Chapter 8

PROJECT-JOIN MAPPINGS, TABLEAUX, AND THE CHASE

We did not present a set of inference axioms for JDs in Chapter 7. Instead, in this chapter we present a method for deciding if a given FD or JD is implied by a set of FDs and JDs.

8.1 PROJECT-JOIN MAPPINGS

The criterion for a relation $r(R)$ decomposing losslessly onto a database scheme $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ is that $r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_p}(r)$. The right side of this equation is rather cumbersome, so we give a shorter notation for it.

Definition 8.1 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a set of relation schemes, where $R = R_1R_2 \cdots R_p$. The *project-join mapping* defined by $\mathbf{R}$, written $m_{\mathbf{R}}$, is a function on relations over R defined by

$$m_{\mathbf{R}}(r) = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_p}(r).$$

Example 8.1 Let $R = ABCDE$ and let $\mathbf{R} = \{ABD, BC, ADE\}$. Consider the relation $r(R)$ in Figure 8.1. The result of applying $m_{\mathbf{R}}$ to r is the relation $s(R)$ shown in Figure 8.2. Applying $m_{\mathbf{R}}$ to s gives back relation s.

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

Figure 8.1

s(A	B	C	D	E)
a	b	c	d	e
a	b'	c	d'	e
a	b'	c	d'	e'
a	b	c	d'	e'
a	b	c	d'	e

Figure 8.2

Saying that a relation $r(R)$ satisfies the JD $*[\mathbf{R}]$ is the same as saying $m_{\mathbf{R}}(r) = r$.

Definition 8.2 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$, where $R = R_1R_2 \cdots R_p$. Relation $r(R)$ is a *fixed-point* of the mapping $m_{\mathbf{R}}$ if $m_{\mathbf{R}}(r) = r$. The set of all fixed-points of $m_{\mathbf{R}}$ is denoted *FIX*($\mathbf{R}$).

Example 8.2 If $\mathbf{R} = \{ABD, BC, ADE\}$, then the relation r in Figure 8.1 is not in *FIX*($\mathbf{R}$), while the relation s in Figure 8.2 is in *FIX*($\mathbf{R}$).

We present some other properties of project-join mappings.

Lemma 8.1 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a set of relation schemes where $R = R_1R_2 \cdots R_p$ and let r and s be relations over R. The project-join mapping $m_{\mathbf{R}}$ has the following properties:

1. $r \subseteq m_{\mathbf{R}}(r)$;
2. if $r \subseteq s$, then $m_{\mathbf{R}}(r) \subseteq m_{\mathbf{R}}(s)$ (monotonicity);
3. $m_{\mathbf{R}}(r) = m_{\mathbf{R}}(m_{\mathbf{R}}(r))$ (idempotence).

Proof The proof of part 1 is left to the reader (see Exercise 8.2). Part 2 follows from the observation that $r \subseteq s$ implies $\pi_{R_i}(r) \subseteq \pi_{R_i}(s)$, $1 \le i \le p$. Let $r' = m_{\mathbf{R}}(r)$; part 3 follows from the property that $\pi_{R_1}(r)$, $\pi_{R_2}(r)$, $\ldots$, $\pi_{R_p}(r)$ join completely (see Exercise 2.16), hence $\pi_{R_i}(r) = \pi_{R_i}(r')$, $1 \le i \le p$.

We would like to know when relations on a relation scheme R can be represented as databases on a database scheme $\mathbf{R}$ such that

1. there is no loss of information, and
2. redundancy is removed.

In practice, we are not interested in all possible relations on scheme R, only some subset. Call it $\mathbf{P}$. The first point above corresponds to saying that for every relation r in **P**, $m_{\mathbf{R}}(r) = r$. That is, $\mathbf{P} \subseteq FIX(\mathbf{R})$. The second point seems to require that if we project a relation r in **P** into the schemes in **R**, some of the projections have fewer tuples than r.

The set **P** will usually be infinite, hence it cannot be described by enumeration. Rather, **P** will frequently be specified by a set of constraints (such as FDs or JDs) on relations on R.

Definition 8.3 Let **C** be a set of constraints on a relation scheme R. $SAT_R(\mathbf{C})$ is the set of all relations r on R that satisfy all the constraints in **C**. We write $SAT(\mathbf{C})$ for SAT_R (**C**) when R is understood, and we write $SAT_R(c)$ for $SAT_R(\{c\})$, where c is a single constraint.

We can now state precisely the notion of implication we have been using informally in our discussions of MVDs and JDs.

Definition 8.4 Let **C** be a set of constraints over relation scheme R. **C** *implies* c, written $\mathbf{C} \models c$, if $SAT_R(\mathbf{C}) \subseteq SAT_R(c)$.

If $\mathbf{P} = \mathrm{SAT}(\mathbf{C})$ for some set of constraints **C**, then our condition requiring no loss of information for databases on database scheme **R** can be stated as

$$SAT(\mathbf{C}) \subseteq FIX(\mathbf{R}) \quad \text{or}$$
$$\mathbf{C} \models *[\mathbf{R}]$$

In subsequent sections we shall develop a test for this condition, when **C** is composed of JDs and FDs.

8.2 TABLEAUX

In this section we present a tabular means of representing project-join mappings; a *tableau*. A tableau is similar to a relation, except, in place of values, a tableau has variables chosen from a set V. V is the union of two sets, V_d and V_n. V_d is the set of *distinguished variables*, denoted by subscripted a's, and V_n is the set of *nondistinguished variables*, denoted by subscripted b's. (We shall use variable and symbol synonymously in this context.) A tableau, T, is shown in Figure 8.3. The set of attributes labeling columns in the tableau, in this case $A_1 A_2 A_3 A_4$, is the *scheme* of the tableau. What would be tuples in a relation are referred to as *rows* of the tableau.

$T(A_1$	A_2	A_3	$A_4)$
a_1	b_1	a_3	b_2
b_3	a_2	a_3	b_4
a_1	b_5	a_3	a_4

Figure 8.3

We restrict the variables in a tableau to appear in only one column. We make the further restriction that at most one distinguished variable may appear in any column. By convention, if the scheme of a tableau is $A_1 A_2 \cdots A_n$, then the distinguished variable appearing in the A_i-column will be a_i.

A tableau T with scheme R can be viewed as a pattern or template for a relation on scheme R. We get a relation from the tableau by substituting domain values for variables. Assume $R = A_1 A_2 \cdots A_n$ and let

$$\mathbf{D} = \cup_{i=1}^{n} dom(A_i).$$

A *valuation* for tableau T is a mapping ρ from V to $\mathbf{D}$ such that $\rho(v)$ is in $dom(A_i)$ when v is a variable appearing in the A_i-column. We extend the valuation from variables to rows and thence to the entire tableau. If $w = \langle v_1 v_2 \cdots v_n \rangle$ is a row in a tableau, we let $\rho(w) = \langle \rho(v_1)\ \rho(v_2) \cdots \rho(v_n) \rangle$. We then let

$$\rho(T) = \{\rho(w) \mid w \text{ is a row in } T\}.$$

Example 8.3 Let ρ be the valuation listed in Figure 8.4. The result of applying ρ to tableau T in Figure 8.3 is the relation r in Figure 8.5.

$\rho(a_1) = 1$	$\rho(b_1) = 4$
$\rho(a_2) = 3$	$\rho(b_2) = 8$
$\rho(a_3) = 5$	$\rho(b_3) = 2$
$\rho(a_4) = 7$	$\rho(b_4) = 7$
	$\rho(b_5) = 4$

Figure 8.4

$r(A_1$	A_2	A_3	$A_4)$
1	4	5	8
2	3	5	7
1	4	5	7

Figure 8.5

8.2.1 Tableaux as Mappings

We can interpret a tableau T with scheme R as a function on relations with scheme R. Let w_d be the row of all distinguished variables. That is, if $R = A_1A_2 \cdots A_n$, $w_d = \langle a_1\, a_2 \cdots a_n \rangle$. (Row w_d is not necessarily in T.) If r is a relation on scheme R, we let

$$T(r) = \{\rho(w_d) \mid \rho(T) \subseteq r\}.$$

This definition says that if we find a valuation ρ that takes every row in T to a tuple in r, then $\rho(w_d)$ is in $T(r)$.

Example 8.4 Let r be the relation shown in Figure 8.6 and let T be the tableau in Figure 8.3. The valuation ρ in Figure 8.4 shows us that the tuple $\langle 1\ 3\ 5\ 7 \rangle$ must be in $T(r)$. The valuation ρ' in Figure 8.7 puts $\langle 2\ 4\ 5\ 7 \rangle$ in $T(r)$. All of $T(r)$ is given as relation s in Figure 8.8.

$r(A_1$	A_2	A_3	$A_4)$
1	4	5	8
2	3	5	7
1	4	5	7
2	3	6	7

Figure 8.6

$\rho'(a_1) = 2$	$\rho'(b_1) = 3$
$\rho'(a_2) = 4$	$\rho'(b_2) = 7$
$\rho'(a_3) = 5$	$\rho'(b_3) = 1$
$\rho'(a_4) = 7$	$\rho'(b_4) = 8$
	$\rho'(b_5) = 3$

Figure 8.7

$T(r) = s(A_1$	A_2	A_3	$A_4)$
1	4	5	8
2	4	5	7
1	4	5	7
1	3	5	8
1	3	5	7
2	3	5	7
2	3	6	7

Figure 8.8

When evaluating $T(r)$, if the A_i-column in T has no distinguished variable in it, then there is no restriction on the value of $\rho(a_i)$. If $\rho(T) \subseteq r$, then $\rho'(T) \subseteq r$, for any ρ' that agrees with ρ on V except on a_i. Thus, if $dom(A_i)$ is infinite, $T(r)$ can have infinitely many tuples and hence will not be a relation. Whenever we want to consider a tableau T as a function from relations to relations, we require that T have a distinguished symbol in every column (see Exercise 8.5).

8.2.2 Representing Project-Join Mappings as Tableaux

It is always possible to find a tableau T that represents the same function as any project-join mapping $m_{\mathbf{R}}$. Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a set of relation schemes, where $R = R_1R_2 \cdots R_p$. The *tableau for* $\mathbf{R}$, $T_{\mathbf{R}}$, is defined as follows: The scheme for $T_{\mathbf{R}}$ is R. $T_{\mathbf{R}}$ has p rows, $w_1, w_2, \ldots, w_p$. Assume $R = A_1 A_2 \cdots A_n$. Row w_i has the distinguished variable a_j in the A_j-column exactly when $A_j \in R_i$. The rest of w_i is unique nondistinguished symbols—nondistinguished symbols that appear in no other rows of $T_{\mathbf{R}}$.

Example 8.5 Let $\mathbf{R} = \{A_1A_2, A_2A_3, A_3A_4\}$. The tableau $T_{\mathbf{R}}$ is shown in Figure 8.9.

$T_{\mathbf{R}}(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4

Figure 8.9

Lemma 8.2 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a set of relation schemes, where $R = R_1R_2 \cdots R_p$. The project-join mapping $m_{\mathbf{R}}$ and the tableau $T_{\mathbf{R}}$ define the same function between relations over R.

Proof Left to the reader (see Exercise 8.7).

Example 8.6 If $\mathbf{R} = \{A_1A_2, A_2A_3, A_3A_4\}$ and r is the relation shown in Figure 8.10, then $m_{\mathbf{R}}(r) = T_{\mathbf{R}}(r) = s$, where s is the relation in Figure 8.11.

$r(A_1$	A_2	A_3	$A_4)$
1	3	5	7
1	4	5	7
2	3	6	8

Figure 8.10

$s(A_1$	A_2	A_3	$A_4)$
1	3	5	/
1	3	6	8
1	4	5	7
2	3	5	7
2	3	6	8

Figure 8.11

8.3 TABLEAUX EQUIVALENCE AND SCHEME EQUIVALENCE

Definition 8.5 Let T_1 and T_2 be tableaux over scheme R. We write $T_1 \sqsupseteq T_2$ if $T_1(r) \supseteq T_2(r)$ for all relations $r(R)$. Tableaux T_1 and T_2 are *equivalent*, written $T_1 \equiv T_2$, if $T_1 \sqsupseteq T_2$ and $T_2 \sqsupseteq T_1$. That is, $T_1 \equiv T_2$ if $T_1(r) = T_2(r)$ for every relation $r(R)$.

Example 8.7 Let T_1 and T_2 be the tableaux in Figures 8.12 and 8.13, respectively. $T_1 \sqsupseteq T_2$. For example, if r is the relation in Figure 8.10, $T_1(r)$ is the relation s in Figure 8.11, while $T_2(r) = r$.

$T_1(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4

Figure 8.12

$T_2(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	b_3	a_3	a_4

Figure 8.13

Definition 8.6 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ and $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be sets of relation schemes, where $R_1R_2 \cdots R_p = S_1S_2 \cdots S_q = R$. $\mathbf{R}$ *covers* $\mathbf{S}$, written $\mathbf{R} \geq \mathbf{S}$, if for every scheme S_j in $\mathbf{S}$, there exists an R_i in $\mathbf{R}$ such that $R_i \supseteq S_j$. We say $\mathbf{R}$ and $\mathbf{S}$ are *equivalent*, written $\mathbf{R} \approx \mathbf{S}$, if $\mathbf{R} \geq \mathbf{S}$ and $\mathbf{S} \geq \mathbf{R}$.

Example 8.8 If $\mathbf{R} = \{A_1A_2, A_2A_3, A_3A_4\}$ and $\mathbf{S} = \{A_1A_2A_3, A_3A_4\}$, then $\mathbf{R} \leq \mathbf{S}$.

Theorem 8.1 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ and $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be sets of relation schemes, where $R_1R_2 \cdots R_p = S_1S_2 \cdots S_q = R$. The following are equivalent:

1. $m_{\mathbf{R}}(r) \supseteq m_{\mathbf{S}}(r)$ for all relations $r(R)$.
2. $\mathrm{T}_{\mathbf{R}} \sqsupseteq \mathrm{T}_{\mathbf{S}}$.
3. $FIX(\mathbf{R}) \subseteq FIX(\mathbf{S})$.
4. $\mathbf{R} \leq \mathbf{S}$.

Proof By Lemma 8.2, 1 and 2 are equivalent. We next show 1 and 3 are equivalent.

Suppose $m_{\mathbf{R}}(r) \supseteq m_{\mathbf{S}}(r)$ for all relations $r(R)$. Let s be in $FIX(\mathbf{R})$. Since $m_{\mathbf{R}}(s) = s$, $s \supseteq m_{\mathbf{S}}(s)$. But, by Lemma 8.1, $s \subseteq m_{\mathbf{S}}(s)$. Therefore $s = m_{\mathbf{S}}(s)$ and $s \in FIX(\mathbf{S})$. Thus we conclude $FIX(\mathbf{R}) \subseteq FIX(\mathbf{S})$.

Now suppose $FIX(\mathbf{R}) \subseteq FIX(\mathbf{S})$. By idempotence, for any relation $r(R)$,

$$m_{\mathbf{R}}(r) = m_{\mathbf{R}}(m_{\mathbf{R}}(r)).$$

Hence $m_{\mathbf{R}}(r)$ is in $FIX(\mathbf{R})$ and $FIX(\mathbf{S})$:

$$m_{\mathbf{S}}(m_{\mathbf{R}}(r)) = m_{\mathbf{R}}(r).$$

From Lemma 8.1 we know $m_{\mathbf{R}}(r) \supseteq r$, so by monotonicity

$$m_{\mathbf{S}}(m_{\mathbf{R}}(r)) \supseteq m_{\mathbf{S}}(r),$$

hence

$$m_{\mathbf{R}}(r) \supseteq m_{\mathbf{S}}(r).$$

Last, we show that 1 and 4 are equivalent.

Suppose $m_{\mathbf{R}}(r) \supseteq m_{\mathbf{S}}(r)$ for all relations $r(R)$. We assume for each attribute A in R, $dom(A)$ has at least two values, which we shall call 0 and 1. We construct a relation $s(R)$ as follows: Relation s has q tuples, $t_1, t_2, \ldots, t_q$. The tuple t_i is defined as

$$t_i(A) = \begin{cases} 0 & \text{if } A \in S_i \\ 1 & \text{otherwise,} \end{cases} \qquad 1 \leq i \leq q.$$

Let t_0 be the tuple of all 0's. It is not hard to see that t_0 must be in $m_{\mathbf{S}}(s)$. Therefore, t_0 is in $m_{\mathbf{R}}(s)$. By the nature of $m_{\mathbf{R}}$, for each relation scheme R_i in

R, there has to be a tuple t_j in s such that $t_j(R_i) = t_0(R_i)$. Thus, $R_i \subseteq S_j$ and $\mathbf{R} \leq \mathbf{S}$.

Now suppose $\mathbf{R} \leq \mathbf{S}$. Let $r(R)$ be an arbitrary relation and let t be any tuple in $m_{\mathbf{S}}(r)$. There must be tuples $t_1, t_2, \ldots, t_q$ in r such that $t_i(S_i) = t(S_i)$, $1 \leq i \leq p$. For any R_j such that $R_j \subseteq S_i$, $t_i(R_j) = t(R_j)$. Since $\mathbf{R} \leq \mathbf{S}$, for any R_j in **R** there is a tuple t_j' in r such that $t_j'(R_j) = t(R_j)$. We see that t is in $m_{\mathbf{R}}(r)$ and hence $m_{\mathbf{R}}(r) \supseteq m_{\mathbf{S}}(r)$.

Example 8.9 Let $\mathbf{R} = \{A_1A_2, A_2A_3, A_3A_4\}$ and $\mathbf{S} = \{A_1A_2A_3, A_3A_4\}$, as in Example 8.8. We see that tableau T_1 in Figure 8.12 is $T_{\mathbf{R}}$ and that Tableau T_2 in Figure 8.13 is $T_{\mathbf{S}}$. Since $\mathbf{R} \leq \mathbf{S}$, by Theorem 8.1, $T_{\mathbf{R}} \sqsupseteq T_{\mathbf{S}}$. For example, if r is the relation in Figure 8.14, then $T_{\mathbf{R}}(r)$ is given in Figure 8.15 and $T_{\mathbf{S}}(r)$ is given in Figure 8.16. Evidently, $T_{\mathbf{R}}(r) \supseteq T_{\mathbf{S}}(r)$.

$r(A_1$	A_2	A_3	$A_4)$
1	4	6	8
2	4	7	9
3	5	7	10

Figure 8.14

$T_{\mathbf{R}}(r)(A_1$	A_2	A_3	$A_4)$
1	4	6	8
1	4	7	9
1	4	7	10
2	4	6	8
2	4	7	9
2	4	7	10
3	5	7	9
3	5	7	10

Figure 8.15

$T_{\mathbf{S}}(r)(A_1$	A_2	A_3	$A_4)$
1	4	6	8
2	4	7	9
2	4	7	10
3	5	7	9
3	5	7	10

Figure 8.16

Corollary Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ and $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be sets of relation schemes, where $R_1R_2 \cdots R_p = S_1S_2 \cdots S_q = R$. The following are equivalent.

1. $m_{\mathbf{R}} = m_{\mathbf{S}}$
2. $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$
3. $FIX(\mathbf{R}) = FIX(\mathbf{S})$
4. $\mathbf{R} \simeq \mathbf{S}$

Condition 1 means $m_{\mathbf{R}}(r) = m_{\mathbf{S}}(r)$ for all relations $r(R)$. Note that conditions 2 and 4 use equivalence rather than equality. Equivalence can hold without equality.

Example 8.10 Let $\mathbf{R} = \{A_1A_2A_3, A_1A_4, A_1A_3A_4\}$ and $\mathbf{S} = \{A_1A_2A_3, A_3A_4, A_1A_3A_4\}$ be sets of relation schemes. $\mathbf{R} \geq \mathbf{S}$ and $\mathbf{S} \geq \mathbf{R}$, so $\mathbf{R} \simeq \mathbf{S}$. By the corollary to Theorem 8.1, $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$. But as we see from Figures 8.17 and 8.18, $T_{\mathbf{R}} \neq T_{\mathbf{S}}$, even if we rename nondistinguished variables.

$T_{\mathbf{R}}(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
a_1	b_2	b_3	a_4
a_1	b_4	a_3	a_4

Figure 8.17

$T_{\mathbf{S}}(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	b_3	a_3	a_4
a_1	b_4	a_3	a_4

Figure 8.18

Although we can have $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$ without $T_{\mathbf{R}} = T_{\mathbf{S}}$, if $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$, then $T_{\mathbf{R}}$ and $T_{\mathbf{S}}$ will exhibit a certain similarity.

Definition 8.7 Let w_1 and w_2 be rows in a tableau T with scheme R. If for every attribute A in R, $w_2(A)$ is a distinguished variable implies $w_1(A)$ is a distinguished variable, then w_1 is said to *subsume* w_2.

Example 8.11 In Figure 8.17, the third row subsumes the second row. In Figure 8.18, the third row also subsumes the second row.

Definition 8.8 Let T be a tableau. T *reduced by subsumption*, denoted $SUB(T)$, is the tableau consisting of the set of rows in T that are not subsumed by any other row of T.

Example 8.12 $SUB(T_{\mathbf{R}})$ is given in Figure 8.19, for the tableau $T_{\mathbf{R}}$ in Figure 8.17.

$SUB(T_{\mathbf{R}})(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
a_1	b_4	a_3	a_4

Figure 8.19

Theorem 8.2 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ and $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be sets of relation schemes where $R_1R_2 \cdots R_p = S_1S_2 \cdots S_q = R$. $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$ if and only if $SUB(T_{\mathbf{R}})$ is identical to $SUB(T_{\mathbf{S}})$, except for possibly a one-to-one renaming of the nondistinguished symbols.

Proof Left to the reader (see Exercise 8.13).

Example 8.13 Let $\mathbf{R} = \{A_1A_2A_3, A_1A_4, A_1A_3A_4\}$ and $\mathbf{S} = \{A_1A_2A_3, A_3A_4, A_1A_3A_4\}$ be sets of relation schemes, as in Example 8.10. $SUB(T_{\mathbf{R}})$, shown in Figure 8.19, is identical to $SUB(T_{\mathbf{S}})$.

Corollary $SUB(T_{\mathbf{R}}) \equiv T_{\mathbf{R}}$.

8.4 CONTAINMENT MAPPINGS

As we see from Theorem 8.2, there is a simple test for equivalence of tableaux that come from sets of schemes, namely, identity of subsumption-reduced versions. Any tableau where no nondistinguished variable occurs more than once comes from some set of schemes. Unfortunately, Theorem 8.2 does not hold for tableaux where some nondistinguished variables are duplicated.

Example 8.14 Consider the tableau T in Figure 8.20 and its subsumption-reduction $SUB(T)$ in Figure 8.21. Let r be the relation in Figure 8.22. Certainly $SUB(T)$ is identical to $SUB(SUB(T))$. However, $T(r) = r$, whereas $SUB(T)(r) = r'$, where r' is the relation given in Figure 8.23.

$T(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	a_4
b_4	a_2	a_3	b_2

Figure 8.20

$SUB(T)(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	a_4

Figure 8.21

$r(A_1$	A_2	A_3	$A_4)$
1	3	4	6
2	3	5	7

Figure 8.22

$r'(A_1$	A_2	A_3	$A_4)$
1	3	4	6
1	3	5	7
2	3	4	6
2	3	5	7

Figure 8.23

We want to formulate a condition for equivalence of arbitrary tableaux. To do so we introduce containment mappings of tableaux. A containment mapping is quite similar to a valuation, but instead of mapping tableau variables to domain values, it maps them to variables in a second tableau, in such a way that rows are mapped to rows.

Definition 8.9 Let T and T' be tableaux on scheme R, with variable sets V and V'. A mapping ψ: $V \rightarrow V'$ is a *containment mapping from* T *to* T$'$ if the following conditions hold:

1. If variable v is in the A-column of T then $\psi(v)$ is in the A-column of T'.
2. If variable v is distinguished, then $\psi(v)$ is distinguished. (By our naming convention, $\psi(v) = v$.)

3. $\psi(T) \subseteq T'$. That is, when ψ is extended to rows of T and thence to T itself, it maps every row of T to a row in T'.

Example 8.15 Let T and T' be the tableaux in Figures 8.24 and 8.25. There is a containment mapping from T to T', namely ψ, where

$$\begin{aligned}
&\psi(a_i) = a_i, \quad 1 \leq i \leq 4\\
&\psi(b_1) = a_3\\
&\psi(b_2) = b_1\\
&\psi(b_3) = a_1\\
&\psi(b_4) = b_2\\
&\psi(b_5) = a_2.
\end{aligned}$$

The first two rows of T are mapped to the first row of T' by ψ; ψ maps the third row of T to the second row of T'. There is no containment mapping from T' to T, since, for example, the first row of T' would have to map to a row with at least the distinguished variables a_1, a_2 and a_3.

$T(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_2
b_4	b_5	a_3	a_4

Figure 8.24

$T'(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	a_2	a_3	a_4

Figure 8.25

Theorem 8.3 Let T and T' be tableaux over scheme R. $T \sqsupseteq T'$ if and only if there is a containment mapping from T to T'.

Proof (if) Let ψ be a containment mapping from T to T'. Take any relation $r(R)$ and look at $T(r)$ and $T'(r)$. If ρ is a valuation for T' such that $\rho(T') \subseteq r$, then $\rho \circ \psi$ is a valuation for T such that $\rho \circ \psi\,(T) \subseteq r$. The inclusion follows from $\psi(T) \subseteq T'$ by applying ρ to both sides. If w_d is the row of all distinguished variables, since $\psi(w_d) = w_d$, $\rho \circ \psi(w_d) = \rho(w_d)$, so $T(r) \supseteq T'(r)$.

(only if) Suppose $T \sqsupseteq T'$. Consider T' also as a relation. We have $T(T') \supseteq T'(T')$. Consider the valuation ρ' that is the identity on the variables V' of T'. Clearly $\rho'(T') = T' \subseteq T'$, so $\rho'(w_d) = w_d \in T'(T')$. There must be a valuation ρ for T such that $\rho(T) \subseteq T'$ and $\rho(w_d) = w_d$. We see that ρ can also be construed as a containment mapping from T to T'.

Example 8.16 We see that $T \sqsupseteq T'$, where T and T' are the tableaux in Figures 8.24 and 8.25. For example, if r is the relation in Figure 8.26, then $T(r) = r'$, where r' is given in Figure 8.27, while $T'(r) = r$, so $T(r) \supseteq T'(r)$.

$r(A_1$	A_2	A_3	$A_4)$
1	4	6	8
2	4	7	8
3	5	7	9

Figure 8.26

$r'(A_1$	A_2	A_3	$A_4)$
1	4	6	8
1	4	7	8
1	4	7	9
2	4	7	8
2	4	7	9
3	5	7	8
3	5	7	9

Figure 8.27

Example 8.17 Let T'' be the tableau in Figure 8.28. There is a containment mapping from T'' to T (What is it?), so $T'' \sqsupseteq T$. For the relation r in Figure 8.26, $T''(r) = r''$, where r'' is given in Figure 8.29. We see $T''(r) \supseteq T(r)$.

$T''(A_1$	A_2	A_3	$A_4)$
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4

Figure 8.28

$r''(A_1$	A_2	A_3	$A_4)$
1	4	6	8
1	4	7	8
1	4	7	9
2	4	6	8
2	4	7	8
2	4	7	9
3	5	7	8
3	5	7	9

Figure 8.29

Corollary Let T and T' be tableaux over scheme R. $T \equiv T'$ if and only if there is a containment mapping from T to T' and a containment mapping from T' to T.

Example 8.18 Let T be the tableau consisting of only the row w_d of all distinguished variables. Let T' be any tableau that contains w_d. $T \equiv T'$. The containment mapping from T to T' maps w_d to w_d. The containment mapping from T' to T maps every row to w_d.

8.5 EQUIVALENCE WITH CONSTRAINTS

We are trying to characterize when a relation can be faithfully represented by its projections. From the corollary to Theorem 8.1 and Theorem 8.2, we see that if $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ is a database scheme over R, then $FIX(R)$ is the set of all relations over R only if $R_i = R$ for some i. If $R_i = R$, there is no need for the other relation schemes in R, so R ends up being a single relation scheme. Thus, in general, the answer to the question, "When can relations over R be represented faithfully as database over a nontrivial database scheme $\mathbf{R}$?" is never.

We seldom deal in the most general case. We usually want to represent a set of relations over scheme R where some set of constraints is imposed. We can use those constraints to find nontrivial database schemes on which to represent the relations.

Definition 8.10 Let **P** be a set of relations over scheme R. If T_1 and T_2 are tableaux over R, then T_1 *contains* T_2 *on* **P**, written $T_1 \sqsupseteq_{\mathbf{P}} T_2$, if $T_1(r) \supseteq T_2(r)$ for every relation r in P. T_1 *and* T_2 *are equivalent on* **P**, written $T_1 \equiv_{\mathbf{P}} T_2$, if $T_1 \sqsupseteq_{\mathbf{P}} T_2$ and $T_2 \sqsupseteq_{\mathbf{P}} T_1$.

The set **P** will most often be expressed as $\mathbf{P} = SAT(\mathbf{C})$ for some set of constraints **C**. We abbreviate $\equiv_{SAT(C)}$ as $\equiv_{\mathbf{C}}$. Recall that we are interested in when $SAT(\mathbf{C}) \subseteq FIX(\mathbf{R})$ for a database scheme **R**. That is, for a given database scheme **R**, can every relation in $SAT(\mathbf{C})$ be losslessly decomposed onto **R**? In terms of constraints, we are asking whether $\mathbf{C} \models *[\mathbf{R}]$. If T_I is a tableau for the identity mapping (T_I contains the row of all distinguished variables), then we want to know if $T_{\mathbf{R}}$ behaves as T_I on $SAT(\mathbf{C})$. That is, is $T_{\mathbf{R}} \equiv_{\mathbf{C}} T_I$? Theorem 8.3 gives a test for $\sqsupseteq$; we need a test for $\sqsupseteq_{\mathbf{C}}$.

For the next lemma, we need to view a tableau as a relation. We have already used this device in the proof of Theorem 8.3. We must be more precise now, since we want to know when tableau T, considered as a relation, is in set **P**. What we mean by this condition is that for any valuation ρ, $\rho(T) \in \mathbf{P}$. For an arbitrary set of relations **P**, this conditions is hard to test. However, when $\mathbf{P} = SAT(\mathbf{C})$, where **C** consists of FDs and JDs, if for some one-to-one valuation ρ, $\rho(T) \in \mathbf{P}$, then for any other valuation ρ', $\rho'(T) \in \mathbf{P}$ (see Exercise 8.20).

Lemma 8.3 Let T_1 and T_2 be tableaux over scheme R and let **P** be a set of relations over R. Let T_1' and T_2' be tableaux such that

1. $T_1 \equiv_{\mathbf{P}} T_1'$ and $T_2 \equiv_{\mathbf{P}} T_2'$, and
2. T_1' and T_2' considered as relations are both in **P**.

Then $T_1 \sqsubseteq_{\mathbf{P}} T_2$ if and only if $T_1' \sqsubseteq T_2'$.

Proof The if direction is immediate. Clearly $T_1' \sqsubseteq T_2'$ implies $T_1' \sqsubseteq_{\mathbf{P}} T_2'$, so $T_1' \sqsubseteq T_2'$, $T_1 \equiv_{\mathbf{P}} T_1'$, and $T_2 \equiv_{\mathbf{P}} T_2$ imply $T_1 \sqsubseteq_{\mathbf{P}} T_2$. For the only if direction, $T_1 \sqsubseteq_{\mathbf{P}} T_2$, $T_1 \equiv_{\mathbf{P}} T_1'$ and $T_2 \equiv_{\mathbf{P}} T_2'$ imply $T_1' \sqsubseteq_{\mathbf{P}} T_2'$. We now show that $T_1' \sqsubseteq_{\mathbf{P}} T_2'$ implies $T_1' \sqsubseteq T_2'$.

Consider $T_1'(T_1')$. (We are treating T_1' simultaneously as a tableau and as a relation.) Since T_1', as a relation, is in **P**, $T_1'(T_1') \subseteq T_2'(T_1')$. Let w_d be the row of all distinguished variables and let ρ be the identity valuation for T_1'. Obviously, $\rho(T_1') \subseteq T_1'$, so $\rho(w_d) = w_d$ is in $T_1'(T_1')$ and hence in $T_2'(T_1')$. There must be a valuation η for T_2' such that $\eta(T_2') \subseteq T_1'$ and $\eta(w_d) = w_d$. The valuation η can be viewed as a containment mapping from T_2' to T_1'. Hence, by Theorem 8.3, $T_1' \sqsubseteq T_2'$.

Corollary For the hypotheses of Lemma 8.3, $T_1 \equiv_{\mathbf{P}} T_2$ if and only if $T_1' \equiv T_2'$.

Let us take stock. We are seeking a test for $T_1 \sqsubseteq_{\mathbf{C}} T_2$. We know how to test $T_1 \sqsubseteq T_2$. By Lemma 8.3, we could test $T_1 \sqsubseteq_{\mathbf{C}} T_2$ if we had a way to take an arbitrary tableau T and find a tableau T' such that $T \equiv_C T'$ and T' as a relation is in $SAT(\mathbf{C})$. We shall introduce *transformation rules* for tableaux. A transformation rule for a set of constraints $\mathbf{C}$ is a means to modify a tableau T to a tableau T' so that $T \equiv_C T'$.

We have seen a limited type of transformation rule in subsumption. For a tableau T with no duplicated nondistinguished variables, removing a subsumed row preserves equivalence. We shall look at transformation rules for a set of constraints $\mathbf{C}$ composed of FDs and JDs. The different transformation rules will actually correspond to individual FDs (F-rules) and JDs (J-rules). Repeated application of these transformation rules will yield a tableau that, as a relation, satisfies all the dependencies in $\mathbf{C}$.

For the rest of this chapter, $\mathbf{C}$ will always be a set of FDs and JDs over a set of attributes $\mathbf{U}$. $\mathbf{U}$ will be the scheme for all relations and tableaux.

8.5.1 F-rules

For every FD $X \rightarrow A$ in $\mathbf{C}$ there is an associated F-rule. The F-rule for $X \rightarrow A$ represents a class of transformations that can be applied to a tableau, depending on which rows are chosen.

Let tableau T have rows w_1 and w_2, where $w_1(X) = w_2(X)$. Let $w_1(A) = v_1$ and $w_2(A) = v_2$ and suppose $v_1 \neq v_2$. We apply the F-rule for $X \rightarrow A$ to T by identifying variables v_1 and v_2, to form a new tableau T'. Variables v_1 and v_2 are identified by renaming one of them to be the other. If one of v_1 and v_2 is distinguished, say v_1, then every occurrence of v_2 is replaced by v_1. If v_1 and v_2 are both non-distinguished, every occurrence of the one with the larger subscript is replaced by the one with the smaller subscript. Since a tableau is a set of rows, some rows may be identified by renaming.

Example 8.19 Let T be the tableau in Figure 8.30 and let $\mathbf{C} = \{A_1A_2 \rightarrow A_4, A_2A_4 \rightarrow A_3\}$. Applying the F-rule for $A_2A_4 \rightarrow A_3$ to the first and second rows of T identifies variables a_3 and b_3. Since a_3 is distinguished, it replaces b_3, to yield the tableau T' in Figure 8.31. The F-rule for $A_1A_2 \rightarrow A_4$ can be applied to the first and third rows of T' to identify variables b_1 and b_4. Since b_1 has the lower subscript, it replaces b_4. The first and third rows are now the same, so the result, T'' in Figure 8.32, has only two rows.

$T(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	a_2	b_3	b_1
a_1	a_2	b_3	b_4

Figure 8.30

$T'(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	a_2	a_3	b_1
a_1	a_2	a_3	b_4

Figure 8.31

$T''(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	a_2	a_3	b_1

Figure 8.32

Theorem 8.4 Let T' be the result of applying the F-rule for the FD $X \to A$ to tableau T. T and T' are equivalent on $SAT(X \to A)$.

Proof Left to the reader (see Exercise 8.23).

8.5.2 J-rules

Let $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be a set of relation schemes and let $*[\mathbf{S}]$ be a JD over $\mathbf{U}$. Let T be a tableau and let $w_1, w_2, \ldots, w_q$ be rows of T that are joinable on $\mathbf{S}$ with result w. Applying the J-rule for $*[\mathbf{S}]$ to T allows us to form the tableau $T' = T \cup \{w\}$.

Example 8.20 Let T be the tableau in Figure 8.33 and let $\mathbf{C} = \{*[A_1A_2A_4, A_1A_3A_4], *[A_1A_2, A_2A_3, A_3A_4]\}$. We can apply the J-rule for $*[A_1A_2, A_2A_3, A_3A_4]$ to the second row and the third row of T to generate the row $\langle a_1 a_2 b_3 a_4 \rangle$. The resulting tableau T' is given in Figure 8.34. The J-rule for $*[A_1A_2A_4, A_1A_3A_4]$ can be applied to the first and fourth rows of T' to generate the row $\langle a_1 b_1 b_3 a_4 \rangle$. Tableau T'' in Figure 8.35 is the result of this application.

$T(A_1$	A_2	A_3	$A_4)$
a_1	b_1	b_2	a_4
a_1	a_2	b_3	b_4
b_5	a_2	b_3	a_4

Figure 8.33

$T'(A_1$	A_2	A_3	$A_4)$
a_1	b_1	b_2	a_4
a_1	a_2	b_3	b_4
b_5	a_2	b_3	a_4
a_1	a_2	b_3	a_4

Figure 8.34

$T''(A_1$	A_2	A_3	$A_4)$
a_1	b_1	b_2	a_4
a_1	a_2	b_3	b_4
b_5	a_2	b_3	a_4
a_1	a_2	b_3	a_4
a_1	b_1	b_3	a_4

Figure 8.35

Theorem 8.5 Let $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$. Let T' be the result of applying the J-rule for $*[\mathbf{S}]$ to tableaux T. T and T' are equivalent on $SAT(*[\mathbf{S}])$.

Proof We must show that $T(r) = T'(r)$ for an arbitrary relation $r \in SAT(*[\mathbf{S}])$.

Let t' be any tuple in $T'(r)$. Let ρ be the valuation with $\rho(w_d) = t'$ (w_d is the all-distinguished row) and $\rho(T') \subseteq r$. We have $\rho(T) \subseteq \rho(T')$, since $T \subseteq T'$ (set containment), so $\rho(T) \subseteq r$, and $\rho(w_d) = t' \in T(r)$. Hence $T'(r) \subseteq T(r)$.

Now let t be any tuple in $T(r)$ and let ρ be the valuation with $\rho(w_d) = t$ and $\rho(T) \subseteq r$. The only tuple that could possibly be in $\rho(T')$ but not in $\rho(T)$ is $\rho(w)$, where w is the row generated by the J-rule for $*[\mathbf{S}]$ from rows $w_1, w_2, \ldots, w_q$ of T. It is left to the reader to show that if $w_1, w_2, \ldots, w_q$ are joinable on $\mathbf{S}$ with result w, then $\rho(w_1), \rho(w_2), \ldots, \rho(w_q)$ are joinable on $\mathbf{S}$ with result $\rho(w)$ (see Exercise 8.25). Since r is in $SAT(*[\mathbf{S}])$, and $\{\rho(w_1), \rho(w_2), \ldots, \rho(w_q)\} \subseteq \rho(T) \subseteq r$, $\rho(w)$ is in r. Therefore $\rho(T') \subseteq r$, and $\rho(w_d) = t \in T'(r)$. Hence $T(r) \subseteq T'(r)$ and $T(r) = T'(r)$.

8.6 THE CHASE

In this section we give a computation method, the *chase*, that finds, given a tableau T and set of dependencies $\mathbf{C}$, a new tableau T^* such that $T \equiv T^*$ and

T^* as a relation is in $SAT(\mathbf{C})$. Thus, using Lemma 8.3 and Exercise 8.18, we shall be able to test tableaux for equivalence under $\mathbf{C}$.

The chase computation is simply described. Given T and $\mathbf{C}$, apply the F- and J-rules associated with the FDs and JDs in $\mathbf{C}$, *as long as they make a change*. We shall prove that the order of application of the transformation rules is immaterial. By Theorems 8.4 and 8.5, if the computation terminates, it always yields a tableau $T^* \equiv_{\mathbf{C}} T$. What is harder to show is that the computation always halts and that the resulting tableau, T^*, is in $SAT(\mathbf{C})$.

Example 8.21 Let T be the tableau in Figure 8.36 and let $\mathbf{C} = \{*[ABC, BCD], B \to C, AD \to C\}$. (We use A,B,C,D for A_1, A_2, A_3, A_4 for readability.) Tableau $T = T_{\mathbf{R}}$ where $\mathbf{R} = \{AB, BD, ACD\}$. Applying the F-rule for $B \to C$ yields tableau T_1 in Figure 8.37. We then apply the J-rule for $*[ABC, BCD]$ to get T_2 in Figure 8.38 and apply the F-rule for $AD \to C$ to get T_3 in Figure 8.39. One more application of the J-rule for $*[ABC, BCD]$ yields tableau T^* in Figure 8.40. No more transformation rules that correspond to dependencies in $\mathbf{C}$ can be applied to change T^*. Also, T^*, as a relation, is in $SAT(\mathbf{C})$.

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

Figure 8.36

$T_1(A$	B	C	$D)$
a_1	a_2	b_1	b_2
b_3	a_2	b_1	a_4
a_1	b_6	a_3	a_4

Figure 8.37

$T_2(A$	B	C	$D)$
a_1	a_2	b_1	b_2
b_3	a_2	b_1	a_4
a_1	b_6	a_3	a_4
a_1	a_2	b_1	a_4

Figure 8.38

$T_3(A$	B	C	$D)$
a_1	a_2	a_3	b_2
b_3	a_2	a_3	a_4
a_1	b_6	a_3	a_4
a_1	a_2	a_3	a_4

Figure 8.39

$T^*(A$	B	C	$D)$
a_1	a_2	a_3	b_2
b_3	a_2	a_3	a_4
a_1	b_6	a_3	a_4
a_1	a_2	a_3	a_4
b_3	a_2	a_3	b_2

Figure 8.40

Definition 8.11 A *generating sequence* for tableau T under constraints **C** is a sequence of tableaux $T_0, T_1, T_2, \ldots$ where $T = T_0$ and T_{i+1} is obtained from T_i by applying an F- or J-rule for a dependency in **C**, $0 \leq i$. We require $T_i \neq T_{i+1}$. If the generating sequence has a last element T_n such that no F- or J-rules for **C** can be applied to T_n to make a change, then T_n is called a *chase of* T *under* **C**. $Chase_{\mathbf{C}}(T)$ represents all such chases.

Example 8.22 Let T and **C** be as in Example 8.21. T, T_1, T_2, T_3, T^* is a generating sequence for T under **C**. Therefore, $T^* \in chase_{\mathbf{C}}(T)$.

We need to keep track of rows during the chase computation for some of our subsequent proofs. Let tableau T' be derived from tableau T by the application of a J-rule. If w is a row in T, the *row corresponding to* w in T' is w itself. Let T' be derived from T by an F-rule that changes variable v to variable v'. If w is a row in T, the *row corresponding to* w in T' is w', where w' is row w with v replaced by v'. (If w does not contain v, then $w = w'$.)

If $T_0, T_1, \ldots, T_i, \ldots, T_j, \ldots$ is a generating sequence, and w_i is a row in T_i, we can extend the "corresponds" relation transitively, and write of the row w_j in T_j corresponding to w_i. That is, there are rows $w_{i+1}, w_{i+2}, \ldots, w_{j-1}$ where $w_k \in T_k$, such that w_{i+1} corresponds to w_i, w_{i+2} corresponds to $w_{i+1}, \ldots, w_j$ corresponds to w_{j-1}.

Example 8.23 In the generating sequence T, T_1, T_2, T_3, T^* of Example 8.22, the first rows of tableaux T_1, T_2, T_3, T^* all correspond to the first row of T. Also, the fourth row of T_3 corresponds to the fourth row of T_2.

For any row w in a tableau in a generating sequence, there is always a row corresponding to w in any later tableau in the sequence. However, w does not necessarily correspond to some row in an earlier tableau in the sequence, since w could have been generated by a J-rule. Distinct rows in one tableau may correspond to the same row in a later tableau (see Exercise 8.27).

Theorem 8.6 Given a tableau T and constraints **C**, every generating sequence for T under **C** is finite. Thus, $chase_{\mathbf{C}}(T)$ is never empty.

Proof Since tableaux are sets of rows, and no F- or J-rule introduces new variables, there are only a finite number of tableaux that can appear in a generating sequence for T under **C**. If we can show that no tableau appears twice in a generating sequence, we are done.

Let T_i and T_j be tableaux in a generating sequence, where $i < j$. If at some point in the subsequence $T_i, T_{i+1}, \ldots, T_j$ an F-rule was used, then T_i has some variable that T_j lacks, so $T_i \neq T_j$. If only J-rules were used in the subsequence, then T_j has at least one more row than T_i, so $T_i \neq T_j$.

Theorem 8.7 For any tableau T^* in $chase_{\mathbf{C}}(T)$, T^*, as a relation, is in $SAT(\mathbf{C})$.

Proof If T^* violates an FD $X \rightarrow A$ in **C**, there must be two rows w_1 and w_2 in T^* with $w_1(X) = w_2(X)$, but $w_1(A) \neq w_2(A)$. The F-rule for $X \rightarrow A$ can be applied to rows w_1 and w_2 to change T^*, which means T^* cannot be the last tableau in a generating sequence under **C**. Hence T^* satisfies $X \rightarrow A$. Similarly, if T^* violates a JD in **C**, then the J-rule for that JD can be applied to T^* to make a change.

Example 8.24 The tableau T in Figure 8.41 is $T_{\mathbf{R}}$ for $\mathbf{R} = \{AE, ADE, BCD\}$. The tableau T^* in Figure 8.42 is in $chase_{\mathbf{C}}(T)$, where $\mathbf{C} = \{AE \rightarrow D, D \rightarrow C, *[AB, BCDE]\}$. The J-rule for the JD in **C** is never used. We see that T^* satisfies **C**.

T(A	B	C	D	E)
a_1	b_1	b_2	b_3	a_5
a_1	b_4	b_5	a_4	a_5
b_6	a_2	a_3	a_4	b_7

Figure 8.41

$T^*(A$	B	C	D	$E)$
a_1	b_1	a_3	a_4	a_5
a_1	b_4	a_3	a_4	a_5
b_6	a_2	a_3	a_4	b_7

Figure 8.42

Corollary $Chase_{\mathbf{C}}(T) = \{T\}$ if and only if T, as a relation, is in $SAT(\mathbf{C})$.

8.6.1 The Finite Church-Rosser Property

The chase computation is an example of a replacement system. A *replacement system* is a pair $(Q, \Rightarrow)$, where Q is a set of objects and $\Rightarrow$ is an antireflexive binary relation on Q, called the *transformation relation*.* In our case, the chase computation is a replacement system for every set of constraints **C**. Q is the set of tableaux over **U**, and $T \Rightarrow T'$ if T' is obtained from T by applying an F- or J-rule corresponding to a dependency in **C**.

Definition 8.12 The relation $\overset{*}{\Rightarrow}$ is the reflexive, transitive closure of $\Rightarrow$. We read $T \overset{*}{\Rightarrow} T'$ as "T goes to T'" or "T' is reachable from T."

Definition 8.13 Given the replacement system $(Q, \Rightarrow)$, object $p \in Q$ is *irreducible* if $p \overset{*}{\Rightarrow} q$ implies $p = q$. That is, for no $q \neq p$ does $p \Rightarrow q$.

Definition 8.14 The replacement system $(Q, \Rightarrow)$ is *finite* if for every $p \in Q$ there is a constant c, depending on p, such that if $p \overset{*}{\Rightarrow} q$ in i steps, then $i \leq c$. That is, for any object p in Q, only a finite number of transformations can be applied to p before reaching an irreducible object.

Using Theorem 8.6, it follows that the replacement system for a given chase computation is finite. $Chase_{\mathbf{C}}(T)$ is all the irreducible tableaux reachable from T using F- and J-rules for **C**.

Definition 8.15 A finite replacement system $(Q, \Rightarrow)$ is *finite Church-Rosser* (FCR) if for any object $p \in Q$, if $p \overset{*}{\Rightarrow} q_1$ and $p \overset{*}{\Rightarrow} q_2$ and q_1 and q_2 are both irreducible, then $q_1 = q_2$. That is, starting with any p, no matter how we apply transformations, we eventually end up at the same irreducible object.

*Replacement systems also sometimes include an equivalence relation over Q. Equivalence is then used in place of equality in the definition of Finite Church-Rosser and in Theorem 8.8.

Example 8.25 Let **B** be the set of all well-formed Boolean expressions using the symbols 0, 1, (,), $\vee$ or $\wedge$. We assume the expressions are completely parenthesized. The pair $(B, \Rightarrow)$ is a replacement system, where $\Rightarrow$ is the relation summarized in Figure 8.43. We have $T \Rightarrow T'$ whenever T' is T with one of the strings in the left column replaced by the associated string in the right column.

string	replacement
(0)	0
(1)	1
$0 \wedge 0$	0
$0 \wedge 1$	0
$1 \wedge 0$	0
$1 \wedge 1$	1
$0 \vee 0$	0
$0 \vee 1$	1
$1 \vee 0$	1
$1 \vee 1$	1

Figure 8.43

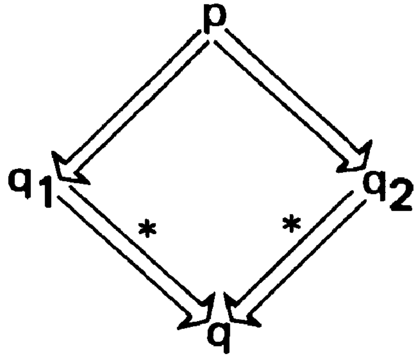
We have, for example,

$$
\begin{aligned}
&(((0 \vee 0) \vee 1) \wedge 0) \Rightarrow \\
&(((0) \vee 1) \wedge 0) \Rightarrow ((0 \vee 1) \wedge 0) \Rightarrow \\
&((1) \wedge 0) \Rightarrow \\
&(1 \wedge 0) \Rightarrow \\
&(0) \Rightarrow 0.
\end{aligned}
$$

The strings 0 and 1 are the only irreducible expressions in **B**. Every expression in **B** goes to exactly one of 0 or 1 under $\overset{*}{\Rightarrow}$, and does so in a finite number of steps. Hence, $(\mathbf{B}, \Rightarrow)$ is FCR.

We shall show that the chase computation for a set of constraints **C** is FCR. That result implies that $chase_{\mathbf{C}}(T)$ always contains exactly one element. To show the chase computation is FCR, we cite the following theorem, which is a special case of a theorem due to Sethi.

Theorem 8.8 (Sethi) A replacement system $(Q, \Rightarrow)$ is FCR if and only if it is finite and, for any object $p \in Q$, if $p \Rightarrow q_1$ and $p \Rightarrow q_2$, then there is a q in Q such that $q_1 \overset{*}{\Rightarrow} q$ and $q_2 \overset{*}{\Rightarrow} q$. Diagramatically, we have



Example 8.26 For the replacement system (**B**, $\Rightarrow$) of Example 8.26,

$((0 \vee 0) \vee (1 \vee 1)) \Rightarrow ((0) \vee (1 \vee 1))$ and
$((0 \vee 0) \vee (1 \vee 1)) \Rightarrow ((0 \vee 0) \vee (1))$.

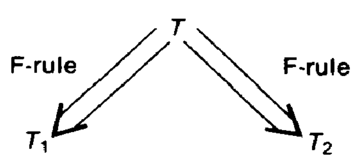
As required by the theorem

$((0) \vee (1 \vee 1)) \overset{*}{\Rightarrow} (0 \vee 1)$ and
$((0 \vee 0) \vee (1)) \overset{*}{\Rightarrow} (0 \vee 1)$.

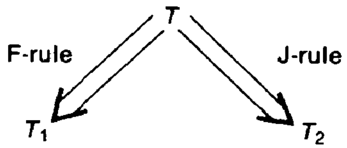
Theorem 8.9 The chase computation for a set of constraints is an FCR replacement system. Therefore, $chase_{\mathbf{C}}(T)$ is always a singleton set.

Proof We use Theorem 8.8. We have already observed that the chase is a finite replacement system. We must show that if we can obtain either tableau T_1 or tableau T_2 from tableau T by a single application of a transformation rule for **C**, then there is some tableau T^* that can be obtained from both T_1 and T_2 by 0 or more applications for the rules for **C**. We treat three cases:

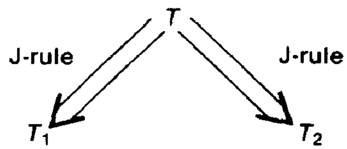
Case 1:



Case 2:



Case 3:



Observe that J-rules leave existing rows in a tableau unchanged, and that an F-rule cannot change one occurrence of a given variable without changing all other occurrences. Let w_1 and w_2 be rows in tableau T, and let u_1 and u_2 be the rows in tableau T' corresponding to w_1 and w_2, where T' can be obtained from T by application of F- and J-rules. By the observation, if $w_1(X) = w_2(X)$, then $u_1(X) = u_2(X)$. Thus, if some F-rule or J-rule is applicable to a set of rows in T, then the same rule applies to the corresponding set of rows in T'. We now treat the cases.

Case 1 Let T_1 be T with variables v_1 and v_2 identified using the F-rule for $X \to A$. Let T_2 be T with variables v_3 and v_4 identified using $Y \to B$.

If $A \neq B$, use the F-rule for $X \to A$ on T_2 to identify v_1 and v_2. The result is T^*, which is T with v_1 and v_2 identified, and v_3 and v_4 identified. T^* can also be obtained from T_1 by using $Y \to B$ to identify v_3 and v_4.

If $A = B$, the argument for $A \neq B$ holds when v_1, v_2, v_3, and v_4 are all distinct. If not, assume v_1 is a distinguished variable, or, if none of v_1, v_2, v_3 or v_4 is distinguished, v_1 is the nondistinguished variable with lowest subscript among the four variables. (We may have to reverse the roles of T_1 and T_2.) Also assume $v_3 = v_1$ or $v_3 = v_2$. We claim T^* is T with v_2, v_3, and v_4 replaced by v_1.

If $v_3 = v_1$, the argument above works again, or $T_1 = T_2 = T^*$, if $v_2 = v_4$. If $v_3 = v_2$, the argument is more involved. In T_1, v_2 ($= v_3$) has been replaced by v_1. Since the F-rule for $Y \to B$ was applied to T to identify v_3 and v_4, the rule for $Y \to B$ also applies to T_1 to replace v_4 by v_1. This replacement yields T^*. In T_2, v_3 replaced v_4; or vice-versa. If v_3 replaced v_4, then the F-rule for

X $\rightarrow A$ can replace v_3 with v_1 in T_2. If v_4 replaced v_3, the F-rule for $X \rightarrow A$ will let v_1 replace v_4 in T_2. In either case, v_2, v_3, and v_4 are replaced by v_1, and the result is T^*.

Example 8.27 Let T be the tableau in Figure 8.44. Applying F-rules for $A \rightarrow B$ and $C \rightarrow B$, we get the tableaux T_1 and T_2, respectively, shown in Figures 8.45 and 8.46. Applying $C \rightarrow B$ to T_1 or $A \rightarrow B$ to T_2 gives tableau T^* in Figure 8.47.

$T(A$	B	$C)$
a_1	b_1	b_2
a_1	b_3	a_3
b_4	a_2	a_3

Figure 8.44

$T_1(A$	B	$C)$
a_1	b_1	b_2
a_1	b_1	a_3
b_4	a_2	a_3

Figure 8.45

$T_2(A$	B	$C)$
a_1	b_1	b_2
a_1	a_2	a_3
b_4	a_2	a_3

Figure 8.46

$T^*(A$	B	$C)$
a_1	a_2	b_2
a_1	a_2	a_3
b_4	a_2	a_3

Figure 8.47

Proof of Theorem 8.9 continued

Case 2 Assume the F-rule replaces variable v_1 by variable v_2 in T to form T_1. Assume the J-rule creates row w to add to T to form T_2. If w has no occurrence of v_1, then apply the J-rule to T_1 to generate w. The application is possible, because the portions of the rows that went into forming w are unchanged from T to T_1. Similarly, applying the F-rule to T_2 replaces v_1 by v_2, since addition of a row cannot bar application of a rule. The result of either rule application is tableau T^*, which is T with variable v_1 replaced by v_2 and row w added.

If row w contains v_1, T^* will be T with v_1 replaced by v_2 and row w' added, where w' is row w with v_1 replaced by v_2. Applying the F-rule used to transform T_1 to T_2 still changes v_1 to v_2, thereby changing row w to w'. The result is T^*. The J-rule used to generate w from T can be applied to the rows in T_1 that correspond to the rows in T to which the rule was originally applied. The resulting row from T_1 will be w' and so the result of the application is T^*. Note that some rows in T_1 may correspond to more than one row in T.

Example 8.28 Let T be the tableau in Figure 8.48. Applying the F-rule for $A \rightarrow B$ yields tableau T_1 in Figure 8.49; applying the J-rule for $*[AB, BC]$ yields tableau T_2 in Figure 8.50. Applying the J-rule to T_1 or the F-rule to T_2 will yield tableau T^*.

$T(A$	B	$C)$
a_1	b_1	b_2
b_3	b_1	a_3
a_1	a_2	a_3

Figure 8.48

$T_1(A$	B	$C)$
a_1	a_2	b_2
b_3	a_2	a_3
a_1	a_2	a_3

Figure 8.49

$T_2(A$	B	$C)$
a_1	b_1	b_2
b_3	b_1	a_3
a_1	a_2	a_3
a_1	b_1	a_3

Figure 8.50

Proof of Theorem 8.9 continued Case 3 is left to the reader (see Exercise 8.30). Since we are able to find an appropriate T^* in all three cases, the chase computation is FCR.

Since $chase_{\mathbf{C}}(T)$ is always a singleton set, we modify our notation to let $chase_{\mathbf{C}}(T)$ represent its only element.

Corollary If $SAT(\mathbf{C}) = SAT(\mathbf{C}')$, then $chase_{\mathbf{C}}(T) = chase_{\mathbf{C}'}(T)$ for any tableau T.

Proof We prove here the special case where $\mathbf{C}' = \mathbf{C} \cup \{c\}$ for any c such that $\mathbf{C} \models c$. Let $T^* = chase_{\mathbf{C}}(T)$. The same applications of rules will take us from T to T^* under $\mathbf{C}'$, since $\mathbf{C}' \supseteq \mathbf{C}$. Furthermore, Theorem 8.7 shows us that we cannot apply any rules for $\mathbf{C}'$ to T^*, because T^* as a relation is in $SAT(\mathbf{C})$ and hence in $SAT(\mathbf{C}')$. We see $chase_{\mathbf{C}'}(T) = T^*$.

The proof of the general version of the corollary is left to the reader (see Exercise 8.31). If $\mathbf{C}$ and $\mathbf{C}'$ are arbitrary equivalent sets of constraints, then $\mathbf{C} \models c'$ for any constraint $c' \in \mathbf{C}$. Likewise, for any c in $\mathbf{C}$, $\mathbf{C}' \models c$. If $\mathbf{C}'' = \mathbf{C} \cup \mathbf{C}'$, then $SAT(\mathbf{C}'') = SAT(\mathbf{C}) = SAT(\mathbf{C}')$. It can be shown, using the special case, that $chase_{\mathbf{C}}(T) = chase_{\mathbf{C}''}(T) = chase_{\mathbf{C}'}(T)$.

8.6.2 Equivalence of Tableaux under Constraints

We can now test equivalence of tableaux under constraints, which gives us a test for cases when a project-join mapping $m_{\mathbf{R}}$ is lossless on $SAT(\mathbf{C})$. By the remarks at the beginning of this section, we know $T \equiv_{\mathbf{C}} chase_{\mathbf{C}}(T)$. Theorem 8.7 tells us $chase_{\mathbf{C}}(T)$, as a relation, is in $SAT(\mathbf{C})$. Using Lemma 8.3, we have the following results.

Theorem 8.10 Let T_1 and T_2 be tableaux, and let $\mathbf{C}$ be a set of constraints. $T_1 \sqsubseteq_{\mathbf{C}} T_2$ if and only if $chase_{\mathbf{C}}(T_1) \sqsubseteq chase_{\mathbf{C}}(T_2)$.

Corollary $T_1 \equiv_{\mathbf{C}} T_2$ if and only if $chase_{\mathbf{C}}(T_1) \equiv chase_{\mathbf{C}}(T_2)$.

Example 8.29 Consider tableaux T_1 and T_2 in Figures 8.51 and 8.52. T_1 is the tableau for the set of schemes $\{AB, BC, AD\}$. T_2 is the tableau for the set $\{AB, BC, CD\}$. Let $\mathbf{C} = \{A \rightarrow D, *[AB, BCD]\}$. Figures 8.53 and 8.54 show $T_1^* = chase_{\mathbf{C}}(T_1)$ and $T_2^* = chase_{\mathbf{C}}(T_2)$.

T_1(A	B	C	D)
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
a_1	b_5	b_6	a_4

Figure 8.51

T_2(A	B	C	D)
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_4
b_5	b_6	a_3	a_4

Figure 8.52

T_1^*(A	B	C	D)
a_1	a_2	b_1	a_4
b_3	a_2	a_3	a_4
a_1	b_5	b_6	a_4
a_1	a_2	a_3	a_4
b_3	a_2	b_1	a_4

Figure 8.53

T_2^*(A	B	C	D)
a_1	a_2	b_1	b_2
b_3	a_2	a_3	b_2
b_5	b_6	a_3	a_4
a_1	a_2	a_3	b_2
b_3	a_2	b_1	b_2

Figure 8.54

Since T_1^* contains the row of all distinguished variables, it is not hard to find a containment mapping from T_2^* to T_1^*. Hence $T_2^* \sqsubseteq T_1^*$ and therefore $T_2 \sqsubseteq_{\mathbf{C}} T_1$.

8.6.3 Testing Implication of Join Dependencies

We desire a means to test when all the relations in $SAT(\mathbf{C})$ can be faithfully represented by their projections onto the relation schemes in some database scheme **R**. This condition, equivalently stated as $\mathbf{C} \models *[\mathbf{R}]$ or $m_{\mathbf{R}}$, is the identity mapping on $SAT(\mathbf{C})$. In terms of tableau equivalence, $T_{\mathbf{R}} \equiv_{\mathbf{C}} T_I$ where

T_I is the tableau consisting only of w_d, the row of all distinguished variables. T_I is the identity mapping on all relations. By Theorem 8.10, we can test the equivalence above by checking if $chase_{\mathbf{C}}(T_{\mathbf{R}}) \equiv chase_{\mathbf{C}}(T_I)$. $Chase_{\mathbf{C}}(T_I) = T_I$ (why?), so we are checking whether $chase_{\mathbf{C}}(T_{\mathbf{R}}) \equiv T_I$. The test for that condition is simply whether or not $chase_{\mathbf{C}}(T_{\mathbf{R}})$ contains w_d (see Exercise 8.33).

Example 8.30 T_1 in Figure 8.51 is the tableau for the database scheme $\mathbf{R} = \{AB, BC, AD\}$. Let $\mathbf{C} = \{A \rightarrow D, *[AB, BCD]\}$. Since $chase_{\mathbf{C}}(T_1)$, given in Figure 8.53, contains w_d, any relation in $SAT(\mathbf{C})$ decomposes losslessly onto $\mathbf{R}$. T_2 in Figure 8.52 is the tableau for database scheme $\mathbf{S} = \{AB, BC, CD\}$. Since $chase_{\mathbf{C}}(T_2)$, given in Figure 8.54, does not contain w_d, there are some relations in $SAT(\mathbf{C})$ that have lossy decompositions onto $\mathbf{S}$.

Example 8.31 As promised in Section 6.5.4, we shall now show that if $\mathbf{R}$ is a database scheme over $\mathbf{U}$ that completely characterizes a set F of FDs and some scheme $R \in \mathbf{R}$ is a universal key for $\mathbf{U}$, then any relation in $SAT(F)$ decomposes losslessly onto $\mathbf{R}$.

Let G be the set of FDs expressed by the keys of the relation schemes in R. We know $G \models R \rightarrow \mathbf{U}$. Let $T^*_{\mathbf{R}} = chase_G(\mathrm{T}_{\mathbf{R}})$. Let w be the row for R in $T_{\mathbf{R}}$ and let w^* be the corresponding row in $T^*_{\mathbf{R}}$. We claim w^* is the row of all distinguished variables.

Let H be a G-based DDAG for $R \rightarrow \mathbf{U}$. There is a computation for $chase_G(\mathrm{T}_{\mathbf{R}})$ that mimics the construction of H. The correspondence will be that if Y is the set of node labels at some point in the construction of H, then the row corresponding to w in some tableau in the generating sequence for $T^*_{\mathbf{R}}$ has distinguished variables in all the Y-columns. More formally, let $H_0, H_1, \ldots, H_n = H$ be the successive DDAGs in the construction of H. We shall describe a generating sequence $T_{\mathbf{R}} = T_0, T_1, \ldots, T_n$ for $T^*_{\mathbf{R}}$.

Let w_i be the row in T_i corresponding to w in T_0. If Y_i is the set of node labels for DDAG H_i, we want w_i to have distinguished variables in all the Y_i-columns. Initially, the desired relationship holds. Y_0 is just R, and $w_0 = w$ is the row for R in T_R. Suppose the relationship holds for H_i and T_i. Suppose also that H_{i+1} is derived from H_i by adding a node labeled A, using the FD $K \rightarrow A$ from G. K must be a key for some relation scheme R_j in $\mathbf{R}$, where $A \in R_j$. There is a row u for R_j in T_0. Let u_i be the corresponding row in T_i. Row u_i has distinguished variables in the R_j-columns at least (see Lemma 8.4, to follow).

Since $K \rightarrow A$ was used to extend H_i, $K \subseteq Y_i$, hence w_i is distinguished in all the K-columns. Since $K \subseteq R_j$, u_i is distinguished in the K-columns. The F-rule for $K \rightarrow A$ is applicable to T_i on rows w_i and u_i, because $w_i(K) = u_i(K)$. Applying the F-rule sets $w_i(A) = u_i(A)$, which means that $w_i(A)$ is made distinguished, if it is not already. Hence in T_{i+1}, w_{i+1} is distinguished at least on $Y_iA = Y_{i+1}$.

As the result of our induction, we see that w_n in T_n is the row of all distinguished variables, since $H_n = H$ has all the attributes in **U** as node labels. One minor detail remains. There may be more rules for G that can be applied to T_n. Let the chase computation continue until it terminates: T_{n+1}, T_{n+2}, ..., $T^*_{\mathbf{R}}$. The row w^* in $T^*_{\mathbf{R}}$ corresponding to w_n in T_n is still all distinguished.

We see that $chase_G(T_{\mathbf{R}})$ contains the row of all distinguished variables, so $G \models *[R]$. Thus, any relation r in $SAT(G) = SAT(F)$ decomposes losslessly onto **R**.

8.6.4 Testing Implication of Functional Dependencies

We have a test for implication of JDs by a set **C** of FDs and JDs. We now turn to a test for implication of FDs by **C**. To test implication of JDs, we interpreted tableaux as mappings from relations to relations. For the FD test, we shall view tableaux as relations, or, more accurately, templates for relations. Before presenting the test, we need two lemmas.

Lemma 8.4 Let T be a tableau and let **C** be a set of constraints. Let ρ be a valuation for T such that $\rho(T) \subseteq r$, where r is chosen from $SAT(\mathbf{C})$. If $T = T_0, T_1, T_2, \ldots, T_n$ is a generating sequence for $chase_{\mathbf{C}}(T)$, then for $0 \le i \le n$,

1. $\rho(w_0) = \rho(w_i)$, where w_0 is any row in T_0 and w_i is the corresponding row in T_i. Also, w_i subsumes w_0.
2. $\rho(T_i) \subseteq r$.
3. $T_i \sqsupseteq T_{i+1}$ $i \neq n$.

Proof Parts 1 and 2. It suffices to say that if w_j is a row in T_j and w_{j+1} is the corresponding row of T_{j+1} then

$\rho(w_j) = \rho(w_{j+1})$ and
w_{j+1} subsumes w_j;

and if w is a row in T_{j+1} that corresponds to no row in T_j, then

$\rho(w) \in r$.

If T_{j+1} is obtained by an F-rule that changes no variable in w_j, or a J-rule, then $w_j = w_{j+1}$ and obviously $\rho(w_j) = \rho(w_{j+1})$ and w_{j+1} subsumes w_j. Otherwise, in going from T_j to T_{j+1}, for some attribute A, $w_j(A)$ changes from v_1 to v_2.

The change must be through the applications of an F-rule for an FD $X \to A$ to two rows u_1 and u_2 in T_j, where $u_1(X) = u_2(X)$, $u_1(A) = v_1$ and $u_2(A) = v_2$. By induction $\rho(u_1) = t_1$ and $\rho(u_2) = t_2$, where t_1 and t_2 are tuples in r. We must have $t_1(X) = t_2(X)$. Since r is in SAT(**C**), $t_1(A) = t_2(A)$. Now $\rho(v_1) = \rho(u_1(A)) = t_1(A) = t_2(A) = \rho(u_2(A)) = \rho(v_2)$. Hence $\rho(w_j) = \rho(w_{j+1})$. Also, if one of v_1 or v_2 is distinguished, it must be v_2, so w_{j+1} subsumes w_j.

If w is a row in T_{j+1} that corresponds to no row in T_j, then w must be the result of joining rows $u_1, u_2, \ldots, u_q$ of T_j on **S**, where *[**S**] $\in$ **C**. By Exercise 8.25, $\rho(u_1), \rho(u_2), \ldots, \rho(u_q)$, which are all in r, are joinable on **S** with result $\rho(w)$. Since $r \in SAT(\mathbf{C})$, $\rho(w) \in r$.

The proof of part 3 is left to the reader (see Exercise 8.36).

Suppose we have a non-trivial FD $X \to A$, and we want to test whether $\mathbf{C} \models X \to A$. We construct a tableau T_X as follows. T_X has two rows, w_d and w_X. Row w_d is all distinguished symbols; w_X has distinguished symbols in the X-columns and distinct nondistinguished symbols elsewhere. That is, $T_X = \mathrm{T}_\mathbf{R}$ for $\mathbf{R} = \{\mathbf{U}, X\}$.

Example 8.32 Figure 8.55 shows T_{BC} for $\mathbf{U} = ABCD$.

$T_{BC}(A$	B	C	$D)$
a_1	a_2	a_3	a_4
b_1	a_2	a_3	b_2

Figure 8.55

Theorem 8.11 $\mathbf{C} \models X \to A$ if and only if $chase_\mathbf{C}(T_X)$ has only distinguished variables in the A-column.

Proof Let $T^* = chase_\mathbf{C}(T_X)$. Suppose T^* has a nondistinguished symbol in the A-column. T^* considered as a relation is a counterexample to $\mathbf{C} \models X \to A$. By Theorem 8.7, T^* satisfies **C**. However, every row of T^* has all distinguished symbols in the X-columns, since chase computation does not create new symbols. Row w_d remains unchanged throughout the chase, by Lemma 8.4. Thus T^* has two rows that agree on X but disagree on A: w_d and the row with a nondistinguished symbol in the A-column. Hence, T^* violates $X \to A$.

Suppose now that T^* has only a distinguished variable in the A-column, and let r be an arbitrary relation in SAT(**C**). Let t_1 and t_2 be any pair of tuples in r with $t_1(X) = t_2(X)$. Consider the valuation ρ for T_X such that

$\rho(w_d) = t_1$ and $\rho(w_X) = t_2$. Such a valuation exists, because $w_d(X) = w_X(X)$. We just saw that w_d is the row in T^* corresponding to w_d in T_X. Let w_X^* be the row corresponding to w_X. By Lemma 8.4, $\rho(w_X^*) = \rho(w_X)$. Since T^* has only one variable in the A-column, $w_X^*(A) = w_d(A)$. Thus we see

$$t_1(A) = \rho(w_d(A)) = \rho(w_X^*(A)) = \rho(w_X(A)) = t_2(A).$$

Any two tuples in r that agree on X also agree on A. Since r was arbitrary, $SAT(\mathbf{C}) \subseteq SAT(X \to A)$ or $\mathbf{C} \models X \to A$.

Example 8.33 Suppose we wish to test $\mathbf{C} \models BC \to D$. If $\mathbf{C} = \{A \to D\}$, then $chase_{\mathbf{C}}(T_{BC}) = T_{BC}$. There is a b_2 in the D-column, so $BC \to D$ is not implied by $\mathbf{C}$. If $\mathbf{C}' = \{A \to D, *[ABC, CD]\}$, then $chase_{\mathbf{C}'}(T_{BC})$ is the tableau T^* in Figure 8.56. T^* has only a_4 in the D-column, so $\mathbf{C}' \models BC \to D$.

$T^*(A$	B	C	$D)$
a_1	a_2	a_3	a_4
b_1	a_2	a_3	a_4

Figure 8.56

We originally defined X^+ as the closure of a set of attributes X with respect to a set of FDs F. We can extend the definition consistently to include JDs as well as FDs.

Definition 8.16 Let $\mathbf{C}$ be a set of FDs and JDs and let X be a set of attributes. The *closure of* X with respect to $\mathbf{C}$, denoted X^+, is the largest set of attributes Y such that $\mathbf{C} \models X \to Y$. Note that if $\mathbf{C}$ is only FDs, the new definition reduces to the old definition.

Corollary For a given $\mathbf{C}$, X^+ is the set of all attributes A such that the A-columns of $chase_{\mathbf{C}}(T_X)$ has only distinguished variables.

Corollary If J is a set of JDs, then $J \models X \to Y$ implies $X \supseteq Y$. That is, a set of JDs implies only trivial FDs.

Proof $Chase_J(T_X)$ will have a nondistinguished variable in every column corresponding to an attribute in $\mathbf{U} - X$, since J-rules do not identify symbols.

8.6.5 Computing a Dependency Basis

Since MVDs are a special case of JDs, we can always test $\mathbf{C} \models X \twoheadrightarrow Y$ by testing $\mathbf{C} \models *[XY, XZ]$, where $Z = \mathbf{U} - XY$. However, the next theorem shows an alternate way to use the chase to find *all* sets Y such that $C \models X \twoheadrightarrow Y$, for a given X.

Theorem 8.12 Let $\mathbf{C}$ be a set of constraints, and let Y be a set of attributes disjoint from X^+ under $\mathbf{C}$. $\mathbf{C} \models X \twoheadrightarrow Y$ if and only if $chase_{\mathbf{C}}(T_X)$ contains a row u_Y with distinguished variables exactly in all the YX^+-columns.

Proof (if) Let T_X^* be $chase_{\mathbf{C}}(T_X)$. Let u_d and u_X be the rows in T_X^* corresponding to w_d and w_X. (We know $w_d = u_d$.) Let $\mathbf{R}$ be the database scheme $\{XY, XZ\}$ where $Z = \mathbf{U} - YX^+$. We shall show that $T_{\mathbf{R}}^* = chase_{\mathbf{C}}(T_{\mathbf{R}})$ must contain w_d, hence $\mathbf{C} \models *[XY, XZ]$, which is equivalent to $\mathbf{C} \models X \twoheadrightarrow Y$.

Let p_{XY} and p_{XZ} be the rows in $T_{\mathbf{R}}$ for relation schemes XY and XZ. Let q_{XY} and q_{XZ} be the corresponding rows in $T_{\mathbf{R}}^*$. Consider a mapping δ from variables in T_X to variables in $T_{\mathbf{R}}^*$ such that $\delta(w_d) = q_{XY}$ and $\delta(w_X) = q_{XZ}$. The mapping δ can be viewed as a valuation if $T_{\mathbf{R}}^*$ is considered as a relation; δ exists because $p_{XY}(X) = p_{XZ}(X)$, so $q_{XY}(X) = q_{XZ}(X)$. Since $T_{\mathbf{R}}^*$ as a relation is in $SAT(\mathbf{C})$, by Lemma 8.4, $\delta(T_X^*) \subseteq T_{\mathbf{R}}^*$, $\delta(w_d) = \delta(u_d)$, and $\delta(w_X) = \delta(u_X)$. Since $u_d(X^+) = u_X(X^+)$, $q_{XY}(X^+) = q_{XZ}(X^+)$. We see that δ maps distinguished variables in the X^+-columns of T_X^* to distinguished variables in the X^+ columns of $T_{\mathbf{R}}^*$.

We shall show that for row u_Y of T_X^*, with distinguished symbols in exactly the YX^+-columns, $\delta(u_Y)$ is the row w_d of all distinguished symbols in T_R^*. Since u_Y is distinguished in the X^+-columns, $\delta(u_Y)$ is distinguished in the X^+-columns by the argument in the previous paragraph. Since q_{XY} subsumes p_{XY}, q_{XY} is distinguished in all the Y-columns. We know $\delta(w_d) = q_{XY}$, so δ must map distinguished variables in the Y-columns of T_X^* to distinguished variables in the Y-columns of $T_{\mathbf{R}}^*$. Row u_Y is distinguished in all the Y-columns, so $\delta(u_Y)$ is distinguished in all the Y-columns. Since q_{XZ} subsumes p_{XZ}, q_{XZ} is distinguished in all the Z-columns. We know $\delta(w_X) = q_{XZ}$, so δ must map nondistinguished variables in the Z-columns of T_X^* to distinguished variables in the Z-columns of $T_{\mathbf{R}}^*$. Row u_Y is nondistinguished in all the Z-columns, so $\delta(u_Y)$ is distinguished in the Z-columns. $\mathbf{U} = YX^+Z$, so $\delta(u_Y)$ is distinguished everywhere. Therefore $T_{\mathbf{R}}^*$ contains w_d, so $\mathbf{C} \models X \twoheadrightarrow Y$.

(only if) Assume $\mathbf{C} \models X \twoheadrightarrow Y$. Let $\mathbf{C}' = \mathbf{C} \cup \{*[XY, XZ]\}$, where $Z = \mathbf{U} - YX^+$, as before. By the corollary to Theorem 8.9, $chase_{\mathbf{C}}(T_X) = chase_{\mathbf{C}'}(T_X)$ because $SAT(\mathbf{C}) = SAT(\mathbf{C}')$. Consider the computation for

$chase_{C'}(T_X)$ where the first step is to apply the J-rule for $*[XY, XZ]$ to rows w_d and w_X. The result is a row w that is distinguished exactly in the XY-columns. During the remainder of the chase computation, any nondistinguished variables in the X^+-columns of w will be made distinguished. Thus the row in $chase_{C'}(T_X)$ corresponding to w will have distinguished variables in exactly the YX^+-columns. (Why are there no distinguished variables elsewhere?) $Chase_C(T_X) = chase_{C'}(TX)$, so we are done.

Example 8.34 Let $\mathbf{C} = \{B \rightarrow C, *[ABC, CDE]\}$. Tableau T_B is given in Figure 8.57 and $T_B^* = chase_C(T_B)$ is given in Figure 8.58. We see that $B^+ = BC$ and that $\mathbf{C}$ implies the MVDs $B \twoheadrightarrow ADE$, $B \twoheadrightarrow \emptyset$, $B \twoheadrightarrow A$ and $B \twoheadrightarrow DE$.

T_B(A	B	C	D	E)
a_1	a_2	a_3	a_4	a_5
b_1	a_2	b_2	b_3	b_4

Figure 8.57

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

Figure 8.58

From $chase_C(T_X)$, then, we can determine the set $Q = \{Y \mid \mathbf{C} \models X \twoheadrightarrow Y$ and $X^+ \cap Y = \emptyset\}$. Referring to Section 7.4.2, by replication, $\mathbf{C} \models X \twoheadrightarrow A$ for any $Z \in X^+$. Exercise 8.38 will show that $\mathbf{C} \models X \twoheadrightarrow Y$ if and only if Y can be written as $X'Y'$, where $X' \subseteq X^+$ and $Y' \in Q$. We can extend our definition of dependency basis to include JDs.

Definition 8.17 Let $\mathbf{C}$ be a set of constraints and let X be a set of attributes. The *dependency basis of* X with respect to $\mathbf{C}$, denoted $DEP(X)$, is $mdsb(\{Y \mid \mathbf{C} \models X \twoheadrightarrow Y\})$. (Recall that $mdsb$ is minimum disjoint set basis—see Section 7.4.3.)

As before, $\mathbf{C} \models X \twoheadrightarrow Y$ if and only if Y is the exact union of sets in $DEP(X)$. $DEP(X)$ can be calculated directly from Q and X^+ as $mdsb(Q) \cup \{\{A\} \mid A \in X^+\}$.

Example 8.35 Let $\mathbf{C} = \{B \rightarrow C, *[ABC, CDE]\}$, as in Example 8.34. We saw in that example that $B^+ = BC$ and $Q = \{ADE, \emptyset, A, DE\}$. We can calculate $DEP(B) = \{A, B, C, DE\}$.

8.7 TABLEAUX AS TEMPLATES

In this section we shall formalize the idea of a tableau as a template for relations.

Definition 8.18 Let $\mathbf{P}$ be a set of relations, and let r be any relation. A *completion of* r *under* $\mathbf{P}$ is a relation s in $\mathbf{P}$ such that $r \subseteq s$ and there is no relation s' in $\mathbf{P}$ such that $r \subseteq s' \subsetneq s$. $COMP_{\mathbf{P}}(r)$ is the set of all such completions; $COMP_{\mathbf{C}}(r)$ is shorthand for $COMP_{SAT(\mathbf{C})}(r)$.

Completions do not always exist.

Example 8.36 Let r be the relation in Figure 8.59. If $F = \{A \rightarrow C\}$, then $COMP_F(r)$ is empty. If $J = \{*[AB, BCD]\}$, then $COMP_J(r) = \{s\}$, where s is the relation in Figure 8.60.

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

Figure 8.59

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

Figure 8.60

Completions are not unique, given they exist.

Example 8.37 Let r be the relation of Figure 8.59. Let $\mathbf{P} = SAT(*[AB, BC])$. The dependency $*[AB,BC]$ is an embedded JD for the given relation scheme. $COMP_{\mathbf{P}}(r)$ contains relation s in Figure 8.60, and also the relation q in Figure 8.61. In fact, $COMP_{\mathbf{P}}(r)$ contains one relation for every value in the domain of attribute D.

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

Figure 8.61

A set **P** of relations is closed under intersection if for every pair of relations r and s in **P**, $r \cap s$ is in **P**.

Lemma 8.5 **P** is closed under intersection if and only if completions under **P** are unique.

Proof Suppose **P** is closed under intersection. Let s and s' be completions of r under **P**. By closure, $s \cap s'$ is in **P**, and $s \cap s' \supseteq r$, so $s = s \cap s' = s'$. For the converse, suppose completions under **P** are unique. Let r and s be in **P**, and let $q = r \cap s$. There must be some subset r' of r (perhaps r itself) such that r' is a completion of q under **P**. Likewise, there is a subset s' of s that is a completion of q. By uniqueness of completion $r' = s'$, so $r' = q = s'$ and q is in **P**.

Corollary If **C** is a set of FDs and JDs, then completions under $SAT(\mathbf{C})$ are unique.

Proof Left to the reader (see Exercise 8.40).

Completions always exist for a set J of JDs only. Completions can be found in a manner similar to the chase computation. However, if **C** contains both FDs and JDs, completions do not always exist, even for relations that satisfy the FDs (see Exercise 8.41). For a set of FDs F, $COMP_F(r)$ exists exactly when $r \in SAT(F)$. In that case, $COMP_F(r) = r$. (We use $COMP_P(r)$ to stand for its only member when P is closed under intersection.)

We now give the set of relations a tableau represents.

Definition 8.19 Let T be a tableau and let **P** be a set of relations. The *representation set of* T *under* **P**, denoted $REP_{\mathbf{P}}(T)$, is

$$\{r \mid r \in COMP_{\mathbf{P}}(\rho(T)) \text{ for some valuation } \rho\}.$$

As usual, $REP_{\mathbf{C}}(T)$ stands for $REP_{SAT(\mathbf{C})}(T)$.

Lemma 8.6 Let **P** be a set of relations closed under intersection and let T_1 and T_2 be tableaux. If $T_1 \sqsubseteq_{\mathbf{P}} T_2$, then for every relation r in $REP_P(T_1)$, there is a relation s in $REP_{\mathbf{P}}(T_2)$ such that $s \subseteq r$.

Proof Let $r \in REP_{\mathbf{P}}(T_1)$, where r is $COMP_{\mathbf{P}}(\rho_1(T_1))$, and let w_d be the row of all distinguished variables. $T_1(r)$ contains $\rho_1(w_d)$, since $r \supseteq \rho_1(T_1)$. Since $T_1 \sqsubseteq_{\mathbf{P}} T_2$, $\rho_1(w_d) \in T_2(r)$. There must be a valuation ρ_2 such that $\rho_2(w_d) = \rho_1(w_d)$ and $\rho_2(T_2) \subseteq r$. Let $s = COMP_{\mathbf{P}}(\rho_2(T_2))$. Relation s exists because $\rho_2(T_2) \subseteq r \in \mathbf{P}$. It follows that $s \subseteq r$.

Example 8.38 Lemma 8.6 is quite weak when $\mathbf{P} = SAT(\mathbf{C})$, for **C** a set of FDs and JDs. No matter what **C** is, $SAT(\mathbf{C})$ contains all relations consisting of a single tuple. Suppose we have $T_1 \sqsubseteq_{\mathbf{C}} T_2$ and $r \in REP_{\mathbf{C}}(T_1)$. Let t be a tuple in r, let s be the relation consisting only of t, and let ρ be the valuation such that $\rho(T_2) = s$. Since $COMP_{\mathbf{C}}(s) = s$, $s \in REP_{\mathbf{C}}(T_2)$ and clearly $s \subseteq r$.

However, when $\mathbf{P} = SAT(\mathbf{C})$, we can prove a fairly strong result.

Theorem 8.13 Let **C** be a set of constraints and let T be a tableau. If $T^* = chase_{\mathbf{C}}(T)$, then $REP_{\mathbf{C}}(T) = REP_{\mathbf{C}}(T^*)$.

Proof Suppose $r \in REP_{\mathbf{C}}(T)$. Let ρ be the valuation such that $r = COMP_{\mathbf{C}}(\rho(T))$. Clearly, $\rho(T) \subseteq r$. Since $r \in SAT(\mathbf{C})$, from Lemma 8.4 we have

1. $\rho(T) \subseteq \rho(T^*)$ and
2. $\rho(T^*) \subseteq r$.

We see $COMP_{\mathbf{C}}(\rho(T^*)) = r$, so $REP_{\mathbf{C}}(T) \subseteq REP_{\mathbf{C}}(T^*)$.

Now suppose $r \in REP_{\mathbf{C}}(T^*)$. Let ρ be a valuation such that $r = COMP_{\mathbf{C}}(\rho(T^*))$. Since T^* as a relation is in $SAT(C)$, $\rho(T^*) \in SAT(\mathbf{C})$, so $r = \rho(T^*)$. T may have more variables than T^*, but ρ can be consistently extended to T in such a way that $\rho(T) \subseteq \rho(T^*)$. Let w be any row in T, and let w^* be the corresponding row in T^*. Set $\rho(w) = \rho(w^*)$. Let $T = T_0, T_1, T_2, \ldots, T_n = T^*$ be a generating sequence for T^*. By Lemma 8.4, we know that

$$\rho(T_1) \subseteq \rho(T_2) \subseteq \cdots \subseteq \rho(T_n).$$

Since $SAT(C)$ has the intersection property,

$$COMP_{\mathbf{C}}(\rho(T_1)) \subseteq COMP_{\mathbf{C}}(\rho(T_2)) \subseteq \cdots \subseteq COMP(\rho(T_n)).$$

(Here $COMP_{\mathbf{C}}(s)$ stands for a relation.) Suppose one of the containments is proper:

$$COMP_{\mathbf{C}}(\rho(T_i)) \subsetneq COMP_{\mathbf{C}}((T_{i+1})).$$

There must be a tuple $\rho(w)$ in $\rho(T_{i+1})$ that is not in $COMP_{\mathbf{C}}(\rho(T_i))$, otherwise $\rho(T_{i+1}) \subseteq COMP_{\mathbf{C}}(\rho(T_i))$ and the two completions are equal. Therefore, $w \in T_{i+1}$, $w \notin T_i$. Row w must have been generated by a J-rule from rows in T_i, say rows $w_1, w_2, \ldots, w_q$ and the J-rule for *[S]. Now $\rho(w_1)$, $\rho(w_2), \ldots, \rho(w_q)$ are in $\rho(T_i)$, hence in $COMP_{\mathbf{C}}(\rho(T_i))$. But $COMP_{\mathbf{C}}(\rho(T_i)) \in SAT(\mathbf{C})$ and hence must satisfy *[S], so $\rho(w)$ is in $COMP_{\mathbf{C}}(\rho(T_i))$, a contradiction. None of the containments are proper, so $COMP_{\mathbf{C}}(\rho(T)) = COMP_{\mathbf{C}}(\rho(T^*)) = r$.

We see that $REP_{\mathbf{C}}(T) \subseteq REP_{\mathbf{C}}(T^*)$, and so $REP_{\mathbf{C}}(T) = REP_{\mathbf{C}}(T^*)$.

Corollary For a set of constraints $\mathbf{C}$ and tableau T,

$$REP_{\mathbf{C}}(T) = \{\rho(T^*) \mid T^* = chase_{\mathbf{C}}(T) \text{ and } \rho \text{ is a valuation}\}.$$

Proof $REP_{\mathbf{C}}(T) = REP_{\mathbf{C}}(T^*) = \{COMP_{\mathbf{C}}(\rho(T^*)) \mid \rho \text{ is a valuation}\}$. As we saw in the proof of the theorem, $COMP_{\mathbf{C}}(\rho(T^*)) = \rho(T^*)$.

In light of the last theorem, we might expect some connection between the conditions $T_1 \equiv_{\mathbf{C}} T_2$ and $REP_{\mathbf{C}}(T_1) = REP_{\mathbf{C}}(T_2)$. However, the first does not imply the second (Exercise 8.42), nor does the second imply the first, as the next example shows.

Example 8.39 Let T_1 and T_2 be the tableaux in Figures 8.62 and 8.63. Let $\mathbf{C} = \{A \rightarrow B\}$. Both the tableaux, as relations, are in $SAT(\mathbf{C})$, hence they are their own chases under $\mathbf{C}$. There is no containment mapping from T_1 to T_2, so $T_1 \not\equiv_{\mathbf{C}} T_2$. However, we see that for any valuation ρ_1 for T_1 there is a valuation ρ_2 for T_2 such that $\rho_1(T_1) = \rho_2(T_2)$, and vice-versa. By the corollary to Theorem 8.13, $REP_{\mathbf{C}}(T_1) = REP_{\mathbf{C}}(T_2)$.

T_1(A	B	C)
a_1	a_2	a_3
a_1	a_2	b_2
b_3	b_4	a_3

Figure 8.62

$T_2(A$	B	$C)$
a_1	b_1	a_3
a_1	b_1	b_2
b_3	a_2	a_3

Figure 8.63

8.8 COMPUTATIONAL PROPERTIES OF THE CHASE COMPUTATION

In general, the chase computation has exponential time complexity. If tableau T has k columns and m rows, $chase_{\mathbf{C}}(T)$ can have m^k rows (see Exercise 8.44). If we are using the chase computation to test for a lossless join, we need not always compute the entire chase. As soon as w_d, the all-distinguished row, is encountered, there is no need to continue. If w_d occurs in any tableau in a generating sequence, it will appear in the final tableau in the sequence. However, the problem of determining whether $w_d \in chase_{\mathbf{C}}(T)$ probably does not have a polynomial-time solution, because the problem of testing $\mathbf{C} \models *[\mathbf{S}]$ is known to be NP-hard. There are methods, other than the chase, that can be used to test $\mathbf{C} \models c$ in polynomial time, where c is an FD or MVD.

$Chase_F(T)$, for a set F of FDs, never has more rows that T, since F-rules do not create new rows. It is not suprising, then, that $chase_F(T)$ can be computed in polynomial time. We assume that the input to the problem is the tableau T and the set F. For simplicity, assume that one attribute or one tableau variable takes one unit of space to express. Let

$k = |\mathbf{U}| =$ the number of the columns in T.
$m =$ the number of rows in T, and
$p =$ the amount of space to express F.

The size of our input is

$$n = O(k \cdot m + p).$$

We now indicate how to compute $chase_{\mathbf{C}}(T)$ in $O(n^3)$ time. We shall make repeated passes through the set of FDs. For each FD $X \rightarrow A$, we do a bucket sort on the rows of the tableau to bring rows with equal X-components together. If $|X| = q$, the sort takes $O(q \cdot m)$ time. Once the rows are sorted, in $O(q \cdot m)$ time again, we can find rows with equal X-components and make them identical in their A-columns. Over all the FDs in F, the sum of the sizes of their left sides is no more than p. Thus, one pass through all the FDs takes $O(p \cdot m)$ time.

We continue to make passes through F until we make a pass where no changes occur. At that point, we are done. T can have at most $k \cdot m$ distinct variables to begin with. Every pass except the last decreases the number of variables by one, so we make $O(k \cdot m)$ passes at most. The total time spent on the chase is $O(k \cdot p \cdot m^2)$, which is no more than $O(n^3)$.

If the tableau corresponds to a database scheme, and only the relation schemes are given as input, the procedure above requires $O(n^4)$ time, where n is the size of the input (see Exercise 8.45). Other methods for computing the chase exist that can bring the time complexity down to $O(n^2/\log n)$.

Up to this point we have assumed all our FDs have single attributes on their right sides, in order to make the F-rule simple to state. The F-rule can be generalized to handle multiple attributes on the right side of an FD. If w_1 and w_2 are rows in a tableau such that $w_1(X) = w_2(X)$, and $X \rightarrow Y$ is an FD in the set of constraints, we can identify $w_1(A)$ and $w_2(A)$ for each attribute A in Y.

There is also an extension of the J-rule that allows us to generate more than one row at a time. If *[S] is a JD in the set of constraints, we may apply the project-join mapping m_S to a tableau and use the result as the next tableau in the generating sequence.

Example 8.40 Suppose T_1 in Figure 8.64 is a tableau in a generating sequence for $chase_C(T)$, where C contains $*[AB, BC, CD]$. T_2 in Figure 8.65 can be the next tableau in the generating sequence.

T_1(A	B	C	D)
b_1	a_2	b_2	a_4
a_1	a_2	a_3	b_3
b_4	b_5	a_3	a_4

Figure 8.64

The astute reader may be wondering if the subscripts on nondistinguished variables can be dispensed with and these variables could be considered distinct until identified with a distinguished variable. The next example shows a tableau where nondistinguished variables must be equated to perform the chase.

T_2

A	B	C	D
b_1	a_2	b_2	a_4
b_1	a_2	a_3	b_3
b_1	a_2	a_3	a_4
a_1	a_2	b_2	a_4
a_1	a_2	a_3	b_3
a_1	a_2	a_3	a_4
b_4	b_5	a_3	b_3
b_4	b_5	a_3	a_4

Figure 8.65

Example 8.41 Let T be the tableau in Figure 8.66 and let $\mathbf{C} = \{A \to C, B \to C, CD \to E\}$. In order to compute $chase_{\mathbf{C}}(T)$, we must be able to identify b_2, b_4 and b_8.

T

A	B	C	D	E
a_1	b_1	b_2	a_4	b_3
a_1	a_2	b_4	b_5	b_6
b_7	a_2	b_8	a_4	a_5
b_9	b_{10}	a_3	b_{11}	a_4

Figure 8.66

The reader should check that the chase in Example 8.41 cannot proceed without equating nondistinguished variables, even if the closure of the FDs is used.

We shall briefly turn our attention to embedded join dependencies (EJDs). Let $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be a set of relation schemes where $S_1S_2 \cdots S_q = S \subseteq \mathbf{U}$. To test $\mathbf{C} \models *[\mathbf{S}]$, form the tableau $T_{\mathbf{S}}$ over $\mathbf{U}$. Compute $T_{\mathbf{S}}^* = chase_{\mathbf{C}}(T_{\mathbf{S}})$. If $T_{\mathbf{S}}^*$ contains a row that is distinguished in all the S-columns, then $\mathbf{C} \models *[\mathbf{S}]$.

Example 8.42 Let $\mathbf{S} = \{AD, AB, BDE\}$, let $\mathbf{U} = A\,B\,C\,D\,E$, and let $\mathbf{C} = \{A \to C, B \to C, CD \to E, E \to B\}$. We form the tableau $T_{\mathbf{S}}$, as shown in Figure 8.67, and compute $T_{\mathbf{S}}^* = chase_{\mathbf{C}}(T_S)$, as shown in Figure 8.68. Since $T_{\mathbf{S}}^*$ contains a row distinguished in the $ABDE$-columns, the implication holds. Note that the C-column must be included. If $T_{\mathbf{S}}$ were formed over just $A\,B\,D\,E$, as shown in Figure 8.69, $chase_{\mathbf{C}}(T_{\mathbf{S}})$ would not contain the row of all distinguished variables.

$T_S(A$	B	C	D	$E)$
a_1	b_1	b_2	a_4	b_3
a_1	a_2	b_4	b_5	b_6
b_7	a_2	b_8	a_4	a_5

Figure 8.67

$T_S^*(A$	B	C	D	$E)$
a_1	a_2	b_2	a_4	a_5
a_1	a_2	b_2	b_5	b_6
b_7	a_2	b_2	a_4	a_4

Figure 8.68

$T_S^*(A$	B	D	$E)$
a_1	b_1	a_4	b_2
a_1	a_2	b_3	b_4
b_5	a_2	a_4	a_5

Figure 8.69

The chase computation does not generalize to include EJDs as part of **C**. The J-rule for an EJD would only generate a partial row. The partial row could be padded out with new nondistinguished variables, but then the proof of finiteness of the chase fails (Theorem 8.6).

8.9 EXERCISES

8.1 Let $\mathbf{R} = \{AB, BCD, AE\}$. Compute $m_{\mathbf{R}}(r)$ and $m_{\mathbf{R}}(s)$ for the relations r and s in Figures 8.1 and 8.2.

8.2 Prove part 1 of Lemma 8.1.

8.3 Let $\mathbf{R} = \{R_1, R_2, \ldots, R_p\}$ be a database scheme where $R = R_1R_2 \cdots R_p$. Show that for any relation $r(R)$

$$m_{\mathbf{R}}(r) \in FIX(\mathbf{R}).$$

8.4 Prove that for any tableau T with scheme R and any relation $r(R)$, $r \subseteq T(r)$.

8.5 Let T be a tableau with scheme R, and let $r(R)$ be a relation. Show that if T has a distinguished variable in every column, then $T(r)$ is a relation. That is, $T(r)$ is a *finite* set of tuples.

8.6 Apply the tableau

$T(A_1$	A_2	A_3	$A_4)$
a_1	a_2	a_3	b_1
b_2	a_2	a_3	a_4
a_1	b_3	b_4	a_4

to the relation

$r(A_1$	A_2	A_3	$A_4)$
1	3	5	7
1	3	5	8
2	4	6	8
1	4	6	7

8.7* Prove Lemma 8.2. Hint: Show that if tuples $t_1, t_2, \ldots, t_p$ are joinable on **R**, then there is a valuation ρ for $T_{\mathbf{R}}$ that maps w_i to t_i, $1 \leq i \leq p$, where w_i is the row with distinguished variables in the R_i-columns.

8.8 Show that if tableau T contains the row of all distinguished variables, then $T(r) = r$ for any relation r.

8.9 Let $\mathbf{R} = \{A_1A_2A_3A_4, A_2A_3A_4A_5\}$ and let $\mathbf{S} = \{A_1A_2A_3, A_2A_3A_4, A_4A_5\}$. How many sets of relation schemes **Q** are there such that $\mathbf{R} \geq \mathbf{Q} \geq \mathbf{S}$?

8.10 For the sets **R** and **S** of Exercise 8.9, show that the containment $FIX(\mathbf{R}) \supseteq FIX(\mathbf{S})$ is proper.

8.11* Prove a version of Theorem 8.1 where all the containments are proper.

8.12 What is the maximum number of rows a tableau T can have subject to the constraint $SUB(T) = T$?

8.13 Prove Theorem 8.2. Hint: Use the result that $T_{\mathbf{R}} \equiv T_{\mathbf{S}}$ if and only if $\mathbf{R} \simeq \mathbf{S}$.

8.14 Show that for an arbitrary tableau T, $SUB(T) \sqsubseteq T$.

8.15 Prove or disprove: $T_1 \sqsubseteq T_2$ implies $SUB(T_1) \sqsubseteq SUB(T_2)$.

8.16 For the tableaux

$T_1(A_1$	A_2	$A_3)$
a_1	a_2	b_1
b_2	a_2	a_3

and

$T_2(A_1$	A_2	$A_3)$
a_1	a_2	b_1
b_2	a_2	a_3
b_2	b_3	b_1

find a relation r such that the containment

$$T_1(r) \supseteq T_2(r)$$

is proper.

8.17 Given tableau T and rows w_1 and w_2 in T, say w_1 *supersedes* w_2 if w_1 subsumes w_2 and $w_1(A) \neq w_2(A)$ implies $w_2(A)$ is a nondistinguished variable appearing nowhere else in T. Let $SUP(T)$ be T with all superseded rows removed. Prove $SUP(T) \equiv T$.

8.18 Let T_1 and T_2 be tableaux. Prove that if $T_1 \supseteq T_2$ *as sets of rows*, then $T_1 \sqsubseteq T_2$ as mappings.

8.19 Given tableaux T_1 and T_2, give an algorithm to test if there is a containment mapping from T_1 to T_2. What is the time-complexity of your algorithm?

8.20 Let T be a tableau and **C** a set of FDs and JDs. Prove: If $\rho(T) \in SAT(\mathbf{C})$ for some 1-1 valuation ρ, then $\rho'(T) \in SAT(\mathbf{C})$ for any other valuation ρ'.

8.21 What equivalence preserving transformation rules exist for $\mathbf{C} = \emptyset$?

8.22 Apply the F-rules for the FDs $A_1 \rightarrow A_3$ and $A_3A_4 \rightarrow A_2$ to the tableau

$T(A_1$	A_2	A_3	$A_4)$
a_1	b_1	a_3	b_2
b_3	a_2	a_3	a_4
a_1	b_4	b_5	a_4

as many times as possible.

8.23 Prove Theorem 8.4. Hint: Show that if ρ is a valuation such that $\rho(T) \subseteq r$ for $r \in SAT(X \rightarrow A)$ and w_1 and w_2 are rows of T where $w_1(X) = w_2(X)$, then $\rho(w_1(A)) = \rho(w_2(A))$.

8.24 Continue applying the J-rules for **C** in Example 8.20 to tableau T'' until no more changes can be made.

8.25 Let $\mathbf{S} = \{S_1, S_2, \ldots, S_q\}$ be a set of relation schemes, let T be a tableau and let ρ be a valuation for T. Show that if $w_1, w_2, \ldots, w_q$ are rows of T joinable on $\mathbf{S}$ with result w, then $\rho(w_1), \rho(w_2), \ldots, \rho(w_q)$ are joinable on $\mathbf{S}$ with result $\rho(w)$.

8.26 Compute the chase of tableau

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

under the set of constraints $\mathbf{C} = \{A \rightarrow B, E \rightarrow D, *[A\,B\,C\,D, D\,E]\}$.

8.27 Give an example of a generating sequence where two distinct rows in one tableau have the same corresponding row in a subsequent tableau.

8.28 Let $T_0, T_1, \ldots, T_n$ be a generating sequence for an arbitrary chase computation. Show that $T_0 \sqsupseteq T_1 \sqsupseteq \cdots \sqsupseteq T_n$.

8.29 Consider the replacement system of Example 8.25. Show that if the condition that parentheses explicitly express the precedence of $\wedge$ over $\vee$ is removed, then the system is not FCR.

8.30 Complete case 3 of the proof of Theorem 8.9.

8.31 Prove the general case of the corollary to Theorem 8.9.

8.32 Prove that the tableaux

$T_1(A$	B	C	$D)$
a_1	b_1	a_3	b_2
a_1	a_2	b_3	b_4
b_5	a_2	a_3	a_4

and

$T_2(A$	B	C	$D)$
a_1	a_2	b_1	b_2
b_3	a_2	b_4	a_4
b_5	b_6	a_3	a_4

are equivalent on $SAT(\mathbf{C})$, where $\mathbf{C} = \{A \rightarrow B, D \rightarrow C, *[AB, BC, CD]\}$.

8.33 Show that for a tableau T, if $T \equiv T_I$, where T_I is the tableau with just row w_d (of all distinguished variables), then T contains w_d.

8.34 For Example 8.30, find a relation in $SAT(\mathbf{C})$ that has a lossy decomposition onto database scheme **S**.

8.35 (a) Consider the database scheme $\mathbf{R} = \{ABC, ADEI, BDEI, CDEI\}$ and the set of constraints $\mathbf{C} = \{A \to D, B \to E, C \to I\}$. Show that $\mathbf{C} \models *[\mathbf{R}]$, but that for no proper subset **S** of **R** does $\mathbf{C} \models *[\mathbf{S}]$.

(b)* Generalize part (a) to show that for any $n \geq 3$ there is a set **R** of n relation schemes and a set **C** of functional dependencies such that $\mathbf{C} \models *[\mathbf{R}]$, but for no proper subset S of **R** does $\mathbf{C} \models *[\mathbf{S}]$.

(c) Show that if **R** consists of two relation schemes, X and Y, and **C** is only FDs,

$$\mathbf{C} \models *[X, Y] \text{ if and only if}$$
$$\mathbf{C} \models X \cap Y \to X \text{ or } \mathbf{C} \models X \cap Y \to Y.$$

8.36 Prove part 3 of Lemma 8.4.

8.37 What is $(AB)^+$ under the set of constraints

$$\mathbf{C} = \{*[ABC, BCD, DE], B \to D\}?$$

8.38 Let **C** be a set of constraints, X a set of attributes and $Q = \{Y \mid \mathbf{C} \models X \to\to Y \text{ and } X^+ \cap Y = \emptyset\}$. Show that $\mathbf{C} \models X \to\to Y$ if and only if Y can be written $X'\,Y'$, with $X' \subseteq X^+$ and $Y' \in Q$.

8.39 Find $DEP(BC)$ under the set of constraints $\{*[ABD, ACEI], *[ACDI, BCEI], B \to I\}$.

8.40 Show that if **C** is a set of FDs and JDs, then $SAT(\mathbf{C})$ is closed under intersection, but if **C** also has EJDs, it is not necessarily closed under intersection.

8.41 Show that if **C** contains only JDs, then $COMP_{\mathbf{C}}(r)$ always exists, but that if **C** also contains FDs, then $COMP_{\mathbf{C}}(r)$ does not necessarily exist, even if r satisfies all the FDs.

8.42 Given the set of constraints **C** and tableaux T_1 and T_2, show that $T_1 \equiv_{\mathbf{C}} T_2$ does not necessarily imply $REP_{\mathbf{C}}(T_1) = REP_{\mathbf{C}}(T_2)$. Note: In light of Theorem 8.13, you may assume T_1 and T_2, as relations, are in $SAT(\mathbf{C})$.

8.43 Construct an example along the lines of Example 8.39, where **C** consists only of JDs.

8.44 Give a general example of a tableau T with m rows and k columns, and a set **C** of constraints, such that $chase_{\mathbf{C}}(T)$ has m^k rows.

8.45 Show that the procedure for computing the chase given in Section 8.8 has time-complexity $O(n^4)$ if the input is given as a set of relation schemes and a set of FDs, rather than a complete tableau and FDs.

8.46* Suppose we generalize the J-rule to include EJDs, as described at the end of Section 8.8. Partial rows are padded with new nondistinguished variables. Give a set **C** of constraints, which will include EJDs, such that an infinite generating sequence T_0, T_1, T_2 ... exists under **C**. Moreover, the generating sequence must have the property that $T_i \not\supseteq T_{i+1}$, $i \geq 0$.

8.10 BIBLIOGRAPHY AND COMMENTS

Most of the material from Sections 8.1–8.4 is due to Beeri, Mendelzon, et al. [1979] and Aho, Sagiv, and Ullman [1979a, 1979b]. Tableaux and the chase process with FDs alone are due to Aho, Beeri, and Ullman [1979], who used it to test when a set of FDs implies a JD. The extension of the chase to JDs, its use to solve other dependency problems, and the treatment of tableaux as templates are by Maier, Mendelzon, and Sagiv [1979].

Theorem 8.8 is from Sethi [1974]. Graham [1980] offers another proof that the chase is finite Church-Rosser. Liu and Demers [1978] and Downey, Sethi, and Tarjan [1980] have offered fast algorithms for the chase computation with FDs alone. With JDs, fast algorithms probably do not exist. It is NP-hard to test if a JD is implied by a JD and a set of FDs (Maier, Sagiv, and Yannakakis [1981]), a JD and a set of MVDs (Beeri and Vardi [1980b]), or a set of MVDs alone (Tsou [1980]). The first two problems have been shown to be in NP. Kanellakis [1980] has shown intractability when doing inferences from FDs where domain sizes are restricted.

The "if" part of Exercise 3.5c was first noted by Delobel and Casey [1973]. The "only if" part was noted by Rissanen [1977].