MTA SZTAKI Közlemények 36/1987 pp. 49-78

The Equivalence Problem For Relational Database Schemes

Joachim Biskup and Uwe Räsch
Institut für Informatik
Hochschule Hildesheim
Samelsonplatz 1
D-3200 Hildesheim
Federal Republic of Germany

Abstract

Mappings between the sets of instances of database schemes are used to define different degrees of equivalence. The available class of mappings and the set of dependencies allowed for defining schemes deal here as parameters. A comparison of the equivalences shows that there is only one natural kind of equivalence. For various cases we prove its decidability or undecidability. Besides we get a characterization of mappings expressible in the relational algebra without the difference.

1. Introduction and Conventions

An intuitive definition of equivalence is given by [Gee]: "Two databases are equivalent if they represent the same set of facts about a certain piece of world". We will try to get a more exact definition of what is meant by equivalence. We will consider only relational databases.

The motivation for a comparison of the information capacity of database schemes stems from different areas:

- design of conceptual schemes, especially the so-called database normalization
- integration of different userviews into a single global scheme
- translations between different databases
- extensions and transformations of databases
- evaluation of different approaches in database theory.

We will develop a general model (chapter 1) to formalize some kinds of equivalence and to study differences between them (chapter 2). Then we are concerned with the decision problem for the chosen kind of equivalence. In chapter 3 we present some decidable cases, whereas in chapter 4 we prove various undecidability results.

We will neither assume a universal scheme assumption or a universal relation assumption (see [AP]), nor consider the update facilities used in a database (see [Codd] for update-equivalence).

More powerful mapping classes are NGEN* and FGEN*. They are related to the " M-internal mappings " of [Hull] and defined as follows:

q1 $\in$ NGEN* iff SYMB(q1(A1)) $\subseteq$ SYMB(A1) for all A1 $\in$ TYPES1
(" no generation of new values "),

q1 $\in$ FGEN* iff there exists a finite set M such that
SYMB(q1(A1)) - SYMB(A1) $\subseteq$ M for all A1 $\in$ TYPES1
(" generation of only a finite set of new values ").

Every mapping class Q is assumed to contain the identity w.r.t. the set of all states of an arbitrary database scheme and to be closed under composition, that means:

$\forall$ q1: TYPE1 -> TYPE2 $\forall$ q2: TYPE2 -> TYPE3:
q1 $\in$ Q and q2 $\in$ Q ==> q2 $\circ$ q1 $\in$ Q.

These assumptions are obviously satisfied by RALG*, RANP*, RAND*, NGEN* and FGEN*.

We will also define two classes of database dependencies.

ALL denotes the set of dependencies which can be transformed into a sentence in prenex-normalform without existential quantifiers. Often the adjective "full" is used to characterize some subsets of ALL, see [FV] or [CLM]. Most prominent examples of such subsets are the functional dependencies, see e.g. [Ullm], the full inclusion dependencies, see [KCV], and the exclusion dependencies of [CV].

EX denotes the set of dependencies which can be transformed to a sentence in prenex-normalform without universal quantifiers. Additionally every predicate symbol other than "=" appears only under an even number of negation signs. An example would be " $\exists$ x1, x2, y1, y2: 1(x1,x2) and 1(y1,y2) and (x1 $\neq$ y1 or x2 $\neq$ y2) " , which demands that the first relation should contain at least two different tuples.

2. A Hierarchy of Equivalences

Most of the approaches to define equivalence of database schemes use the ability to construct mappings between their states as a criterion (see for example [AABM], [Biller], [CV], [IL1], [Hull], [KK], [Koba], [Riss]). Whereas the intention of these papers is sometimes a very special one we will try to be as general as it is possible.

As usual we only want to consider instances instead of arbitrary states, since there seems to be no reason to regard database states which do not correspond to a possible real world. Furthermore, we will not consider arbitrary mappings for the definition of equivalence. For if two schemes both have an infinite set of instances, then there always exists an (mostly pathological) bijection between their instances. One way would be to consider only renaming of values, but this seems much too restrictive.

Approaches which are engaged in database normalization (as [BBG], [BMSU], [IL1], [Riss]) consider only mappings built up by natural join, projection and selection. [Koba] uses four special kinds of mappings. [Hull] is interested in

Like we have done above, schemes will always be denoted as Si (where i is a subscript), states as Ai, Bi or Ci, instances as Ii, Ji or Gi, the set of states as TYPEi, the set of instances as INSi (where the subscription shows to which scheme they belong). SYMB(Ai) stands for the set of all values appearing in the state Ai.

For two states A1,B1 $\in$ TYPE1 A1 $\subseteq$ B1 holds iff for all i with $1 \leq i \leq |S1|$ A1[i] $\subseteq$ B1[i] holds. The number of tuples of A1 is denoted by |A1|.

<u>Database mappings</u> are denoted by q1, p1, r1 : TYPE1 -> TYPE2 or q2, p2, r2 : TYPE2 -> TYPE1. If not defined explicitly, the schemes S1 and S2 and the mapping class Q, to which all these mappings are assumed to belong, are given globally. The obvious conventions about the subscriptioning holds if not stated something else.

q1[i] should denote the i-th part of q1, that means q1 = < q1[1], ..., q1[m] > (for |S2| = m).

Mapping classes or <u>query classes</u> are denoted by Q, Q1 or Q2. The most interesting class is RALG*, the set of sequences of queries expressed in the relational algebra. More precisely q1 $\in$ RALG* iff for all i q1[i] $\in$ RALG holds, where RALG is the usual relational algebra, see [Ullm]. The operators are symbolized in the following way:

- " i " stands for the i-th relation scheme
- " $\cup$ " is the union sign
- " - " is the difference sign
- " [i1, ..., ik] " symbolizes a projection
  (ij are natural numbers, assumed to be different)
- " [i1 -> i2 ->...-> ik -> i1] " symbolizes a permutation
  (ij are different natural numbers)
- " [i comp_op j] " symbolizes a restriction
  (here i, j are natural numbers, comp_op $\in$ { "=", " $\neq$ " })
- " [i comp_op 'c'] " symbolizes a selection
  (here i is a natural number, comp_op $\in$ { "=", " $\neq$ " } and c $\in$ VALUES).

If the arity of a subexpression is lower than the arity of a projection, permutation, restriction or selection applied to it, or if the arities of the both subexpressions involved in a union or difference are not the same, or if "i" is greater than the number of relations of the database schemes, we assume that the expression always yields the empty relation state of arity 1. This is to avoid partially defined mappings.

RANP* (resp. RANP) is the subclass of the <u>r</u>elational <u>a</u>lgebra which do <u>n</u>ot contain any <u>p</u>rojection. RAND* (resp. RAND) contains the expressions without the difference sign.

It should be noted, that, for notational convenience, we do not make a clear distinction between an expression and the denoted mapping.

Our general model is based on typed database schemes with dependencies defined in a (subset of the) first-order logic. Both the class of dependencies and the class of queries will play an important role in separating decidable and undecidable cases of the equivalence problem.

The following definitions and notations are used.

TYPES is the set of <u>types</u> , i.e. attribute domains allowed in the definitions of database schemes. The following property holds:

$\forall$ T1, T2 $\in$ TYPES: T1 $\cap$ T2 = ϕ or T1 $\subset$ T2 or T2 $\subset$ T1.

A type can be finite or infinite. It should be a countable set.

VALUES is the set of all <u>values</u> appearing in such a type, that means: VALUES = { v $\in$ T: T $\in$ TYPES }.

A <u>relation scheme</u> R is a finite sequence of types: R1 = < T1, ..., Tk >, where all Ti $\in$ TYPES. A <u>state</u> of the relation scheme is a finite set of tuples, where each tuple is a sequence of values corresponding to the types of that relation scheme.

A <u>database scheme</u> S1 consists of a finite sequence of relation schemes and a finite set of dependencies:

S1 = < R1, ..., Rm : D >
    = < <T11,...,T1a_1>,...,<Tm1,...,Tma_m> : D >.

We use the notation |S1| = m, S1[i] = Ri, |S1[i]| = ai (the arity of the i-th relation scheme of S1).

The set of <u>dependencies</u> D of S1 is a finite set of first-order sentences over a signature

- with the ai-ary predicate symbol "i" ($1 \leq i \leq m$), which corresponds to the i-th relation scheme S1[i] of S1,
- the binary identity sign, which always will be interpreted as the identity over VALUES,
- a finite set of individual constants "c1", ..., "cn", where all ci $\in$ VALUES; such a constant "ci" will always be interpreted by ci.

Quantifiers in such a sentence range over VALUES (and not over the set of values appearing in a database state). Otherwise a sentence will be interpreted as usual, see [GMN], [Reiter] or [FV].

The set of <u>states</u> of S1 is given by TYPE1 = { < A1[1], ..., A1[m] > :
    for all $1 \leq i \leq m$ the set A1[i] is a finite subset of Ti1 * ... * Tia_i }.

The set of <u>instances</u> of S1 is given by INS1 = { I1 $\in$ TYPE1 : I1 satisfies all dependencies of D }. It is not assumed that a dependency is domain independent, see [FV], but the set of instances of a scheme is assumed to be decidable.

the whole relational algebra and shows the important role of the database dependencies for the definition of equivalence. To cover all cases we will define equivalence with respect to a given class of mappings Q. So our approach is very similar to that of [ABM] and [AABM].

First we will define some properties of mappings between states of two database schemes.

Definition 1

q1 is consistent iff $q1(INS1) \subset INS2$.

q1 is injective iff $\forall I1, J1 \in INS(S1): I1 \neq J1 ==> q1(I1) \neq q1(J1)$.

q1 is surjective iff $q1(INS1) \supset INS2$.

q2 is inverse to q1 iff $\forall I1 \in INS1: q2(q1(I1)) = I1$.

You should note that these properties depend on the set of instances of both schemes. It is easy to show the following facts.

Theorem 2

1. If q2 is inverse to q1, then q1 is injective.
2. If q2 is inverse to q1 and q2 is consistent, then q1 is injective and q2 is surjective.
3. If q2 is inverse to q1 and q1 is surjective, then q2 is consistent and injective and q1 is injective and inverse to q2.

Proof: omitted. ▪

Now we are able to present some kinds of conceptual inclusion and equivalence of database schemes.

Definition 3

S1 <1< S2 wrt.Q iff there exists a consistent and injective $q1 \in Q$.

S1 <2< S2 wrt.Q iff there exists a surjective $q2 \in Q$.

S1 <3< S2 wrt.Q iff there exits a consistent $q1 \in Q$ and a $q2 \in Q$ which is inverse to q1.

S1 <4< S2 wrt.Q iff there exists a surjective $q2 \in Q$ and a $q1 \in Q$ which is inverse to q2.

For i = 1,2,3,4 let

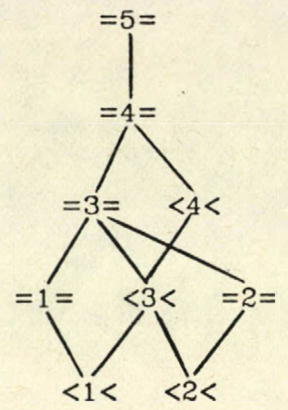
S1 =i= S2 wrt.Q iff S1 <i< S2 wrt.Q and S2 <i< S1 wrt.Q.

S1 =5= S2 wrt.Q iff there exist consistent, injective and surjective $q1, q2 \in Q$ each one being inverse to the other. ▪

Using the properties of a mapping class defined in chapter 1, it is easy to show, that all of the =i= predicates are reflexive, transitive and symmetrical.

The "weak - inclusion" of database schemes (see [AABM]) is exactly the same as our " <2< ". The "inclusion" of [AABM] is equivalent to " <3< " of our definition.

Using theorem 2 and some examples showing distinctness we are able to prove the following Hasse-diagram (the strongest property is at the top) :



It seems that all of these predicates only consider the possibility of translating any instance of one scheme into an instance of the other. A result of [ABM] however can be used to show that this can be equivalent to a comparison of the set of answers to queries.

Theorem 4

Since Q is assumed to contain the identity (on the set of states) and is closed under composition of mappings, the following holds: S1 <2< S2 wrt.Q iff $\forall\, q1 \in Q \;\exists\, q2 \in Q \;\forall\, I1 \in INS1 \;\exists\, I2 \in INS2 : q1(I1) = q2(I2)$.

Proof: see [ABM], theorem 2.1 for the idea.

We reject equivalence =1=, because it is not compatible with such a comparison of answers to queries. But any of the equivalences =2=, =3=, =4=, =5= is proved to be as restrictive as the query-equivalence of [Codd]. Which of them should one choose?

An example will suggest us to reject the weakest of them.

Example 5

Let Q be RALG , $S1 := \langle \langle AB\rangle, \langle BC\rangle, \langle AC\rangle : \phi \rangle$, $S2 := \langle \langle AB\rangle, \langle B\rangle, \langle BC\rangle : \phi \rangle$ and A, B, C $\in$ TYPES be disjoint.

Nobody would consider these schemes as being equivalent. But it turns out that S1 =2= S2 wrt.Q holds. To show this we have to construct two surjective mappings:

q1 := < "(1*3)[1=3][1,2]", "(1 - (1*3)[1=3][1,2])[2]","2">

q2 := < "1-(1*2)[2=3][1,2]", "3 - (2*3)[1=2][2,3]", " (1*2*3*)[2=3][3=4][1,5]" >

It should be noted, that none of the other equivalences holds.

So we will choose only one of the equivalences =3=, =4= and =5= to be the best. However, all our examples which show a difference between these properties look very unnatural. The next theorem states that for all usual query classes there is no real choice.

Theorem 6

If Q is a subset of FGEN, then S1 =3= S2 wrt.Q iff S1 =5= S2 wrt.Q.

Proof:

"<==" follows directly by theorem 2.3.

"==>"

We may assume four mappings in Q:

- q1 consistent and injective
- q2 inverse to q1
- p2 consistent and injective
- p1 inverse to p2.

Using the theorem of Cantor and Bernstein, especially with the more constructive proof of Koenig, one is able to construct a bijection between the set of instances of the schemes, see e.g. [Sier]. For our purpose this does not suffice, because we are looking for such a bijection in the class Q only. Furthermore, it must have an inverse mapping in Q.

In the following we will show that we don't need to construct a new mapping. It suffices to show, that

- q1 is consistent, injective and surjective
- q2 is inverse to q1.

Using theorem 2 this implies S1 =5= S2 wrt.Q.

Let [INS1] denote the set of classes of the partition of INS1 generated by the reflexive and transitive closure of the condition that two $I1, J1 \in$ INS1 with $I1 = p2(q1(J1))$ belong to the same class (symbolized by $[I1] = [J1]$).

Using $q1 \circ p2$ (instead of $p2 \circ q1$) we get a definition of [INS2] in an analogous way.

Every class consists of only a finite number of instances. Any instance of a class is mapped into another instance of the class by some iteration of $p2 \circ q1$ (respectively $q1 \circ p2$) and both p2 and q1 belong to FGEN.

(1)

For every $[I1] \in [INS1]$ the class $[q1(I1)] \in [INS2]$ is well-defined and $q1([I1]) \subseteq [q1(I1)]$.

$[q1(I1)]$ is well-defined since q1 is consistent.

Now, let $J1 \in [I1]$, say $J1=(p2 \circ q1)^m (I1)$.

Then $q1(J1) = q1 \circ (p2 \circ q1) \circ ... \circ (p2 \circ q1)\ (I1) = (q1 \circ p2) \circ ... \circ (q1 \circ p2)\ (q1(I1))$.

i.e. $q1(J1) \in [q1(I1)]$.

(2)

For every $[I2] \in [INS2]$ the class $[p2(I2)] \in [INS1]$ is well-defined and $p2([I2]) \subseteq [p2(I2)]$.

This can be proved in an analogous way.

(3)

For every $[I2] \in [INS2]$ it holds that $q1([p2(I2)]) = [I2]$.

" $\subseteq$ ":

Using (1), respectively the definition of [INS2], we get $q1([p2(I2)]) \subseteq [q1(p2(I2))] \subseteq [I2]$.

" = ":

Using (2) we get $p2([I2]) \subseteq [p2(I2)]$ and therefore we can conclude that $q1(p2([I2])) \subseteq q1([p2(I2)])$ holds. Using the " $\subseteq$ " proof it follows that $q1(p2([I2])) \subseteq q1([p2(I2)]) \subseteq [I2]$.

Because both q1 and p2 are injective and [I2] is a finite set, it follows that these inclusions are really identities.

(4)

It follows directly from (3) that q1 is surjective. Since q1 is consistent and injective and q2 is inverse to q1, we can finish this proof. ▪

We argue that is reasonable to consider only query classes which do not contain mappings being able to generate an unrestricted set of new values. Therefore our attention is now directed on =5=, the sharpest formalization of equivalence.

3. The Decidability of Equivalence

In this chapter we are concerned with cases in which the equivalence =5= is decidable. One has to restrict both the set of dependencies in the schemes and the queryclass Q which determines the sharpness of equivalence. First we will introduce a simple algorithm.

Definition 7

Input: database schemes S1, S2; query class Q;

method:

```
FOR ALL q1 ∈ Q1 DO
  IF q1 is consistent
  THEN
    FOR ALL q2 ∈ Q2 DO
```

```
    IF q2 is consistent AND
       q2 is inverse to q1 AND
       q1 is inverse to q2
    THEN write ( "Proof of equivalence by", q1, q2 );
         GOTO endmark;
    FI;
   OD;
  FI;
 OD ;
 write ( "The schemes are not equivalent." );
 endmark:
```

The sets Q1 and Q2 must be subsets of Q.

Theorem 6 implies that a jump to the endmark only appears if S1 =5= S2 wrt.Q holds. The converse is usually not true. To cover exactly the equivalence we have to provide a lot more:

- an effective construction of Q1 and Q2
- which ensures that they are finite sets
- and contain mappings for the proof of equivalence iff Q contains such ones;
- an algorithm which is able to decide whether a mapping in Q is consistent;
- an algorithm which is able to decide whether a mapping is inverse to an other mapping.

In the following subchapters we will show, that for schemes with dependencies in ALL $\cup$ EX and query classes included in RANP* or in RAND* all these demands can be fullfilled.

The algorithm for consistency and the algorithm for inversion are based on the following theorem of logic on the Bernay - Schoenfinkel Class (BSC) of first-order logic sentences. The sentences of BSC are equivalent to sentences in prenex normal - form with no existential quantifiers on the right of an universal quantifier.

Theorem 8

There exists an effective algorithm which decides for an arbitrary finite subset of BSC without equality and function symbols whether it has a finite model or not.

Proof: see [DG], pp. 79.

3.1 Algorithm for Consistency

The consistency is closely related with the implied constraint problem of [Klug] and [JAK]. A mapping q1 is consistent iff all the dependencies of scheme S2 are satisfied for all states in q1(INS1).

In [Klug] the allowed dependencies are functional dependencies and the so-called equality statements. The mappings are restricted to be in RAND*.

[JAK] consider relational mappings built up by restrictions and products and generalized dependency constraints, see [GJ], as allowed dependencies.

As [JAK] we will use theorem 8 but in a different way. For the following of this chapter let $D1 \subseteq ALL \cup EX$ be the set of dependencies of S1, $d \in ALL \cup EX$ a dependency of S2 and q1 a mapping in RANP* or in RAND*. We have to decide whether d is satisfied by all q1(I1) where $I1 \in INS1$. To use theorem 8 we will construct a set of sentences IC in BSC such that every finite model of IC corresponds to an instance $I1 \in INS1$ for which $\neg d$ is satisfied by q1(I1) and vice versa.

First we will mix $\neg d$ and q1.
Let q1' be the transformation (see [Ullm]) of **q1** into the domain relational calculus:

$q1' = < \{ x11,...,x1m_1 : f_1(x11,...,x1m_1 \}, ..., \{ xn1,...,xnm_n : f_n(xn1,...,xnm_n) \} >$

Substitute every occurence of an "i(y1,...,ymi)" in $\neg d$ by the formula "f_i(y1,...,ymi)" using appropriate renaming of variable symbols if needed. The resulting sentence is denoted by $\neg dq1$. If we add $\neg dq1$ as a dependency to those of S1 then for every instance I1 of this new scheme its image q1(I1) satisfy $\neg d$.

Now we want to show that $\neg dq1$ is expressible as a sentence in prenex normalform where no existential quantifier appears on the right of an universal quantifier. If $q1[i] \in RANP$ then the substitution of f_i doesn't change anything, because in f_i there are no quantifiers at all. If $q1[i] \in RAND$ then in f_i there appear only existential quantifiers. If $d \in ALL$, then in $\neg d$ there are only existential quantifiers and no problems arise. In the other case, if $d \in EX$ then in $\neg d$ only universal quantifiers appear. The substitution of f_i behaves well because we assumed in the definition of EX that every "i(...)" appears only under an even number of negations.

Set $IC' := D1 \cup \{ \neg dq1 \}$.

To get a set of sentences in BSC we have to

- avoid constant symbols (as preinterpreted function symbols)
- simulate the typing of S1
- avoid the equality sign (as a preinterpreted predicate symbol).

For the first task we will introduce a special predicate symbol c of arity 1 for every constant c' appearing in IC'.

Every sentence s of IC' with constants c1,...,ck will be transformed into " $\exists$ x1,...,xk: c1(x1) and ... and ck(xk) and s " (here xi is different from the other variable symbols of s and from other xj). To cover the semantics of constants we add

" $\exists$ x: c(x)",

" $\forall$ x,y: c(x) and c(y) ==> x=y " and

" $\forall$ x,y: c(x) and d(y) ==> $x \neq y$ " for every constant c' (and every constant d' different from c') to IC'.

To simulate the concept of typed schemes we will introduce a special predicate symbol T of arity 1 for every type T' in TYPES appearing in S1. Let

$\{T1,...,Tn\}$ be set of all these new predicate symbols.

We add the following sentences to IC':

" $\forall$ x: T1(x) or ... or Tn(x) "

" $\forall$ x: $\neg$Ti(x) or $\neg$Tj(x) " for all Ti' $\cap$ Tj' = ϕ .

" $\forall$ x: Ti(x) ==> Tj(x) " for all Ti' $\subseteq$ Tj'.

" $\forall$ x1,...,xm: i(x1,...,xm) ==> Ti1(x1) and ... and Tim(xm) "

for all i with S1[i] = < Ti1, ..., Tim >.

For every finite type T' $\in$ TYPES (with exactly n elements) we need additional sentences:

" $\exists$ x1,...,xn: T(x1) and ... and T(xn) and $\neg$ (x1=x2 or x1=x3 or ... or x1=xn or x2=x3 or ... or x2=xn or ... or x(n-1)=xn) ",

" $\forall$ x0,x1,...,xn: T(x0) and ... and T(xn) ==> x0=x1 or x0=x2 or or x1=x2 or ... or x1=xn or ... or x(n-1)=xn ".

At last we must bind constants to their types:

" $\forall$ x: c(x) ==> T(x) " for all predicate symbols c corresponding to the constant c' and all types T which contain c'.

To handle the equality sign as a normal predicate symbol we will use the axioms of equality of [Reiter]. Only universal quantifiers are needed here.

By construction it should be clear, that the resulting set IC of sentences is a (finite) subset of BSC. It should also be clear, how to use this construction for an algorithm for the decision of consistency. So we get :

Theorem 9

The consistency is effectively decidable with respect to

- mappings in RANP* or in RAND*
- schemes with dependencies in ALL $\cup$ EX.

3.2 Algorithm for Inversion

The decision whether a mapping is inverse to another one can be handled as a special case of the decision of the equivalence of two mappings (exactly: of their syntatical description).

Definition 10

q1 is <u>equivalent</u> to p1 iff $\forall$ I1 $\in$ INS1: q1(I1) = p1(I1).

Note that we use here equivalence restricted to the set of instances, in [Klug] called 'equivalence', and not equivalence with respect to all states, in [Klug] called 'strong equivalence'. [ASU] distinguish between 'algebraic equivalence', 'weak equivalence' and 'strong equivalence' of mappings. The last one as defined in [GM] is the same as our equivalence. All these references present algorithms to decide equivalence in special cases. [SY] generalize the results of [ASU]. [IL2] is concerned with the undecidability of the equivalence of mappings, whereas [IL1] shows how to decide it under the open-world assumption.

Theorem 11

q2 is inverse to q1 iff (q2 ○ q1) is equivalent to id_INS1. Here id_INS1 denote the identity on INS1.

Proof: obvious.

Using the result of the previous chapter we easily get the following fact.

Theorem 12

The equivalence of relational mappings

- in RANP* or in RAND*
- between schemes with dependencies in ALL ∪ EX is effectively decidable.

Proof:

Let q1, p1 be given and let |S2| = n.

Define S3 := < S2[1], ..., S2[n], S2[1], ..., S2[n] : D3 >
and r1 := < q1[1], ..., q1[n], p1[1], ..., p1[n] >, where D3 consists of a set of dependencies, such that

$\forall$ I3 ∈ TYPE3: I3 ∈ INS3 <==> $\forall$ 1 ≤ i ≤ n : I3[i] = I3[i+n]

holds. It is obvious,that D can be constructed as a set of full inclusion dependencies, which can be expressed as " $\forall$ x1,...,xn: i(x1,...,xn) ==> j(x1,...,xn) " and thus D is in ALL. The mapping r1: TYPE1 -> TYPE3 is constructed in a way, such that q1 and p1 are equivalent iff r1 is consistent.

Since D3 is a subset of ALL ∪ EX, the dependencies of S1 are assumed to be in ALL ∪ EX and r1 is in RANP* or in RAND* we can finish this proof with a reference to Theorem 9. ▪

3.3 Finite Mapping Sets

The algorithm of definition 7 only works if we are able to construct finite subsets Q1 and Q2 of Q which contain mappings for the proof of the equivalence =5= (if Q contains such mappings at all). Our attention is directed on RANP* and RAND*. First we will define their normalforms.

Definition and Theorem 13

Every mapping q in RANP* is expressible in a way such that

- no selection refers to a subexpression in which a product appears (for short: selection before product)
- product before restriction
- restriction before permutation

- permutation before difference
- difference before union
- no projection appears.

An expression of this form will be called to be in <u>normalform</u> (for RANP*).

Every mapping q in RAND* is expressible in a way such that

- selection before product
- product before restriction
- restriction before projection
- projection before union
- no permutation and no difference appears.

An expression of this form will be called to be in <u>normalform</u> (for RAND*).

Proof:

For the proof for RAND* see [Klug].

Then for RANP* it suffices to show how to shift the difference on its right place (let Ei be arbitrary subexpressions):
(E1 - E2) [i='v'] --> E1 [i='v'] - E2 [i='v']
(E1 - E2) * E3 --> (E1 * E3) - (E2 * E3)
(E1 - E2) [i=j] --> E1 [i=j] - E2 [i=j]
(E1 - E2) [perm] --> E1 [perm] - E2 [perm] ,
where perm = " i1->i2->...->ik->i1"
(E1 ∪ E2) - E3 --> (E1 - E3) ∪ (E2 - E3)
E1 - (E2 ∪ E3) --> (E1 - E2) - E3

Next we will show what must be assumed to get finite mapping classes.

Theorem 14

Let C ⊂ VALUES be a finite set.

Then the set of mappings TYPE1 -> TYPE2 in RANP* with selections using only constants of C is finite and can be enumerated in an effective way.

Proof:

The proof is based on the following observations, whereas the details are left to the reader. Every such mapping has a normalform according to theorem 13. Since projection is not allowed the number of products is restricted by TYPE1 and TYPE2. ▪

In chapter 3.4 we will show how to construct such a finite set C of values.

To get an analogous result for RAND* we have to make an additional assumption to restrict the arity of a subexpression. For example there is a state of a binary relation with 100 tuples, whose transitive closure (see [AU]) is expressible in RAND*, but needs at least 99 product signs. So we would not get a finite set of mappings if we don't exclude such cases.

Theorem 15

Let $C \subseteq$ VALUES be a finite set and l be a natural number.

Then the set of mappings TYPE1 -> TYPE2 in RAND* with selection constants in C and no more than l product signs in every subexpression which doesn't contain a union sign or a difference sign is finite and can be enumerated in an effective way.

Proof: Similar to the proof of theorem 14. ▪

In chapter 3.5 we are concerned with the question how to get such a natural number l with respect to the given schemes.

3.4 Relevant Selections

First we will characterize the selection constants appearing in a relational expressing in a better way. In the following K, L, M are finite subsets of VALUES.

Definition 16

A mapping f : VALUES -> VALUES is called a K-isomorphism iff

- f is bijective and totally computable
- $\forall k \in K$: $f(k) = k$
- $\forall T \in$ TYPES: $f(T) = T$. ▪

In the canonical way isomorphisms are extended to tuples, states and sets of states. See [CH] or [Hull] for similar definitions. The following properties hold:

- if f, g are K-isomorphisms, then $f \circ g$, f^{-1} are K-isomorphisms
- for any scheme S1: f(TYPE1) = TYPE1.

Definition 17

A scheme mapping q1 is compatible with K-isomorphisms iff for every K-isomorphism f $q1 \circ f = f \circ q1$ holds. ▪

In [Banc] and [CH] a similar property is used in the definition of completeness of a query language. It is obvious that every relational expression whose selection constants are contained in K is compatible with K-isomorphisms. The other direction is not as simple. This is because we allow types with a finite number of values. The construction in the proof of theorem 19 shows how to get a weaker result.

Corresponding to the compatibility of mappings we will formulate an analogous property for database schemes.

This property is related to the monotonicity (see e.g. [SY]) and the additivity (see e.g. [AU]) of mappings. It should be noted, that every mapping with breadth l is also monotone. The converse does not hold. The transitive closure (see e.g. [AU]) is a prominent counterexample for it.

The next theorem will state how we can use the breadth to restrict the number of products.

Theorem 21

Let TYPES contain only disjoint sets.

If there exist a natural number l and a finite subset M of VALUES, such that q1 is totally computable, compatible with M-isomorphisms, member of NGEN and has breadth l

then q1 is expressible in the normalform of RAND*, where no more than l + m - 2 product signs appear in every of its subexpressions which have no union or difference sign. Here $m := \max \{ |S2[j]| : 1 \le j \le |S2| \}$.

Proof:

Choose a finite set of states $B1, \ldots, Bp \in TYPE1$ as substitutes of all instances with no more than l tuples. That means:
$\{ A1 \in TYPE1 : |A1| \le l \} = \{ f(Bi) : f \text{ is an M-isomorphism}, 1 \le i \le p \}$.

To do this one can define a finite set $L \subseteq VALUES - M$ (where $|L| \ge \max \{ SYMB(A1) : A1 \in TYPE(S1), |A1| \le l \}$) and enumerate all states with values in $L \cup M$ and with no more than l tuples.

Then for every $A1 \in TYPE1$:

$$q1(A1) = \bigcup_{1 \le i \le p} \ \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} q1(f(Bi))$$ by construction of B1,...,Bp and since q1 has breadth l

$$= \bigcup_{1 \le i \le p} \ \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f(q1(Bi))$$ since q1 is compatible with M-isomorphisms

$$= \bigcup_{1 \le i \le p} \ \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f\Big(\bigcup_{\substack{D \subseteq q1(Bi) \\ |D| = 1 \\ D \in TYPE2}} D \Big)$$

$$= \bigcup_{1 \le i \le p} \ \bigcup_{\substack{D \subseteq q1(Bi) \\ |D| = 1 \\ D \in TYPE2}} \ \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f(D)$$ since f is a canonical extension

For every $1 \le j \le |S2|$ then

Now f is defined by exchanging every $w \in SYMB(A1) \cap (L - M)$ with its associated element, i. e.

$$f(x) := \begin{cases} w' & \text{if } x = w \in SYMB(A1) \cap (L - M) \\ w & \text{if } x = w' \text{ with } w \in SYMB(A1) \cap (L - M) \\ x & \text{else.} \end{cases}$$

By definition $f = f^{-1}$ holds.

(3)

We will show for an arbitrary $I1 \in INS1$: $p1(I1) \in INS2$. Let f be the M-isomorphism for I1 as constructed in (2).

Because of $K \subseteq M$ we get $f(I1) \in INS1$ and $f(INS2) \subseteq INS2$, since both schemes are compatible with K-isomorphisms. So

$$\begin{aligned} p1(I1) &= p1(f(f(I1))) \\ &= f(p1(f(I1))) && \text{,since } p1 \text{ is compatible with M-isomorphisms} \\ &= f(q1(f(I1))) && \text{,using (1)} \\ &\in f(INS2) && \text{,since } q1 \text{ is consistent} \\ &\subseteq INS2 \text{ (4)} \end{aligned}$$

We will show $p2(p1(I1)) = I1$ for an arbitrary $I1 \in INS1$.

Using the argumentation of (3) we know that $p1(I1) = f(q1(f(I1)))$ and so $p2(p1(I1)) = p2(f(q1(f(I1))))$.

Since p2 is compatible with M-isomorphisms we get $p2(p1(I1)) = f(p2(q1(f(I1))))$.

Because $q1 \in NGEN$ we know that $SYMB(q1(f(I1))) \cap (L - M) = \phi$
and by (1) we get $p2(p1(I1)) = f(q2(q1(f(I1))))$.
Since q2 is inverse to q1 $p2(p1(I1)) = f(f(I1)) = I1$. ▪

Using theorem 6 we see that this approximation can also be used for =5=. In the case of RANP* we then get finite Q1 and Q2 for the algorithm of definition 7 by theorem 14.

3.5 Number of Products

In chapter 3.4 we are concerned with the compatability with isomorphisms to characterize the set of selections needed to describe a mapping as a relational expression. Now we will formulate a property which corresponds to the number of products.

Definition 20

Let l be a natural number.

A mapping q1 is of <u>breadth</u> l iff for all $A1 \in TYPE1$

$$q1(A1) = \bigcup_{\substack{B \in TYPE1 \\ B \subseteq A1 \\ |B| \leq l}} q1(B) \quad \text{holds.}$$

Definition 18

A scheme S1 is <u>compatible with K-isomorphisms</u> iff for every K-isomorphism f $f(INS1) \subseteq INS1$ holds. ▪

From our definition of database schemes (especially of their dependencies) it follows that a scheme is compatible with K-isomorphisms if in its dependencies only constants out of K appears. Therefore we are able to construct such a finite set of values for a given scheme in a simple way.

Now we are able to give the main theorem of this chapter.

Theorem 19

Let $Q \subseteq$ RALG* be a mapping class which contains the restriction (e.g. RANP* or RAND*). Let $K \subseteq$ VALUES be a finite set and S1, S2 be both compatible with K-isomorphisms.

Then we can effectively construct a finite set M, such that
S1 <3< S2 wrt. Q iff S1 <3< S2 wrt. M_Q,
where M_Q := { $q \in Q$: in q appears no selection constant not contained in M }.

Proof:

"<==" is trivial for every M.

"==>"

Let $q1 \in Q$ be consistent and $q2 \in Q$ be inverse to q1. We will construct two new mappings p1, p2 $\in$ M_Q having the same properties.

Let L := { w : w is a selection constant of q1 or q2 } and
M := K $\cup$ { $w \in T$: $T \in$ TYPES, T appears in S1 or S2, T is finite }.

Both L and M are finite sets. Obviously q1 and q2 are both compatible with L-isomorphisms.

Let p1 (resp. p2) be the transformation of q1 (resp. q2) implied by the following rules :

r [i='w'] --> r [i$\neq$i] for $w \in L - M$,

r [i$\neq$'w'] --> r for $w \in L - M$, here r stands for a relational subexpression.

Since in p1 and p2 only selection constants of M appear, both of the mappings belong to M_Q. So they are compatible with M-isomorphisms.

We have to show that p1 is consistent and that p2 is inverse to p1.

(1)

Obviously qi(Ai) = pi(Ai) holds for all Ai $\in$ TYPEi (i = 1,2) with SYMB(Ai) $\cap$ (L - M) = ϕ .

(2)

For a given A1 $\in$ TYPE1 there exists a M-isomorphism f such that SYMB(f(A1)) $\cap$ (L - M) = ϕ .

In order to define such a M-isomorphism we consider any w $\in$ SYMB(A1) $\cap$ (L - M). Let T be the smallest type containing w. By definition of M T is infinite and thus we can associate w with a new element w' $\in$ T - ((L - M) $\cup$ SYMB(A1)).

$$q1(A1)[j] = \bigcup_{1 \le i \le p} \bigcup_{\substack{s \in q1(Bi)[j] \\ s \text{ is a tupel}}} \Big(\bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f(s) \Big) \quad \text{holds.}$$

The number of different i and s in that formula is finite. Let i and s be fixed. It suffices to show that the mapping

$$:= \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f(s) \text{ does not need more than } l + m - 2 \text{ product signs.}$$

To do this we first choose an enumeration of the tuples of Bi such that $Bi = < \{t11,...,t1n_1\}, \{t21,...,t2n_2\}, ..., \{tv1,...,tvn_v\} >$.

Let $t := < t11, ..., t1n_1, t21, ..., t2n_2, ..., tv1,..., tvn_v >$.

Define the sequence of products, corresponding to t, which maps Bi into t. Since $|Bi| \le l$ there appear no more than l-1 product signs. Then use a sequence of selections, so that
for all $w \in M$ and $1 \le i \le |t|$ there is a " [i = 't[i]'] " iff $t[i] \in M$,
and there is a " [$i \ne$ 'w'] " iff $t[i] \notin M$.

Finally we need a sequence of restrictions, such that for all $i,j \in \{ 1,...,|t| \}, i < j$ there is a " [i = j] " iff $t[i] = t[j]$ and there appears a " [$i \ne j$] " iff $t[i] \ne t[j]$.

Let p1 denote the constructed relational mapping.

Obviously $\forall A1 \in TYPE1$: $Bi \subseteq A1 \iff t \in p1(A1)$ holds.
Since p1 is compatible with M-isomorphisms f
$\forall A1 \in TYPE1$: $f(Bi) \subseteq A1 \iff f(t) \in p1(A1)$ holds.

Therefore we know that $p1(A1) \supseteq \bigcup_{\substack{f \text{ is M-isom.} \\ f(Bi) \subseteq A1}} f(s)$.

The other inclusion is obtained by
$\forall A1 \in TYPE1$: $p1(A1) \subseteq \{ f(t) : f \text{ is M-isomorphism} \}$.

This only holds, because we have assumed that all types contain different values. In this case we are able to construct a M-isomorphism f for a given $t' \in p1(A1)$.

The last step of the construction of the relational mapping must build up f(s) from f(t). Because of $q1 \in NGEN$, a projection usually suffices to describe it. But if a value appears in s more times than it does in t, we need additional products (and restrictions to link such a copied tuple tij with t). A simple reflection shows that no more than m - 1 additional product signs are needed. The normalform can be reached without new products. ▪

A simple induction would show that the converse implication of theorem 21 is also true. So one would get a characterization of RAND* not using syntactic criterions, provided that TYPES contains only disjoint sets of values.

We will now define a suitable property for database schemes.

Definition 22

Let l be a natural number.
The database scheme S1 is of <u>breadth</u> l iff

$$\text{for all } I1 \in INS1 \quad I1 = \bigcup_{\substack{J \in INS1 \\ J \subset I1 \\ |J| \leq l}} J \quad \text{holds.}$$

The breadth of a scheme is a special case of the locality of [IS] and the distributivity of [IS]. Although looking similar the boundedness of [GV] and the boundedness of [AV] are scarcely related to the breadth.

We will now formulate the main theorem of this chapter.

Theorem 23

Let S1 and S2 be database schemes with breadth l.
Then one can effectively compute a natural number w such that
S1 =5= S2 wrt. NGEN* ∩ { q is monotone } iff
S1 =5= S2 wrt. NGEN* ∩ { q is monotone, totally computable and has breadth w }.

Proof:

"<==" is trivial for every number w.

"==>"

Let q1, q2 ∈ NGEN* be monotone, consistent, injective and surjective, each one being inverse to the other.

Choose w such that

(1) $\forall$ I1 ∈ INS1: $|I1| \leq l$ ==> $|q1(I1)| \leq w$ and
(2) $\forall$ I2 ∈ INS2: $|I2| \leq l$ ==> $|q2(I2)| \leq w$ holds.

One way to do this is the following:

$ms1 := \max \{ |SYMB(A1)| : A1 \in TYPE1, |A1| \leq l \}$
$w1 := \max \{ |A2| : |SYMB(A2)| \leq ms1 \}$
Since q1 ∈ NGEN* holds, w1 can be used as w to fullfill (1).
In the same manner we get w2 to fullfill (2) and we finally set
$w := \max \{ w1, w2 \}$ to fullfill both (1) and (2).

We will define substitutes p1, p2 for q1, q2 which are in the demanded mapping class:

$$\text{for all } A1 \in TYPE1 \quad p1(A1) := \bigcup_{\substack{B \in TYPE1 \\ B \subset A1 \\ |B| \leq w}} q1(B)$$

and

for all $A2 \in TYPE2$ $\quad p2(A2) := \bigcup_{\substack{B \in TYPE2 \\ B \subseteq A2 \\ |B| \leq w}} q2(B)$.

It suffices to show that $q1(I1) = p1(I1)$ holds for all instances $I1 \in INS1$ (but not necessarily for all states). The analogous result for p2 can be obtained in the same way.

" $\subseteq$ ":

Since q1 is consistent and S2 has the breadth 1 we get for an

arbitrary $I1 \in INS1$: $\quad q1(I1) = \bigcup_{\substack{G2 \in INS2 \\ G2 \subseteq q1(I1) \\ |G2| \leq 1}} G2$

Since q1 is surjective each G2 is represented by q1(J1) for an $J1 \in INS1$:

$$q1(I1) = \bigcup_{\substack{q1(J1) \in INS2 \\ q1(J1) \subseteq q1(I1) \\ |q1(J1)| \leq 1 \\ J1 \in INS1}} q1(J1)$$

$$= \bigcup_{\substack{q1(J1) \in INS2 \\ q1(J1) \subseteq q1(I1) \\ |q1(J1)| \leq 1 \\ J1 \in INS1 \\ |J1| \leq w}} q1(J1) \qquad \text{using (2), q2 is inverse to q1}$$

$$\subseteq \bigcup_{\substack{q1(J1) \in INS2 \\ q1(J1) \subseteq q1(I1) \\ J1 \in INS1 \\ |J1| \leq w}} q1(J1)$$

$$= \bigcup_{\substack{q1(J1) \in INS2 \\ J1 \subseteq I1 \\ J1 \in INS1 \\ |J1| \leq w}} q1(J1) \qquad \text{using the monotonicity of q2, which is inverse to q1}$$

$$= \bigcup_{\substack{J1 \in INS1 \\ J1 \subseteq I1 \\ |J1| \leq w}} q1(J1) \qquad \text{since q1 is consistent}$$

$$\subseteq \bigcup_{\substack{J1 \in TYPE1 \\ J1 \subseteq I1 \\ |J1| \leq w}} q1(J1) \qquad \text{since } INS1 \subseteq TYPE1$$

$$= p1(I1)$$

" $\supseteq$ ": By monotonicity of q1. ▪

The property of a scheme to have finite breadth is essentially a demand on its dependencies. We must really restrict the class of allowed dependencies as the following example will show.

Example 24

S := < <N,N>, <N,N> : { " $\forall$ x,y: 1(x,y) ==> $\exists$ z: 2(x,z) ",
" $\forall$ x,z: 2(x,z) ==> $\exists$ y: 1(y,z) " } >,
where N should denote the set of all natural numbers.

I := < { <n,n> : $1 \leq n \leq m$ }, { <n-1,n> : $2 \leq n \leq m$ } >,
where m is a given natural number.

Any instance J with $J \subseteq I$ and $<1,1> \in J[1]$ is identical to I. This is implied by the two inclusion dependencies of S.

Therefore S cannot have breadth l if l < 2*m - 1.

Because m is choosen arbritrary S does not have finite breadth at all. ▪

If we restrict our attention on dependency classes already shown to be good-natured with respect to the decision algorithms, we are able to compute an approximation of the breadth of a scheme.

Theorem 25

Let S1 be a scheme with dependencies in ALL $\cup$ EX. Let TYPES contain only disjoint sets.

Then one can effectively compute a number l such that S has breadth l.

Sketch of the proof:

Set l := (m + k + e) ** a , where

- a is the number of attributes of S,
- m is the greatest number of attributes of a relation scheme of S1,
- k is the number of constants appearing in dependencies of S1,
- e is the number of occurences of existential quantification in the dependencies of S, provided they are written in prenex normalform.

Since for all $I1 \in INS1$ $I1 = \cup$ { B : $B \subseteq I1$, $|B| = 1$, $B \in TYPE1$ } holds, it suffices to show that for given $I1 \in INS1$, $B \in TYPE1$, $|B| = 1$ there is an instance $J1 \in INS1$, so that $B \subseteq J1 \subseteq I1$ and $|J1| \leq l$ holds.

We will build a set of sentences in BSC to characterize such an J1. If they have a finite model at all, they have a model with no more than m + k +

e individuals. The correspondence of suc[illegible] del to a state with no more than l tuples is obvious.

K := { k : k $\in$ SYMB(B) or k appears as a constant in a dependency of S }

(1)

Let in_I $\in$ ALL be a sentence, such that for every finite model M there exist a corresponding state A1 $\in$ TYPE1 and a K-isomorphism f such that f(A1) $\subseteq$ I1. These sentences can be constructed in an obvious manner. The proof of theorem 21 contains a similar construction (if translated into the relational calculus). Only constants in K appear in this sentence.

(2)

Let B_in be a sentence without quantifier, so that every finite model M corresponds to a state A1 such that B $\subseteq$ A1 holds.
A sentence "i(v1,...,vk)" suffices, if <v1,...,vk> $\in$ B[i]. Only constants of K appear in B_in.

(3)

Let D be the set of dependencies of S. It is assumed that D $\subseteq$ ALL $\cup$ EX.

Let E := D $\cup$ { in_I, B_in }.

Every model of E corresponds to a state A1 $\in$ TYPE1, so that

- there exists a M-isomorphism f with f(A1) $\subseteq$ I1 (by (1))
- A1 $\in$ INS1 (by D $\subseteq$ E)
- for the above chosen f f(A1) $\in$ INS1 holds (since S is compatible with K-isomorphisms)
- B $\subseteq$ A1 (by (2))
- for the above chosen f B $\subseteq$ f(A1) holds (since SYMB(B) $\subseteq$ K and so f(B) = B holds).

Therefore we know of an f(A1) $\in$ INS1 with B $\subseteq$ f(A1) $\subseteq$ I1 and must finally show that there is a model of E with no more than m + k + e individuals.

(4)

We want to use the theorem of Herbrand.

To do this we have to eliminate the equality and the constants as preinterpreted objects.

First we will eliminate the constants. We build the conjunction of all sentences of E, substitute every constant symbol with a new specific variable symbol, bind these variables globally with existential quantifiers and add atoms which demand that they all have different values. Let E' denote this new sentence.
Since E $\in$ ALL $\cup$ EX it is obvious that E' belongs to BSC.
Every model of E is also a model of E'. For every model of E' there is a model of E having the same number of individuals.

Next we use the axioms of equality of [Reiter] to handle the equality symbol as a normal predicat symbol. Let E" $\subseteq$ BSC denote the constructed set of sentences. Every model of E' is also a model of E". For every model of E" there is a model of E' having no more individual symbols.

The number of existential quantification in E'' is not greater than m + k + e. This is also the number of terms of the Herbrand universe of E'', since the Skolemization of E'' only generate function symbols with arity 0. This is because E'' is in BSC.

Using Herbrand's theorem we can conclude that there is a model of E with no more than m + k + e individuals if there is a model at all. ▪

3.6 Summary

Theorem 26

Let S1 and S2 be schemes with dependencies in ALL ∪ EX.
Let TYPES contain only disjoint sets.
Let Q be a subset of RANP* or of RAND*, which contains the restriction.

Then the algorithm of definition 7 can be used to decide whether S1 =5= S2 wrt.Q or not.

Proof:

In chapter 3.1, theorem 9 we have suggested a way to decide if a mapping is consistent. In chapter 3.2, theorem 12 we have shown how to decide the equivalence of two mappings. Using theorem 11 it is obvious how to use this for a decision whether a mapping is inverse to another mapping. In chapter 3.3, theorem 14 it is shown how to enumerate finite sets Q1 and Q2 in the case of Q ⊂ RANP* if we know a finite set of selection constants relevant for the proof of equivalence. In chapter 3.4, theorem 19 can be used to get such a set of constants. In chapter 3.3, theorem 15 we have seen, that in the case of Q ⊂ RAND* we need additionally an approximation of the number of products to get finite sets Q1 and Q2. In chapter 3.5, theorem 21 we have formulated the breadth of a mapping as a sufficient criterion for this matter. In chapter 3.5, theorem 25 we suggested a way to approximate the breadth of a scheme. In chapter 3.5, theorem 23 we finally have proved that there is no need to consider mappings with a breadth not related to the breadth of the schemes. ▪

4. Undecidability of Equivalence

In chapter 3 we have only considered schemes with dependencies in ALL ∪ EX and mapping classes below the relational algebra. Now we want to show the reason for these restrictions.

4.1 More Powerful Dependency Classes

In the following we will use "implication" as "finite implication" (see [CFP], [CLM]), because we only deal with finite database states. We use a known result concerning with the implication problem for database dependencies.

Theorem 27

It is not decidable whether S1 =3= S2 wrt.Q holds

- where S1, S2 ranges over all schemes with functional dependencies and (binary) inclusion dependencies
- Q is a mapping class which includes the identity mapping (as assumed in chapter 1) and is contained in FGEN.

Proof:

See [Mitch] for the undecidability of the (finite) implication problem for functional dependencies and binary inclusion dependencies.

Let D, { d } be arbitrary sets of dependencies of these classes, expressed in the notation of chapter 1. It should be noted, that inclusion dependencies are not included in ALL $\cup$ EX.

Using the signature of both dependency sets it is simple to construct two schemes S1 and S2 so that TYPE1 = TYPE2 and their set of dependencies is D resp. D $\cup$ {d}. Only one infinite type should appear in the scheme definitions.

We want to show that this is already a correct reduction of the implication problem into the equivalence problem for database schemes.

"==>"

If d is implied by D, then INS1 = INS2 holds. The identity mapping can be used to prove S1 =3= S2 wrt.Q.

"<=="

If S1 =3= S2 wrt.Q holds, then there is a mapping q1 in Q which is consistent and injective.
For finite subsets V $\subseteq$ VALUES let (for i = 1,2)
INSSYMB(Si,V) := { Ii $\in$ INSi: SYMB(Ii) $\subseteq$ V }.

Since q1 $\in$ FGEN there is a finite N $\subseteq$ VALUES such that q1 only generates new values in N.
Since q1 is consistent for any finite V $\in$ VALUES
q1 (INSSYMB(S1,V$\cup$N)) $\subseteq$ INSSYMB(S2,V$\cup$N).

By definition of S1 and S2 it is obvious, that
INSSYMB(S2,V$\cup$N) $\subseteq$ INSSYMB(S1,V$\cup$N).
Because q1 is injective and INSSYMB(S1,V$\cup$N) is a finite set, it follows that INSSYMB(S1,V$\cup$N) = INSSYMB(S2,V$\cup$N).

Clearly every instance of S1 is contained in INSSYMB(S1,V$\cup$N) for some finite V (since any instance is assumed to be finite). So we get INS1 = INS2.

This means that d is implied by D.

4.2 More Powerful Mapping Classes

We are not aware of a result concerning the equivalence directly. We will show the undecidability of the consistency, injectivity and the surjectivity of mappings ranging over the whole relational algebra. The following theorem deals as the base for it.

Theorem 28

Let T $\in$ TYPES be an infinite set.

It is not decidable whether q1(A1) = ϕ holds for all A1 $\in$ TYPE1, where

- S1 ranges over all database schemes without dependencies
- q1 ranges over all mappings TYPE1 -> T in the relational algebra.

Proof:

A simple construction reduces the equivalence problem for relational expressions on the above mentioned problem since we can use the difference operator.

It is known that the equivalence problem of relational expressions is not decidable, see [IL2]. [Klug] cites a result of [Solo] considering expressions without selections. A direct way to prove it can be based on the undecidability of the first-order logic (finite models only). ▪

Theorem 29

The consistency with respect to

- mappings in RALG
- from a database scheme without dependencies
- into a database scheme with one functional dependency, one exclusion dependency (see [CV]), or one unary inclusion dependency (see [KCV])

is undecidable.

Proof:

Let q1: TYPE1 = INS1 -> T be given. We will construct a scheme S2 and a mapping p1, which is consistent iff q1(A1) = ϕ for all A1 $\in$ TYPE1. This suffices to use theorem 28.

one functional dependency:

p1 := < (SYM * SYM * q1)[1,2] >, where SYM is a relational expression which supply the relation consisting of all values appearing in a state A1,

d := " $\forall$ x, y, z: 1(x,y) and 1(x,z) ==> y = z ", which is a functional dependency,

S2 := < < T, T > : { d } >.

Then p1 is consistent iff p1(INS1) $\subseteq$ INS2 iff $\forall$ I1 $\in$ TYPE1: | SYMB(I1) | $\leq$ 1 or q1(I1) = ϕ .

Since we are able to decide whether q1(I1) = ϕ holds for all I1 $\in$ INS1 with |SYMB(I1)| $\leq$ 1, this suffices to use theorem 28.

one exclusion dependency:

p1 := < q1, q1 >,
d := " $\forall$ x: $\neg$1(x) or $\neg$2(x) ", which is an exclusion dependency,
S2 := < < T >, < T > : { d } >.

Then p1 is consistent iff q1(I1) = ϕ for all I1 $\in$ INS1.

one unary inclusion dependency:

p1 := < q1, q1 - q1 >,
d := " $\forall$ x: 1(x) ==> 2(x) ", which is a unary inclusion dependency,
S2 := < < T >, < T > : { d } >.

Then p1 is consistent iff q1(I1) = ϕ for all I1 $\in$ INS1.

Theorem 30

The injectivity with respect to

- mappings in RALG
- schemes without dependencies

is undecidable.

Proof:

Let the mapping q1: TYPE1 = INS1 -> T be given, where w.l.o.g. |S1| = n, |S1[i]| = ai (for $1 \le i \le n$).

For $1 \le i \le n$ define
p1[i] := (i * (SYM - (SYM * q1)[1]))[1, 2, ..., ai],
where SYM is the same as in the proof of theorem 35.
It is obvious, that for every A1 $\in$ TYPE1 p1(A1) $\in$ { ϕ , A1 } holds.

Since p1 does not generate values we know that p1(ϕ) = ϕ.

Therefore p1 is injective iff p1(A1) = A1 for all A1 $\in$ TYPE1. This means that q1(A1) = ϕ for all A1 $\in$ TYPE1 and theorem 28 can be used to show the undecidability of injectivity.

Theorem 31

The surjectivity with respect to

- mappings in RALG
- schemes without dependencies

is undecidable.

Proof:

The same construction as for the proof of theorem 30 is used.

It is obvious that p1 is surjective
iff p1(A1) = A1 for all A1 $\in$ TYPE1 holds, i.e.
iff q1(A1) = ϕ for all A1 $\in$ TYPE1 holds.

5. Conclusions

There remain some open questions. Using theorem 21 (and assuming that the typing of database schemes is not of interest or behaves well) we get an exact characterization of mappings expressible by the relational algebra without the difference operator. They are totally computable, are compatible with M-isomorphisms (where M is a finite set of values), does not generate new values and have finite breadth. There is no idea of a set of properties characterizing the relational algebra without the projection operator.

When we considered the whole relational algebra in chapter 4 we have only shown the undecidability of some properties necessary for equivalence. There is no result dealing with the equivalence itself.

In chapter 3 we have shown the decidability of consistency and inversion. We do not know whether there are algorithms for the injectivity and surjectivity, too. [IL2] deals with the losslessness (the same as injectivity) under the open-world assumption, but this is a much more weaker property than our injectivity. Restricting the attention on the algebra without projection (and schemes with dependencies in ALL $\cup$ EX) the decidability of the surjectivity can be proved in a similar manner as the decidability of the consistency.

In chapter 3 we only consider mappings in RANP* or in RAND*, but do not handle with (RANP $\cup$ RAND)*, which would be the "mixing of both classes". We do not know how to change chapter 3.5 for this purpose.

Although defining TYPES as a hierarchy in chapter 1 in theorem 21 we have assumed that it contains only disjoint sets of values. There are some possibilities to change the definition of an M-isomorphism in a way that this assumption would not be necessary, but other difficulties would appear elsewhere in chapter 3.5.

At last it should be mentioned that this paper does not deal with very efficient ways to decide the equivalence. Full use of the typing of database schemes will speed up our algorithm. Cardinality comparisons as described in [Hull] can probably be used in more advanced decision algorithms.

References:

[ABM] G.Ausiello, C.Batini, M.Moscarini: 'Conceptual relations between databases transformed under join and projection', in: Proc. Symp. Math. Found. of Comp. Sc., 9, 1980, pp. 123 - 136

[AABM] P.Atzeni, G.Aussiello, C.Batini, M.Moscarini: 'Inclusion and equivalence between relational database schemes', in: Theoretical Computer Science, 19, 1982, pp. 267 - 285

[AP] P.Atzeni, D.S.Parker: 'Assumptions in relational database theory', in: ACM Symp. on Princ. of Database Systems, 1, 1982, pp. 1 - 9

[ASU] A.V.Aho, Y.Sagiv, J.D.Ullman: 'Equivalences among relational expressions', in: SIAM Journ. of Computing, 8, 1979, pp. 218 - 246

[AU] A.V.Aho, J.D.Ullman: 'Universality of data retrieval languages', in: ACM Symp. on Princ. of Programming Languages, 6, 1979, pp. 110 - 117

[AV] S.Abiteboul, V.Vianu: 'Transactions in relational databases', in: Proc. ACM Int. Conf. on Very Large Data Bases, 1984

[Banc] F.Bancilhon: 'On the completeness of query languages for relational data bases', in: Proc. Symp. Math. Found. of Computer Science, 7, 1978, pp. 112 - 123

[BBG] P.A.Bernstein, C.Beeri, N.Goodman: 'A sophisticated introdution to database normalization theory', in: Proc. ACM Int. Conf. on Very Large Data Bases, 4, 1978, pp. 113 - 124

[Biller] H.Biller: 'On the equivalence of database schemes - a semantic approach to data translation', in: Information Systems 4. 1979 , pp. 35 - 47

[BMSU] C.Beeri, A.O.Mendelzon, Y.Sagiv, J.D.Ullman: 'Equivalence of relational database schemes', in: SIAM Journ. of Computing, 10, 1981, pp. 352 - 370

[CFP] M.Casanova, R.Fagin, C.H.Papadimitiou: 'Inclusion dependencies and their interaction with functional dependencies', in: ACM Symp. on Princ. of Database Systems, 1, 1982, pp. 171 - 176

[CH] A.K.Chandra, D.Harel: 'Computable queries for relational data bases', in: Journ. of Computer and System Sciences, 21, 1980, pp. 156 - 178

[CLM] A.K.Chandra, H.R.Lewis, J.A.Makowsky: 'Embedded implicational dependencies and their inference problem', in: ACM Symp. on Theory of Computing, 13, 1981, pp. 342 - 354

[Codd] E.F.Codd: 'Further normalizations on data base relational model', in: Data Base Systems (R.Rustin ed.), Prentice-Hall, Englewood Cliffs, 1972, pp. 33 - 64

[CV] M.A.Casanova, V.M.P.Vidal: 'Towards a sound view integration methodology', in: ACM Symp. on Princ. of Database Systems, 2, 1983, pp. 36 - 47

[DG] B.Dreben, W.D.Goldfarb: 'The decision problem: solvable classes of quantificational formulas', Addison-Wesley Publishing Company, Reading, 1979

[FV] R.Fagin, M.Y.Vardi: 'The theory of data dependencies - a survey', IBM Research Report 4321, San Jose, 1984

[Gee] W.C.Mc Gee: 'A contribution to the study of data equivalence', in: Database Management (J.W.Klimbie etc. ed.), Cargese, Amsterdam, 1974, pp. 123 - 148

[GJ] J.Grant, B.E.Jacobs: 'On the family of generalized dependency constraints', in: Journ. of the ACM, 29, 1982, pp. 986 - 997

[GM] J.Graham, A.O.Mendelzon: 'Strong equivalence of relational expressions under dependencies'. in: Information Processing Letters, 14, 1982, pp. 57 - 62

[GMN] H.Gallaire, J.Minker, J.M.Nicolas: 'Logic and databases: a deductive approach', in: ACM Computing Surveys, 16, 1984, pp. 153 - 185

[GV] M.H.Graham, M.Y.Vardi: 'On the complexity and axiomatizability of consistent database states', in: Proc. ACM Symp. on Princ. of Database Systems, 3, 1984, pp. 281 - 289

[Hull] R.Hull: 'Relative information capacity of simple relational database schemata', Techn. Rep. 84-300, Comp. Science Department, Univ. South. Calif., Los Angeles, 1984

[Hull] R.Hull: 'Relative information capacity of simple relational database schemata',in: Proc. ACM Symp. on Princ. of Database Systems, 3, 1984, pp. 97 - 109

[IL1] T.Imielinski, W.Lipski: 'A technique for translating states between database schemata', in: ACM Int. Conf. on Management of data, 1982, pp. 61 - 68

[IL2] T.Imielinski, W.Lipski: 'On the undecidability of equivalence problems for relational expressions', in: Advances in Data Base Theory, 2, 1985, pp. 393 - 409

[IS] T.Imielinski, N.Spyratos: 'On lossless transformation of database states not necessarily satisfying universal instance assumption', in: Proc. ACM Symp. on Princ. of Database Systems, 3, 1984, pp. 258 - 265

[Koba] I.Kobayashi: 'Losslessness and semantic correctness of database scheme transformation: another look of schema equivalence', in: Information Systems, 11, 1986, pp. 41 - 59

[KK] P.Kandzia, H.J.Klein: 'On the equivalence of relational data bases in connection with normalization', Techn. Rep. 7901, Univ. Kiel, 1979

[JAK] B.E.Jacobs, A.R.Aronson, A.C.Klug: 'On interpretations of relational languages and solutions to the implied constraint problem', in: ACM Transactions on Database Systems, 7, 1982, pp. 291 - 315

[KCV] P.C.Kanellakis, S.S.Cosmodakis, M.Y.Vardi: 'Unary inclusion dependencies have polynomial time inference problem', in: Proc. ACM Symp. Theory of Computing, 15, 1983, pp. 264 - 277

[Klug] A.Klug: 'Calculating constraints on relational expressions', in: ACM Transactions on Database Systems, 5, 1980, pp. 260 - 290

[Mitch] J.C.Mitchell: 'The implication problem for functional and inclusion dependencies', in: Information and Control, 56, 1983, pp. 154 - 173

[Reiter] R.Reiter: 'Equality and domain closure on first - order databases', in: Journ. of the ACM, 27, 1980, pp. 235 - 249

[Riss] J.Rissanen: 'On the equivalence of database schemes', in: Proc. ACM Symp. Princ. of Database Systems, 1, 1982, pp. 23 - 26

[Solo] M.K.Solomon: 'Undecidability of the equivalence problem for relational expressions', in: Bell Lab. Memo

[Sier] W.Sierpinski: 'Cardinal and ordinal numbers', PWN Polish Scientific Publishers, Warschau, 1965

[SY] Y.Sagiv, M.Yannakakis: 'Equivalence among relational expressions', in: Proc. ACM Int. Conf.on Very Large Data Bases, 4, 1978, pp. 535 - 548

[Ullm] J.D.Ullman: 'Principle of Database Systems', Computer Science Press, Rockville, 1982

Проблема эквивалентности в схемах реляционных базах данных

Й. Бискуп, У. Реш

Резюме

Авторы используют отображения между схемами для определения эквивалентности, и так самые схемы играют роль параметров. Одним из результатов этого подхода есть то, что существует только одно естественное понятие эквивалентности. Они изучают также разрешимость или неразрешимость этого понятия, а также описывают отображения как реляционную алгебру без дифференции.

EKVIVALENCIA PROBLÉMA A RELÁCIÓS ADATBÁZIS SÉMÁKBAN

J. Biskup, U. Räsch

Összefoglaló

Az adatbázis sémák előfordulásai közötti leképezéseket fel lehet használni az ekvivalencia különböző fokainak definiálására. Ez lehetővé teszi, hogy maga a séma paraméterként fogható fel. Az ekvivalenciák összehasonlitásai azt eredményezték, hogy csak egy természetes ekvivalencia-fogalom létezik. Különböző esetekben a szerzők bebizonyitják ennek az ekvivalencia-fogalomnak az eldönthetőségét ill. eldönthetetlenségét. Emellett a szerzők a leképezéseket különbség nélküli reláció-algebrák segitségével is jellemezték.