October 1982

Report No. STAN-CS-82-940

PB96-150842

The Equivalence of Universal Relation Definitions

by

Jeffrey D. Ullman, Moshe Y. Vardi and David Maier

Department of Computer Science

Stanford University
Stanford, CA 94305

THE EQUIVALENCE OF UNIVERSAL RELATION DEFINITIONS

David Maier†
Oregon Graduate Center

Jeffrey D. Ullman‡
Moshe Y. Vardi††
Stanford Univ.

ABSTRACT

The universal relation model aims at achieving complete access path independence by relieving the user of the need for logical navigation among relations. It assumes that for every set of attributes there is a basic relationship that the user has in mind. Two fundamentally different approaches to the universal relation model have been taken. The first approach sees the universal relation as a user view, about which he poses queries. Specifically, a representative instance is constructed, and queries are answered based on its non-null part. The second approach sees the model as having query-processing capabilities that relieve the user of the need to specify the logical access path. The relationship between the user's view and the computation answering a query is a central issue that systems supporting a universal view of data must handle.

We introduce "lossless" and "monotone" expressions and show that the representative instance construction has these properties. Also, every lossless monotone expression produces a result that is a subset of what the representative instance produces. We show that the existence of any first-order formula to simulate the representative instance is equivalent to a "boundedness" condition on the dependencies defining the database scheme. In addition, whenever there is a first-order formula to simulate the representative instance, then we can do so with an expression of simple form: the union of tableau mappings. We close with a discussion of some of the problems with the representative instance approach that suggest better universal relation models may be possible.

I. Underlying Assumptions

We assume the reader is familiar with relational database terminology to the extent covered in <Mal, U1> and with the idea of implementing a universal relation view of data as discussed in those works and <KS, MW, M*, U2>.

Goals of Universal Relation Database Systems

A primary justification used by Codd for the introduction of the relational model was his view that earlier models were not adequate to the task of boosting the productivity of programmers <C1, C2>. One of his stated motivations was to free the application programmer and the end user from the need to specify access paths (the so-called "navigation problem"). A second motivation was to eliminate the need for program modification to accommodate changes in the database structure, i.e., to eliminate access path dependence of programs.

† Supported by NSF grant IST-81-04834 and AFOSR grant 80-0212 in accordance with NSF agreement IST-80-21358. Some of this material was developed while the first author was at SUNY, Stony Brook.
‡ Supported by AFOSR grant 80-0212 in accordance with NSF agreement IST-80-21358.
†† Supported by a Weizmann fellowship; a Fulbright award, and NSF grant DCS-80-12907.

Though being a significant step forward, the relational model by itself fails to achieve complete freedom from user-supplied navigation and from access path dependence. That is, the relational model was successful in removing the need for *physical navigation*; no access paths need to be specified within the storage structure of a relation. However, the relational model has not yet provided independence from *logical navigation*, since access paths among the relations must still be specified.

For example, consider a database that has relations ED(Employee, Department) and DM(Department, Manager). If we are interested in the relationship between employees and managers through departments, then we have to specify the natural join of the ED and DM relations, projected onto EM. This expression is an access path specification, and if the database were reorganized to have a single relation EDM, then the program would have to be modified accordingly.

The universal relation model aims at achieving complete access path independence by letting us ask the system in an appropriate language "give me the relationship between employees and their managers," expecting the system to figure out the correct access path for itself.† Of course, we cannot expect the system always to select the correct relationship between employees and managers automatically, because the user might have something other than the simplest connection, through departments, in mind, e.g., the manager of the manager of the employee, or the manager(s) of all departments that come alphabetically later than the department of the employee. We shall, in a universal relation system, have to settle for eliminating the need for logical navigation along the most direct paths, while allowing the user to navigate in more convoluted ways explicitly.

Unlike the relational model, the universal relation model was not introduced as a clearly defined model, but rather evolved during the 1970's through the independent work of several researchers. Moreover, these researchers were not only concerned with the universal relation as a data model, but some, like <B> and <BBG>, saw the concept primarily as a vehicle to discuss interrelational data dependencies. The issues of dependency satisfaction are rather different from those of supporting a user view, and the multiplicity of assumptions led to considerable confusion and to some attacks on the universal relation model (<K, AP>) that we regard as not germane to the subject.

Synopsis of Paper

We shall try in this paper to unify and clarify the various "universal relation assumptions," as they pertain to data modeling and the support of a user view. We first indicate the assumptions that are so fundamental to the universal relation model that they are common to all the different approaches to the model. Then,

† A similar approach was taken in the RENDEZVOUS system, which generates possible access paths and lets the user choose the desired one <C*>.

we consider what we regard as the two basic approaches to the model. The first approach sees the universal relation as a user view, about which he poses queries. Thus, in order to answer those queries we must first define the semantics of this user view. The second approach sees the model as having a query processing ability that enables the user to pose queries about the actual database relations (rather than some abstract universal relation), without specifying an access path. Thus, in order to answer queries, we must first define a computational procedure that produces the desired answer; that procedure includes inferring the access path.

The main technical contribution of this paper is an exploration of the relationship between these two basic approaches. We establish broad conditions under which the approaches are the same, i.e., the result of the query on the abstract universal relation is equivalent to a computation in relational algebra.

The Universal Connection Assumption

Perhaps the most basic assumption is that there is a *universal relation scheme*, a set of attributes about which queries may be posed. Further, attributes in this set are assumed to play only one "role," and puns are not allowed. Thus, an attribute like NAME cannot stand for names of employees, customers, suppliers, and managers in the same universal relation scheme.

A seldom acknowledged assumption, but one that underlies all known universal relation systems, is that query processing consists of two steps.

1. *Binding.* From the set of attributes X mentioned in the query, form a relation $[X]$, called the *connection* of X, over set of attributes X. Technically, $[X]$ is a function from database states d to relations $[X](d)$. We shall use $[X]$ to stand for the relation $[X](d)$ when the database state d is understood or irrelevant. We allow the possibility that $[X]$ is the empty function, i.e., no connection over set of attributes X is permitted.
2. *Evaluation.* Whatever operations must be applied to answer the query are then applied to $[X]$.

The binding and evaluation phases are independent. Different functions $[X]$ can be used to produce different relations over X, without changing the way evaluation works on the resulting relation, although the answer may, of course, be changed.

For example, the queries

```
retrieve (EMP)
    where MGR="Jones"
```

and

retrieve (MGR)

where EMP="Smith"

are each answered by forming from the database some relation r over (EMP, MGR) in the binding phase. For the evaluation phase, in the first case, we select from r those tuples with MGR="Jones" and project onto EMP. In the second case, we select for EMP="Smith" and project onto MGR.

Underlying the assumptions that queries may be answered in this two step way is the assumption that for all sets of attributes X (or at least for many of them) there is a unique relationship on this set X that the user has in mind. That does not mean there can be only one relationship on X, but rather, one relationship is the most basic one, so we can assume that this relationship is what the user has in mind unless he explicitly specifies otherwise. In the above example of employees, departments and managers, the most basic relationship between managers and employees is that of "manages," while the relationship "manages the manager of" we intuitively feel is less basic. This underlying assumption is called the *relationship uniqueness assumption*. The origin of the concept is in the "window" concept of <Ma2>.

In practice, systems such as <U2, KS, MW, M*> that support a universal relation view of data permit queries with several tuple variables, each of which may range over a separate "copy" of the universal relation. In that case, there is one set of attributes associated with each tuple variable, and for each such set, X, we allow the corresponding tuple variable to range over $[X]$.

The One Flavor Assumption

There is another rather fundamental assumption that underlies much of the work on universal relation systems. Unfortunately, this assumption seems impossible to derive from more basic principles. It has roughly the intellectual status of a belief like "entity-relationship diagrams are adequate to model the real world." That is, there is substantial empirical evidence for the assumption, but we know of no deep reason why it must be valid.

Our assumption, called the *one flavor assumption*, is that all tuples in $[X]$ represent the same "flavor" of relationship among the attributes in X. That is, the meaning to the user of the fact that tuple t is in $[X]$ does not depend on the details of the construction that put it in $[X]$. Another way to state this assumption is that we must, when selecting the attributes in the universal relation scheme and picking an algorithm to compute $[X]$, arrange that the same attribute does not play two different "roles" in one relation $[X]$.

Evidently, "flavor" and "role" are defined only intuitively. We regard it as a fundamental hypothesis of universal relation systems that it is always† possible to rename attributes so that the single-flavoredness of

† or at least sufficiently often that it is worth trying to implement a universal relation support system.

any relationship will be apparent to the user. Perhaps an example will make these ideas clearer.

Example 1: Suppose we have a universal relation scheme ABC with relations AB, BC, and AC. According to the functional dependencies assumed, different systems such as <U2, KS> would make different responses to a query with $X = AB$. The two responses we are most likely to get from a universal relation system are

i) Apply the query to the AB relation; that is, $[AB]$ is simply the relation AB.

ii) Take the union of the connection in AB and the connection through C, that is,

$$[AB] = AB \cup \pi_{AB}(AC \bowtie BC)$$

It appears that neither is right all the time, and the question of which is right hinges on whether tuples in AB are of the same "flavor" as tuples in $\pi_{AB}(AC \bowtie BC)$. We shall consider two interpretations for ABC, and try thereby to illustrate the distinction.

Suppose that A = COURSE, B = STUDENT, and C = ENROLLMENT. That is, we may imagine that a relation (COURSE, STUDENT), representing graduate courses, was at some time merged with a network database representing the many-many relationship between undergraduate courses and students by means of dummy ENROLLMENT records, each of which is owned by a STUDENT record and a COURSE record. Then we intuitively feel that the student-course pairs obtained from the (STUDENT, COURSE) relation are of the same flavor as pairs that are related via the (STUDENT, ENROLLMENT) and (ENROLLMENT, COURSE) relations. That is, whichever the source, the student is taking the course. Thus, in this case we would prefer interpretation (*ii*) in response to a query like

retrieve (COURSE)
where STUDENT = "Jones"

The response according to choice (*ii*) would be all courses taken by Jones, not just the graduate courses.

Now consider an interpretation of ABC where A = STUDENT, B = LEVEL, and C = COURSE. The (STUDENT, LEVEL) relation gives the level (Freshman, etc.) of each student, the (STUDENT, COURSE) relation tells what courses the student is taking, and the (COURSE, LEVEL) relation gives the nominal level of each course.

Here, we believe that the proper response to the query

retrieve (LEVEL)
where STUDENT = "Jones"

is given by (*i*), that is, just tell what level Jones is on, not the set of levels of Jones and all his courses.

Note that the two queries above are both of the same form: retrieve (B) where A = constant. We feel the difference in proper interpretation is due to the fact that (student, level) pairs meaning the student is

at that level are not of the same flavor as pairs meaning the student is taking a course at that level. In more conventional terms, the attribute LEVEL is "semantically overloaded," and should be split into two attributes, say STUDENT_LEVEL and COURSE_LEVEL. □

The one flavor assumption is a generalization of an assumption made by <B> to the effect that attribute names were split adequately so that there would not be two different functional relationships between sets of attributes. That is, if a functional dependency $X \rightarrow Y$ holds in a database scheme, and different ways of deriving this dependency yield different values y_1 and y_2 for the attributes of Y associated with value x for the attributes of X, then surely the tuples xy_1 and xy_2 over the set of attributes XY cannot be regarded as of the same flavor in any reasonable sense. However, there could be one-flavor violations that do not involve FD's. For example, the (COURSE, LEVEL) relation in Example 1 could allow courses to be at several levels, such as Senior/Grad. The FD COURSE→LEVEL disappears, but the one-flavor violation remains.

II. Universal Relations as the User View

The other notions that have at times been referred to as "the universal relation assumption" fall into two categories. First come assumptions that imply data is treated as if it were all in a single relation over all the attributes. Presumably, $[X]$ will be the projection of this relation onto X. The second group consists of various assumptions about how $[X]$ is to be calculated, without explicit reference to a universal relation. The relationship between the two forms of definition is important because the first group, defining a user view, is intellectually justifiable, while the second group provides the efficient computation needed to respond to the user in the way he expects based on his view. In general, computing the view explicitly is far too expensive, and we must resort to computing $[X]$ only in response to a query about X.

We shall deal with the first group in this section. Historically, the first "assumption" in this class was the pure universal relation assumption. By this approach, we restrict ourselves to cases where there exists a relation u over the universal set of attributes U, such that, for every relation scheme R in the database scheme, the current relation for R is $\pi_R(u)$. In this case we take u as the user view, and $[X]$ is simply $\pi_R(u)$.

This approach has the advantage that we can view the database as a physical representation of a single universal relation, and it facilitates dealing with the semantic constraints on the database. In fact, it was exactly for that reason that the pure universal relation was implicitly taken for the first time by Bernstein <B>, when he developed a design theory for relational databases with FD's.

Clearly, in order for this approach to work, we have to ensure that the universal relation is unique, and that we have an effective way of computing it. These issues were investigated in <BR, MMSU, Ri, V1>. However, even with these issues solved, it was widely accepted that the pure universal relation approach is

not widely enough applicable <BBG>. In fact, in <HLY> it is shown that testing whether a database satisfies the assumption or whether an update maintains the assumption is NP-complete, i.e., probably exponential in the size of the database.

A much more promising assumption, which has become known as the *weak universal relation* assumption, is that an appropriate universal relation to serve as a user view is any relation u over U such that

1. u satisfies whatever dependencies are given, and
2. $\pi_R(u)$ is a superset of the current relation for R in the database, for each relation scheme R.

Since we cannot know which of the infinity of weak universal relations truly represents the "real world" at any given moment, one assumes that the only facts that can be deduced about the universal relation from the given relations of the database are those that hold in all weak universal relations. That is, we take the user's view to be a collection of sets of tuples, one for each set of attributes X. Let S_X be the set of tuples for set of attributes X. In S_X appear exactly those tuples t such that for every weak instance u, there is a tuple t' in u that agrees with t on X.

Weak instances were first studied by <H> as a means to define satisfaction of functional dependencies by a collection of relations. They have since been studied by <Me> as a way to define information content in relational database schemes and by <Sa1, Sa2, Y> as a model of what the user should see as the universal relation about which he is to pose queries.

An important property of weak instances is that as long as the dependencies are of a type for which the chase process (<ABU, MMS>) is a partial decision procedure, even dependencies as general as embedded implicational dependencies <F, BV, YP>, we can construct from a set of database relations a single relation that embodies the information present in all weak instances. The desired relation, which is the relation S_X mentioned above, is constructed as follows.

1. We construct a relation u over U. To begin, for each relation r over one of the database relation schemes, say R, and for each tuple t in r, place in u a tuple that agrees with t on the attributes of R, and that in the other attributes has a new "null" symbol appearing nowhere else. We use $\perp_i$ for nulls.
2. Apply dependencies to "chase" u, that is, generate new tuples and equate symbols, as required by the dependencies. However, when equating a null and nonnull symbol, replace the null by the nonnull. For certain kinds of dependencies, in particular full dependencies, where no new symbols are generated by the chase, this step terminates. However, we introduce new nulls when embedded dependencies are applied, so the process may not terminate if there are embedded dependencies. In this case the result of the present step should be taken as the infinite relation that results from chasing "forever."
3. If during the chase process, we are ever forced to equate two symbols, neither of which is null, then

the process stops. We interpret this situation as saying that the actual relations in the database do not satisfy the given dependencies. To be exact, in this case there is no universal relation that satisfies the dependencies and produces supersets of the database relations when projected onto the schemes of those relations; that is, there is no weak instance.

The relation constructed as above from a list of database relations $r_1, \ldots, r_n$, is called the *representative instance* for these relations (with respect to the given dependencies), and we denote it by $\mathrm{RI}(r_1, \ldots, r_n)$. The weak universal relation assumption says that the representative instance is a suitable model of the data as stored in one relation. The representative instance differs from a pure universal relation in that the latter consists only of total tuples. In contrast, the representative instance extends tuples of one relation to have nonnull values in whatever components are justified by the dependencies. However, in general, the representative instance consists of relationships defined on subsets of the universal set of attributes—subsets that are as large as make sense.

Whenever a tuple of u has nonnull symbols in the components for set of attributes X, these values are presumed to be related in a significant way, and therefore belong in $[X]$. Put another way, if we let $\pi{\downarrow_R}(u)$ stand for the projection of u onto R after throwing away all tuples that have nulls in one or more of the components corresponding to the attributes in R, then the weak universal relation assumption says that $[X] = \pi{\downarrow_X}(u)$, where u is the representative instance defined above. We call $\pi\downarrow$ the *restricted projection*.

As this way of defining connections is only one of many possibilities, we should, strictly speaking, use a special notation for this definition. We shall use $[X]^\Delta$ to denote the set of tuples obtained by computing the representative instance using set of dependencies Δ, and then performing the restricted projection onto X. A fundamental property of the representative instance is that it always produces the intersection of all the weak instances, that is, the sets S_X consisting of those tuples that are in the projection of every weak instance onto X. While this relationship is generally believed, we provide a proof in the following theorem.

Theorem 1: For all X, $[X]^\Delta = S_X$.

Proof: Since $\mathrm{RI}(r_1, \ldots, r_n)$ is a weak universal relation, it is clear that $S_X \subseteq [X]^\Delta$. It remains to prove the other direction.

Let u be the universal relation constructed in the beginning of the construction of the representative instance. That is, u is constructed by taking all tuples in the database and padding them with distinct nulls. Now let v be any weak universal relation for the database. That is, v satisfies the dependencies in Δ, and $\pi_R(u)$ is a superset of the current relation for R in the database for each relation scheme R. It is easy to see that we can define a mapping h on the entries in u that is the identity on all entries that come from the

E	C	D	M
Jones	Ann	$\perp_1$	$\perp_2$
Jones	Jim	$\perp_3$	$\perp_4$
Green	Sue	$\perp_5$	$\perp_6$
Jones	$\perp_7$	Shoes	$\perp_8$
Smith	$\perp_9$	Toys	$\perp_{10}$
$\perp_{11}$	$\perp_{12}$	Toys	Green

Fig. 1. Initial table.

database (i.e., the non-null entries), such that $h(u) \subseteq v$. Now we can show by induction on the chase steps, as in <BV, SU, MMS>, that application of the dependencies preserves this statement. That is, let u' be a universal relation in an intermediate step of the algorithm. Then we can define a mapping h on the entries in u' that is the identity on all entries that come from the database, and such that $h(u') \subseteq v$.

It follows that we also have such a mapping h for which $h(\mathrm{RI}(r_1, \ldots, r_n)) \subseteq v$. Now let t be a tuple in $RI(r_1, \ldots, r_n)$ with non-null entries in the components corresponding to attributes in X. Then we have that $h(t)$ is in v and $h(t)[X] = t[X]$. It follows that $t[X]$ is in S_X. Thus, $[X]^\Delta \subseteq S_X$. □

Example 2: Suppose we have attributes E (employee), C (child), D (department), and M (manager), with relation schemes EC, ED, and DM with current values

E	C
Jones	Ann
Jones	Jim
Green	Sue

E	D
Jones	Shoes
Smith	Toys

D	M
Toys	Green

Suppose that the given dependencies are $E \twoheadrightarrow C$, $E \rightarrow D$, and $D \rightarrow M$.† The initial value of u constructed by step (1) above is shown in Fig. 1.

We could apply $E \twoheadrightarrow C$ to deduce that the tuple

(Jones, Ann, $\perp_3$, $\perp_4$)

was in u. However, we would later discover that this tuple contains the same information as the first tuple in Fig. 1, so we shall instead apply the FD's, discovering after we do that the MVD $E \twoheadrightarrow C$ is satisfied as a result (which must be the case because the two FD's logically imply the MVD here). Thus, $E \rightarrow D$ applied to rows 1, 2, and 4 of Fig. 1 tells us that $\perp_1 = \perp_3 =$ Shoes. Then $D \rightarrow M$ tells us that $\perp_{10} =$ Green,

† The first of these is redundant and is included only to illustrate a point.

E	C	D	M
Jones	Ann	Shoes	$\perp_2$
Jones	Jim	Shoes	$\perp_2$
Green	Sue	$\perp_5$	$\perp_6$
Jones	$\perp_7$	Shoes	$\perp_2$
Smith	$\perp_9$	Toys	Green
$\perp_{11}$	$\perp_{12}$	Toys	Green

Fig. 2. The representative instance.

and $\perp_2 = \perp_4 = \perp_8$; we shall replace them all by $\perp_2$. At this point no further changes can be made. No nonnull symbols were equated, so the data is deemed to satisfy the given dependencies. Figure 2 shows the resulting representative instance. It serves as an adequate universal relation, telling us all the facts that can be deduced from the given data and the dependencies. For example, we know that Ann is the child of someone in the Shoe Department, because $[CD] = \{$(Ann, Shoes), (Jim, Shoes)$\}$. □

Representative Instances and Logical Theories

Another way to look at the weak universal relation approach is to view the database as a logical theory. From this point of view (<GaMi, Ko , Re>) a database is a set of sentences in a first order language without function symbols, whose relation names denote the relations of the database and whose constants denote the elements of the domain of the database. Let T be such a theory. Then, in answer to a query about a relation R, we must produce the set of tuples $\{t \mid T \models R(t)\}$, i.e., the set of all tuples whose membership in the relation for R is implied by the theory. This approach has the advantage of working even in the case that the given constraints are not dependencies, so the chase would not applicable.

Let us now see what the theory T is in our case. Our construction is similar to the construction in <GrMe>, though they had a somewhat different intention in mind. The language we use has a relation name X for every attribute set X. In particular, it has a relation name R for every relation scheme R in the database scheme, and it has the universal relation name U. For constants we use the elements of the database, which denote themselves.

The theory has five kinds of sentences. First, we have a set DB of atomic sentences describing the relations in the database. That is, for every tuple t in some relation r over relation scheme R, we have in DB the sentence $R(t)$. Secondly, we have a set INC of sentences saying that the relation r for a relation scheme R is included in the projection of the universal relation on R. For example, suppose that $U = ABCD$ and

$R = BCD$. Then put in INC the sentence

$$(\forall b)(\forall c)(\forall d)(\exists a)(R(b,c,d) \rightarrow U(a,b,c,d))$$

Thirdly, we have a set CON of sentences saying that the relation x for an attribute set X contains the projection of the universal relation on X. In the above example we have the sentence

$$(\forall a)(\forall b)(\forall c)(\forall d)(U(a,b,c,d) \rightarrow R(b,c,d))$$

In addition we have the set DIS stating that all elements are distinct, i.e, for each pair of distinct elements a and b we have the sentence $a \neq b$. Finally, we have Δ, which is the given set of dependencies written as first-order sentences (<BV, F>) with the universal relation name as the only relation name. The theory T is taken to be the union

$$DB \cup INC \cup CON \cup DIS \cup \Delta$$

Consider now a model of this theory. Since all constants must be interpreted as distinct elements (by the sentences in DIS), we can assume that they are interpreted as themselves. Let r be a relation in the database for the relation scheme R, and let r' be the interpretation of R in the model. For every tuple t in r we have a sentence $R(t)$ in DB, so t must belong to r'. Let u be the interpretation of the universal relation U. By the sentences in INC, we have $r' \subseteq \pi_R(u)$, and consequently, $r \subseteq \pi_R(u)$. Also, u satisfies Δ. It follows that u is a weak universal relation for the database. Conversely, given a weak universal relation u for the database, we can construct a model for the theory, by taking u as the interpretation of U, and by taking $\pi_X(u)$ as the interpretation of X for each attribute set X. We have just proven:

Theorem 2: A database satisfies the given dependencies if and only if its theory is satisfiable. □

Now, given an attribute set X, we define $[X]_T$ to be $\{t \mid T \models X(t)\}$. The next theorem shows that $[X]_T$ is the desired relation.

Theorem 3: $[X]_T = S_X$.

Proof: We have seen that every model of T corresponds to a weak universal relation for the database and vice versa. Let t be a tuple in S_X. Then in every weak universal relation u there is a tuple t' such that $t = t'[X]$. Thus, in every model of T there is such a tuple t'. By the sentences in CON, in every model of T, t is contained in the interpretation of X. Thus, $T \models X(t)$, and t is in $[X]_T$. The opposite direction is similar. □

III. Computational Definitions of the Universal Relation

Now we come to a variety of assumptions that start from the point of view that the user may query about

any set of attributes X, and the system will perform some computation on the relations of the database to compute $[X]$. This approach was originally taken by <B>, where it is implicitly assumed that $[X]$ is computed by joins simulating functional dependencies. It was taken again in <ABU>, where it was assumed that $[X]$ should be computed by a natural join. There, the notion of a "correct join" was identified with the notion of lossless join; in fact the lossless join seems to be the basic computational procedure in all works taking the computational approach.

The first implemented system to behave this way, the UNIX command q <Ah>, uses a list of sets of attributes; the list is established once and for all, for each database. $[X]$ is computed by searching the list for the first set of attributes Y that includes X, computing a relation over Y in some specified way, and projecting onto X. If no set on the list is found, then the join of all the relations is taken and projected. Several papers have investigated ways to find "correct" joins, and if possible, "optimal" joins to compute $[X]$ for a given attribute set X. See for example <Ar, L, V2>.

Most other proposals compute $[X]$ by taking the union of one or more terms, each of which is a lossless join. (In practice, the sets of relations on the list in a q application are likely to be lossless as well.) For example, <O, Sa1, Sa2, KS> discuss taking the union of extension joins as a way to compute $[X]$. The paper <MU> proposes taking the union of joins each of which lives inside some "maximal object," where the join is not only lossless, but where the losslessness follows from particular rules, like the FD or MVD rules for testing lossless joins. Variants of this approach have been implemented in System/U <U2> and PITS <MW, M*>.

The idea of using for $[X]$ a union of projections of lossless joins can be generalized considerably. For example, the System/U algorithm, since it requires a lossless join of "objects," which may be proper subsets of relations, really uses a union of projections of lossless joins of projections of relations.

Lossless Expressions

Suppose $E(R_1, \ldots, R_n)$ is any expression whose operands are relation schemes $R_1, \ldots, R_n$ that are subsets of some universal scheme U. Suppose also that the result of E is a relation over set of attributes $X \subseteq U$. We say E is *lossless* with respect to a set of dependencies Δ if for each relation u over U that satisfies Δ, when we substitute for each operand R of E the relation $\pi_R(u)$, the value of E is a subset of $\pi_X(u)$. For the usual sorts of expressions we deal with, such as joins and the "tableau mappings" to be defined formally later, it is easy to show that containment always holds in the opposite direction, so "is a subset of" could be replaced by "equals" in the definition of losslessness.

Example 3: Suppose we have a universal relation scheme $EDOP$, representing employees, departments,

offices and phones. We shall assume the FD $E \rightarrow D$, but no other dependencies except the join dependency

$$\bowtie(ED, EO, EP, OP)$$

that we suppose corresponds to the formal definition of the universal relation according to the style of <FMU>. We shall, for this example, take the relation schemes to be EDP, EO, and OP.

Suppose we take $X = DO$, that is, we are interested in the department–office relationship. One lossless expression we might use is the join of all the relations projected onto DO, that is

$$\pi_{DO}(EDP \bowtie EO \bowtie OP)$$

The losslessness of this expression follows from the given join dependency.

A simpler lossless expression is $\pi_{DO}(EDP \bowtie EO)$. The losslessness of this expression follows from the FD $E \rightarrow D$, and it is important to note that $EDP \bowtie EO$ is not a lossless join. That is, to test the losslessness of $\pi_{DO}(EDP \bowtie EO)$ we must check whether it is contained in the expression $\pi_{DO}(EDOP)$, using the test of <ASU>. The tableaux for the two expressions are

E	D	O	P
	d	o	
e	d		p
e		o	

and

E	D	O	P
	d	o	
	d	o	

respectively. We here and throughout the paper use blanks for symbols that appear nowhere else in the given tableau.

The first of these tableaux can be "chased" using the FD $E \rightarrow D$, yielding

E	D	O	P
	d	o	
e	d		p
e	d	o	

whereupon the containment $\pi_{DO}(EDP \bowtie EO) \subseteq \pi_{DO}(EDOP)$ follows, since the row of the tableau for $\pi_{DO}(EDOP)$ can map into the second row of the above tableau.

A third lossless expression we might use is $\pi_{DO}(\pi_{ED}(EDP) \bowtie EO)$. We can show this expression to be lossless by an argument similar to the one used above. □

In fact, we shall later prove a general rule for testing the losslessness of expressions, such as those in Example 3, that are representable as tableaux. Simply chase the tableau, and see if a row with all the distinguished symbols is created. Note how this rule generalizes the lossless join test of <ABU>, since there we want a row with the distinguished symbol in every column. In the present case, we do not care about symbols not in the X-columns, where X is the scheme for the result of the expression.

Monotone Expressions

There is another condition on expressions that plays a role in characterizing computational ways to define universal relations. Say an expression $E(R_1, \ldots, R_n)$ is *monotone* if whenever $r_i \subseteq s_i$ for $1 \leq i \leq n$, it follows that $E(r_1, \ldots, r_n) \subseteq E(s_1, \ldots, s_n)$. For example, all expressions of relational algebra using the operators select, project, Cartesian product, and union, i.e., all those that do not involve set difference, are monotone.

Motivation for Losslessness and Monotonicity

There is a natural motivation for restricting our attention to lossless, monotone expressions. Let us first consider losslessness.

Suppose that the user actually has a universal relation u in mind. Then he would like to have $[X] = \pi_X(u)$. However, because of the structure of the database, the user cannot store u, and he is forced instead to store its projections $\pi_{R_1}(u), \ldots, \pi_{R_n}(u)$ onto the relation schemes of the database scheme. Thus, he would like the function f that computes $[X]$ to be such that $f(\pi_{R_1}(u), \ldots, \pi_{R_n}(u)) = \pi_X(u)$. Well, perhaps asking for "the whole truth" is too much, because the database scheme may not support the reconstruction of certain connections in the universal relation. But surely the user would like "nothing but the truth"; that is, $f(\pi_{R_1}(u), \ldots, \pi_{R_n}(u)) \subseteq \pi_X(u)$. In other words, f should be lossless.

Indeed, the function f defined by the representative instance is lossless. To see informally why this is so, suppose we start with a universal relation u_1 satisfying a set of dependencies Δ, and we project u_1 onto some relation schemes to get relations $r_1, \ldots, r_k$. Whatever the chase process for the dependencies in Δ is, we expect that no combination of nonnull symbols will be generated by the chase unless it is a consequence of Δ. Since u_1 has all the combinations of symbols found among the tuples in the r_i's and satisfies Δ, every combination found in the result, u_2, of the chase, will be found in u_1. Thus

$$\pi_{\downarrow X}(u_2) \subseteq \pi_{\downarrow X}(u_1) = \pi_X(u_1)$$

which proves that the representative instance construction is lossless.

Note that we cannot in general prove that equality holds, since u_1 may contain tuples in its projection

onto X that cannot be reconstructed by the chase. However, there is a broad class of dependencies for which we can almost prove equality. An implicational dependency is said to be *typed* if symbols do not appear in more than one column. That is, the domains for the various attributes are regarded as disjoint, and dependencies can neither be predicated on the same value appearing in more than one column, nor can they infer the presence in one column of a symbol appearing in another.

Theorem 4: Let Δ be a set of typed implicational dependencies, u_1 a universal relation satisfying Δ, and u_2 the result of projecting u_1 onto relation schemes $R_1, \ldots, R_n$ and constructing the representative instance from these projections. Then $\pi{\downarrow}_X(u_2)$ is either empty or is exactly $\pi_X(u_1)$. The latter occurs exactly when the expression $\pi_X(\bowtie_{1 \le i \le n} R_i)$ is lossless with respect to Δ.

Proof: We argued above why $\pi{\downarrow}_X(u_2) \subseteq \pi_X(u_1)$. Conversely, suppose $\pi{\downarrow}_X(u_2)$ is not empty. Then, Δ allows us to infer, from the fact that certain tuples are in the relations over the R_i's, that some tuple t with non-null components for X exists in the representative instance u_2.

Now consider any tuple s in u_1, and consider its projection onto the relation schemes. Since the dependencies in Δ are typed, all equalities that Δ requires to infer t are satisfied by the projections of s, since all these projections agree in components that they have in common. Therefore, Δ will imply the existence in u_2 of some tuple with non-null components for X, and these components must agree with the corresponding components of s, since Δ is typed. Thus, $s[X]$ is in $\pi{\downarrow}_X(u_2)$. Also, taking the projections of a tuple on $R_1, \ldots, R_n$, padding them with nulls, and chasing them with Δ, is exactly the test for losslessness of the expression $\pi_X(\bowtie_{1 \le i \le n} R_i)$. Since we showed that we get a tuple with non-null components for X, it follows that the expression is lossless.

Finally, if the expression is lossless, then the losslessness test will produce a tuple with non-null components for X, and that means that, starting with a tuple s from u_1, we get $\pi_X(s)$ in $\pi{\downarrow}_X(u_2)$. □

Corollary 1: Let Δ be a set of implicational dependencies, u_1 a universal relation satisfying Δ, and u_2 the result of projecting u_1 onto relation schemes $R_1, \ldots, R_n$ and constructing the representative instance from these projections. Then $\pi{\downarrow}_X(u_2) = \pi_X(u_1)$ if and only if the expression $\pi_X(\bowtie_{1 \le i \le n} R_i)$ is lossless with respect to Δ.

Proof: The proof is similar to the proof of the theorem. □

When the dependencies are not typed, then Theorem 4 does not necessarily hold, as the next example shows.

Example 4: Let the universal relation scheme be $ABCD$ and the relation schemes be AB, AC, and AD.

Suppose we have the following implicational dependency.

A	B	C	D
a	b	c	d
a'	a	d	d'

That is, whatever symbols appear together in the A and D components must also appear together in the B and C components.

Then consider the following universal relation u.

a_1	b_1	c_1	d_1
a_1	a_1	d_1	d_1

Let $X = BC$, so the projection of u onto X is $\{b_1c_1, a_1d_1\}$. If we project u onto AB, AC, and AD and chase the representative instance, we can infer the existence of a tuple with middle components a_1d_1, but not the existence of one with middle components b_1c_1. □

The motivation for monotonicity comes from our hope to duplicate by an expression the connection $[X]$ defined by the representative instance, and from the fact that the function defined by the representative instance is monotone.† It is straightforward to show, whenever the representative instance is defined for the two databases $(r_1, \ldots, r_k)$ and $(s_1, \ldots, s_k)$, i.e., both databases satisfy the given dependencies Δ, that the condition $r_i \subseteq s_i$ for all i implies $[X]^{\Delta}(r_1, \ldots, r_k) \subseteq [X]^{\Delta}(s_1, \ldots, s_k)$. Thus $[X]$ defined by the restricted projection of the representative instance has the monotonicity property.

IV. Representative Instances, Lossless-Monotone Expressions, and Tableau Mappings

The broadest observation we can make is that the lossless, monotone expression approach to defining $[X]$ can only produce tuples that we get from the representative instance. In this section we also introduce tableau expressions and explore their relationship to the representative instance.

Containment of Lossless-Monotone Expressions within Representative Instances

Theorem 5: Let E be an expression that is monotone, lossless with respect to some set of dependencies Δ, and produces a relation over X. We do not constrain Δ, except that it must consist only of dependencies for which the "chase" process succeeds in producing a (finite or infinite) representative instance that satisfies Δ, e.g., the implicational dependencies. Then if tuple t is in $E(r_1, \ldots, r_n)$, it follows that t is in $\pi_{\downarrow X}(u)$, where

† Another such property, which does not play a role here, is the *containment condition*, which says that if X and Y are two sets of attributes, and $X \subseteq Y$, then for all database states d, $\pi_X([Y](d)) \subseteq [X](d)$. That is, whatever connection among the attributes of Y is represented by the database, the connection for X is an essential part of it.

$u = \mathrm{RI}(r_1, \ldots, r_n)$.

Proof: Let R_i be the relation scheme for r_i, and let $s_i = \pi_{R_i}(u)$. Then $r_i \subseteq s_i$, since u is $\mathrm{RI}(r_1, \ldots, r_n)$. By monotonicity, t is in $E(s_1, \ldots, s_n)$. As E is lossless with respect to Δ, and u satisfies Δ, t must be in $\pi_X(u)$, and since t has no null symbols, it is in $\pi{\downarrow}_X(u)$. □

Tableau Mappings

We wish to deal with the question of when there is an expression, particularly a first-order formula (i.e., an expression of relational algebra), to simulate the effect of the representative instance.† To make this characterization we need to introduce two concepts. The first is tableau mappings, which are expressions that can be denoted by tableaux as in <ASU>, and the second, which we call "bounded" database schemes, involves a strong limit on the length of the chase needed to deduce that a particular tuple is in $[X]$ during the construction of the representative instance.

For our purposes, both tableaux and (embedded) implicational dependencies will be represented in the same notation, $(t_1, \ldots, t_k)/t$, where the t_i's and t are rows of abstract symbols. The components of these rows correspond in a fixed, understood order, to the attributes of the universal relation scheme. The positions of t are either blank or are symbols that appear at least once among the t_i's. If a dependency is represented, t could alternatively be an equality $a = b$ between two symbols appearing among the t_i's. If a tableau is represented, then the t_i's could be tagged by relation schemes or relation names; we write t_i (R_i) to indicate that row t_i is tagged by relation R_i. In that case, we expect that every position of t_i that does not correspond to an attribute in the relation scheme for R_i will have a *unique* symbol, one that appears nowhere else in the tableau. Normally, we represent unique symbols by blanks.

As dependencies, we call the t_i's *hypothesis rows* and t the *conclusion row*. The notation being used here, and the meaning attributed to these dependencies is defined further in <SU, U1>, and in <BV, F>, although different notation is used in the latter papers. Roughly, the dependency says that whenever we see tuples that look like the hypothesis rows, in the sense that there is a mapping of symbols that makes all the hypothesis rows become tuples of the relation, then there is also some tuple in the relation that is the conclusion row after mapping of the symbols, with blank positions mapped to arbitrary symbols; if the conclusion is $a = b$, then instead we require that this arbitrary symbol mapping has in fact mapped a and b to the same symbol, because the relation allowed no other possibility.

As an expression, the t_i's are called *rows*, and t is called the *summary*. The meaning of such a mapping

† Note that there is always a second-order formula to simulate the representative instance, since we may thus express the condition that a relation with the properties of the representative instance exists.

E	D	O	P	
e	d		p	(EDP)
e		o		(EO)
		o	p	(OP)
	d	o		

Fig. 3. A tableau expression or implicational dependency.

is described in <ASU>. Informally, it means that the result of the mapping applied to a universal relation u is found by taking each possible mapping of symbols that makes each row a tuple in u and placing in the result the tuple that is formed by applying this symbol mapping to the summary. We can also apply a tableau mapping not to the universal relation but to a collection of relations $r_1, \ldots, r_n$ over relation schemes $R_1, \ldots, R_n$ that are each subsets of the universal relation scheme. In this case we require that for each row t_j, tagged by some relation R, we have $R = R_i$ for some i, $1 \leq i \leq n$, and that the symbol mapping sends t_j, restricted to R_i, to some tuple of r_i.

Lemma 1: Let $E = (t_1, \ldots, t_n)/t$ be a tableau expression, and Δ a set of implicational dependencies. Then E is lossless with respect to Δ if and only if $\Delta \models E$, when E is treated as an implicational dependency.

Proof: E is lossless if and only if it is contained in the expression that is the projection onto those attributes in which t has a nonblank, that is, in the result of the tableau mapping t'/t, where t' is t with blanks replaced by new symbols. By the test of <ASU>, this containment holds only if, after chasing $\{t_1, \ldots, t_n\}$ by the dependencies in Δ, t' can map to one of the resulting rows. But that is exactly the condition under which $\Delta \models E$. □

Example 5: In Fig. 3 we see an expression or dependency based on the database of Example 3. We follow the convention of using blanks not only in the summary/conclusion, but everywhere that a symbol appearing only once is found. As a tagged tableau mapping, it produces the natural join of the three relations EDP, EO, and OP, projected onto DO. As a dependency, it says that d and o appear together in a tuple of each universal relation in which for some e and p, there is a tuple in which e, d, and p appear together, another tuple in which e and o appear, and a third in which o and p appear, all in their appropriate columns. As a consequence of this dependency, the above expression $\pi_{DO}(\bowtie(EDP, EO, OP))$ is lossless. □

Bounded Database Schemes

A *database scheme* is a finite set of relation schemes and a finite set of dependencies that apply to the

universal relation scheme that is the union of the given relation schemes. We denote the database scheme by $D = (\Delta, \{R_1, \ldots, R_n\})$, where Δ is the dependencies and $R_1, \ldots, R_n$ the relation schemes. We say that two database schemes are *equivalent* if

1. their relation schemes are the same, and
2. their dependencies are logically equivalent, i.e., the same universal relations satisfy both sets of dependencies.

Let $D = (\Delta, \{R_1, \ldots, R_n\})$ be a database scheme. We say D is *k-bounded* (for set of attributes X) if for any relations $r_1, \ldots, r_n$ over the R_i's, if $u = \mathrm{RI}(r_1, \ldots, r_n)$, and t is in $\pi{\downarrow}_X(u)$, then we can deduce that fact by a sequence of at most k applications of dependencies in Δ, starting with the r_i's (padded with blanks as in Fig. 1). D is *bounded* if it is k-bounded for some k. Note that "bounded" says more than that the chase terminates in a finite relation for any r_i's. It says that sequences of k dependency applications suffice independently of the initial r_i's. As we shall see, "bounded," "k-bounded," and "1-bounded" all are equivalent statements, in the sense that any bounded database scheme is equivalent to a 1-bounded scheme.

V. Lossless Tableau Mappings and Representative Instances

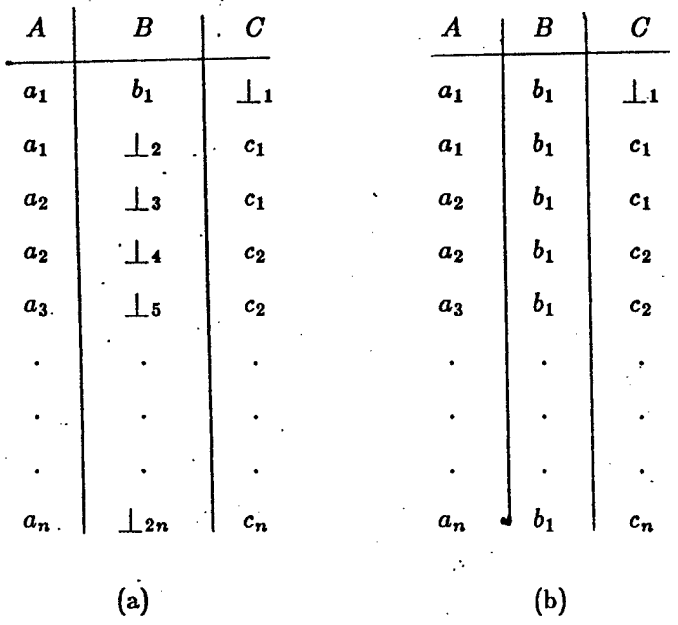
We shall now develop our characterization of when the representative instance can be simulated by a first-order formula. In particular, we show the equivalence of the following three statements about a database scheme D, whose dependencies are implicational, and a set of attributes X.

1. D is bounded for X (and in fact 1-bounded).
2. There is a first-order formula that computes $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$, i.e., there is an expression of relational algebra that simulates the representative instance.
3. $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$ is computed by a finite union of tableau mappings.

Theorem 6: Let $D = (\Delta, \{R_1, \ldots, R_n\})$ be a database scheme, where Δ is a set of implicational dependencies. Then there is, for each set of attributes X, a finite set of lossless tableau expressions whose union yields the same relation as $\pi{\downarrow}_X(u)$, where u is $\mathrm{RI}(r_1, \ldots, r_n)$ if and only if D is equivalent to some bounded scheme.

Proof:

If: We may as well assume that D itself is bounded. Let k be the bound on the number of dependencies that need to be applied, and let m be the maximum number of hypothesis rows in any member of Δ. Then consider every tagged tableau expression E whose summary has nonblanks exactly in the positions for the attributes in X, and that has at most km rows. Depending on the equalities of various symbols among the



A	B	C
a_1	b_1	$\perp_1$
a_1	$\perp_2$	c_1
a_2	$\perp_3$	c_1
a_2	$\perp_4$	c_2
a_3	$\perp_5$	c_2
.	.	.
.	.	.
.	.	.
a_n	$\perp_{2n}$	c_n

(a)

A	B	C
a_1	b_1	$\perp_1$
a_1	b_1	c_1
a_2	b_1	c_1
a_2	b_1	c_2
a_3	b_1	c_2
.	.	.
.	.	.
.	.	.
a_n	b_1	c_n

(b)

Fig. 4. Chase of unbounded length.

rows of the tableau, E may or may not be guaranteed to produce tuples over X that will be produced when the representative instance is chased by Δ. If $\Delta \models E$, then we have such a guarantee; if not then we don't.

Moreover, since D is bounded, all tuples found to be in the representative instance projected onto X will be generated by some such expression that is logically implied by Δ when the expression is treated as a dependency. Also, there is evidently only a finite number of such dependencies. We therefore can compute $\pi{\downarrow}_X(u)$ by taking the union of all those expressions whose tableaux' summaries have distinguished symbols in exactly the columns of X, and that are logically implied by Δ, when treated as dependencies.

Only if: Suppose that there is a lossless expression E_X that is the finite union of tableau expressions and produces the same result as $\pi{\downarrow}_X(u)$. Convert the set of E_X's to a set of dependencies, say Δ'. Since the E_X's are lossless, $\Delta \models \Delta'$. It follows that Δ is equivalent to $\Delta \cup \Delta'$.

Clearly, one step of dependency application using Δ' serves to obtain enough tuples in u to prove that $\pi{\downarrow}_X(u)$ will contain whatever E_X produces, as D is 1-bounded. □

Example 6: Consider the database scheme $D = (\{A{\rightarrow}B, C{\rightarrow}B\}, \{AB, AC\})$. We claim that D is not bounded, and therefore the universal relation defined by its representative instance cannot be simulated by any finite union of lossless tableau expressions. To see informally why D is not bounded, consider the relations $\{a_1, b_1\}$ for AB and $\{a_1c_1, a_2c_1, a_2c_2, a_3c_2, \ldots, a_ic_{i-1}, a_ic_i, \ldots, a_nc_n\}$ for AC. The initial representative instance is shown in Fig. 4(a), and the chased version in Fig. 4(b). Note that deleting any tuple from AC means that $a_nb_1c_n$ will no longer appear in the chase.

Suppose now that D is bounded. Whatever set of dependencies equivalent to $\{A \rightarrow B, C \rightarrow B\}$ we choose, the existence of a tuple $a_n b_1 c_n$ will have to be inferred from only a finite number of the tuples in AC. However, by making n large enough, we can force the inference to be made without looking at some $a_i c_i$. Then, by deleting this tuple from AC, we can force a situation where the dependencies $\{A \rightarrow B, C \rightarrow B\}$ no longer imply the presence of tuple $a_n b_1 c_n$, yet the chase using the equivalent set of dependencies still produces that tuple. □

Now let us turn our attention to the second equivalence, that between arbitrary first-order formulas and finite unions of lossless tableau mappings.

Theorem 7: Let $D = (\Delta, \{R_1, \ldots, R_n\})$ be a database scheme, where Δ is a set of implicational dependencies, and let X be a set of attributes. Then $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$ is expressed by a first-order formula if and only if it is expressed by some finite union of lossless tableau mappings.

Proof: The "if" portion is trivial. For the "only if" part, we observe that every time we add a tuple in the chase, we can find a tableau mapping that yields the projection of this tuple onto X. We find this tableau by first expressing as an implicational dependency the fact that this tuple is inferred from a finite set of tuples of the original relations. The hypotheses of this dependency will each have a particular relation scheme in whose columns all the nonunique symbols appear; thus it can be viewed as a tagged tableau mapping. By Lemma 1, the losslessness of this mapping follows from Δ. It follows that there is some (possibly infinite) set of lossless tableau mappings whose union yields $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$.

Let these mappings be $T_1, T_2, \ldots$, and let $Q_1, Q_2, \ldots$ be the relations over X produced by these mappings.† We know that $Q_1 \cup Q_2 \cup \cdots$ is equal to $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$. There is an unexpected difficulty here because we cannot refer to this infinite union by a first-order sentence. However, we can say that the relation R is a superset of this union by using an infinite set of sentences. In what follows, we use R_i as a relation symbol that stands for the i^{th} relation in the database scheme, and we use R as an arbitrary relation over X.

By <GV>, we can find a (possibly infinite) set of implicational dependencies Δ' over $R_1, \ldots, R_n$ asserting that these relations are consistent, in the sense that when we chase these relations, we do not attempt to equate two different nonnull symbols, and therefore, the representative instance exists. Further, we can construct for each T_i a first-order formula $\psi_i(t, R_1, \ldots, R_n)$ that asserts that tuple t is in Q_i. Let $\phi(t, R_1, \ldots, R_n)$ be the hypothetical first-order formula that says of tuple t that t is in $\pi{\downarrow}_X(u)$, where u is the representative instance constructed from the relations for which the R_i's stand. Then we have the following

† The reader should be aware that in this proof, we represent relations by predicate symbols, Q's and R's. Thus, Q really stands for the predicate $Q(t)$ that is true if and only if tuple t is in the relation we call Q. We shall continue to treat such predicate symbols as if they were relations, e.g., by applying algebraic operators to them.

logical implication.

$$\Delta' \cup \{ (\forall t)(\psi_i \rightarrow R(t)) \mid i = 1, 2, \ldots \} \models (\forall t)(\phi \rightarrow R(t))$$

The reason that this implication holds is as follows. In a model of the left-hand side, the relations $r_1, \ldots, r_n$ for $R_1, \ldots, R_n$ constitute a database that has a weak universal relation, and the relation for R contains all the Q_i's. Thus it contains $\pi{\downarrow}_X(\mathrm{RI}(r_1, \ldots, r_n))$, and therefore it contains the relations produced by ϕ. Thus the model satisfies $\forall t(\phi \rightarrow R(t))$.

By compactness, there is a finite subset of the ψ_i's,† which we may take to be $\psi_1, \ldots, \psi_k$, such that

$$\Delta' \cup \{ (\forall t)(\psi_1 \rightarrow R(t)), \ldots, (\forall t)(\psi_k \rightarrow R(t)) \} \models (\forall t)(\phi \rightarrow R(t))$$

Now let R be $Q_1 \cup \cdots \cup Q_k$, that is, the union of the results of applying the k tableau mappings to the given relations. Thus $R(t)$ is logically equivalent to $Q_1(t) \vee \cdots \vee Q_k(t)$. Then $\psi_i(t, R_1, \ldots, R_n) \rightarrow R(t)$ is surely true for $1 \leq i \leq k$, so

$$\Delta' \models (\forall t)(\phi(t, R_1, \ldots, R_n) \rightarrow (Q_1(t) \vee \cdots \vee Q_k(t)))$$

That is, $\pi{\downarrow}_X(u) \subseteq (Q_1 \cup \cdots \cup Q_k)$, where u is, as before, the representative instance constructed from the relations of the database.

Containment in the opposite direction is obvious, since the T_i's were constructed to mimic what the chase does, and each Q_i is the result of applying T_i to the relations of the database. Thus for all database relations $r_1, \ldots, r_n$ that satisfy Δ' (i.e., we can successfully chase the r_i's to construct a representative instance u) we have that $\pi{\downarrow}_X(u)$ equals the union of the tableau mappings $T_1, \ldots, T_k$ applied to $r_1, \ldots, r_n$. □

Query Optimization

Our main interest in the computational approach to the universal relation model comes from a practical consideration of computational efficiency; we do not want the expressions computing the $[X]$'s to be too complicated. Thus, naturally, the issue of optimizing the expression to compute $[X]$ is of paramount interest. For example, Sagiv <Sa1, Sa2> takes only minimal extension joins to produce the answers to queries, and in limited cases, proves that these simple expressions suffice to compute $[X]^{\Delta}$.

An interesting consequence of Theorem 7 is that we can use the weak optimization technique of <ASU> to optimize our expression. By that theorem, we have to deal only with unions of lossless tableau mappings. Let T be such a tableau. We can view T as a tagged tableau that defines a mapping on relations over $R_1, \ldots, R_n$, as an untagged tableau that defines a mapping on universal relations, or as a dependency on

† and incidentally, a finite subset of Δ'.

universal relations. Suppose that T' is another tagged tableau, obtained by removing some of the rows of T, that is weakly equivalent to T. That is, T' is equivalent to T when T and T' are considered as untagged tableaux. Clearly, T is contained in T' when they are considered as tagged tableaux; i.e., when applied to relations over $R_1, \ldots, R_n$, T' produces all tuples that T produces. Now, T is lossless with respect to the set of dependencies Δ, so $\Delta \models T$, when T is considered as a dependency, and consequently, $\Delta \models T'$, when T' is considered as a dependency. Thus, T' as a tagged tableau produces only tuples that are produced by the representative instance. As a consequence, T' can replace T in the union of tableau mappings.

Weakly optimized joins on maximal objects are used in System/U in order to compute connections <U2>. The motivation there is given by appealing to the way dangling tuples are treated; this argument is intuitively reasonable but without mathematical foundations.

Storage of Query Interpretation Information

Naturally, we do not wish to store, for the current database state d, all the views $[X](d)$, where X ranges over all sets of attributes. However, might it be feasible to store expressions for calculating $[X](d)$ from d for all X? The implication of our developments is that if any first-order way of computing connections exists, then we can establish for each set of attributes X a finite set of tableaux whose mappings together produce $[X]$. However, if there are, say, 100 attributes in the universal scheme, it does not seem realistic to store all the expressions needed to reconstruct the $[X]$'s.

Existing universal relation systems have mechanisms for constructing the expressions for $[X]$ "on the fly." For example, System/U <U2> stores only the maximal objects, and obtains $[X]$ by reductions of the expressions for the maximal objects.

It appears that we can take something like this approach in general. If connections are defined by the representative instance, then for any set of attributes X and attribute A not in X, $[X] \subseteq \pi_X([XA])$. Thus, $[X]$ is at least the union of the projections of all $[XA]$'s. If $[X]$ is exactly equal to the union of these projections, then we need not store an expression for $[X]$. The only X's for which we need to store a formula are those for which $[X]$ is a proper superset of $\cup_A \pi_X([XA])$; these were called *implicit objects* in <Mal>, because they generalized the idea of constructing maximal objects as in <MU>.

VI. Concluding Remarks

We have explored computational methods that might be used to simulate the effect of the representative instance. Three overlapping classes of expressions were considered as possible computation methods:

1. monotone,

2. lossless, and

3. first-order.

We also identified a class of expressions that is in the intersection of all three of these classes: unions of lossless tableau mappings.

We showed that monotonicity and losslessness are properties that we should expect of any computational method that simulates the representative instance, for the simple reason that the representative instance has these properties. If we want a first-order method, i.e., an expression of relational algebra, then we find that we need only consider finite unions of lossless tableau mappings, since all first-order methods are equivalent to one of these. As a consequence, condition (3) above implies (1) and (2).

We would like to close by pointing out three shortcomings of the theory. First, though we have identified the class of database schemes where the representative instance can be simulated by a first-order expression, our characterization, the boundedness condition, is not effective; we do not know how to test whether a schema is bounded or not. In fact, we do not even know whether this problem is solvable at all, even in simple cases where only functional dependencies are given.

Secondly, in showing that if there is any first-order expression then it must be a union of lossless tableau mappings (Theorem 7), we used the compactness theorem. But in order to use compactness, we have to take into account both finite and infinite databases. What happens if we restrict ourselves to finite databases? Conceivably, there could be a first-order expression that is equivalent to an infinite union of lossless tableau mappings, but is not equivalent to any finite union of such. In some limited cases, such as Example 6, we can show that this is not the case by more involved arguments. But these arguments do not lead themselves to generalization.

Third, there are reasons why the representative instance approach does not support all the semantics that we might wish for in a universal relation system, and we doubt that it will serve as the "ultimate" universal relation model. Some problems that we see as forcing awkwardness in the way universal relation systems are used are the following.

1. A representative instance system answers queries by intersecting the weak instances, and then applying the query (Theorem 1). However, if the weak instances are all the possible universal relations that the user might see, it may make more sense to apply the query to all the weak instances, and then take the intersection of the results. This approach cannot produce more than the method of query interpretation in which we compute the representative instance first, but there are examples where it produces less, notably when a join on "not equal" is involved in the query.

2. The representative instance allows us to infer equalities among nulls, but, since nulls are projected out

before we compute the answer to the query, these equalities cannot influence the answer. For example, we might deduce that employees Smith and Jones have the same manager because they are in the same department, yet not know their manager, because the information is not in the database. In answer to the query "list pairs of employees with the same manager," we would not list the pair (Smith, Jones).

3. The representative instance approach supports only one notion of nulls, generally referred to as "missing value nulls." We might wish to restrict the ability assumed in the representative instance approach to extend any tuple in any relation of the database to the universal set of attributes. Perhaps it is better to extend tuples only in limited ways, leaving certain positions in tuples "blank" and allowing no dependency applications at all involving blanks. The effect of this restriction is that the universal relation is split into several relations with overlapping, but distinct sets of attributes.

We hope to discuss these issues and propose an "improved" representative instance that supports many of the concepts developed here in a forthcoming paper <MUV>.

References

[Ah] Aho, A. V., private communication.

[ABU] Aho, A. V., C. Beeri, and J. D. Ullman, "The theory of joins in relational databases," *ACM Transactions on Database Systems* 4:3 (1979), pp. 297–314.

[Ar] Arazi-Gonczarowski, Z., "A high-level interface for naive users in a relational database," Dept. of C. S., Hebrew Univ., Jerusalem, 1981.

[AP] Atzeni, P. and D. S. Parker, "Assumptions in relational database theory," *Proc. ACM Symposium on Principles of Database Systems*, pp. 1–9, 1982.

[ASU] Aho, A. V., Y. Sagiv, and J. D. Ullman, "Equivalence of relational expressions," *SIAM J. Computing* 8:2 (1979), pp. 218–246. See also "Efficient optimization of a class of relational expressions," *ACM Transactions on Database Systems* 4:4 (1979), pp. 435–454.

[BBG] Beeri, C., P. A. Bernstein, and N. Goodman, "A sophisticates introduction to database normalization theory," *Proc. International Conference on Very Large Data Bases*, pp. 113–124, 1978.

[BR] Beeri, C. and J. Rissanen, "Faithful representation of relational database schemes," IBM research report, San Jose, CA, 1980.

[BV] Beeri, C. and M. Y. Vardi, "Formal systems for tuple and equality generating dependencies," unpublished memorandum, Hebrew University, 1981. To appear in *SIAM J. Computing.*

[B] Bernstein, P. A., "Synthesizing third normal form relations from functional dependencies," *ACM Transactions on Database Systems* 1:4 (1976), pp. 277–298.

[C1] Codd, E. F., "A relational model for large shared data banks," *Comm. ACM* 13:6 (1970), pp. 377–387.

[C2] Codd, E. F., "Relational databases: a practical foundation for productivity," *Comm. ACM* 25:2 (1982), pp. 109–117.

[C*] Codd, E. F., R. S. Arnold, J. M. Cadiou, C. L. Chang, and N. Roussopoulos, "Rendezvous version I: an experimental English language query formulation system for casual users of relational databases," RJ2144, IBM, San Jose, CA, 1978.

[F] Fagin, R., "Horn clauses and database dependencies," *Proc. Twelfth Annual ACM Symposium on the Theory of Computing*, pp. 123–134, 1980. To appear in *J. ACM*.

[FMU] Fagin, R., A. O. Mendelzon, and J. D. Ullman, "A simplified universal relation assumption and its properties," *ACM Transactions on Database Systems* **7**:3 (1982), pp. 343–360.

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

[GrMe] Graham, M. and A. O. Mendelzon, "Notions of dependency satisfaction," *Proc. ACM Symposium on Principles of Database Systems*, pp. 177–188, 1982.

[GV] Graham, M. and M. Y. Vardi, "On the axiomatizability of consistent database states," manuscript in preparation.

[H] Honeyman, P., "Testing satisfaction of functional dependencies," to appear in *J. ACM*.

[HLY] Honeyman, P., R. E. Ladner, and M. Yannakakis, "Testing the universal instance assumption," *Inf. Proc. Letters*, **10**:1 (1980), pp. 14–19.

[Ke] Kent, W., "Consequences of assuming a universal relation," *ACM Transactions on Database Systems* **6**:4 (1981), pp. 539–556.

[Ko] Kowalski, R., "Logic as a database language," Dept. of Computing, Imperial College, London, 1981.

[KS] Kuck, S. M. and Y. Sagiv, "A universal relation database system implemented via the network model," *Proc. ACM Symposium on Principles of Database Systems*, pp. 147–157, 1982.

[L] Lozinski, E. L., "Construction of relations in relational databases," *ACM Transactions on Database Systems* **5**:2 (1980), pp. 208–224.

[Ma1] Maier, D., *The Theory of Relational Databases*, Computer Science Press, Rockville, Md., 1983.

[Ma2] Maier, D., "Discarding the universal instance assumption," *Proc. XP/1 Workshop*, Stony Brook, N. Y., June, 1980.

[MMSU] Maier, D., A. O. Mendelzon, F. Sadri, and J. D. Ullman, "Adequacy of decompositions in relational databases," *J. Computer and System Sciences* **21**:3 (1980), pp. 368–380.

[MMS] Maier, D., A. O. Mendelzon, and Y. Sagiv, "Testing implications of data dependencies," *ACM Transactions on Database Systems* **4**:4 (1979), pp. 455–469.

[M*] Maier, D., D. Rozenshtein, S. Salveter, J. Stein, and D. S. Warren, "Toward logical data independence: a relational query language without relations," *ACM SIGMOD International Symposium on Management of Data*, pp. 51–60, 1982.

[MU] Maier, D. and J. D. Ullman, "Maximal objects and the semantics of universal relation databases," to appear in *TODS*.

[MUV] Maier, D., J. D. Ullman, and M. Y. Vardi, "Beyond representative instances," manuscript in preparation, Stanford Univ., Stanford, CA.

[MW] Maier, D. and D. S. Warren, "Specifying connections for a universal relation scheme database," *ACM SIGMOD International Symposium on Management of Data*, pp. 1–7, 1982.

[Me] Mendelzon, A. O., "Database states and their tableaux," *Proc. XP2 Workshop*, State College, Pa., June, 1981.

[O] Osborn, S. L., "Towards a universal relation interface," *Proc. International Conference on Very Large Data Bases*, pp. 52–60, 1979.

[Re] Reiter, R., "Towards a logical reconstruction of relational database theory," Dept. of C. S., Univ. of British Columbia, 1981.

[Ri] Rissanen, J., "Independent components of relations," *ACM Transactions on Database Systems* **2**:4 (1977), pp. 317–325.

[SU] Sadri, F. and J. D. Ullman, "Template dependencies: a large class of dependencies in relational databases and its complete axiomatization" *J. ACM* **29**:2 (1982), pp. 363–372.

[Sa1] Sagiv, Y., "Can we use the universal instance assumption without using nulls?," *ACM SIGMOD International Symposium on Management of Data*, pp. 108–120, 1981.

[Sa2] Sagiv, Y., "A characterization of globally consistent databases and their correct access paths," unpublished memorandum, Univ. of Illinois, Dept. of C. S., 1981.

[U1] Ullman, J. D., *Principles of Database Systems*, Computer Science Press, Potomac, Md., 1983.

[U2] Ullman, J. D., "The U. R. strikes back," *Proc. ACM Symposium on Principles of Database Systems*, pp. 10–22, 1982.

[V1] Vardi, M. Y., "On decomposition of relational databases," to appear in *IEEE Twenty-Third Annual Symp. on Foundations of Computer Science*, Nov., 1982.

[V2] Vardi, M. Y., "Axiomatization of functional and join dependencies in the relational model," M. Sc. Thesis, Dept. of Applied Math., Weizmann Inst. of Science, Rehovot, 1980.

[Y] Yannakakis, M., "Algorithms for acyclic database schemes," *Proc. International Conference on Very Large Data Bases*, pp. 82–94, 1981.

[YP] Yannakakis, M. and C. H. Papadimitriou, "Algebraic dependencies," *Proc. Twenty-First Annual IEEE Symposium on Foundations of Computer Science*, pp. 328–332, 1980. To appear in *J. Computer and System Sciences*.