AD-A135 303 NOTIONS OF DEPENDENCY SATISFACTION(U) STANFORD UNIV CA 1/1
DEPT OF COMPUTER SCIENCE M H GRAHAM ET AL. AUG 83
STAN-CS-83-979 AFOSR-TR-83-0961 AFOSR-80-0212

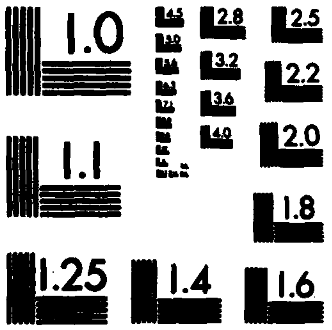
UNCLASSIFIED F/G 9/2 NL

END

FILMED

1-84

DTIC


1.0
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

August 1983 Report No. STAN-CS-83-979 (4)

AFOSR-TR- 83-0961

AD-A135 303

Notions of Dependency Satisfaction

by

M. H. Graham, A. O. Mendelzon, and M. Y. Vardi

Department of Computer Science

Stanford University
Stanford, CA 94305

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 83-0961 · 2. GOVT ACCESSION NO. AD-A135303	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) NOTIONS OF DEPENDENCY SATISFACTION	5. TYPE OF REPORT & PERIOD COVERED TECHNICAL
	6. PERFORMING ORG. REPORT NUMBER STAN-CS-83-979
7. AUTHOR(s) M.H. Graham, A.O. Mendelzon and M.Y. Vardi	8. CONTRACT OR GRANT NUMBER(s) AFOSR-80-0212
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science Stanford University Stanford CA 94305	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE61102F; 2304/A2
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematical & Information Sciences Directorate Air Force Office of Scientific Research /NM Bolling AFB DC 20332	12. REPORT DATE AUG 83
	13. NUMBER OF PAGES 36
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Two notions of dependency satisfaction, consistency and completeness, are introduced. Consistency is the natural generalization of weak-instance satisfaction and seems appropriate when only equality-generating dependencies are given, but disagrees with the standard notion in the presence of tupe-generating dependencies. Completeness is based on the intuitive semantics of tuple-generating dependencies but differs from the standard notion for equality-generating dependencies. It is argued that neither approach is the correct one, but rather that they correspond to different circumstances. (CONTINUED)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ITEM #20, CONTINUED: Consistency and completeness of a state are characterized in terms of the tableau associated with the state and in terms of logical properties of a set of first-order sentences associated with the state. A close relation between the problems of testing for consistency and completeness and of testing implication of dependencies is established, leading to lower and upper bounds for the complexity of consistency and completeness. The possibility of formalizing dependency satisfaction without using a universal relation scheme is examined.

Notions of Dependency Satisfaction

Marc H. Graham

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

Alberto O. Mendelzon

Computer Systems Research Group
University of Toronto
Toronto, Canada M5S 1A4

Moshe Y. Vardi[†]

Computer Science Department
Stanford University
Stanford, California 94305

ABSTRACT

Two notions of dependency satisfaction, *consistency* and *completeness*, are introduced. Consistency is the natural generalization of weak-instance satisfaction and seems appropriate when only equality-generating dependencies are given, but disagrees with the standard notion in the presence of tuple-generating dependencies. Completeness is based on the intuitive semantics of tuple-generating dependencies but differs from the standard notion for equality-generating dependencies. It is argued that neither approach is the *correct* one, but rather that they correspond to different policies on constraint enforcement, and each one is appropriate in different circumstances. Consistency and completeness of a state are characterized in terms of the tableau associated with the state and in terms of logical properties of a set of first-order sentences associated with the state. A close relation between the problems of testing for consistency and completeness and of testing implication of dependencies is established, leading to lower and upper bounds for the complexity of consistency and completeness. The possibility of formalizing dependency satisfaction without using a universal relation scheme is examined.

August 1983

AFOSR-80-0212

[†] Research supported by a Weizmann Post-Doctoral Fellowship and [redacted] Address: IBM Research Laboratory, 5600 Cottle Rd., San Jose CA 95193.

Notions of Dependency Satisfaction

Marc H. Graham

School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332

Alberto O. Mendelzon

Computer Systems Research Group
University of Toronto
Toronto, Canada M5S 1A4

Moshe Y. Vardi[†]

Computer Science Department
Stanford University
Stanford, California 94305

1. Introduction

The starting point of dependency theory is the notion that certain database states are *legal*, or *correct*, and others are not. The class of formal statements called *dependencies* is studied as a language for the specification of the allowable states. It is therefore of great importance to have a precise notion of when a database state *satisfies* a set of dependencies.

When the database state consists of a single relation, dependency theory does provide an adequate notion of satisfaction. Dependencies are restated as first order sentences on a language with a single predicate letter, and a relation satisfies a set of dependencies if it provides a model for the associated set of sentences. The generalization of this notion to multi-relation databases has until recently followed one of two paths. In the first approach, a dependency is said to hold in a particular relation, so inter-relation dependencies are eliminated by definition. The second approach conveniently assumes that the database consists of the set of projections of some universal relation; questions of satisfaction are dealt with by examining this universal relation.

More recent work [H, V] has pointed out the limitations of these approaches and proposed a more appealing one. A database state is said to satisfy a set of functional dependencies if there exists a universal relation I such that I satisfies the dependencies and is a *containing instance*, that is, the projections of I on the given relation schemes contain each of the given relations. Any such

[†] Research supported by a Weizmann Post-Doctoral Fellowship and AFOSR 80-0212 ~~grant 80-12907.~~ Address: IBM Research Laboratory, 5600 Cottle Rd., San Jose CA 95193.

I is called a ***weak instance*** for the database state; this notion is often called ***weak satisfaction.*** Detailed justification and discussion of this approach can be found in [G,H,V]. It is worth noting that, when only functional dependencies are present, a single-relation database is weakly satisfying precisely when it is satisfying in the standard sense.

Given the useful properties of weak satisfaction [G,M,S,Y], it is tempting to generalize it directly to other kinds of dependencies by saying that a database state satisfies a set of (functional, multivalued, join, template, etc.) dependencies if there exists a containing instance for it that satisfies the dependencies. In this case we shall say that the state is *consistent* with the dependencies.

Our first objection to this proposal is that weak satisfaction now becomes different from standard satisfaction for single-relation databases. For example, if all the dependencies are total tuple-generating dependencies (such as multivalued and join dependencies), then any database state satisfies any set of dependencies. In particular, every single relation can be made into a weak instance for itself by adding a finite number of tuples to it [MMS,BV1].

A second, related, objection is that this proposal does not seem to capture the intuitive semantics of tuple-generating dependencies in multi-relation databases either. Consider for example the following database.

Example 1:

R_1

Student	*Course*
Jack	*CS378*

R_2

Course	*Room*	*Hour*
CS378	*B215*	*M10*
CS378	*B213*	*W10*

R_3

Student	*Room*	*Hour*
Jack	*B215*	*M10*

and dependencies $\{SH \rightarrow R,\ RH \rightarrow C,\ C \twoheadrightarrow S \mid RH\}$.

The multivalued dependency $C \twoheadrightarrow S \mid RH$ is intended to express the fact that a student is associated with every (r,h) pair such that some course that the student takes meets at room r at time h. This constraint is intuitively violated in the example, since Jack is not associated with room B215 on Wednesdays at 10 although he is taking CS378.

An alternative approach to defining satisfaction is through the notion of *complete states*, introduced for a different purpose in [M]. We say that a state is complete if it contains each tuple that appears in the projections of every weak instance for the state. The state in Example 1 is not complete, because every weak instance for it contains the sub-tuple $\langle Jack, B213, W10 \rangle$, which does not appear in R_3. When only tuple-generating dependencies are given, completeness coincides with the standard notion of satisfaction on single-relation databases.

An objection to the completeness criterion is that it seems unnatural for equality-generating dependencies such as functional dependencies. Consider the following example

Example 2:

R_1

Student	*Course*
Jack	*CS378*

R_2

Course	*Room*	*Hour*
CS378	*B215*	*M10*

R_3

Student	*Room*	*Hour*
John	*B320*	*F12*

with the only dependency $C \rightarrow RH$.

This is not a complete state, since the sub-tuple $\langle Jack, B215, M10 \rangle$ will be forced by $C \rightarrow RH$ to appear in every weak instance, but it does not appear in R_3. However, it is hard to argue that this state *violates* the $C \rightarrow RH$ dependency, which simply requires that each course be associated with a unique room and time.

In sum, we have described two notions of dependency satisfaction, *consistency* and *completeness*. Consistency is the natural generalization of weak satisfaction and seems appropriate when only equality-generating dependencies are given, but disagrees with the standard notion in the presence of tuple-generating dependencies. Completeness is based on the intuitive semantics of tuple-generating dependencies but appears unnatural for equality-generating dependencies. It is our thesis that neither approach is the *correct* one, but rather that they correspond to different policies on constraint enforcement, and each one is appropriate in different circumstances.

After introducing definitions and notation in Section 2, Section 3 presents the notions of consistency and completeness. We show how to construct for a database state ρ two sets of first order sentences, C_ρ and K_ρ, such that ρ is consistent with the given dependencies if and only if C_ρ is finitely satisfiable and ρ is complete with respect to the given dependencies if and only if K_ρ is finitely satisfiable. In Section 4, we characterize consistency and completeness in terms of the chase of the associated tableaux [M]. When all dependencies are total, our results provide a decision procedure for testing consistency and completeness of a state. However, we show that testing whether a state is inconsistent with a typed equality-generating dependency and testing whether a state is incomplete with respect to a join dependency are both NP-complete problems. Furthermore, the general problems of testing consistency and completeness under full dependencies are shown to be complete in exponential time. In Section 5, we study the decision problem for consistency and completeness when embedded dependencies are present. By relating consistency and completeness to the well-studied question of dependency implication, we show that both are undecidable in the general case. Finally, in Section 6 we examine the construction of sets of sentences similar to C_ρ

and K_ρ, but without using a predicate that stands for the universal relation scheme. We show that this can be done when the database scheme is weakly cover-embedding, a necessary condition for independence of the database scheme [GY].

Our results deal with *untyped* relations and dependencies, that is, a value may appear in different columns of a relation. However, all of the results, except for Theorems 8, 9 and 15, can be specialized to the typed case.

2. Definitions and Notation

2.1. Relations, database states and tableaux

We fix a finite set of *attributes* called the *universe*, $U = \{A_1, \ldots, A_n\}$. Each attribute A_i has an associated infinite set called its *domain* and denoted dom(A_i). Since we deal with *untyped* databases, we shall let all the domains be the same, say the integers. A *relation scheme* R is a subset of U. A *database scheme* $\mathbf{R} = \{R_1, \ldots, R_k\}$ is a collection of relation schemes such that the union of the R_i's is U. A *tuple* defined on relation scheme R is a function that maps each attribute in R to a value. The value can be either an integer or a *variable* taken from an infinite set of uninterpreted symbols. A tuple can be visualized as a row of table where the columns are labeled by the attributes. A *tableau* on R is a finite set of tuples defined on R. If t is a tuple on R and X is a subset of R, $t[X]$ denotes the restriction of t to X. If $t(A)$ is an integer for every A in X, we say that t is *total on X*. A *relation* on R is a tableau on R such that every tuple is total on R. A relation on U is called a *universal relation.*

For r a tableau on R and X a subset of R, the *projection of r on X* is

$$\pi_X(r) = \{\, t[X] \mid t \in r \text{ and } t \text{ is total on } X \,\}.$$

Note that our definition of projection corresponds to what is sometimes called "total projection" in the literature, so the projection of any tableau on any set of attributes is always a relation. When $\mathbf{R} = \{R_1, \ldots, R_k\}$, we write $\pi_{\mathbf{R}}(r)$ for $\langle \pi_{R_1}(r), \ldots, \pi_{R_k}(r) \rangle$.

A state of a database scheme $\mathbf{R}$ is a function ρ that maps every relation scheme R in $\mathbf{R}$ to a relation on R. We write

$$\rho = \langle r_1, \cdots, r_k \rangle = \langle \rho(R_1), \cdots, \rho(R_k) \rangle$$

We associate with each state ρ of **R** a tableau T_ρ defined on the universe. T_ρ contains exactly one tuple for each tuple in each relation of ρ. The tuple r corresponding to tuple t in $\rho(R)$ is constructed by letting $r[R]=t[R]$ and filling the rest of r with distinct variables that appear nowhere else in T_ρ.

Example 3: Let $\mathbf{R}=\{AB,BCD,AD\}$, and ρ is the state below.

$\rho(AB)$

A	B
1	2
1	3

$\rho(BCD)$

B	C	D
2	5	8
4	6	7

$\rho(AD)$

A	D
1	9

Then T_ρ is:

A	B	C	D
1	2	b_1	b_2
1	3	b_3	b_4
b_5	2	5	8
b_6	4	6	7
1	b_7	b_8	9

A *valuation* v for a tableau R is a mapping from the symbols in the tableau into variables and constants such that $v(c)=c$ for every constant c that appears in T.

2.2. Dependencies

Following [BV1], we use tableaux to represent implicational dependencies. A *template dependency* (td) is a pair $d=\langle T,w\rangle$, where T is a tableau containing no constants and w is a tuple containing no constants. We say d is *full* or *total* if $w[A]$ appears in T for every attribute A. Otherwise, d is said to be *embedded* or *partial*. A relation I satisfies d if for every valuation v such that $v(T)\subseteq I$, there exists an extension v' of v to all the symbols of w such that $v'(w)\in I$. Informally, a template dependency says that if certain tuples appear in I, then some other tuple must also appear. Template dependencies are a special case of the *tuple-generating dependencies* (*tgd's*), where a set of tuples is allowed instead of the single tuple w. For total dependencies, one can assume without loss of generality that a single tuple appears on the right hand side [BV1]; thus total template dependencies are no less general than total tuple-generating dependencies. *Join dependencies* [ABU,Ri] are a specia case of total td's. An *equality-generating dependency* (egd) is a pair $d = \langle T,(a_1,a_2)\rangle$, where T is a tableau containing no constants, and a_1,a_2 are variables that appear in T for some A. A tableau S satisfies an egd $d = \langle T,(a_1,a_2)\rangle$ if for every valuation v such that $v(T)\subseteq S$, $v(a_1) = v(a_2)$. *Functional dependencies* are a special case of egd's.

Egd's also act like tgd's, since by generating new equalities they generate new tuples. This action can be simulated by total td's. Beeri and Vardi [BV1, BV2] show how to construct, given a set D of dependencies, a set $\overline{D}$ of tgd's that has the following properties:

(1) $\overline{D}$ is obtained from D by replacing each egd by some td's.

(2) $D \models \overline{D}$.

(3) Let d be a tgd. If $D \models d$ then $\overline{D} \models d$.

We call $\overline{D}$ the *egd-free version* of D.

3. Consistency and Completeness

In this section we define two properties of a database state with respect to a set of dependencies, consistency and completeness, which we consider to be two different aspects of dependency satisfaction. We characterize these properties in terms of the satisfiability of two first-order theories associated with the state.

Let us fix a database scheme $\mathbf{R}=\{R_1,\ldots,R_n\}$. Let WEAK$(D,\rho)$ be the set of all weak instances for a database state ρ under a set of dependencies D. That is, WEAK(D,ρ) is the set of all universal relations satisfying D such that their projections contain each relation in ρ. Say a state ρ is *consistent* with set of dependencies D if WEAK$(D,\rho)\neq\emptyset$. The *completion* of a state ρ, ρ^+, is defined by

$$\rho^+ = \bigcap_{I\in \text{WEAK}(\overline{D},\rho)} \{\pi_{\mathbf{R}}(I)\},$$

where the intersection is taken relation-wise. Note that $\rho\subseteq\rho^+$ for any ρ. Say a state ρ is *complete* with respect to set of dependencies D if ρ equals its completion, that is, $\rho = \rho^+$.

Intuitively, a state ρ is consistent if there is some way of adding tuples to relations of ρ that will transform it into the set of projections of some satisfying universal instance. If ρ is consistent, then there are many different sets of tuples which can be added to it to demonstrate its consistency. However, there are certain tuples which will have to appear in every such extension of ρ. If all these necessary tuples are already in ρ to begin with, then we say ρ is complete. Note that the definition of completeness is based on the egd-free version of D, $\bar{D}$. This is done to allow consistency and completeness to be independent notions. While WEAK(D,ρ) could be empty (for inconsistent ρ), WEAK$(\bar{D},\rho)$ is never empty. We will show later that, for consistent states, it does not matter whether D or $\bar{D}$ is used.

Several workers [GM,Ni] have advocated the use of first order logic to express dependencies and other constraints. As we explained in the introduction, their approach does not easily generalize to dependencies in multi-relation databases. Given a dependency statement such as $X \rightarrow Y$ in a database scheme where X and Y may not appear together in one relation scheme, or may appear in more than one, it is not clear how satisfaction of the dependency can be formalized as satisfaction of some first order sentence.

The notions of satisfaction proposed above do provide a means of using first order logic to formalize dependencies. However, a rather drastic shift in point of view is required. It is no longer possible to write down a sentence for each dependency and ask whether the database provides a model for each of these sentences. Consider for example the notion of consistency as satisfaction, and let $d_1 = A \rightarrow C$, $d_2 = B \rightarrow C$, with the database scheme $\{AB, BC\}$. Let $\rho(AB) = \langle 00,01 \rangle$ and $\rho(BC) = \langle 01,12 \rangle$. It is easy to see that ρ is consistent with d_1 and with d_2, but it is not consistent with $\{d_1, d_2\}$.

Our approach is to construct two sets of sentences, C_ρ and K_ρ, for each state ρ. We will show that ρ is consistent exactly when C_ρ is finitely satisfiable and that ρ is complete exactly when K_ρ is finitely satisfiable. Thus we reduce both notions of satisfaction to the standard logical notion of finite satisfiability. Note that in this approach consistency and completeness of a state are not first order notions; they are statements about first order theories rather than statements in these theories.

Before constructing the sets of sentences C_ρ and K_ρ, we fix a linear ordering on the elements of the universe U. We now write U as the sequence $\langle A_1, \ldots, A_n \rangle$. Each relation scheme R in **R** may be written as the sequence $\langle A_{i_1}, \ldots, A_{i_m} \rangle$, where $i_j < i_k$ for $j < k$.

C_ρ and K_ρ each contain two subsets of sentences, the *database scheme axioms* and the *state* axioms. The scheme axioms depend only on the database scheme **R** and the set of dependencies D;

the state axioms depend on the state ρ.

The scheme axioms of both C_ρ and K_ρ include the *containing instance* axioms. For each relation scheme R, there is one containing instance axiom that says that every tuple in the R-relation of the state must be the projection on R of some tuple of the universal relation. In other words, the containing instance axioms assert the existence of a containing instance for the state. Formally, for each relation scheme $R = \langle A_{i_1}, \ldots, A_{i_m} \rangle$ in **R**, there is a sentence of the form

$$\forall \mathbf{a} \exists \mathbf{y} (R(a_1, \ldots, a_m) \rightarrow U(\mathbf{y}_0, a_1, \mathbf{y}_1, a_2, \ldots, a_m, \mathbf{y}_m))$$

where **a** is the sequence $a_1, a_2, \ldots, a_m$ and **y** is the sequence $\langle \mathbf{y}_0, \ldots, \mathbf{y}_m \rangle$, no a_j appears in **y** and no symbol appears more than once in **y**. The sequence $\mathbf{y}_j$ is of length $i_{j+1} - (i_j + 1)$, where $i_0 = 0$ and $i_{m+1} = n + 1$.

The scheme axioms of C_ρ also include the *dependency axioms*, which are just the dependencies in D encoded as implicational sentences as described by Fagin [F]. K_ρ also contains dependency axioms, but in this case we use the egd-free version of D, $\tilde{D}$, rather than D itself.

The state axioms of both C_ρ and K_ρ contain the state ρ encoded as a set of quantifier-free sentences. For each tuple $\langle a_1, \ldots, a_m \rangle$ in $\rho(R)$, we include the sentence $R(a_1, \ldots, a_m)$, where the a_i's are constants.

Finally, the state axioms of C_ρ include the *distinctness axioms*, and those of K_ρ include the *completeness* axioms. The distinctness axioms are the set of inequalities $c \neq d$, where c and d are distinct constants appearing in ρ. The completeness axioms contain, for each tuple $\langle a_1, \ldots, a_m \rangle$ such that each a_i appears in ρ but the tuple itself does not appear in $\rho(R)$, the sentence

$$\forall \mathbf{y} (\neg U(\mathbf{y}_0, a_1, \ldots, a_m, \mathbf{y}_m))$$

where **y** is constructed as in the containing instance axioms. Intuitively, the completeness axioms say that only tuples appearing in $\rho(R)$ can be in the projection of the universal relation on R.

Example 4: We construct C_ρ and K_ρ for the state shown in Example 1. U is the sequence $\langle S,C,R,H \rangle$; **R** contains the schemes $R_1 = \langle S,C \rangle$, $R_2 = \langle C,R,H \rangle$, and $R_3 = \langle S,R,H \rangle$. The dependencies are the functional dependencies $SH \rightarrow R$, $RH \rightarrow C$, and the multivalued dependency $C \twoheadrightarrow S \mid RH$.

- Containing Instance Axioms:

$$\forall\, s,c \;\exists\, r,h\; (R_1(s,c) \rightarrow U(s,c,r,h))$$

$$\forall\, c,r,h \;\exists\, s\; (R_2(c,r,h) \rightarrow U(s,c,r,h))$$

$$\forall\, s,r,h \;\exists\, c\; (R_3(s,r,h) \rightarrow U(s,c,r,h))$$

- Dependencies:

$$(\forall\, s_1c_1c_2h_1r_1r_2)(U(s_1,c_1,r_1,h_1) \wedge U(s_1,c_2,r_2,h_1) \rightarrow r_1 = r_2)$$

$$(\forall\, s_1s_2c_1c_2h_1r_1)(U(s_1,c_1,r_1,h_1) \wedge U(s_2,c_2,r_1,h_1) \rightarrow c_1 = c_2)$$

$$(\forall\, s_1s_2c_1r_1r_2h_1h_2)(U(s_1,c_1,r_1,h_1) \wedge U(s_2,c_1,r_2,h_2) \rightarrow U(s_2,c_1,r_1,h_1))$$

- Egd-free Dependency Axioms:

$$(\forall\, s_1s_2c_1c_2c_3h_1h_2r_1r_2)(U(s_1,c_1,r_1,h_1) \wedge U(s_1,c_2,r_2,h_1) \wedge U(s_2,c_3,r_1,h_2) \rightarrow U(s_2,c_3,r_2,h_2))$$

$$(\forall\, s_1s_2s_3c_1c_2h_1h_2r_1r_2)(U(s_1,c_1,r_1,h_1) \wedge U(s_2,c_2,r_1,h_1) \wedge U(s_3,c_1,r_2,h_2) \rightarrow U(s_3,c_2,r_2,h_2))$$

$$(\forall\, s_1s_2c_1r_1r_2h_1h_2)(U(s_1,c_1,r_1,h_1) \wedge U(s_2,c_1,r_2,h_2) \rightarrow U(s_2,c_1,r_1,h_1))$$

etc.

- State axioms:

$R_1(Jack, CS378)$

$R_2(CS378, B215, M10)$

$R_2(CS378, B213, W10)$

$R_3(Jack, B215, M10)$

- Distinctness axioms:

$B215 \neq B213$

$M10 \neq W10$

$Jack \neq CS378$

$Jack \neq B215$

etc.

- Completeness axioms:

For R_1:

$$\forall\, r,h \neg U(CS378,CS378,r,h)$$

$$\forall\, r,h \neg U(B215,CS378,r,h)$$

etc.

For R_2:

$$\forall\, s \neg U(s,CS378,B213,M10)$$

$$\forall\, s \neg U(s,CS378,B215,W10)$$

etc.

For R_3:

$$\forall\, c \neg U(Jack,c,B213,W10)$$

etc.

C_p consists of the containing instance axioms, the dependency axioms, the state axioms and the distinctness axioms. K_p consists of the containing instance axioms, the egd-free dependency axioms, the state axioms and the completeness axioms.

Before proceeding to our results, we introduce some basic definitions and notation of model theory. A *structure* for a language L consists of a *domain* of elements and an *interpretation* of each predicate and constant of L. A structure with a finite domain is *finite*. If M is a structure with domain A and P is a k-ary predicate symbol, then $M(P) \subseteq A^k$ is the interpretation of P in M, and if c is a constant, $M(c) \in A$ is the interpretation of c in M.

A structure M for a language L is a *model* of a set of sentences Σ if for every $\sigma \in \Sigma$, σ is true in M, written $M \models \sigma$ (we assume the reader to be familiar with the notion of truth in a model). A set of sentences is *finitely satisfiable* if it has a finite model.

The next two theorems establish that finite satisfiability of C_ρ and K_ρ are identical respectively to consistency and completeness of ρ with respect to D.

Theorem 1: C_ρ is finitely satisfiable if and only if ρ is consistent with D.

Proof: If ρ is consistent, it is clear that for every $I \in \text{WEAK}(D,\rho)$ the structure M with $M(R) = \rho(R)$ and $M(U) = I$ is a finite model of C_ρ. For the converse, let M be a finite model of C_ρ. For each pair of constants c, d in the language of C_ρ, we have $M \models c \neq d$, so $M(c) \neq M(d)$. Thus we can assume without loss of generality that constants are interpreted as themselves, i.e. $M(c) = c$. It is easy to verify that $M(U)$ is a containing instance for ρ that satisfies D. That is, $M(U) \in \text{WEAK}(D,\rho)$, so ρ is consistent with D. □

Theorem 2: K_ρ is finitely satisfiable if and only if ρ is complete with respect to D.

Proof: Suppose K_ρ is satisfiable with a finite model M. We claim that K_ρ has a finite model M$'$ where no two constants have the same interpretation. To construct M$'$, we replace each element of the domain of M with many distinct copies of that element. For example, if $\langle a,b,c \rangle \in M(R)$, then $M'(R)$ would contain tuples $\langle a_1,b_1,c_1 \rangle$, $\langle a_1,b_1,c_2 \rangle$, $\langle a_1,b_2,c_3 \rangle$, etc. If two constants are interpreted as the same element in M, they would be interpreted as two distinct copies of that element in M$'$. Since K_ρ does not have equality in it, M$'$ is still a model of K_ρ. Thus, without loss of generaity, we can assume that in M$'$ constants are interpreted as themselves, so $M'(U) \in \text{WEAK}(\overline{D},\rho)$. By the completeness axioms, $\pi_R(I)$ does not contain any tuple constructed from values appearing in ρ but not itself appearing in ρ. Hence, ρ^+ cannot contain any such tuple, so ρ is complete.

For the converse, suppose that ρ is complete. Consider the set S of all tuples t on some relation scheme R such that t is constructed from values appearing in ρ but t is not in ρ. Since ρ is complete, if t is an R-tuple in S, then there is a universal relation $I_t \in \text{WEAK}(\overline{D},\rho)$ such that $t \notin \pi_R(I_t)$. We use now the *direct product* construction [F] to produce a universal relation $I \in \text{WEAK}(\overline{D},\rho)$ such that if t is an R-tuple from S, then $t \notin \pi_R(I)$. I is the direct product $\bigtimes_{t \in S} I_t$, constructed as follows. Let $S = \{t_1, \ldots, t_m\}$. The values in I are m-sequences $c = \langle c_1, \ldots, c_m \rangle$

of constants from ρ, where we identify the m-sequence $\langle c,c,\cdots,c\rangle$ with the constant c. Now a tuple s is in I if and only if the tuple s_i, which is obtained from s by projecting each m-sequence on its i-th component, is in I_{t_i}. It is straightforward to verify that I is a containing instance for ρ. Furthermore, since dependencies are preserved under direct product [F], I must satisfy $\overline{D}$. The finite structure M with $\mathrm{M}(R) = \rho(R)$ for each R and $\mathrm{M}(U) = I$ is a model of K_ρ. □

4. Testing Satisfaction under Full Dependencies

In this section we show that both consistency and completeness of a state can be tested by chasing the associated tableau, when embedded dependencies are not present. Thus the upper bounds on complexity known for chasing tableaux under various special kinds of full dependencies apply also to testing satisfaction. Throughout this section, D is a set of full dependencies. We start by defining the chase of a tableau under full dependencies and establishing preliminary results about chasing with the egd-free version of D.

The *chase* of a tableau T with respect to a set of dependencies D, denoted by $\text{CHASE}_D(T)$, is the result of applying the following two transformation rules exhaustively to T.

Td-rule:

If $\langle S,w\rangle$ is a td in D, and there exists a valuation v such that $v(S)\subseteq T$, add $v(w)$ to the rows of T.

Egd-rule:

Suppose $\langle S,(a_1,a_2)\rangle$ is an egd in D, and there exists a valuation v such that $v(S)\subseteq T$, and $v(a_1),v(a_2)$ are not both constants. If only one of $v(a_1),v(a_2)$ is a constant, rename all occurrences of the other one in T to that constant. If both are variables, rename all occurrences of the higher numbered variable to the lower numbered one.

Given a tableau T_ρ and set of dependencies D, we will write

$$T_\rho^* = \text{CHASE}_D(T_\rho)$$

and

$$T_\rho^+ = \text{CHASE}_{\overline{D}}(T_\rho)$$

where $\overline{D}$ is the egd-free version of D.

We need some preliminary results before characterizing consistent states.

Lemma 1: For each $I \in \text{WEAK}(D,\rho)$, there is a valuation ν such that $\nu(T_\rho^*) \subseteq I$.

Proof: By induction on the computation of T_ρ^*. □

Lemma 2: For any consistent state ρ, $\bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\} = \pi_{\mathbf{R}}(T_\rho^*)$.

Proof: Let $I \in \text{WEAK}(D,\rho)$. Let ν be the valuation of the previous lemma, $\nu(T_\rho^+) \subseteq I$. Then

$$\pi_{\mathbf{R}}(T_\rho^*) \subseteq \pi_{\mathbf{R}}(\nu(T_\rho^*)) \subseteq \pi_{\mathbf{R}}(I).$$

It follows that $\pi_{\mathbf{R}}(T_\rho^*) \subseteq \bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\}$. For the other inclusion, let t be a tuple in the R_i-component of $\bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\}$. Let ν be an injective valuation for T_ρ^* that maps each variable to a constant not appearing in t. Since $\nu(T_\rho^*) \in \text{WEAK}(D,\rho)$, $t \in \pi_{R_i}(\nu(T_\rho^*))$. By construction of ν, this implies that t must come from some R_i-total tuple of T_ρ^*, so $t \in \pi_{R_i}(T_\rho^*)$. □

Consistency and completeness of a state can be characterized in terms of the associated tableau T_ρ as follows.

Theorem 3: The following are equivalent.

(a) ρ is consistent with D.

(b) T_ρ^* satisfies D.

Proof:

(a) implies (b): Let $I \in \text{WEAK}(D,\rho)$, and let ν be the valuation of Lemma 1. Suppose that T_ρ^* does not satisfy some $d \in D$. d cannot be a td, since in that case a td-rule is applicable to T_ρ^*. Thus d must be an egd $\langle S,(a_1,a2)\rangle$, and there is a valuation ν' such that $\nu'(S) \subseteq T_\rho^*$ and $\nu'(a_1) \neq \nu'(a_2)$. Both $\nu'(a_1)$ and $\nu'(a_2)$ must be constants, otherwise an egd-rule is applicable to T_ρ^*. But now $\nu(\nu'(S)) \subseteq I$ and $\nu(\nu'(a_1)) \neq \nu(\nu'(a_2))$, so I does not satisfy D - a contradiction.

(b) implies (a): Let ν be an injective valuation for T_ρ^* that map each variable to a constant not appearing in ρ. Then $\nu(T_\rho^*)$ satisfies D and it is a containing instance for ρ. Thus $\text{WEAK}(D,\rho) \neq \emptyset$. □

We need two more preliminary results before characterizing complete states.

Lemma 3: For each $I \in \text{WEAK}(\overline{D},\rho)$, there is a valuation ν such that $\nu(T_\rho^+) \subseteq I$.

Proof: By induction on the computation of T_ρ^+. □

The next lemma shows that the completion of a state can be obtained from T_ρ^+.

Lemma 4: For any state ρ, $\rho^+ = \pi_R(T_\rho^+)$.

Proof: Let $I \in \text{WEAK}(\bar{D},\rho)$. Let ν be the valuation of the previous lemma, $\nu(T_\rho^+) \subseteq I$. Then

$$\pi_R(T_\rho^+) \subseteq \pi_R(\nu(T_\rho^+)) \subseteq \pi_R(I).$$

It follows that $\pi_R(T_\rho^+) \subseteq \rho^+$. For the other inclusion, let t be a tuple in the R_i-component of ρ^+. Let ν be an injective valuation for T_ρ^+ that maps each variable to a constant not appearing in t. Since $\nu(T_\rho^+) \in \text{WEAK}(\bar{D},\rho)$, $t \in \pi_{R_i}\nu(T_\rho^+)$. By construction of ν, this implies that t must come from some R_i-total tuple of T_ρ^+, so $t \in \pi_{R_i}(T_\rho^+)$. □

Theorem 4: The following are equivalent.

(a) ρ is complete with respect to D.

(b) ρ is complete with respect to $\bar{D}$.

(c) $\rho = \pi_R(T_\rho^+)$.

Proof: The equivalence of (a) and (b) is immediate from the fact that $\bar{\bar{D}} = \bar{D}$. The equivalence of (a) and (c) follows directly from Lemma 4. □

We have defined completeness and consistency to be independent notions. However, it is interesting to note that, for consistent states, the notion of completeness can be simplified as follows.

Theorem 5: For state ρ consistent with dependencies D, the following are equivalent:

(a) ρ is complete with respect to D.

(b) $\rho = \pi_R(T_\rho^*)$.

(c) $\rho = \bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_R(I)\}$.

Proof:

(b) equivalent to (c): Follows from Lemma 2.

(a) equivalent to (b): By Theorem 2, ρ is complete wrt D iff if $\rho = \pi_R(T_\rho^+)$. We claim that $\pi_R(T_\rho^*) = \pi_R(T_\rho^+)$ for consistent states. Since ρ is consistent, by Theorem 1, T_ρ^* satisfies D. By property (2) of $\bar{D}$, we also have that T_ρ^* satisfies $\bar{D}$. Hence, by Lemma 3, there is a valuation ν such that $\nu(T_\rho^+) \subseteq T_\rho^*$. Consequently, $\pi_R(T_\rho^+) \subseteq \pi_R(T_\rho^*)$. For the other inclusion, let t be a tuple in the R_i-component of $\pi_R(T_\rho^*)$. Let ν be a map for T_ρ that sends distinct constants to distinct variables. Let $T = \nu(T_\rho)$, and let s be a tuple such that $s[R_i] = \nu(t)$ and the rest of s consists of distinct new variables. We claim that $D \models \langle T,s \rangle$. Indeed, there is a tuple t_1 in T_ρ^* such that

$t_1[R_i]=t$. Therefore, there is a tuple s_1 in $\text{CHASE}_D(T)$ such that $s_1[R_i]=s[R_i]$. By the results in [BV1], it follows that $D \models \langle T,s \rangle$. But then also $\bar{D} \models \langle T,s \rangle$, by property (3) of $\bar{D}$. Thus, by the results in [BV1], there is a tuple s_2 in $\text{CHASE}_{\bar{D}}(T)$ such that $s_2[R_i]=s[Ri]$. It follows that $t \in \pi_{\mathbf{R}}(T_\rho^+)$. □

Corollary 1: The following are equivalent:

(a) ρ is consistent and complete with respect to D.

(b) T_ρ^* satisfies D and $\rho = \pi_{\mathbf{R}}(T_\rho^*)$.

(c) $\rho = \bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\}$. □

The next theorem relates consistency and completeness to standard satisfaction for single relations.

Theorem 6: For $\mathbf{R} = \{U\}$, $\rho(U)$ satisfies D if and only if ρ is consistent and complete with respect to D.

Proof: (Only if) Let $J = \rho(U) \in \text{SAT}(D)$. Since $J \in \text{WEAK}(D,\rho)$, ρ is consistent with D. We claim also that $\rho = \bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\}$. Since $\rho = \pi_{\mathbf{R}}(J)$, clearly $\bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\} \subseteq \rho$. Furthermore, for every $I \in \text{WEAK}(D,\rho)$, $J \subseteq I$, that is, $\rho \subseteq \pi_{\mathbf{R}}(I)$, so $\rho \subseteq \bigcap_{I \in \text{WEAK}(D,\rho)} \{\pi_{\mathbf{R}}(I)\}$. Therefore ρ is complete wrt D.

(If) Let ρ be consistent and complete with respect to D. Then T_ρ^* satisfies D by Theorem 3, and $\rho = \pi_{\mathbf{R}}(T_\rho^*)$ by Theorem 5. Since $\mathbf{R} = \{U\}$ and all dependencies are total, clearly $\pi_{\mathbf{R}}(T_\rho^*) = T_\rho^*$, hence $\rho = T_\rho^*$ satisfies D. □

As a consequence of Theorems 3 and 4, the chase is a decision procedure for consistency and completeness under full dependencies. In the rest of this section we shall give upper and lower complexity bounds for these problems.

We first give NP-completeness results that follow from Theorem 6.

Theorem 7:

(1) For $\mathbf{R}=\{U\}$, testing whether a state ρ is inconsistent with a typed egd or whether it is incomplete with respect to a jd is NP-complete.

(2) For $\mathbf{R}=\{U\}$, testing whether a state ρ is not complete with respect to a set D of full dependencies is NP-complete.

(3) Testing whether a state ρ is inconsistent with a set D of egd's is NP-complete.

Proof:

(1) In [MSY] it is shown that testing whether a relation violates a jd is NP-complete, and in [BV3] it is shown that testing whether a relation violates a typed egd is NP-complete. The claim now follows by Theorem 6.

(2) The claim follows by the above mentioned NP-completeness results in [BV3, MSY] and Theorem 6.

(3) NP-hardness follows from the first claim; we have to show that the problem is in NP. To test for inconsistency, one constructs T_ρ and chases it by D. If at any stage the chase require identifying two constants, then ρ is inconsistent with D. By [BV3] chasing by egd's can be done in nondeterministic polynomial time. □

We now refer to the general case and prove lower and upper exponential time bounds.

Theorem 8: Testing whether a state ρ is consistent with a set D of full dependencies is EXPTIME-complete.

Proof: As observed before, to test for inconsistency, one constructs T_ρ and chases it by D. If at any stage the chase require identifying two constants, then ρ is inconsistent with D. Otherwise, it is consistent. An analysis of the chase in [BV3] shows that it can be done in exponential time. It remains to show that the problem is EXPTIME-hard. We show it by reduction from the implication problem for full td's, which was shown in [CLM] to be EXPTIME-complete. That is, given a set D of full td's and a full td d, we construct in polynomial time a set D' of full dependencies and a state ρ such that $D \models d$ iff ρ is inconsistent with D'.

Let U be the relation scheme for the dependencies $D \cup d$. Let d be $\langle T, w \rangle$, with $T = \{w_1, \ldots, w_m\}$. Without loss of generality assume that there are at least two variables in T. To test whether $D \models d$, we chase T by D and see whether w is generated. The idea of the reduction is to have a state ρ that "looks like" T and a set D' of dependencies that simulate D and in addition force identification of two constants if w is generated. In order to do that we need to mark the tuples in the original state and the tuples that are generated by the chase. The marking is done by equalities satisfied by the tuples. For that we add new attributes: the database scheme is $\mathbf{R} = \{U'\}$, where

$$U' = U \cup \{A, A_1, \ldots, A_m, B, B_1, \ldots, B_m\}.$$

Let α be a one-to-one valuation that maps the variables in T to constants. $\rho(U')$ has tuples $u_1, \ldots, u_m$ that correspond to the tuples $w_1, \ldots, w_m$ in T in the following way:

(1) $u_i[U] = \alpha(w_i)$,

(2) $u_i[A]$ and $u_i[A_i]$ are the same new constant, and

(3) u_i has distinct new constants elsewhere.

Note that, since **R** has a single relation scheme, T_ρ is just $\rho(U')$.

Let now $\langle S, v \rangle$ be a full td in D. We construct a full td $\langle S', v' \rangle$ on U' and put it in D'. For each tuple v_i in S we have a tuple v'_i in S' defined as follows:

(1) $v'_i[U] = v_i$,

(2) v'_i has distinct new variables elsewhere.

v' is defined as follows:

(1) $v'[U] = v$, and

(2) $v'[A, A_1, \ldots, A_m] = v'[B, B_1, \ldots, B_m] = v_1[B, B_1, \ldots, B_m]$.

For example, if $\langle S, v \rangle$ is:

	F	G	H
v:	f	g	h
v_1:	f	g	$h1$
v_2:	f	$g1$	h
v_3:	$f1$	g	g

Then $\langle S', v' \rangle$ is:

	A	A_1	A_2	A_3	B	B_1	B_2	B_3	F	G	H
v':	b	$b1$	$b2$	$b3$	b	$b1$	$b2$	$b3$	f	g	h
v'_1:	.	.	.	.	b	$b1$	$b2$	$b3$	f	g	$h1$
v'_2:	.	.	.	.	.	.	.	.	f	$g1$	h
v'_3:	.	.	.	.	.	.	.	.	$f1$	g	h

(Dot represents variables with unique occurrences).

In addition we put in D' an egd $\langle T',(a_1,a_2)\rangle$, where a_1 and a_2 are two distinct variables from T. T' has tuples $w_1', \ldots, w_m', w'$ that correspond to $w_1, \ldots, w_m, w$.

w'_i is defined as follows:

(1) $w'_i[U]=w_i$,

(2) $w'_i[A]$ and $w'_i[A_i]$ are the same new variable, and

(3) w'_i has new distinct variables elsewhere.

w' is defined as follows:

(1) $w'[U]=w$, and

(2) w' has new distinct variables elsewhere.

For example, if $\langle T,w\rangle$ is:

	F	G	H
w:	f	g	h
w_1:	f	g	$h1$
w_2:	f	$g1$	h
w_3:	$f1$	g	g

Then the constructed egd is $\langle T',(f,f1)\rangle$, where T' is:

	A	B	A_1	A_2	A_3	B_1	B_2	B_3	F	G	H
w'_1:	$a1$	.	$a1$	.	.	.	.	.	f	g	$h1$
w'_2:	$a2$	.	.	$a2$	.	.	.	.	f	$g1$	h
w'_3:	$a3$	.	.	.	$a3$	.	.	.	$f1$	g	h
w':	.	.	.	.	.	.	.	.	f	g	h

(Dot represents variables with unique occurrences).

To prove that $D \models d$ iff ρ is inconsistent with D' we show that a chase of T by D can be simulated by a chase of T_ρ by D' and vice versa.

Consider first a chase of T by D. We claim that for any tuple t generated by a td $\langle S,v \rangle$ in a chase of T by D, one can generate a tuple t' by the td $\langle S',v' \rangle$ in a chase of T_ρ by D' such that:

(1) $t'[U]=\alpha(t)$, and

(2) $t'[A] \neq t'[A_i]$ for $1 \leq i \leq m$.

We leave the verification of this claim to the reader. If $D \models d$, then w is generated by the chase of T by D. Therefore, a chase of T_ρ by D' generates a tuple u such that $u[U]=\alpha(w)$. Let us now apply the egd $\langle T',(a_1,a_2) \rangle$ with a valuation β that maps w'_i to u_i and w' to u. β agrees with α on the variables of T, so we are forced to identify $\alpha(a_1)$ and $\alpha(a_2)$. That means that ρ is inconsistent with D'.

Consider now a chase of T_ρ by D'. We claim that for any tuple t' generated by a td $\langle S',v' \rangle$ in a chase of T_ρ by D', one can generate a tuple t by the td $\langle S,v \rangle$ in a chase of T by D such that:

(1) $t'[U]=\alpha(t)$, and

(2) $t'[A] \neq t'[A_i]$ for $1 \leq i \leq m$.

We leave the verification of this claim to the reader. If ρ is inconsistent with D' then the egd $\langle T',(a_1,a_2) \rangle$ must be applied with some valuation β. But since $w'_i[A]=w'_i[A_i]$, w'_i cannot be mapped by β to any other tuple but u_i. Thus β agrees with α on the variables of T. In particular, $\beta(w')[U]=\alpha(w)$. That is, w' must be mapped to a tuple u generated by the chase of T_ρ such that $u[U]=\alpha(w)$. But then w is generated by a chase of T by D, so $D \models d$.

To complete the proof we note that the reduction from D and d to ρ and D' can be done in polynomial time. □

Corollary 2: For $\mathbf{R}=\{U\}$, testing whether a state ρ is consistent with a set D of full dependencies is EXPTIME-complete. □

The corollary should be contrasted with clause (2) of Theorem 7. While consistency is EXPTIME-complete even for database schemes with a single relation scheme, for completeness there is a complexity gap between the case of database schemes with a single relation scheme and the case of database schemes with two relation schemes.

Theorem 9: Testing whether a state ρ is complete with respect to a set D of full td's is EXPTIME-complete.

Proof: To test for incompleteness, one constructs T_ρ and chases it by D. If at any stage a tuple t is generated such that $t[R_i]$ has no variables and $t[R_i]$ is not in $\rho(R_i)$ for some of the relation schems R_i in the database scheme R, then ρ is incomplete with respect D. Otherwise, it is complete. An analysis of the chase in [BV3] shows that it can be done in exponential time. It remains to show that the problem is EXPTIME-hard. We show it by reduction from the implication problem for full td's which was shown in [CLM] to be EXPTIME-complete. That is, given a set D of full td's and a full td d, we construct in polynomial time a set D' of full td's and a state ρ such that $D \models d$ iff ρ is incomplete with D'.

Let U be the relation scheme for the dependencies $D \cup d$. Let d be $\langle T,w \rangle$, with $T=\{w_1, \ldots, w_m\}$. Without loss of generality assume that w is not in T. To test whether $D \models d$, we chase T by D and see whether w is generated. The idea of the reduction is to have a state ρ that "looks like" T and a set D' of dependencies that simulate D and in addition generate a "forbidden" tuple if w is generated. Unlike the reduction in the proof of Theorem 8, we have to be careful not to generate "forbidden" tuples too early. For that we add new attributes: the database scheme is $\mathbf{R}=\{R_1,R_2\}$, where $R_1=U\cup\{A,B,A_1, \ldots, A_m\}$ and $R_2=\{C,D\}$. The new universe is $U'=R_1 \cup R_2$.

Let α be a one-to-one valuation that map the variables in T to constants. $\rho(R_1)$ has tuples $u_1, \ldots, u_m$ that correspond to the tuples $w_1, \ldots, w_m$ in T in the following way:

(1) $u_i[U]=\alpha(w_i)$,

(2) $u_i[A]$, $u_i[B]$, and $u_i[A_i]$ are the same new constant, and

(3) u_i has distinct constants elsewhere.

$\rho(R_2)$ has a single tuple u_0 such that $u_0[C]$ and $u_0[D]$ are the same new constant.

In T_p, the tuples $u_1, \ldots, u_m$ are extended with distinct new variables for the attributes C and D, and u_0 is extended with unique variables for all attributes other than C and D. The idea is that for every tuple t generated by a chase of T_p, $t[A_1]$ and $t[D]$ are variables, so no "forbidden" tuple is generated until the very last step.

Let now $\langle S,v \rangle$ be a full td in D. We construct a full td $\langle S',v' \rangle$ on U' and put it in D'. For each tuple v_i in S we have a tuple v'_i in S' defined as follows:

(1) $v'_i[U] = v_i$,

(2) $v'_i[A]$ and $v'_i[B]$ are the same new variable, and

(3) v'_i has distinct new variables elsewhere.

In addition S' has a tuple v'_0 defined as follows:

(1) $v'_0[C]$ and $v'_0[D]$ are the same new variable, and

(2) v'_0 has distinct new variables elsewhere.

Finally, v' is defined as follows:

(1) $v'[U] = v$,

(2) $v'[A_1, \ldots, A_m] = v'_0[A_1, \ldots, A_m]$,

(3) $v'[A]$ and $v'[B]$ are the same old variable (any variable from v will do), and

(4) $v'[CD] = v'_1[CD]$.

For example, if $\langle S,v \rangle$ is:

	F	G	H
v:	f	g	h
v_1:	f	g	$h1$
v_2:	f	$g1$	h
v_3:	$f1$	g	h

Then $\langle S',v' \rangle$ is:

	A	B	A_1	A_2	A_3	C	D	F	G	H
v':	f	f	$a1$	$a2$	$a3$	$c1$	$d1$	f	g	h
v'_0:	.	.	$a1$	$a2$	$a3$	c	c	$f2$	$g2$	$h2$
v'_1:	$a4$	$a4$	.	.	.	$c1$	$d1$	f	g	$h1$
v'_2:	$a5$	$a5$	.	.	.	.	.	f	$g1$	h
v'_3:	$a6$	$a6$	.	.	.	.	.	$f1$	g	h

(Dot represents variables with unique occurrences).

In addition we put in D' a full td $\langle T',w'\rangle$. T' has tuples $w'_0, w_1', \ldots, w_m'$ that correspond to $w, w_1, \ldots, w_m$.

w'_0 is defined as follows:

(1) $w'_0[U]=w$, and

(2) w'_0 has new distinct variables elsewhere.

w'_i is defined as follows:

(3) $w'_i[U]=w_i$,

(4) $w'_i[A]$ and $w'_i[A_i]$ are the same new variable, and

(5) w'_i has new distinct variables elsewhere.

Finally, w' is defined as follows:

(1) $w'[U]=w$, and

(2) $w'[A,B,A_1,\ldots,A_m,C,D]=w'_1[A,B,A_1,\ldots,A_m,C,D]$.

For example, if $\langle T,w\rangle$ is:

	F	G	H
v:	f	g	h
v_1:	f	g	$h1$
v_2:	f	$g1$	h
v_3:	$f1$	g	h

Then $\langle T',w'\rangle$ is:

	A	B	A_1	A_2	A_3	D	E	F	G	H
w':	$a1$	$a2$	$a1$	$a3$	$a4$	$c1$	$d1$	f	g	h
w'_0:	.	.	.	.	.	.	.	f	g	h
w'_1:	$a1$	$a2$	$a1$	$a3$	$a4$	$c1$	$d1$	f	g	$h1$
w'_2:	$a5$	.	.	$a5$	.	.	.	f	$g1$	h
w'_3:	$a6$	.	.	.	$a6$	.	.	$f1$	g	h

(Dot represents variables with unique occurrences).

To prove that $D \models d$ iff ρ is incomplete with respect to D' we show that a chase of T by D can be simulated by a chase of T_ρ by D' and vice versa.

Consider first a chase of T by D. We claim that for any tuple t generated by a td $\langle S, v\rangle$ in a chase of T by D, one can generate a tuple t' by the td $\langle S', v'\rangle$ in a chase of T_ρ by D' such that:

(1) $t'[U] = \alpha(t)$,

(2) $t'[A]$ and $t'[B]$ are the same constant,

(3) $t'[A_1], \ldots, t'[A_m], t'[C]$, and $t'[D]$ are distinct variables.

We leave the verification of this claim to the reader. If $D \models d$, then w is generated by the chase of T by D. Therefore, a chase of T_ρ by D' generates a tuple u such that $u[U] = \alpha(w)$. Let us now apply the td $\langle T', w'\rangle$ in the chase of T_ρ with the valuation β that maps w'_0 to u and maps w'_i to u_i. This generates the tuple $\beta(w')$ with:

(1) $\beta(w')[U] = \alpha(w)$, and

(2) $\beta(w')[A, B, A_1, \ldots, A_m, C, D] = u_1[A, B, A_1, \ldots, A_m, C, D]$.

It follows that $\beta(w')[R_1]$ consists solely of constants and is not in $\rho(R_1)$, since w is not in T. So ρ is incomplete with respect to D'.

Consider now a chase of T_ρ by D'. We claim that for any tuple t' generated by a td $\langle S', v'\rangle$ in a chase of T_ρ by D', one can generate a tuple t by the td $\langle S, v\rangle$ in a chase of T by D such that:

(1) $t'[U]=\alpha(t)$,

(2) $t'[A]$ and $t'[B]$ are the same constant, and

(3) $t'[A_1], \ldots, t'[A_m]$, $t'[C]$, and $t'[D]$ are distinct variables.

We leave the verification of this claim to the reader. If ρ is incomplete with respect D' then the td $\langle T', w' \rangle$ must be applied with some valuation β. But, since $w'_i[A]=w'_i[A_i]$, w'_i cannot be mapped to any other tuple but u_i. Thus β necessarily agrees with α on the variables that are in T. In particular, $\beta(w')[U]=\alpha(w)$, so w'_0 must be mapped to a tuple u generated by the chase of T_ρ such that $u[U]=\alpha(w)$. But then w is generated by a chase of T by D, so $D \models d$.

To complete the proof we note that the reduction from D and d to ρ and D' is polynomial.

□

We note that in the proofs of Theorem 8 and 9 we have untyped dependencies in D' even if the dependencies in D are typed. We believe that the exponential lower bounds for consistency and completeness hold also for typed dependencies.

5. Testing Satisfaction under Embedded Dependencies

In the previous section, we restricted our attention to full dependencies in order to obtain decidability results. In this section we return to arbitrary sets of dependencies. Our main result will be that both consistency and completeness are undecidable in this general setting. To show this, we will prove that consistency and completeness are recursively equivalent to certain dependency implication problems which are known to be undecidable.

The first step is to reduce consistency to the implication problem of egd's by a set of dependencies. Let ρ be a state and D a set of arbitrary dependencies. Construct a set of egd's E_ρ as follows. Let $T = \nu(T_\rho)$ be an isomorphic image of T_ρ in which no constants appear. For every pair of distinct constants c and d in T_ρ, $\langle T, (\nu(c), \nu(d)) \rangle$ is an element of E_ρ.

Theorem 10: ρ is consistent with D if and only if for no egd $e \in E_\rho$ is it the case that $D \models e$.

Proof: Suppose ρ is consistent. Let $I \in \text{WEAK}(D,\rho)$. We can construct from ν a homomorphism η with $\eta(T) \subseteq I$ and for each constant c of ρ, $\eta(\nu(c)) = c$. I certainly satisfies D, but it violates each egd in E_ρ. Therefore no element of E_ρ is implic by D.

For the converse, suppose there is no $e \in E_\rho$ such that $D \models e$. Let $E_\rho = \{e_1, \ldots, e_k\}$ and $e_i = \langle T, (\nu(c_i), \nu(d_i)) \rangle$. Construct from T an atomic sentence $\bar{\tau}$ by letting $\bar{\tau}$ be the conjunction of all sentences $U(t)$ such that t is a tuple in T. Now consider the sentence d given by

$$\exists \mathbf{x}(\tilde{\tau} \wedge \nu(c_1) \neq \nu(d_1) \wedge \cdots \wedge \nu(c_k) \neq \nu(d_k)),$$

where $\mathbf{x}$ is a sequence of all the variables in T. We claim that $D' = D \bigcup d$ is finitely satisfiable. Suppose not; then $D \models \neg d$. Now $\neg d$ is a disjunctive egd of the form

$$\forall \mathbf{x}(\tilde{\tau} \rightarrow \nu(c_1) = \nu(d_1) \vee \cdots \vee \nu(c_k) = \nu(d_k)).$$

We now rely on a finite version of a theorem of McKinsey [McK] due to Graham and Vardi [GV] to conclude that for some $1 \leq i \leq k$, $D \models \forall \mathbf{x}(\tilde{\tau} \rightarrow \nu(c_i) = \nu(d_i)$, that is, $D \models e_i$, contradicting our assumption that D does not imply e for any $e \in E_\rho$.

Since D' is finitely satisfiable, it has a finite model M. Let $s(c)$ be the domain element assigned to each variable $\nu(c)$ of d to make d true in M. Note that $s(c_i) \neq s(d_i)$ for every c_i, d_i appearing in an inequality in d, so we can assume without loss of generality that $s(c) = c$. Then M(U) is a weak instance for ρ, showing that ρ is consistent. □

We now reduce the implication problem for egd's to the consistency problem. Let D be any set of dependencies and let $e = \langle T, (a,b) \rangle$ be an egd. We form the set R_e of states of the universal scheme $\{U\}$ as follows. For each mapping ν from the symbols of T to constants such that $\nu(a) \neq \nu(b)$, $\nu(T)$ is a member of R_e.

Theorem 11: $D \models e$ if and only if no state in R_e is consistent with D.

Proof: Suppose $D \models e$, and let $\nu(T)$ be any state in R_e. Clearly $\nu(T)$ violates e. Any weak instance in WEAK$(D, \nu(T)$ must satisfy D, and hence e. But $\nu(T)$ would have to be a subset of such a weak instance, which is impossible. Hence no such weak instance exists and $\nu(T)$ is inconsistent with D.

For the converse, suppose D does not imply e. Let I be any relation that satisfies D but not e. Such a relation must contain a homomorphic image $\nu(T)$ of T, such that $\nu(a) \neq \nu(b)$. Hence I is a weak instance for $\nu(T)$, which is an element of R_e. □

From the last two theorems we obtain the following immediate corollary, relating the decidability of the membership problem for an egd from a set of dependencies and the decidability of consistency under that set of dependencies.

Corollary 3: Let D be a set of dependencies. Let D_e be the set of egd's implied by D. The following are equivalent.

(a) D_g is recursive.

(b) For every database scheme **R** over the universe on which D is defined, the consistency of every state of **R** is decidable.

(c) The consistency of every state of the universal scheme of D is decidable.

Proof: (a) implies (b) follows from Theorem 10; (b) implies (c) is immediate; (c) implies (a) follows from Theorem 11. □

The development of the last two theorems can be repeated to relate completeness to td implication. For the analogue to Theorem 10 construct a set of exponentially many td's, G_ρ, from a state ρ. Elements of G_ρ are of the form $\langle T,w\rangle$, where T is the image of T_ρ under an injection ν to variables, and w is constructed as follows. Let R_i be a relation scheme in the given database scheme, and let t be a tuple on R_i such that t consists of constants taken from ρ but $t \notin \rho(R_i)$. Then $w[R_i]=\nu(t)$ and the rest of w consists of distinct new variables. Informally, each element $\langle T,w\rangle$ of G_ρ says that a containing instance for ρ must contain a tuple w such that its projection on some relation scheme is not in ρ. Note that G_ρ is a set of embedded td's.

Theorem 12: ρ is complete with respect to D if and only if for no element $g \in G_\rho$ is it the case that $D \models g$.

Proof: If there is some $g = \langle T,w\rangle \in G_\rho$ such that $D \models g$, then let R be the relation scheme that led to include g in G_ρ. There is some tuple t constructed with values from ρ that does not appear in $\rho(R)$. By property (3) of $\overline{D}$, we know that if $D \models g$ then $\overline{D} \models g$. Let I be an element of WEAK$(\overline{D},\rho)$. Since I satisfies $\overline{D}$, it satisfies g. Let μ be a valuation such that $\mu(T_\rho) \subseteq I$. Since T is the image of T_ρ under an injection ν and I satisfies g, I must contain some tuple whose projection on R is t. It follows that $t \in \bigcap_{I \in \mathrm{WEAK}(\overline{D},\rho)} \{\pi_R(I)\}$, so ρ is incomplete.

For the converse, suppose that no $g \in G_\rho$ is implied by D. By property (3) of $\overline{D}$, no $g \in G_\rho$ is implied by $\overline{D}$. Thus, for every $g \in G_\rho$, there is a universal relation I_g such that I_g satisfies $\overline{D}$ but not g. Let $g = \langle T,w\rangle$ and let t be the tuple on relation scheme R that led to the inclusion of g in G_ρ. There is a valuation ν such that $\nu(T) \subseteq I_g$ and $\nu(w[R]) \notin \pi_R(I_g)$. By the multiple copies construction of Theorem 2, we can assume that ν is injective. Thus we can assume without loss of generality that $I_g \in$ WEAK$(\overline{D},\rho)$, and $t \notin \pi_R(I_g)$. It follows that $t \notin \rho^+(R)$. Since this is true for every tuple t constructed from values in ρ but not itself in $\rho(R)$, it follows that ρ is complete. □

For the analogue of Theorem 11, let D be a set of dependencies and $g = \langle T,w\rangle$ a td. We may assume $w \notin T$, else g is trivial. Let U be the relation scheme of $D \cup g$, let

$R = \{A \mid w[A]$ occurs in $T\}$, and let $\mathbf{R} = \{U, R\}$. Let ν be an injection from variables of T to constants. Let S be the set of all relations on U constructed from values in $\nu(T)$ that contain $\nu(T)$. Let K be the set of all states of the form $\pi_{\mathbf{R}}(r)$, where $r \in S$ and $\pi_R(r)$ does not contain $\nu(w)$.

Theorem 13: $D \models g$ if and only if every state of K is incomplete.

Proof: Again the proof parallels that of Theorem 11. If $D \models g$, then the completion σ^+ of σ for each $\sigma \in K$ is such that $\nu(w)[R] \in \sigma^+(R)$, but $\nu(w) \notin \sigma(R)$, hence every such σ is incomplete.

Conversely, if D does not imply g, let I be a relation on U that satisfies D but not g. I is a weak instance for the state $\sigma = \pi_{\mathbf{R}}(I)$ in K. Since σ is exactly the projection of one of its weak instances, it must be complete. □

Corollary 4: Let D_t be the set of all td's implied by a set of dependencies D. D_t is recursive if and only if completeness of any state of any database scheme over the attributes of D is decidable. □

We now state the main result of this section, which is a corollary of the four theorems above.

Theorem 14: There does not exist an algorithm which will determine for every pair $\langle D, \rho \rangle$ whether ρ is consistent nor whether ρ is complete with respect to D.

Proof: The implication problem of egd's from arbitrary dependencies was shown undecidable by Vardi [Va1]. Implication of td's was shown undecidable by Vardi [Va3] and Gurevich and Lewis [GL]. □

Since no general algorithms exist for deciding either completeness or consistency, we become interested in solvable subcases. If implication is decidable for D, for example, if D contains only full dependencies, consistency and completeness are decidable, as shown in the previous section. But there may be specific database schemes for which consistency and completeness are decidable, even if implication is not. However, there is no algorithm that will decide, given a database scheme and a set of dependencies, whether consistency and completeness are decidable for that scheme and those dependencies.

Theorem 15: The set $A = \{\langle D, \mathbf{R} \rangle \mid$ consistency and completeness of states of $\mathbf{R}$ with respect to D are decidable $\}$ is not recursive.

Proof: Vardi [Va2] showed that it is undecidable whether the implication problem for a set of dependencies D is decidable or not. Suppose A were recursive. For a fixed set of dependencies D, the predicate of these dependencies has a fixed arity, that is, a fixed set of attributes. There are only finitely many database schemes over this set of attributes. As implication is decidable for D if and only if $\langle D, \mathbf{R} \rangle \in A$ for each of these finitely many R's, an algorithm for membership in A would yield an algorithm to test decidability of the implication problem. So A cannot be recursive. □

6. Discarding the Universal Relation Scheme

The sentences in C_ρ and K_ρ use a predicate letter corresponding to the universal relation scheme for the database. It is interesting to ask whether we can construct a theory with properties similar to Theorems 4 and 5, but using only predicate symbols $R_1, \ldots, R_n$ of the same arities and sorts as the relation schemes, thus avoiding the universal predicate. This amounts to asking whether dependency satisfaction can be expressed in a "local" way, without having to resort to the existence of a universal relation.

There is a special case, the independent schemes, when the question can clearly be answered in the affirmative. Given a set of dependencies D on a database scheme $\{R_1, \ldots, R_n\}$, the *projected dependencies* D_i are all the dependencies that must hold in any relation r_i on R_i such that $r_i = \pi_{R_i}(r)$, where r is a universal relation satisfying D. A state ρ is called *locally satisfying* if every $\rho(R_i)$ satisfies D_i. A database scheme is said to be *independent* if every locally satisfying state is consistent with D. When the database scheme is independent, we can write down the required set of sentences by expressing each dependency in the context of some R_i. For special cases such as functional and multivalued dependencies, projected dependencies can be easily characterized in terms of the original set D, although finding the D_i's is computationally hard [H]. For more general classes of dependencies, we do not even know if the D_i's are finite. In the general case, the results in this section should be viewed as existence proofs for the desired sets of sentences, rather than effective constructions.

Our main result in this section is that in fact it is possible to construct a set of sentences with the desired properties when the database scheme is *weakly cover embedding*. To define this notion, note that the projected dependencies D_i can be viewed as embedded dependencies on U. For D_i defined on R_i, we say a relation on U satisfies D_i if $\pi_{R_i}(I)$ does. We say that a database scheme $\mathbf{R}$ *weakly cover embeds* a set of dependencies D if any state of $\mathbf{R}$ consistent with $\bigcup_{i=1}^{n} D_i$ is consistent with D. In the framework of Section 4, in a weakly cover embedding scheme it suffices to chase using only dependencies local to some relation scheme of $\mathbf{R}$.

It is easy to see that the class of weakly cover embedding schemes contains both the cover embedding or dependency preserving schemes [MMSU] and the independent schemes. Since, for cover embedding schemes, we have $\bigcup_{i=1}^{n} D_i \models D$, such schemes are weakly cover embedding. Since any state consistent with $\bigcup_{i=1}^{n} D_i$ is locally consistent, the independent schemes are weakly cover embedding. A polynomial time algorithm for testing whether a weakly cover embedding database scheme is independent, in the case where all dependencies are fd's, is given in [GY]. Even for this restricted case, no algorithm to test whether a scheme is weakly cover embedding is known.

Given a state ρ of a weakly cover embedding database scheme, we construct a new set of sentences $\mathbf{B}_\rho$ as follows. The language of $\mathbf{B}_\rho$ is the same as the language of A_ρ, except that we do not use the universal predicate letter U. $\mathbf{B}_\rho$ contains four kinds of sentences.

- State axioms: for each R_i and each tuple $t \in \rho(R_i)$, the sentence $R_i(t)$.
- Join-consistency axioms: for each R_i in the database scheme, $\mathbf{B}_\rho$ contains the sentence

$$\forall \mathbf{x}(R_i(\mathbf{x}) \rightarrow (\exists b_1 \cdots b_m)(R_1(\mathbf{v}_1) \wedge \cdots \wedge R_n(\mathbf{v}_n)))$$

where $\mathbf{v}_i = \mathbf{x}$ and the $\mathbf{v}$'s are constructed from values in $\mathbf{x}$ and the b's so that for all $1 \leq p, q \leq n$, if the jth attribute of R_p is the kth attribute of R_q, then $\mathbf{v}_p[j] = \mathbf{v}_q[k]$. Intuitively, the join-consistency axioms, together with the state containment axioms, assert the existence of a join-consistent state that contains ρ.

- Dependencies: for each i, the set of dependencies D_i can be rewritten as a set of first order sentences on R_i.
- Distinctness axioms: as before, these assert that all constants are distinct.

Example 5: We construct $\mathbf{B}_\rho$ for the state of Examples 1 and 4. The universe is $U = \{S, C, R, H\}$, the relation schemes $R_1 = SC$, $R_2 = CRH$, $R_3 = SRH$, and the dependencies $SH \rightarrow R$, $RH \rightarrow C$. The projected dependencies are: $D_1 = \emptyset$, $D_2 = \{RH \rightarrow C\}$, $D_3 = \{SH \rightarrow R\}$.

State axioms:

$R_1(Jack, CS378)$

$R_2(CS378, B215, M10)$

$R_2(CS378, B213, W10)$

$R_3(Jack, B215, M10)$

Join-consistency axioms:

$$(\forall x_1x_2)(R_1(x_1x_2) \rightarrow (\exists b_1b_2)(R_2(x_2b_1b_2) \wedge R_3(x_1b_1b_2)))$$

$$(\forall x_1x_2x_3)(R_2(x_1x_2x_3) \rightarrow (\exists b_1)(R_1(b_1x_1) \wedge R_3(b_1x_2x_3)))$$

$$(\forall x_1x_2x_3)(R_3(x_1x_2x_3) \rightarrow (\exists b_1)(R_1(x_1b_1) \wedge R_2(b_1x_2x_3)))$$

Dependencies:

$$(\forall r_1h_1c_1c_2)(R_2(c_1r_1h_1) \wedge R_2(c_2r_1h_1) \rightarrow c_1 = c_2)$$

$$(\forall s_1r_1r_2h_1)(R_3(s_1r_1h_1) \wedge R_3(s_1r_2h_1) \rightarrow r_1 = r_2)$$

Distinctness axioms:

$$B215 \neq B213$$

$$M10 \neq W10$$

etc.

Theorem 16: For a weakly cover embedding database scheme, $\mathbf{B}_\rho$ is finitely satisfiable if and only if ρ is consistent with D.

Proof: (If) Suppose ρ is consistent, and let $I \in \text{WEAK}(D,\rho)$. Let $r_i = \pi_{R_i}(I)$ for each i. Consider the interpretation for $\mathbf{B}_\rho$ where each constant is mappped to itself and R_i is interpreted as r_i for each i By definition of projected dependencies, each r_i satisfies D_i. Since I is a containing instance for ρ, the r_i's satisfy the state containment axioms. Finally, the r_i's are join-consistent by construction and thus satisfy the join-consistency axioms. It follows that we have a finite model for $\mathbf{B}_\rho$.

(Only if) Suppose $\mathbf{B}_\rho$ is finitely satisfiable. Proceeding as in Theorem 1, we can assume that there is a database state ρ' that satisfies $\mathbf{B}_\rho$. That is, ρ' contains ρ, is join consistent, and $\rho'(R_i)$ satisfies D_i for each i. Let I be a universal relation such that $\pi_{\mathbf{R}}(I) = \rho'$. Clearly I satisfies the

D_i's, so $I \in \text{WEAK}(\bigcup_i D_i, \rho')$, showing that ρ' is consistent with $\bigcup_i D_i$. Since **R** is weakly cover embedding, ρ' is also consistent with D. Since $\rho \subseteq \rho'$, $\text{WEAK}(D,\rho') \subseteq \text{WEAK}(D,\rho)$, so ρ is consistent with D. □

The following example shows that the construction above does not generalize to non-cover embedding schemes; we leave open the question of whether such a set of sentences can be constructed at all for arbitrary schemes.

Example 6: Let $\mathbf{R}=\{AC,BC\}$, $D=\{AB \rightarrow C, C \rightarrow B\}$, $\rho(AC)=\{\langle 0\,1\rangle,\langle 0\,2\rangle\}$, $\rho(BC)=\{\langle 3\,1\rangle,\langle 3\,2\rangle\}$. Note that $D_1=\emptyset$, $D_2=\{C \rightarrow B\}$. It is easy to see that ρ is consistent with $D_1 \bigcup D_2$, but not consistent with D. However, $\mathbf{B}_\rho$ is consistent in this example, since ρ is a join-consistent state that satisfies the local dependencies and hence provides a model for $\mathbf{B}_\rho$.

7. Discussion

We have pointed out that there are two separate sides to the standard notion of dependency satisfaction: consistency and completeness. We view consistency as corresponding to a "lazy evaluation" tactic for constraint maintenance. As long as no violations can be proven, the state is considered legal. The derived tuples not present in the state can be generated on demand, for purposes such as query answering. Note the similarity of this policy to the "deductive databases" approach [GM], where any fact deducible from the stored relations is considered part of the database. Requiring both consistency and completeness corresponds to a constraint maintenance policy that guarantees that all derived tuples will be present in the database at all times. There is a storage-computation tradeoff in the choice of a policy. This tradeoff applies not only to multi-relation databases but also to single relations. Consistency of a relation under a set of, say, fd's and mvd's, is strictly weaker than standard satisfaction.

The combination of our notions of satisfaction with the concept of independence leads to interesting questions. For example, what are the database schemes such that every locally consistent state is consistent and complete? Chan and Mendelzon [CM] have characterized these schemes when the join dependency for the database scheme and a set of functional dependencies are given.

References

[ABU] Aho, A.V., Beeri, C., Ullman, J.D., "The theory of joins in relational databases", *ACM Trans. on Database Systems* 4(1979), 297-314.

[ASU] Aho ,A.V., Sagiv, Y., Ullman, J.D., "Equivalence Among Relational Expressions," *SIAM J. Computing* 8:2 (1979), 218-246.

[BR] Beeri, C., and Rissanen, J., "Faithful Representations of Relational Database Schemes," IBM Research Report RJ2722, 1980.

[BV1] Beeri, C., and Vardi, M.Y., "A Proof Procedure for Data Dependencies," The Hebrew Univ., Dept. of Computer Science, Dec. 1980.

[BV2] Beeri, C., and Vardi, M.Y., "The Implication Problem for Data Dependencies," Proc. 8th ICALP, in *Lecture Notes in Computer Science* 115, Springer-Verlag, 1981, 73-85.

[BV3] Beeri, C., and Vardi, M.Y., "On the Complexity of Testing Implications of Data Dependencies," The Hebrew University, Dept. of Computer Science, Dec. 1980.

[CLM] Chandra, A.K., Lewis, H.R., Makowsky, J.A., "Embedded implicational dependencies and their inference problem", Proc. 13th ACM Ann. Symp. on Theory of Computing, 1981, 342-354.

[Co] Codd, E.F., "Further normalization of the data base relational model", in *Data Base Systems* (R. Rustin, ed.), Prentice-Hall, N.J., 1972, pp. 33-64.

[CM] Chan, E.P.F., and Mendelzon, A.O., "Independent and Separable Database Schemes," Proc. ACM PODS Symp., 1983, 288-296.

[F] Fagin, R., "Horn Clauses and Database Dependencies," *JACM* 29:4 (1982), 952-985.

[G] Graham, M.H., "Satisfying Database States," Report CSRG-137, Univ. of Toronto, December 1981.

[GJ] Garey, M.R., Johnson, D.S., *Computers and Intractability*, Freeman, San Francisco, 1979.

[GL] Gurevich, Y., and Lewis, H.R., "The Inference Problem for Template Dependencies," Proc. ACM PODS Symp., 1982, 221-229.

[GM] Gallaire, H., and Minker, J. (eds.), *Logic and Data Bases*, Plenum Press, New York, 1978.

[GV] Graham, M.H., and Vardi, M.Y., "On the Complexity and Axiomatizability of Consistent Database States," unpublished manuscript, 1983.

[GY] Graham, M.H., and Yannakakis, M., "Independent Database Schemas," Proc. ACM PODS Symp., 1982, 199-204.

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

[M] Mendelzon, A.O., "Database States and their Tableaux," Proc. XP2 Workshop on Relational Database Theory, 1981.

[McK] McKinsey, J.C.C., "The Decision Problem for some Classes of Sentences without Quantifiers," *J. of Symbolic Logic* 8 (1943), 61-76.

[MMS] Maier, D., Mendelzon, A.O., and Sagiv, Y., "Testing Implications of Data Dependencies," *ACM Trans. on Database Systems* 4(1979), pp. 455-469.

[MMSU] Maier, D., Mendelzon, A.O., Sadri, F., and Ullman, J.D., "Adequacy of Decompositions of Relational Databases," In *Advances in Database Theory* (H. Gallaire, J. Minker, and J.M. Nicolas, eds.), Plenum Press, 1981, 101-114.

[MSY] Maier, D., Sagiv, Y., and Yannakakis, M., "On the Complexity of Testing Implications of Functional and Join Dependencies," *JACM* 28:4 (1981), 680-695.

[Ni] Nicolas, J.M., "First Order Logic Formalization for Functional, Multivalued and Mutual Dependencies," Proc. ACM SIGMOD Conf., 1978, 40-466.

[Ri] Rissanen, J., "Theory of relations for databases - a tutorial survey," Proc. 7th Symp. on Math. Found. of Computer Science, Poland, 1978, Lecture Notes in Computer Science 64, Springer-Verlag, 537-551.

[S] Sagiv, Y., "Can We Use the Universal Instance Assumption without Using Nulls?", Proc. ACM SIGMOD Conf., 1981, 108-120.

[V] Vassiliou,Y., "A Formal Treatment of Imperfect Information in Database Management," Ph.D. Thesis, University of Toronto, 1980.

[Va1] Vardi, M., "The Implication Problem for Data Dependencies in Relational Databases," Ph.D. Thesis (in Hebrew), The Hebrew University in Jerusalem, 1981.

[Va2] Vardi, M.Y., "Global Decision Problems for Relational Databases," Proc. 22nd IEEE Symp. on FOCS, 1981, pp.198-202.

[Va3] Vardi, M.Y., "The Implication and Finite Implication Problems for Typed Template Dependencies," Proc. ACM PODS Symp., 1982, 230-238.

[Y] Yannakakis, M., "Algorithms for Acyclic Databases," Proc. 7th. VLDB Conf., 1981, 82-94.