient as the test in the case that $\mathbf{C}$ is only FDs. However, no test at all is known for the join condition when $\mathbf{C}$ is FDs and JDs and $\mathbf{R}$ is arbitrary.

Example 9.17 Let $\mathbf{U} = ABCDE$, $\mathbf{R} = \{ABC, BCD, DE\}$ and let $\mathbf{C} = \{D \rightarrow E, BC \twoheadrightarrow A, AD \twoheadrightarrow E\}$. If F is the set of FDs implied by $\mathbf{C}$, then $\{D \rightarrow E\}$ is a cover for $F_{\mathbf{R}}$. We see that $\mathbf{C} \models *[\mathbf{R}]$ and that $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ imply $\mathbf{C}$, so the join condition is satisfied in this case.

Example 9.18 Let $\mathbf{U} = ABCDE$, $\mathbf{R} = \{ABC, BCD, DE\}$ and let $\mathbf{C} = \{D \rightarrow E, BC \twoheadrightarrow A, A \twoheadrightarrow E\}$. Again, $\{D \rightarrow E\}$ is a cover for $F_{\mathbf{R}}$, but $F_{\mathbf{R}}$ and $*[\mathbf{R}]$ do not imply $A \twoheadrightarrow E$, so the join condition does not hold here.

9.3.3 Testing Data-Equivalence

When $\mathbf{P} = SAT(\mathbf{C})$, and $\mathbf{R}$ and $\mathbf{S}$ are database schemes, in order to test $\mathbf{R} \approx_{\mathbf{P}} \mathbf{S}$, we must determine whether $FIX_{\mathbf{C}}(\mathbf{R}) = FIX_{\mathbf{C}}(\mathbf{S})$. The equality holds, of course, when there is containment in both directions. As previously noted, $FIX_{\mathbf{C}}(\mathbf{R}) \subseteq FIX_{\mathbf{C}}(\mathbf{S})$ exactly when $FIX_{\mathbf{C}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$. That containment holds if and only if $\mathbf{C} \cup \{*[\mathbf{R}]\} \models *[\mathbf{S}]$, which we can test with the chase when $\mathbf{C}$ is FDs and JDs.

When $\mathbf{R}$ preserves $SAT(\mathbf{C})$, a corollary to Theorem 9.5 tells us that $FIX_{\mathbf{C}}(\mathbf{R}) \subseteq FIX(\mathbf{S})$ is equivalent to $T_{\mathbf{S}} \sqsubseteq_{\mathbf{C}} T_{\mathbf{R}}$. The following lemma can be useful for testing that containment.

Lemma 9.14 Let $\mathbf{R}$ and $\mathbf{S}$ be database schemes, let $\mathbf{C}$ be a set of FDs and JDs and let $T_{\mathbf{R}}^* = chase_{\mathbf{C}}(T_{\mathbf{R}})$. Then $T_{\mathbf{R}} \sqsubseteq_{\mathbf{C}} T_{\mathbf{S}}$ if and only if $T_{\mathbf{R}}^* \sqsubseteq T_{\mathbf{S}}$.

Proof Left to the reader (see Exercise 9.34).

Lemma 9.14 is useful because $T_{\mathbf{R}}^* \sqsubseteq T_{\mathbf{S}}$ is quite easy to test. We need to find a containment mapping from $T_{\mathbf{S}}$ to $T_{\mathbf{R}}^*$ in order for the containment to hold. Since $T_{\mathbf{S}}$ has no duplicated nondistinguished variables, we need only check that for each row w in $T_{\mathbf{S}}$ there is a row w' in $T_{\mathbf{R}}^*$ that subsumes w.

Example 9.19 Let $\mathbf{U} = ABCDE$, let $\mathbf{R} = \{ABC, BCD, DE\}$, let $\mathbf{S} = \{ACE, BCD\}$, and let $\mathbf{P} = SAT(\mathbf{C})$ where $\mathbf{C} = \{B \rightarrow E, D \twoheadrightarrow B\}$. $T_{\mathbf{R}}^* = chase_{\mathbf{C}}(T_{\mathbf{R}})$ and $T_{\mathbf{S}}^* = chase_{\mathbf{C}}(T_{\mathbf{S}})$ are shown in Figures 9.20 and 9.21. There is a containment mapping from $T_{\mathbf{S}}$ to $T_{\mathbf{R}}^*$, so $T_{\mathbf{R}} \sqsubseteq_{\mathbf{C}} T_{\mathbf{S}}$. However, there is not a containment mapping from $T_{\mathbf{R}}$ to $T_{\mathbf{S}}^*$, so $\mathbf{R} \not\approx_{\mathbf{C}} \mathbf{S}$. Actually, the simplified test of Theorem 9.5 for $\mathbf{R} \approx_{\mathbf{C}} \mathbf{S}$ does not apply here, since neither $\mathbf{R}$ nor $\mathbf{S}$ preserves $SAT(\mathbf{C})$. However, the negative result carries through (see Exercise 9.35).

$T_{\mathbf{R}}^*$(

A	B	C	D	E
a_1	a_2	a_3	b_1	a_5
b_3	a_2	a_3	a_4	a_5
b_5	b_6	b_7	a_4	a_5
b_3	b_6	a_3	a_4	a_5
b_5	a_2	b_7	a_4	a_5

)

Figure 9.20

$T_S^*(A$	B	C	D	$E)$
a_1	b_1	a_3	b_2	a_5
b_3	a_2	a_3	a_4	b_4

Figure 9.21

In the case that **R** and **S** preserve $SAT(\mathbf{C})$, we know $\mathbf{R} \approx_{\mathbf{C}} \mathbf{S}$ is equivalent to $T_{\mathbf{R}} \equiv_{\mathbf{C}} T_{\mathbf{S}}$. If **C** is a set of FDs only, then we can calculate the chases of $T_{\mathbf{R}}$ and $T_{\mathbf{S}}$ in time polynomial in the space required to represent $T_{\mathbf{R}}$, $T_{\mathbf{S}}$, and **C**. Let $T_{\mathbf{R}}^*$ and $T_{\mathbf{S}}^*$ be those two chases. We can test $T_{\mathbf{R}}^* \sqsubseteq T_{\mathbf{S}}$ and $T_{\mathbf{R}} \sqsupseteq T_{\mathbf{S}}^*$, and hence $T_{\mathbf{R}} \equiv_{\mathbf{C}} T_{\mathbf{S}}$, in polynomial time, because containment mappings are easy to find in these cases, as we stated before. The next result gives us a polynomial time test for whether a database scheme **R** preserves a set of FDs F.

Definition 9.17 Let T be a tableau. *T embeds an FD* $\mathrm{X} \rightarrow \mathrm{Y}$ if some row w in T is distinguished on at least its XY-columns. *T embeds a set of FDs* F if T embeds every FD in F.

Evidently, a set of FDs F is enforceable on database scheme R if and only if $T_{\mathbf{R}}$ embeds some cover G of F.

Example 9.20 The tableau T in Figure 9.22 embeds the set of FDs $F = \{C \twoheadrightarrow AB, E \rightarrow B\}$.

$T(A$	B	C	D	$E)$
a_1	a_2	a_3	b_1	b_2
b_3	a_2	b_4	a_4	b_5
a_1	a_2	a_3	b_6	a_5

Figure 9.22

Theorem 9.10 Let **R** be a database scheme and let F be a nonredundant set of FDs. **R** preserves $SAT(F)$ if and only if $T_{\mathbf{R}}^* = chase_F(T_{\mathbf{R}})$ embeds F.

Proof Suppose **R** preserves $SAT(F)$. We show a property of $T_{\mathbf{R}}^*$. If w is a row of $T_{\mathbf{R}}^*$ and X is the set of columns where w has a distinguished variable, then $X = X^+$ under F. Suppose $X \neq X^+$. Let A be an attribute in $X^+ - X$. We know $F \models X \rightarrow A$. Also, $T_{\mathbf{R}}^*$ as an instance is in $SAT(F)$ so $m_{\mathbf{R}}(T_{\mathbf{R}}^*)$ is in $SAT(F)$, since **R** preserves $SAT(F)$. But $m_{\mathbf{R}}(T_{\mathbf{R}}^*)$ contains both w and w_d, the row of all distinguished variables. (Why?) Now $w(X) = w_d(X)$, but

$w(A) \neq w_d(A)$, which is a violation of $X \rightarrow A$. We conclude from this contradiction that $X^+ = X$. Furthermore, it can be shown that $T^*_{\mathbf{R}}$ contains no duplicated nondistinguished variables (see Exercise 9.36).

Let $X \rightarrow Y$ be an arbitrary FD in F. We want to show that $X \rightarrow Y$ is embedded in $T^*_{\mathbf{R}}$. Let w_X be a row with distinguished variables in exactly the X-columns, and nondistinguished variables that do not appear elsewhere in $T^*_{\mathbf{R}}$. Let T be the tableau $T^*_{\mathbf{R}}$ with row w_X added. Let $T^* = chase_F(T)$ and let w^*_X be the row in T^* that corresponds to w_X in T. Let w^*_X be distinguished in exactly the Z-columns. We claim $Z = X^+$. By the argument in the previous paragraph, $Z = Z^+$, so $Z \supseteq X^+$. By Exercise 9.37, if $A \in Z$, then $F \models X \rightarrow A$. So $Z \subseteq X^+$ and hence $Z = X^+$.

Furthermore, we claim that T^* is just T with w_X changed to w^*_X. That is, none of $T^*_{\mathbf{R}}$ in T changes in the computation of $chase_F(T)$. Consider the first F-rule applied in computing $chase_F(T)$. Say it is the F-rule for $W \rightarrow B$ in F. Row w_X must be involved in the application, since $T^*_{\mathbf{R}}$ satisfies $W \rightarrow B$. For $W \rightarrow B$ to be applicable to T, $W \subseteq X$ and there must be a row w in $T^*_{\mathbf{R}}$ with $w(W) = w_X(W)$. We conclude that w is distinguished on all the W-columns, Since the set of distinguished columns of w is closed under F, w is distinguished on B as well. Thus the application of the F-rule for $W \rightarrow B$ to T changes $w_X(B)$ to a distinguished variable. The rows of $T^*_{\mathbf{R}}$ in T are unchanged. The same argument then applies to the second application of an F-rule, the third application, and so forth. Therefore, none of the rows of $T^*_{\mathbf{R}}$ change during the computation of $chase_F(T)$.

We know that w_X ends up as w^*_X in T^*, where w^*_X is distinguished in exactly the X^+ columns. F is nonredundant, so there is a nonempty subset Y' of Y such that $X \rightarrow Y'$ is not implied by $F - \{X \rightarrow Y\}$. At some point in computing $chase_F(T)$, then, we have to use an F-rule for $X \rightarrow A$, for some $A \in Y'$, in order to distinguish w_X on Y'. By the results of the previous paragraph, there is some row w in $T^*_{\mathbf{R}}$ that is distinguished on its X-columns. Since the set of columns where w has distinguished variables is closed under F, w is distinguished on its Y-columns as well. Hence $T^*_{\mathbf{R}}$ embeds $X \rightarrow Y$.

Now suppose $T^*_{\mathbf{R}}$ embeds F. Let $X \rightarrow Y$ be an FD in F. Let r be an instance in $SAT(F)$. Since r satisfies $X \rightarrow Y$, so does $T^*_{\mathbf{R}}(r)$ (see Exercise 9.39). Further, $T^*_{\mathbf{R}}(r) = T_{\mathbf{R}}(r) = m_{\mathbf{R}}(r)$, so $m_{\mathbf{R}}(r)$ satisfies $X \rightarrow Y$. We conclude that $\mathbf{R}$ preserves $SAT(F)$.

Example 9.21 Let $\mathbf{R}$ be the database scheme $\{ABC, CD, DEI\}$ and let $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A, ADE \rightarrow I\}$. F is nonredundant. $T^*_{\mathbf{R}} = chase_F(T_{\mathbf{R}})$ is shown in Figure 9.23. $T^*_{\mathbf{R}}$ embeds F, so $\mathbf{R}$ perserves $SAT(F)$. Note, however, that F is not enforceable on $\mathbf{R}$ ($D \rightarrow \mathbf{A}$ is the problem) nor does $F \models *[\mathbf{R}]$.

$T_{\mathbf{R}}^{*}$

A	B	C	D	E	I
a_1	a_2	a_3	a_4	b_2	b_3
a_1	b_5	a_3	a_4	b_6	b_7
a_1	b_9	b_{10}	a_4	a_5	a_6

Figure 9.23

The final result of this chapter relates preservation of a set of FDs to the fix-points of a database scheme that naturally arises from the set.

Definition 9.18 Let F be a set of FDs, and let G be a nonredundant cover for F. The *intended database scheme of* F, which we denote as $\boldsymbol{R}_F$, is

$$\{X^+ \mid X \rightarrow Y \in G\}.$$

The definition does not depend on the choice of G (see Exercise 9.40).

Example 9.22 Let $F = \{AB \rightarrow C,\ C \rightarrow E,\ D \rightarrow A,\ ADE \rightarrow I\}$ as in Example 9.21. We have $(AB)^+ = ABC$, $C^+ = ACD$, $D^+ = AD$, and $(ADE)^+ = ADEI$, so $\mathbf{R}_F = \{ABC, ACD, AD, ADEI\}$.

In the case that $*[\mathbf{R}_F]$ is a full JD (mentions all the attributes of $\mathbf{U}$), we have the following result.

Theorem 9.11 Let $\mathbf{R}$ be a database scheme, and let F be a set of FDs. Suppose $*[\mathbf{R}_F]$ is a full JD. Then $FIX_F(\mathbf{R}_F) \subseteq FIX(\mathbf{R})$ if and only if $\mathbf{R}$ preserves $SAT(F)$.

Proof (if) By Exercise 9.36, $chase_F(T_{\mathbf{R}}) = T_{\mathbf{S}}$ for some database scheme $\mathbf{S}$. Furthermore, from the proof of Theorem 9.10, for any relation scheme $S \in \mathbf{S}$, $S = S^+$. Let G be a nonredundant cover for F. $T_{\mathbf{S}}$ embeds G by Theorem 9.10. For any FD $X \rightarrow Y$ in G, there is a relation scheme S in $\mathbf{S}$ such that $XY \subseteq S$. Since $S = S^+$, $X^+ \subseteq S^+$. We conclude $\mathbf{S} \geq \mathbf{R}_F$. Therefore, $FIX(\mathbf{S}) \supseteq FIX(\mathbf{R}_F)$, and so $FIX_F(\mathbf{S}) \supseteq FIX_F(\mathbf{R}_F)$. By the definition of $\mathbf{S}$, $FIX_F(\mathbf{R}) = FIX_F(\mathbf{S})$. We see that $FIX_F(\mathbf{R}_F) \subseteq FIX(\mathbf{R})$.

(only if) Obviously, R_F preserves $SAT(F)$. By the second corollary to Theorem 9.5, $T_{\mathbf{R}} \sqsubseteq_F T_{\mathbf{R}_F}$. We know $m_{\mathbf{R}_F}(r)$ is in $SAT(F)$ for any instance r in $SAT(F)$. Also, $m_{\mathbf{R}}(r) \subseteq m_{\mathbf{R}_F}(r)$, so by Exercise 9.26, $m_{\mathbf{R}}(r)$ is in $SAT(F)$. Hence R preserves $SAT(F)$.

Example 9.23 Let $\mathbf{R} = \{ABC, CD, DEI\}$ and let $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow A, ADE \rightarrow I\}$, as in Examples 9.21 and 9.22. We saw in Example 9.22 that $\mathbf{R}_F = \{ABC, ACD, AD, ADEI\}$. Tableau T in Figure 9.24 is the chase of $T_{\mathbf{R}}$ under F and $*[\mathbf{R}_F]$. T contains the row of all distinguished variables, so F and $*[\mathbf{R}_F]$ imply $*[\mathbf{R}]$. Hence $FIX_F(\mathbf{R}_F) \subseteq FIX(\mathbf{R})$ and $\mathbf{R}$ preserves F, as we previously determined in Example 9.21.

$T(A$	B	C	D	E	I $)$
a_1	a_2	a_3	a_4	b_2	b_3
a_1	b_5	a_3	a_4	b_6	b_7
a_1	b_9	b_{10}	a_4	a_5	a_6
a_1	a_2	a_3	a_4	a_5	a_6
a_1	b_5	a_3	a_4	a_5	a_6
a_1	a_2	a_3	a_4	b_6	b_7
a_1	b_5	a_3	a_4	b_2	b_3
a_1	b_9	b_{10}	a_4	b_6	b_7
a_1	b_9	b_{10}	a_4	b_2	b_3

Figure 9.24

9.4 EXERCISES

9.1 Suppose all domains for attributes in $\mathbf{U}$ are finite. Compare the cardinalities of $\mathbf{Q}$, $\mathbf{P}$, $\mathbf{M}$, $\pi\mathbf{Q}$, $\pi\mathbf{P}$, $\mathbf{L}$ and $\bowtie\mathbf{L}$.

9.2 Prove that $\pi\mathbf{Q}$ is the set of all databases over scheme $\mathbf{R}$ where the relations join completely.

9.3 Show that for arbitrary $n \geq 3$ there is always a set of n relations such that any $n - 1$ join completely, but all n do not.

9.4 Show that the containment $\mathbf{L} \supseteq \pi\mathbf{P}$ is proper.

9.5 Prove that condition IC2 is equivalent to $\pi\mathbf{P} = \mathbf{L}$.

9.6 Show that database scheme $\mathbf{R}$ decomposes $\mathbf{P}$ into independent components, where $\mathbf{R}$ and $\mathbf{P}$ are given in Example 9.6. Describe the inverse mapping to $\pi_{\mathbf{R}}$.

9.7 Give an example of a database scheme $\mathbf{R}$ and an infinite set of instances $\mathbf{P}$ such that $\mathbf{R}$ satisfies AC1 but not AC2.

9.8 Show that the condition $\mathbf{L} = \pi\mathbf{P}$ does not necessitate AC2.

9.9 Show that AC1 and LJ can be inequivalent when AC2 does not hold.

9.10 Let X, Y, and Z be subsets of $\mathbf{U}$. Show that for an instance $r \in \mathbf{Q}$, r satisfies $X \rightarrow Y$ if and only if $\pi_{XYZ}(r)$ satisfies $X \rightarrow Y$. Show that the statement is false if $X \rightarrow Y$ is replaced by $X \twoheadrightarrow Y$.

9.11 Show that for any instance $r \in \mathbf{Q}$ where $\pi(r) = d$, $\bowtie d \supseteq r$.

9.12 Show that the join condition requires $\bowtie$ to be the inverse of π on $\mathbf{P}$.

9.13 Prove Lemma 9.5.

9.14 Prove that given condition J1, $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow} \subseteq \mathbf{P}$ implies $FIX(\mathbf{R}) \cap \mathbf{L}^{\leftarrow} = \mathbf{P}$.

9.15 Show that IC1 and IC2 do not require condition LJ.

9.16 Prove Lemma 9.6.

9.17 Show that conditions LJ and J2 hold if and only if conditions J1 and J2 hold.

9.18 (a) Show that information preservation is equivalent to property J2 and $FIX(\mathbf{R}) \supseteq \mathbf{L}^{\leftarrow}$.

(b) Show that the join condition is equivalent to property J2 and $FIX(\mathbf{R}) \supseteq \mathbf{P}$.

9.19 Prove Lemma 9.9.

9.20 Prove that $\mathbf{R} \approx_{\mathbf{P}} I$ is equivalent to $T_{\mathbf{R}} \equiv_{\mathbf{P}} T_I$, where I is a database scheme containing $\mathbf{U}$ as the only relation scheme.

9.21 Show that $FIX_{\mathbf{P}}(\mathbf{R})$ is contained by $PRES_{\mathbf{P}}(\mathbf{R})$, and that the containment can be proper.

9.22 Prove the first corollary to Theorem 9.5.

9.23 Show that if instance $r \in \mathbf{P}$ satisfies the MVD $X \twoheadrightarrow Y$, then $\pi_{\mathbf{R}}(r)$ satisfies the MVD $X \twoheadrightarrow Y \cap R$ if $X \subseteq R$.

9.24* Characterize the join dependencies that must hold in $\pi_R(r)$ when instance r satisfies some join dependency *[S].

9.25 In Example 9.12, find a data dependency that $\pi_R(r)$ must satisfy if r is in $\mathbf{P}$, but which s violates.

9.26 Show that if r is an instance violating a set of FDs F, then any instance containing r violates F. Show that if r satisfies F, then every relation contained in r satisfies F.

9.27 Let $\mathbf{P} = SAT(F)$ for a set of FDs F. Show that $\mathbf{L}^{\leftarrow} - \mathbf{P}$ contains an instance with two tuples.

9.28 Show properties LJ and J2 are equivalent to the join condition.

9.29* Characterize when $SAT(F_{\mathbf{R}}) = SAT(\pi F)$ for a set F of FDs over $\mathbf{U}$.

9.30 Complete the proof of Lemma 9.10.

9.31 Compute $(AC)^+$ under $F_{\mathbf{R}}$ where $\mathbf{U} = ABCDEI$, $\mathbf{R} = \{ABC, CDE, AEI\}$, and $F = \{AB \rightarrow D, D \rightarrow I, E \rightarrow I, BC \rightarrow A, I \rightarrow B\}$.

9.32* Find a polynomial-time algorithm that given $\mathbf{R}$ and F produces a cover G for $F_{\mathbf{R}}$ such that every FD in G applies to some relation scheme $R \in \mathbf{R}$, *provided* $F \equiv F_{\mathbf{R}}$. Show that your algorithm fails to produce a cover for $F_{\mathbf{R}}$ when $F \not\equiv F_{\mathbf{R}}$. (How did I know that would happen?)

9.33 For Example 9.18, exhibit an instance in $\bowtie \mathbf{L}$ that is not in $SAT(\mathbf{C})$.

9.34 Prove Lemma 9.14.

9.35 For database schemes **R** and **S**, and a set **C** of dependencies, show that $T_{\mathbf{R}} \not\sqsubseteq_{\mathbf{C}} T_{\mathbf{S}}$ implies $\mathbf{R} \not\approx_{\mathbf{C}} \mathbf{S}$.

9.36 Let **R** be a database scheme that preserves $SAT(F)$ for a set F of FDs. Show that $chase_F(T_{\mathbf{R}})$ is $T_{\mathbf{S}}$ for some database scheme **S**. Describe **S** in terms of **R** and F.

9.37 Let **R** be a database scheme and let F be a set of FDs. Prove the following. Let w be a row of $T_{\mathbf{R}}$ and let w^* be the corresponding row of $chase_F(T_{\mathbf{R}})$. If w is distinguished on exactly X and w^* is distinguished on Y, then $F \models X \rightarrow Y$.

9.38 Let **R** be a database scheme, let F be a set of FDs and let $X \subseteq \mathbf{U}$. Show that if **R** preserves $SAT(F)$, then $R \cup \{X\}$ preserves $SAT(F)$.

9.39 Let T be a tableau that embeds the FD $X \rightarrow Y$. Show that for any instance r in $SAT(X \rightarrow Y)$, $T(r) \in SAT(X \rightarrow Y)$.

9.40 Prove that different choices for G in definition of $\mathbf{R}_F$ do not yield different sets of attributes.

9.5 BIBLIOGRAPHY AND COMMENTS

The material from Section 9.1 is largely from Maier, Mendelzon, *et al.* [1980]. Sections 9.2 and 9.3 follow from Beeri, Mendelzon, *et al.* [1979]. Rissanen [1977] proposed the IC conditions; the AC conditions are from Arora and Carlson [1978]. The algorithm for determining enforceability of a set F of FDs is due to Beeri and Honeyman [1981], who also show it is NP-complete to test if a set G is a cover for the subset of F^+ enforceable on a scheme. Beeri and Rissanen [1980] and Graham [1981a, 1981b] have also dealt with equivalence database schemes under constraints and preserving dependencies.

Ginsburg and Hull [1980] and Ginsburg and Zaiddan [1981] have studied the structure of $SAT(F)$ for a set F of FDs. In particular, Ginsburg and Zaiddan noted that $\pi_R(SAT(F))$ is not necessarily $SAT(F')$ for any set of FDs F' over **R**. Sadri [1980a] noted the similar situation for JDs.