GENERALIZED TABLEAUX FOR CHASING EXPRESSION CONSTRAINTS

by

A. Klug
R. Price

Computer Science Technical Report #386

April 1980

Generalized Tableaux for Chasing Expression Constraints

A. Klug
University of Wisconsin

R. Price
Hewlett-Packard

Abstract

The tableaux of Aho, Sagiv and Ullman and the chase computation method of Maier, Mendelzon and Sagiv are generalized. Given a set C of schema constraints consisting of functional and join dependencies and a relational algebra expression composed of projection, restriction, selection and cross product (without the universal instance assumption) the functional & join dependencies on the expression implied by C are determined. The idea of the method is to generate a tableau for the expression whose summary violates the test constraint in a "canonical" way.

Keywords and Phrases: relational algebra, expression constraints, join dependencies, functional dependencies, generalized tableaux, chase, relational model
CR Categories: 4.33, 5.21

Authors' addresses: Anthony Klug, Computer Sciences Dept., University of Wisconsin-Madison, Madison, WI 53706; Rod Price, Hewlett-Packard, 11000 Wolfe Rd., Cupertino, CA 95014. The work of A. Klug was supported in part through contract NB79SBCA0191 with the National Bureau of Standards.

Contents

some symbols used in this paper

symbol	name	meaning
∩	intersect	set intersection
𝔼	bold E	set of expressions
𝕀	bold I	set of instances
∀	inverted A	universal quantifier
□	box	end of proof
⊂	subset	set inclusion
∃	backwards E	existential quantifier
ℕ	bold N	natural numbers
⋈	join	natural join
𝕋	bold T	set of tableaux
𝕍	bold V	set of variables
𝕐	bold Y	set of variables plus natural numbers
∈	greek epsilon	set membership
∘	circle	function composition
τ	greek tau	expression->tableau transform
×	cross	Cartesian product

1. Introduction

In a relational database system, constraints of various types may be known to hold over the relations in the system. The most common constraint types which have been studied are functional dependencies, multivalued dependencies and join dependencies (see [4], for example). An explicitly specified set of constraints may imply more valid constraints on the given relations, and a computation method called the chase [8] has been presented to test for these valid implied dependencies on schema relations. An algorithm for dependencies on schema relations representable as Horn clauses [6] has also been reported [5].

It is also important to know what constraints hold on views over the given relations. This problem is important for a number of reasons: The logic of application programs bound to the view may depend on the validity of certain view constraints. When we bind the view to the base schema, we need to verify that the view constraints will always be valid. Second, consider the problem of embedding relation data types in a programming language, for instance, in Pascal-R [9]. Pascal is a strongly typed language, and the type of relational objects includes key specifications. For the compiler to be able to type-check an expression involving relational operators, it must be able to determine which domains in an expression of type relation are a key.

The problem of determining dependencies over views is decidable for many classes of views ([7] and [2]). In [2], the decidability of $\forall\exists$-sentences is used. This method, however, is computationally difficult. In this paper we present a method which is more intuitive than $\forall\exists$-decidability and more amenable to optimization. For functional dependencies, the method will run in polynomial time. We have, in fact, implemented the algorithm to calculate functional dependencies on expressions.

The method is a generalization of the chase method of [8], and we get a chase computation which determines valid functional dependencies and join dependencies on a general class of relational algebra expressions.

The next section provides the basic relational definitions from which we work. In section 3, the definitions for tableaux are given. Section 4 defines the chase computation, and proves its basic properties. In section 5, we develop the machinery for using chase computations for testing expression constraints, and in section 6 we give some examples and some extensions to the method.

2. Relational Model Definitions

The formal model we use does not make the universal instance assumption. This assumption has been criticized in the past (e.g., [3]), and it is important to avoid it whenever possi-

ble. In addition, the relational algebra we define is not restricted to natural joins of at most one occurrence of each relation as in [1]. We can, for example, form a repeated join of a family tree relation to get information on great grandparents.

A relation scheme is a pair $\langle R,k\rangle$. R is a symbol (the relation name), and k is a positive integer (R's degree) which is denoted deg(R). If $\langle R,k\rangle$ is a relation scheme, the domains of R, doms(R), is the set $\{1,2,\ldots,k\}$ of natural numbers.

A schema is a sequence $\langle\langle R_1,k_1\rangle,\ldots,\langle R_n,k_n\rangle\rangle$ of relation schemes. It is sometimes written simply $\langle R_1,\ldots,R_n\rangle$. Throughout this paper, one fixed schema $\langle R_1,\ldots,R_n\rangle$ is assumed.

An instance I of schema $\langle R_1,\ldots,R_n\rangle$ is an n-tuple $\langle I_1,\ldots,I_n\rangle$ where for each $i=1,\ldots,n$, $I_i \subseteq \mathbb{N}^{deg(R_i)}$. All domains are taken without loss of generality to range over the set $\mathbb{N}$ of natural numbers, and $\mathbb{N}^m$ is the set of all m-tuples over $\mathbb{N}$. We let $\mathbf{I}$ be the set of all instances over our fixed schema of n relations.

A functional dependency (FD) for degree k is a pair $\langle Z,A\rangle$, also written $Z\rightarrow A$, where $Z \subseteq \{1,2,\ldots,k\}$, $A \in \{1,2,\ldots,k\}$ and $A \notin Z$.

A join dependency (JD) for degree k is a sequence

$\langle J_1,\ldots,J_m\rangle$, also written $[J_1,\ldots,J_m]$, in which each J_i is a sublist of $\{1,\ldots,k\}$ and where $\cup J_i = \{1,2,\ldots,k\}$. A sublist of a set X is a nonrepeating sequence whose entries are taken from X.

A constraint is either an FD or a JD.

A schema FD is a pair $\langle R_i, Z\rightarrow A\rangle$, also written $R_i: Z\rightarrow A$, where R_i is a schema relation and $Z\rightarrow A$ is an FD for degree $\deg(R_i)$. Similarly, a schema JD is a pair $\langle R_i, S\rangle$, also written $R_i:S$, where S is a JD for degree $\deg(R_i)$.

An FD $Z\rightarrow A$ for degree k is true in tuple set $X \subseteq \mathbb{N}^k$ if for all tuples t_1, t_2 in X, if $t_1[Z] = t_2[Z]$, then $t_1[A] = t_2[A]$. Brackets '[', ']' indicate projection on the listed domains.

A JD $[J_1,\ldots,J_m]$ for degree k is true in tuple set $X \subseteq \mathbb{N}^k$ if every element $t \in \mathbb{N}^k$ such that $t[J_i]$ is in $X[J_i]$ $(i=1,\ldots,m)$ is in X.

A schema constraint $\langle R_i, c\rangle$ is true in an instance I if c is true in I_i.

If C is a set of schema constraints, the subset sat(C) of $\mathbb{I}$ denotes the set of instances in which C is true.

The set $\mathbb{E}$ of expressions over our fixed schema and the associated functions deg (degree) and doms (domains) for expressions are defined inductively as follows:

If $e \in \mathbb{E}$ has degree k, then doms(e) = {1,...,k}.

(1) $R_i \in \mathbb{E}$ for each R_i in the schema, and deg(R_i) is already defined.

(2) Projection: If $e \in \mathbb{E}$ and deg(e) is k, then $e[X] \in \mathbb{E}$ where X is a sublist of doms(e), and deg(e[X]) = number of elements in X.

(3) Cross Product: If e_1, $e_2 \in \mathbb{E}$ and deg(e_1) = d_1, deg(e_2) = d_2, then $(e_1 \times e_2) \in \mathbb{E}$, and deg($e_1 \times e_2$) = $d_1 + d_2$.

(4) Selection, Restriction: If $e \in \mathbb{E}$, $X,Y \in$ doms(e) and $V \in \mathbb{N}$, then $e[X=V] \in \mathbb{E}$ and $e[X=Y] \in \mathbb{E}$ and deg(e[X=V]) = deg(e[X=Y]) = deg(e).

With these operators we can also define joins and intersections. Set difference was not included because determining validity of dependencies on expressions with set difference is impossible [7]. The union operator will be discussed in the last section of this paper.

For each $e \in \mathbb{E}$ of degree k and for each $I \in \mathbb{I}$, the value of e on I, denoted e(I), is a subset of $\mathbb{N}^k$. The formal definition, which is omitted, gives the usual semantics for relational algebra operators.

An expression constraint is a pair <e,c>, also written e:c, where $e \in \mathbb{E}$ and c is an FD or a JD for degree deg(e). An expression constraint e:c is true in instance I if c is true in e(I).

An expression constraint e:c is valid in a set of instances

P $\subseteq$ $\mathbb{I}$ if for every I $\in$ P, e:c is true in I.

3. Tableaux

Tableaux have been defined by Aho, Sagiv and Ullman [1]. We generalize the notion in several ways: A tableau may have many rows in its summary. There is no division of variables into distinguished and nondistinguished classes[1]. There is also no restriction on the number of columns in which a variable may occur.

The set $\mathbb{V}$ of variables is the set $\{a_1, a_2, a_3, \ldots\}$ of subscripted "a"s. The set $\mathbb{Y}$ of symbols is $\mathbb{V} \cup \mathbb{N}$.

A tableau T of degree m for the schema $\langle R_1,\ldots,R_n\rangle$ is an n+1-tuple $\langle S,T_1,\ldots,T_n\rangle$ such that $S \subseteq \mathbb{V}^m$, for each $i=1,\ldots,n$, $T_i \subseteq \mathbb{Y}^{\deg(R_i)}$, and every variable in S appears in some T_i. S is called the summary[2]. We let $\mathbb{T}$ be the set of all tableaux over the schema.

If X is a tuple, a tuple set or a tableau, we let $\mathbb{Y}(X)$ denote the set of symbols occurring in X.

A valuation r is a partial function $\mathbb{Y}\to\mathbb{N}$ which is the identity on $\mathbb{N} \subseteq \mathbb{Y}$. Valuations are extended to functions $\mathbb{Y}^k\to\mathbb{N}^k$

---

[1] If one wishes, the variables appearing in the summary may be called distinguished. In this paper we only need to distinguish them in one place.

[2] This and subsequent definitions and theorems can easily be extended to allow constants in summaries. This was not done because constant columns in a view add nothing.

and functions $P(\mathbb{V}^k) \rightarrow P(\mathbb{M}^k)$ (P = power set) by component-wise and element-wise extension. A valuation for a tableau T is one which is defined on $\mathbb{V}(T)$. A tableau can be considered an instance by applying a one-to-one valuation to it and omitting the summary. Normally, the one-to-one valuation will not be explicitly mentioned.

A renaming is a one-to-one partial function $r:\mathbb{V} \rightarrow \mathbb{V}$ which is the identity on $\mathbb{N}$.

A tableau $T = \langle S, T_1, \ldots, T_n \rangle$ may also be considered to be a function $\mathbb{I} \rightarrow \mathbb{N}^{deg(T)}$ by defining:

$$T(I) = \{r(s) : r \text{ is a valuation for } T,\ s \in S,\ \forall i\ r(T_i) \subseteq I_i\}$$

Tableau T_1 is equivalent to tableau T_2 with respect to a subset $P \subseteq \mathbb{I}$, written $T_1 \equiv_P T_2$, if $T_1(I) = T_2(I)$ for all $I \in P$.

4. Transformation Rules and Chases

In this section, the chase computation is defined. A chase consists of a sequence of transformations on a tableau which preserves equivalence with respect to a given class of instances. We start by defining three classes of transformation rules. The first two correspond to FDs and JDs, respectively, which are valid in the class of instances. The third type of transformation rule adds rows to the summary when this will not change what the tableau computes,

and these rules are applicable in any set of instances.

A transformation rule for P $\subseteq$ $\mathbb{I}$ is a partial function $f:\mathbb{T}\to\mathbb{T}$ such that $f(T) \equiv_P T$.

F-rules. For each schema FD $R_i:Z\to A$ there is an F-rule which is defined as follows: If $T = \langle S,T_1,\ldots,T_n\rangle$ and there are $t_1, t_2 \in T_i$ such that $t_1[Z] = t_2[Z]$ and $t_1[A] \neq t_2[A]$, then

(a) if one of $t_1[A]$, $t_2[A]$ are constants, replace all occurrences in T of the other by that constant[3], or

(b) if both $t_1[A]$ and $t_2[A]$ are variables, replace all occurrences in T of the one with the smaller subscript by occurrences of the other.

J-rules. For each JD $R_i:[J_1,\ldots,J_k]$ there is a J-rule which is defined as follows: If there is an element $t \in \mathbb{V}^{deg(R_i)}$ such that for each $j=1,\ldots,k$ there is a $t_j \in T_i$ with $t_j[J_j] = t[J_j]$, then add t to T_i if it is not already there.

T-rules. If r is a function $\mathbb{V}(T)\to\mathbb{V}(T)$ such that $r(T_i) \subseteq T_i$ for $i=1,\ldots,n$, then add r(S) to S if not already there.

In Figure 1 some examples applying these rules are given.

Theorem 1. Let $T \in \mathbb{T}$ and let T' be the result of applying the F-rule for $R_i:Z\to A$ to T. Then $T \equiv T'$ wrt $sat(R_i:Z\to A)$.

---

[3] If both are unequal constants, nothing needs to be done; we will already have $T(I) = \emptyset$ for all $I \in sat(R_i:Z\to A)$.

schema: R_1, $\deg(R_1) = 3$

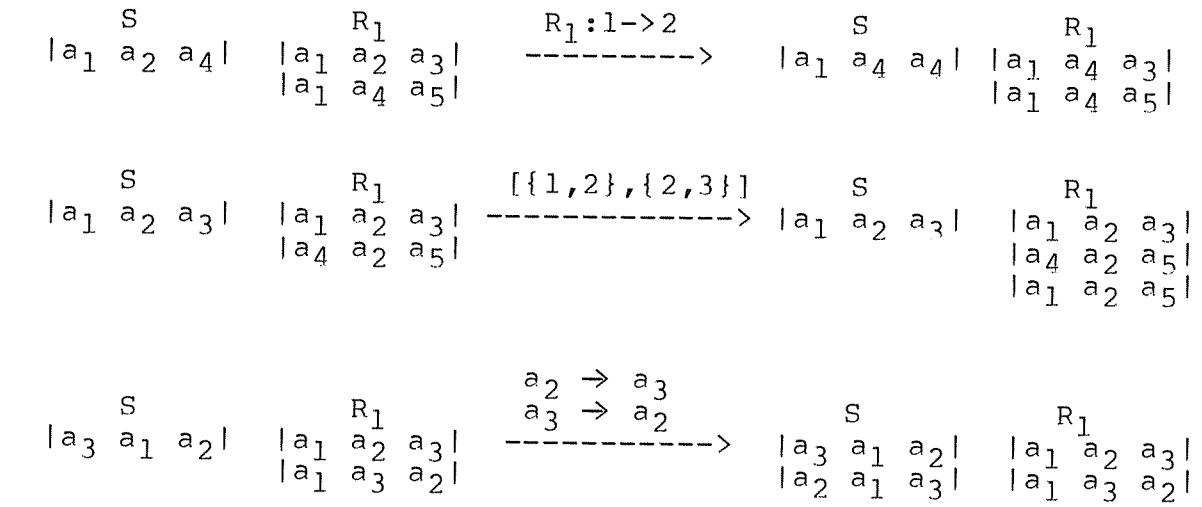

Figure 1. Transformation Examples

Proof. As in [8].

Theorem 2. Let $T \in \mathbb{T}$ and let T' be the result of applying the J-rule for $\langle R_i, S\rangle$ to T. Then $T \equiv T'$ wrt $sat(R_i:S)$.

Proof. As in [8].

Theorem 3. Let $T \in \mathbb{T}$ and let T' be the result of applying the T-rule for the function r to T. Then $T \equiv T'$.

Proof. Let $I \in \mathbb{I}$. Suppose $t \in T(I)$. There is a valuation $p:\mathbb{V}(T)\to\mathbb{N}$ and an $s \in S$ such that $t = p(s)$ and $p(T_i) \subseteq I_i$ $(i=1,\ldots,n)$. We still have $s \in S'$ and $T'_i = T_i$, so $t \in T'(I)$. For the converse, suppose $t \in T'(I)$. There is a

valuation p and an $s' \in S'$ such that $t = p(s')$ and $p(T_i') \subseteq I_i$ for $i=1,\ldots,n$. If $s' \in S$, then, since we also have $p(T_i) \subseteq I_i$, $t = p(s') \in T(I)$. Otherwise we may write $s' = r(s)$ for some $s \in S$. Then because $r(T_i) \subseteq T_i$ for $i=1,\ldots,n$, the valuation por has the property that $t = p(r(s))$ and $p(r(T_i)) \subseteq I_i$. Hence $t \in T(I)$. □

If C is a set of schema constraints, a C-generating sequence for a tableau T is a sequence $T_\emptyset, T_1, T_2, \ldots, T_n$ where $T_\emptyset = T$, T_i is obtained from T_{i-1} by an application of a T-rule or an F-rule or a J-rule for C ($i=1,\ldots,n$), and where there is no applicable rule for T_n.

Suppose T' is obtained from T by some rule. For each tuple $t \in T_i$ (and $s \in S$) there is a corresponding tuple in T_i' ($s' \in S'$) determined as follows: If T' was obtained from T by a J-rule or a T-rule, then for $i=1,\ldots,n$, $T_i \subseteq T_i'$ and $S \subseteq S'$, so all tuples can be made to correspond to themselves. If T' was obtained by an F-rule, then either $t \in T_i'$ ($t \in S'$) too, or there is some variable replacement function $r: \mathbb{V} \to \mathbb{V}$ such that $r(t) \in T_i'$ ($r(t) \in S'$), and this is the corresponding tuple. If T' was obtained by a T-rule, then $T_i = T_i'$ for $i=1,\ldots,n$, and $S \subseteq S'$ and the corresponding tuples in S are themselves.

The following four lemmas similar to lemmas in [8], and their proofs are omitted.

Lemma 1. Let $I \in sat(C)$, $T \in \mathbb{T}$, and let r be a valuation such that $r(T_i) \subseteq I_i$ for $i=1,\ldots,n$. Then for all tableaux T_j in a generating sequence for T, (a) $r(T_{ji}) \subseteq I_i$, and (b) if t in T_i and t_j in T_{ji} correspond, then $r(t) = r(t_j)$, i.e., r maps corresponding tuples to the same instance tuple.

Lemma 2. A given set of F-, J- and T-rules can be applied to a tableau only a finite number of times.

Lemma 3. Suppose no T-rule and no F- or J-rule for C is applicable to tableau T. Then $T \in sat(C)$ (as an instance).

Lemma 4. Let $T,U,V \in \mathbb{T}$ such that U and V are the final tableaux in two C-generating sequences for T. Then U and V are identical.

Lemmas 2 and 4 mean that the following chase function is well-defined: Given a set C of schema constraints and a tableau T, $chase_C(T)$ is the final tableau in a C-generating sequence for T.

Lemma 5. Let $T' = chase_C(T)$. Then (a) $T' \equiv T$ wrt $sat(C)$, and (b) $S' = T'(T')$, where S' is the summary of T'; the first occurrence of T' is the function $\mathbb{I} \to \mathbb{N}^{deg(T)}$, and the second occurrence is T' the instance.

Proof. (a) This is a combination of Theorems 1, 2 and 3 noting that if $c \in C$, then $sat(C) \subseteq sat(c)$.

(b) Let p be a one-to-one valuation for T'. The statement $S' = T'(T')$ formally means $p(S') = T'(\langle p(T'_1),\ldots,p(T'_n)\rangle)$. Checking the definitions shows that the inclusion "$\subseteq$" is true.

To prove the other inclusion, suppose $x \in T'(\langle p(T'_1),\ldots,p(T'_n)\rangle)$. There is a valuation q such that $x = q(s)$ for some $s \in S'$ and $q(T'_i) \subseteq p(T'_i)$ $(i=1,\ldots,n)$. Consider the function $p^{-1} \circ q: \mathbb{V}(T') \to \mathbb{V}(T')$. The above condition means that $p^{-1}(q(T'_i)) \subseteq T'_i$. The T-rule for $p^{-1} \circ q$ is applicable to T', but since T' is already the end of a chase, we must have $p^{-1}(q(S')) \subseteq S'$. Applying p, we get $q(S') \subseteq p(S')$. In particular, $x = p(s) \in p(S')$. □

Theorem 4. If $sat(C) = sat(D)$, then $chase_C(T) = chase_D(T)$ for any tableau T.

Proof. As in [8]. This theorem says that the set of constraints on which the chase is based may be replaced by any convenient equivalent set of constraints. □

5. Chasing Expression Constraints

Our first goal is, given an expression, to find a tableau to represent that expression.

If $e \in \mathbb{E}$ and $T \in \mathbb{T}$, we write $e \equiv T$ if $e(I) = T(I)$ for all $I \in \mathbb{I}$.

We define a transformation $\tau: \mathbb{E} \to \mathbb{T}$ such that $e \equiv \tau(e)$ for all $e \in \mathbb{E}$. The summary of $\tau(e)$ will have only one row.

(1) For $R_i \in \mathbb{E}$ of degree m, $\tau(R_i)$ is the tableau whose summary is $\{\langle a_1,\ldots,a_m\rangle\}$, whose i-th component is also $\{\langle a_1,\ldots,a_m\rangle\}$ and whose other components are empty.

Suppose $\tau(e)$ is defined and is $\langle S,T_1,\ldots,T_n\rangle$ where $S = \{\langle b_1,\ldots,b_m\rangle\}$.

(2) For projection, $\tau(e[X])$ is $\langle S[X],T_1,\ldots,T_n\rangle$, i.e., the summary is projected on the domains in X, and other components are the same.

(3) For selection, $\tau(e[X=V]) = T' = \langle S',T'_1,\ldots,T'_n\rangle$ is obtained as follows: If b_X is a constant $W \neq V$, then S' is empty and $T'_i = T_i$. If b_X is already constant V then $T' = T$. If b_X is a variable a_j, T' is obtained from T by replacing all occurrences of a_j in T by V.

(4) For restriction, $\tau(e[X=Y]) = T' = \langle S',T'_1,\ldots,T'_n\rangle$ is obtained as follows: If $b_X = b_Y$, then $T' = T$. If b_X and b_Y are distinct constants, S' is empty and $T'_i = T_i$. If one of b_X, b_Y are constants, all occurrences of the other in T are replaced by this constant. If both b_X

and b_Y are variables, every occurrence in T of the one with the greater subscript is replaced by an occurrence of the other.

(5) For cross product, suppose $\tau(e_1) = T_1$ and $\tau(e_2) = T_2$. Let k be the largest variable subscript in T_1 and let m be the smallest variable subscript in T_2. Define the renaming function $g:\mathbb{Y}\rightarrow\mathbb{Y}$ by $g(a_j) = a_{j+k+1-m}$. This maps variables of T_2 to the smallest-indexed set variables disjoint from those in T_1. Then
$\tau(e_1 \times e_2) = \langle S_1 \times g(S_2), T_{11} \cup g(T_{21}), \ldots, T_{1n} \cup g(T_{2n}) \rangle$.

Lemma 6. For all $e \in \mathbb{E}$, $e \equiv \tau(e)$.

Proof. The proof by induction on the number of operators in the expression is left to the reader. □

The tableaux of Aho, Sagiv and Ullman [1] are intended to model a relational algebra with natural join, projection and selection over a database with a universal instance. They noted, however, that not all of their tableaux corresponded to expressions. This is not the case here; every tableau represents a relational algebra expression:

Theorem 5. For every $T \in \mathbb{T}$ with a one row summary there is an $e \in \mathbb{E}$ such that $T \equiv e$.

Proof. We will give here only an informal statement of the

proof. For every non-summary row in T_i, there will be a term R_i in the cross product. Whenever two entries in two rows are the same variable, there will be a corresponding restriction on the cross product. For every occurrence of a constant, there is a corresponding selection term on the cross product. Finally, a projection is generated from the summary: If the i-th entry in the summary is a_j, make the i-th entry in the projection list refer to any column in the cross product corresponding to an occurrence of a_j in the tableau. □

In order to test constraints on expressions we need to be able to generate representative tableaux for the expressions which have summaries of a particular form. We arrange that the summary, as a tuple set, will violate the test constraint in a canonical way. That is, we can take the summary as a template and fit it over any tuple set which actually does violate the constraint. If the chase removes this violation, the constraint is valid, otherwise we have a counterexample state. The next lemma describes how to get the desired summaries.

Lemma 7. Let $T \in \mathbb{T}$ and let $r_1,\ldots,r_k$ be renamings on $\mathbb{V}(T)$ such that r_i^{-1} is the identity on $r_i(\mathbb{V}(T)) \cap r_j(\mathbb{V}(T))$ for $j\neq i$ $(i=1,\ldots,k)$. This means that for every symbol b on which r_i is defined, $r_i(b)$ either doesn't appear in the ranges of the other renamings, or it equals b. Then $T \equiv \cup\, r_j(T)$, where the union is done component-wise on the tableau n+1-tuples.

Proof. Let $T' = \cup\, r_j(T)$. Let $I \in \mathbb{I}$ and suppose $t \in T(I)$. There is a valuation $p:\mathbb{V}(T)\rightarrow\mathbb{N}$ such that for some $s \in S$, $t = p(s)$ and such that $p(T_i) \subseteq I_i$ $(i=1,\ldots,n)$. Define a valuation $p':\mathbb{V}(T')\rightarrow\mathbb{N}$ by the rule: $p'(r_j(b)) = p(b)$, where $b \in \mathbb{V}(T)$. This defines a function because if $x \in r_i(\mathbb{V}(T)) \cap r_j(\mathbb{V}(T))$, then $r_i(x) = r_j(x) = x$. Also $p'(T'_i) = p'(\cup\, r_j(T_i)) = \cup\, p'(r_j(T_i)) = \cup\, p(T_i) = p(T_i) \subseteq I_i$. Since there is some $s' \in S'$ of the form $r_j(s)$ (in fact, there are k of them), $p'(s') = p'(r_j(s)) = p(s) = t$. Hence $t \in T'(I)$.

Suppose $t \in T'(I)$. Then there is a valuation $p':\mathbb{V}(T')\rightarrow\mathbb{N}$ and and $s' \in S'$ of the form $r_j(s)$, $s \in S$, such that $t = p'(s')$ and $p'(T'_i) \subseteq I_i$ $(i=1,\ldots,n)$. Define $p:\mathbb{V}(T)\rightarrow\mathbb{N}$ by $p(b) = p'(r_j(b))$. Then $p(s) = p'(r_j(s)) = t$, and $p(T_i) = p'(r_j(T_i)) \subseteq p'(T'_i) \subseteq I_i$. Hence $t \in T'(I)$. □

Once we get a tableau with the desired summary, we will want to treat the summary like a template and map it with a valuation on portions of an expression instance. Every element in the portion of an instance of expression e over

which the template is mapped has an associated valuation which gets it in image of the tableau as a function. But we want a uniform valuation for all tuples matched by the template. The next lemma shows us that this can be done.

Lemma 8. Let T be a tableau whose summary has only one row. Let $r_1,\ldots,r_k$ be renamings such that r_i^{-1} is the identity on $r_i(\mathbb{Y}(T)) \cap r_j(\mathbb{Y}(T))$, $j\neq i$, and such that r_i is not the identity on symbols not in the summary of T. Let $T' = \cup\, r_j(T)$. Given instance I and valuation $p:\mathbb{Y}(S')\to\mathbb{N}$ such that $p(S') \subseteq T'(I)$, p may be extended to $\mathbb{Y}(T')\to\mathbb{N}$ such that $p(T_i') \subseteq I_i$ $(i=1,\ldots,n)$.

Proof. For each $j=1,\ldots,k$, let s_j be the element of S' of the form $r_j(s)$. Since $p(s_j) \in T(I)$ by the last lemma, we know that there is a valuation $p_j:\mathbb{Y}(T)\to\mathbb{N}$ such that $p_j(s) = p(s_j)$ and $p_j(T_i) \subseteq I_i$ $(i=1,\ldots,n)$. Consider $p_j \circ r_j^{-1}$ defined on $\mathbb{Y}(r_j(T))$. We have $p_j(r_j^{-1}(s_j)) = p_j(s) = p(s_j)$, and $p_j(r_j^{-1}(r_j(T_i))) = p_j(T_i) \subseteq I_i$. Also, $p_j \circ r_j^{-1}$ does not conflict on common domain of definition with $p_{j'} \circ r_{j'}^{-1}$ for any $j'\neq j$: For if $b \in \mathbb{Y}(r_j(T)) \cap \mathbb{Y}(r_{j'}(T))$, then r_j^{-1} and $r_{j'}^{-1}$ are both the identity on b, and since b then appears in S', both p_j and $p_{j'}$ agree with p (and hence with themselves) on b. Thus we may define $p_u:\mathbb{Y}(T')\to\mathbb{N}$ which extends each $p_j \circ r_j^{-1}$, $j=1,\ldots,k$. This valuation has the properties that $p_u(s_j) = p_j(r_j^{-1}(s_j)) = p(s_j)$, and $p_u(T_i') = p_u(\cup\, r_j(T_i)) = \cup\, p_j(r_j^{-1}(r_j(T_i)) = \cup\, p_j(T_i) \subseteq I_i$. □

Theorem 6. Let $e \in \mathbb{E}$ and $T = \tau(e)$, and let e:c be a constraint on e (an FD or a JD) and C be a set of schema constraints. Let $T' = \cup\, r_j(T)$ for some set $\{r_1,\ldots,r_k\}$ of renamings as above. If e:c is valid in sat(C) then c is true in the summary of $chase_C(T')$ (under the image of some one-to-one valuation).

Proof. Suppose e:c is valid in sat(C). By Lemma 5, $T'' = chase_C(T')$, as an instance, is in sat(C), so e:c is true in T". But by Lemmas 6, 7 and 5, respectively, $e(T'') = T(T'') = T'(T'') = T''(T'') = S''$, so c is true in the summary of $chase_C(T')$. □

Theorem 7. Let $e \in \mathbb{E}$, $T = \tau(e)$, and C be a set of schema constraints. Then $e:Z\rightarrow A$ is valid in sat(C) iff $Z\rightarrow A$ is true in the summary of $chase_C(T \cup r(T))$, where r is the renaming which is the identity exactly on the variables in the Z-columns of the summary of $\tau(e)$ (and on $\mathbb{N}$).

Proof. We have bent the conditions on the set of renaming functions to allow one of them to be the identity everywhere. This will simplify notation and examples.

(=>) This is the last theorem.

(<=) Let $T' = T \cup r(T)$. Suppose $Z\rightarrow A$ is true in the summary of $T'' = chase_C(T')$. Let $I \in sat(C)$, and let $t_1, t_2 \in e(I)$ such that $t_1[Z] = t_2[Z]$. We can define a valuation p on $\mathbb{V}(S'')$ such that $p(s) = t_1$ and $p(r(s)) = t_2$. By Lemma 8, we

may extend p to $\mathbb{V}(T')$ such that $p(T_i') \subseteq I_i$ (i=1,...,n). By Lemma 2, $p(T_i'') \subseteq I_i$ (i=1,...,n), and $p(S'') = \{t_1, t_2\}$. But $Z \rightarrow A$ is true in S", so $Z \rightarrow A$ is true in $\{t_1, t_2\}$. These tuples were arbitrary, so $Z \rightarrow A$ is true in e(I). □

Theorem 8. Let $e \in \mathbb{E}$, $T = \tau(e)$, and C be a set of schema constraints. Given a JD $e:[J_1,...,J_k]$, define the set $\{r_1,...,r_k\}$ of renamings as in Lemma 8 such that r_i is the identity on $S[J_i]$ (i=1,...,k). Then $e:[J_1,...,J_k]$ is valid in sat(C) iff $[J_1,...,J_k]$ is true in the summary of $chase_C(\cup r_j(T))$.

Proof. (=>) This is Theorem 6.

(<=) Let $T' = \cup r_j(T)$, $T'' = chase_C(T')$, and suppose $[J_1,...,J_k]$ is true in the summary of $T'' = chase_C(T')$. Let $I \in sat(C)$, and suppose $t_1,...,t_k \in e(I)$ such that there is some $t \in \mathbf{N}^{deg(T)}$ with $t[J_j] = t_j[J_j]$. We can define a valuation p on $\mathbb{V}(S'')$ such that $p(s_j) = t_j$ (j=1,...,k). By Lemma 8, we may extend p to $\mathbb{V}(T')$ such that $p(T_i') \subseteq I_i$ (i=1,...,n). By Lemma 2, $p(T_i'') \subseteq I_i$ (i=1,...,n). Thus $p(S'') \subseteq e(I)$, and because $[J_1,...,J_k]$ is true in S", there is an $s \in S''$ with $s[J_j] = s_j[J_j]$ (j=1,...,k). Then $p(s) = t \in e(I)$ and so $[J_1,...,J_k]$ is true in e(I). □

The last point we make in this section is that T-rules are not really needed. We can replace this exponential-time rule by a simple polynomial computation:

Theorem 9. Let T be a tableau and e an equivalent expression. Let $T' = chase_C(T)$ for some constraint set C. Define the function $chase^*$ as the chase function without the T-rules, and let $T^* = chase^*_C(T)$. Then $S' = e(T^*)$, where T^* is considered an instance.

Proof. The proof is left to the reader. □

6. Examples and Summary

We illustrate our generalized chase method in this section with a number of examples.

Consider one relation R with 3 domains and FDs 1,2→3 and 3→2. Does 1,2→3 hold in the "join"[4] (R[1,3]×R[2,3])[2=4][1,3,4]? The tableau T for this expression is the following:

S	R
$\|a_1\ a_5\ a_6\|$	$\|a_1\ a_2\ a_6\|$
	$\|a_4\ a_5\ a_6\|$

To test 1,2→3, we use two renamings: r_1 is the identity; r_2 is $\{a_2 \to a_7, a_4 \to a_8, a_6 \to a_9\}$. Then $r_1(T)$ U $r_2(T)$ is the following tableau:

[4] If we use the notation of [1], and R has domains ABC, this expression corresponds to the natural join R[AC]⋈R[BC].

$|a_1\ a_5\ a_6|$ $|a_1\ a_2\ a_6|$
$|a_1\ a_5\ a_9|$ $|a_4\ a_5\ a_6|$
$|a_1\ a_7\ a_9|$
$|a_8\ a_5\ a_9|$

The computation is given in Figure 2. We see that column 3 in the summary of the final tableau has only one variable, so the FD holds.

Using the same expression, let us test the JD [<1,3>,<2,3>]. The necessary renaming functions are: r_1 defined by $\{a_2 \rightarrow a_7, a_4 \rightarrow a_8, a_5 \rightarrow a_9\}$ and r_2 defined by $\{a_1 \rightarrow a_{10}, a_2 \rightarrow a_{11}, a_4 \rightarrow a_{12}\}$. The chase starts with the tableau:

$|a_1\ a_9\ a_6|$ $|a_1\ a_7\ a_6|$
$|a_{10}\ a_5\ a_6|$ $|a_8\ a_9\ a_6|$
$|a_{10}\ a_{11}\ a_6|$
$|a_{12}\ a_5\ a_6|$

The computation is shown in Figure 3. We see that the JD is true in the final tableau's summary, so it is valid in the expression.

(Ø) $|a_1\ a_5\ a_6|$ $|a_1\ a_2\ a_6|$ (1) $|a_1\ a_5\ a_6|$ $|a_1\ a_5\ a_6|$
$|a_1\ a_5\ a_9|$ $|a_4\ a_5\ a_6|$ (2->5) $|a_1\ a_5\ a_9|$ $|a_4\ a_5\ a_6|$
$|a_1\ a_7\ a_9|$ $|a_1\ a_7\ a_9|$
$|a_8\ a_5\ a_9|$ $|a_8\ a_5\ a_9|$

(2) $|a_1\ a_7\ a_6|$ $|a_1\ a_7\ a_6|$ (3) $|a_1\ a_7\ a_9|$ $|a_1\ a_7\ a_9|$
(5->7) $|a_1\ a_7\ a_9|$ $|a_4\ a_7\ a_6|$ (6->9) $|a_1\ a_7\ a_9|$ $|a_4\ a_7\ a_9|$
$|a_1\ a_7\ a_9|$ $|a_1\ a_7\ a_9|$
$|a_8\ a_7\ a_9|$ $|a_8\ a_7\ a_9|$

Figure 2. Testing an FD. Each step is numbered and labeled with the substitution that got it.

$\|a_1 \ a_9 \ a_6\|$ $\|a_{10} \ a_5 \ a_6\|$	$\|a_1 \ a_7 \ a_6\|$ $\|a_8 \ a_9 \ a_6\|$ $\|a_{10} \ a_{11} \ a_6\|$ $\|a_{12} \ a_5 \ a_6\|$	Use F-rules for FD 3→2 to replace a_7, a_9 and a_5 by a_{11}.
$\|a_1 \ a_{11} \ a_6\|$ $\|a_{10} \ a_{11} \ a_6\|$	$\|a_1 \ a_{11} \ a_6\|$ $\|a_8 \ a_{11} \ a_6\|$ $\|a_{10} \ a_{11} \ a_6\|$ $\|a_{12} \ a_{11} \ a_6\|$	Use the T-rule for the function {1->8,8->1,10->12,12->10}.
$\|a_1 \ a_{11} \ a_6\|$ $\|a_{10} \ a_{11} \ a_6\|$ $\|a_8 \ a_{11} \ a_6\|$ $\|a_{12} \ a_{11} \ a_6\|$	$\|a_1 \ a_7 \ a_6\|$ $\|a_8 \ a_9 \ a_6\|$ $\|a_{10} \ a_{11} \ a_6\|$ $\|a_{12} \ a_5 \ a_6\|$	The final tableau.

Figure 3. Testing for a JD.

This paper has generalized the chase computation method. The method can determine the validity of functional and join dependencies on relational algebra expressions. There are a number of directions future work could take:

The union operator also can be accommodated in this method. For given a set C on constraints, a union $e_1 \cup e_2$ and an FD Z→A to test on this union, we can first test the FD on e_1 and e_2 individually. If it was valid on both components, the only way it could fail to be valid on the union would be to have an instance $I \in sat(C)$ and tuples $t_1 \in e_1$ and $t_2 \in e_2$ which violate Z→A. The method of generating rows in tableau summaries could be modified to generate a tableau which contains a summary row corresponding to e_1 and a summary row for e_2 which agree on the Z-columns. If the chase

identified the A-column, then the FD is valid in the union.

Determining valid JDs on a union would be more complicated. If a test JD had k parts, we would need tableaux with k rows in the summaries. But we would need to test 2^k tableaux, one for each way of getting these k tuples from e_1 and e_2.

The method we have given may be described in these general terms:

> Given a set C of constraints, an expression e, its tableau T and a constraint c in some constraint class to test on e, modify T using renaming functions so that its summary contains tuples which violate c in a "canonical" fashion. Perform the chase computation on this modified tableau, and if the final tableau has a summary which still violates c, then c is not valid on e; otherwise c is valid.

This technique could be applied to constraint types other than just FDs and JDs. As a brief example, consider "pseudo-keys". Given $e \in \mathbb{E}$, we can define a pseudo-key[5] on e to be a set $\{K_1,\ldots,K_m\}$, $K_i \subseteq \text{doms}(e)$, such that every tuple in an instance of e is uniquely identified by its value in columns K_1 or by its value in K_2, ... , or by its

---

[5] Pseudo-keys are defined for illustrative purposes only. It is unlikely that they represent any important semantics.

value in K_m. If e has 3 domains and a pseudo-key {{1},{2}}, then a tableau summary which "canonically" violates this constraint is:

$$\begin{vmatrix} a_1 & a_2 & a_3 \\ a_1 & a_4 & a_5 \\ a_6 & a_2 & a_7 \end{vmatrix}$$

If we do the chase computation and the summary still violates the constraint, then it is not valid, otherwise it is.

Another extension is to use the chase method to answer the opposite question: Given a schema s, an expression e over s and a set C of constraints that must be valid on e, what constraints must necessarily be valid on schema s? To answer this question, we can generate the tableau T for e; get a new tableau from T in which the schema constraint being tested is violated and then do the chase based on the expression constraints. This will surely work for FDs and possibly also for JDs. An example is given in Figure 4.

Schema: R_1, R_2, both binary
Expression: $e = (R_1 \times R_2)[2=3]$ (a join)
Assumed expression constraints: e:1→4
Test schema constraint: R_2:1->2

Tableau for e:

S	R1	R2
$\|a_1\ a_3\ a_3\ a_4\|$	$\|a_1\ a_3\|$	$\|a_3\ a_4\|$

Chase:

(Ø)

S	R_1	R_2	
$\|a_1\ a_3\ a_3\ a_4\|$	$\|a_1\ a_3\|$	$\|a_3\ a_4\|$	Use T-rule with
$\|a_5\ a_3\ a_3\ a_6\|$	$\|a_5\ a_3\|$	$\|a_3\ a_6\|$	$\{a_1 \to a_5,\ a_4 \to a_6\}$

(1)

S	R_1	R_2	
$\|a_1\ a_3\ a_3\ a_4\|$	$\|a_1\ a_3\|$	$\|a_3\ a_4\|$	Use F-rule for
$\|a_5\ a_3\ a_3\ a_6\|$	$\|a_5\ a_3\|$	$\|a_3\ a_6\|$	e:1→4
$\|a_1\ a_3\ a_3\ a_6\|$			
$\|a_5\ a_3\ a_3\ a_4\|$			

(2)

S	R_1	R_2	
$\|a_1\ a_3\ a_3\ a_6\|$	$\|a_1\ a_3\|$	$\|a_3\ a_6\|$	R_2:1→2 is necessary
$\|a_5\ a_3\ a_3\ a_6\|$	$\|a_5\ a_3\|$	$\|a_3\ a_6\|$	
$\|a_1\ a_3\ a_3\ a_6\|$			
$\|a_5\ a_3\ a_3\ a_6\|$			

Figure 4. Testing Necessary Schema Constraints.

References

[1] Aho A.V. Sagiv Y. and Ullman J.D. "Equivalence of Relational Expressions" SIAM J. Comptng. 8, 2, 218-246 (May 1979)

[2] Aronson A.R., Jacobs B.E. and Klug A. "A Note on Constraint Calculation for External Views of a Relational Database", to appear TODS

[3] Beeri C., Bernstein P. and Goodman N. "A Sophisticate's Introduction to Database Normalization

Theory" Proceedings VLDB Conf. 1978, West Berlin

[4] Fagin R. "Normal Forms and Relational Database Operators" Proc. ACM-SIGMOD 1979, Boston

[5] Grant J. and Jacobs B. "On Generalized Dependency Statements", submitted for publication

[6] Horn A. "On sentences which are true on direct unions of algebras" J. Symbolic Logic, 16 (1951), 14-21

[7] Klug A. "Calculating Constraints on Relational Expressions", to appear ACM-TODS

[8] Maier D., Mendelzon A.O. and Sagiv Y. "Testing Implications of Data Dependencies" ACM-TODS 4, 4, 455-469 (1979)

[9] Schmidt J.W. "Some High-Level Constructs for Data of Type Relation" ACM TODS 2, 3, pp.247-261