Algebraic Identities and Query Optimization in a Parametric Model for Relational Temporal Databases

Shashi K. Gadia, *Member*, *IEEE Computer Society*, and Sunil S. Nair

Abstract—This paper presents algebraic identities and algebraic query optimization for a parametric model for temporal databases. The parametric model has several features not present in the classical model. In this model, a key is explicitly designated with a relation, and an operator is available to change the key. The algebra for the parametric model is three-sorted; it includes 1) relational expressions that evaluate to relations, 2) domain expressions that evaluate to time domains, and 3) Boolean expressions that evaluate to TRUE or FALSE. The identities in the parametric model are classified as weak identities and strong identities. Weak identities in this model are largely counterparts of the identities in classical relational databases. Rather than establishing weak identities from scratch, a meta inference mechanism, introduced in the paper, allows weak identities to be induced from their respective classical counterpart. On the other hand, the strong identities will be established from scratch. An algorithm is presented for algebraic optimization to transform a query to an equivalent query that will execute more efficiently.

Index Terms—Relational algebra, algebraic optimization, temporal databases, query optimization, relational model.

1 Introduction

THERE has been extensive research on temporal databases in the last decade and a half. Several models to store and query the temporal information have been proposed in [5], [9], [17], [20], [24], [27], [30], [31], [32]. Several types of index structures have been developed for temporal databases that can be used to improve the performance of query processing. Elmasri et al. [6] have proposed an index structure called a time index, that allows a search on the basis of intervals. The index can be used to process certain selections and joins more efficiently. Kolovson and Stonebraker [19] developed structures based on the R-tree proposed in [15]. In [22], the authors described a structure called a Time-split B-tree that can be used to support queries in a transaction time database. Stream processing algorithms to improve the efficiency of join operation have been described in [21]. In [14], [29], several algorithms have been presented to improve different types of join operations in temporal databases.

In the relational model algebraic identities express the equivalence between algebraic expressions [34]. Algebraic optimization exploits the algebraic identities to transform a given user query to an equivalent query requiring fewer disk accesses [26], [28], [34]. Algebraic optimization is based on heuristics such as "perform selections as early as possible" [26], [34]. McKenzie and Snodgrass [23] listed some identities that they consider desirable to facilitate algebraic optimization in temporal databases. In that paper the authors reviewed several temporal algebras and stated how well the mentioned algebraic identities held in those algebras.

The parametric model and an algebra for temporal databases has appeared in [9], [17], [8]. The algebra for the parametric model gives rise to new expressions and identities necessitating the heuristics for algebraic optimization to be revisited. A preliminary version of this paper appeared in [25].

Central to the pedagogy of the parametric model is the perspective that a temporal database is a time-varying classical snapshot database. This perspective provides the rationale for the term *parametric model*: It is a model containing the set of instants that can be viewed as a parameter space with a database that can be viewed as a parametrization of classical database over the parameter space. In this approach, the concept of parametrization is built from the ground up. It is applied to attribute values, associative navigation (A θ B), relations, and algebraic operators. The parametric approach allows the direct counterparts of classical notions to be extended to temporal databases through snapshots. For example, the counterpart of a classical value of an attribute A in the parametric temporal model is a temporal value that is a function from a time domain into dom(A), the domain of the attribute A. In other words a temporal value is a set of ordered pairs of the form ⟨t, v⟩, where t is an instant of time and v is an ordinary value in the domain of A.

1.1 The Homogeneity Assumption and Nulls

In order to develop a new concept it is customary in the database literature to concentrate on values that are not nulls [34]. Following this practice in the parametric model gives rise to a condition called the homogeneity assumption

- *S.K. Gadia is with the Computer Science Department, Iowa State University, Ames, IA 50011. E-mail: gadia@cs.iastate.edu.*
- *S.S. Nair is with Sybase Inc., 1650 65th St., Emeryville, CA 94608. E-mail: nair@sybase.com.*

Manuscript received 20 Mar. 1996; revised 4 Dec. 1997.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 104490.

[9], [16]. The homogeneity assumption says that the attribute values within the same tuple should be defined over the same domain.

Nulls that represent "values that exist but are unknown" are within the realms of homogeneity. Such nulls have been studied in [13], where a null value that exists at instant t but it is unknown is represented as ⟨t, null⟩. The instant t in ⟨t, null⟩ is not required to be known or even known to exist; the homogeneity condition only requires that if t exists for one attribute value in a tuple, it exists for all attribute values in the tuple. Another kind of null that arises within the realms of homogeneity is the one that is due to the extraneous information introduced in the bitemporal model in [11], [1].[1] *Extraneous information* is information that cannot be removed from the database even though it does not exist in the real world. The nulls that represent "values that do not exist" can be incorporated in the parametric model through snapshots. As in the classical case such nulls would lead to some limitations on the syntax of algebraic expressions. In this paper the implication of this is that an identity that holds in the absence of nulls also holds in the presence of nulls only if the expressions on both sides of the identity are defined in the presence of nulls. This simply amounts to masking some identities from the optimization algorithm in the presence of nulls.

1.2 The Cross Product

The above discussion indicates the resilience of the homogeneity in the presence nulls. Now let's consider resilience of homogeneity in another context: the literal cross product r × s of two homogeneous relations r and s. In a tuple of r × s, the attributes of r may not have the same time domain as the attributes of s. Therefore, the above definition of homogeneity is too stringent to admit r × s as a relation. Here it seems appropriate to consider the time dimensions of r and s to be independent of each other. This leads to multihomogeneity: A tuple is said to be *multihomogeneous* if the snapshots of the tuple at vectors of given instants do not contain nulls. Thus, homogeneity has two tiers: unihomogeneity and multihomogeneity. *Unihomogeneity* is when the parametrization of classical databases is with respect to a single time line; *multihomogeneity* is when the parametrization is along more than one time lines. This paper is confined to unihomogeneity. Multihomogeneity will be addressed in [13]. In the rest of this paper the term homogeneity is synonym for unihomogeneity. This paper will include a limited form of cross product, called a unihomogeneous cross product, that can be considered a placeholder for the literal cross product in [13].

1.3 Weak and Strong Notions

The parametric approach injects the classical concepts into a temporal database through snapshots of that database. Such concepts, in the context of temporal databases, are called *weak*. The *weak equality* between two relations means that the two relations have the same snapshots. The notions of weak operators and weak identities will also be introduced. An example of a weak identity is r ∪ s = s ∪ r. It turns out that all notions in parametric databases are not weak. The notions that are not weak are termed *strong*. One may use the cliches "a weak notion is the sum of its parts" and "a strong notion is more than sum of its parts." The terms homogeneity, and weak and strong notions lead to deeper insights into query languages for temporal data.

Even though a classical relation r has a unique structure as a set of tuples, several structures are possible for the counterpart of r in a temporal database. The important fact is that there are useful queries whose results depend upon the choice of the structure. This fact implies that the structure of a relation plays an important role in temporal databases. But then the question arises: What structures are appropriate? The central goal of parametric model is to produce a query language that is as natural for users as possible. The user thinks in terms of objects, the objects are identified by time invariant keys [24], [17], and the time invariant keys are used to provide the structure to a relation.

1.4 Algebraic Identities and the Inference Theorem

The identities in the parametric model can be classified as weak identities and strong identities. The weak identities are direct counterparts of classical identities. A meta principle called the Representation Theorem, which automatically extends classical operators to temporal databases, has been presented in [9]. Along the same line of thought, the Inference Theorem for Weak Identities, a new meta principle that applies to identities is formulated in this paper. The Inference Theorem allows weak temporal identities to be inferred from their classical counterparts. This removes a level of redundancy in the treatment of identities so that the identities already known from classical databases do not have to be reestablished from scratch. However, unlike weak identities, strong identities have to be established within the scope of this paper.

The rest of the paper is organized as follows: Section 2 presents a summary of the parametric model. Section 3 develops algebraic identities that hold for the model. Section 4 introduces the Inference Theorem for Weak Identities and applies it to prove the weak identities introduced in Section 3. Strong identities are also proved in Section 4. Section 5 gives an algorithm for algebraic optimization. The paper concludes in Section 6.

2 THE PARAMETRIC MODEL FOR TEMPORAL DATABASES

This section gives a brief account of the parametric model for temporal databases. The concepts of time, attribute values, tuples, relations, and algebraic expressions are introduced. The relations in the parametric models have a key designated with them, which allows users to realize real-world objects as tuples in the parametric model.

2.1 Temporal Elements

Let's assume that the universe of time consists of an interval [0, NOW] of instants with a linear order ≤ on it. Here, NOW denotes the current instant of time. For simplicity, it is assumed that [0, NOW] is the discrete set {0, 1, ..., NOW}.

1. The parametric model also extends to belief data. Belief data is covered extensively in our works available in a series of seven technical reports of which [10] serves as an index.

Time intervals are not adequate to model the history of an object in a single tuple, and they lead to query languages that are difficult to use [18]. To obtain timestamps that are closed under the set theoretic operations, the concept of temporal elements is introduced. A *temporal element* is a finite union of time intervals. A time interval is a temporal element. An instant t may be identified with the interval [t, t]; thus, it may be regarded as a temporal element. Examples of temporal elements are $[11, 20] \cup [31, 40]$, or NOW. As expected, the set of all temporal elements is closed under $\cup$, $\cap$, and $-$ (complementation with respect to [0, NOW]). The variables μ and ν, possibly with subscripts are used to denote temporal elements. The set of all temporal elements together with $\cup$, $\cap$, $-$, ϕ, and [0, NOW] satisfies the following identities, and hence it forms a Boolean algebra [33]:

$$\mu_1 \cup \mu_2 = \mu_2 \cup \mu_1$$
$$\mu_1 \cap \mu_2 = \mu_2 \cap \mu_1$$
$$(\mu_1 \cup \mu_2) \cup \mu_3 = \mu_1 \cup (\mu_2 \cup \mu_3)$$
$$(\mu_1 \cap \mu_2) \cap \mu_3 = \mu_1 \cap (\mu_2 \cap \mu_3)$$
$$\mu_1 \cup (\mu_1 \cap \mu_2) = \mu_1$$
$$\mu_1 \cap (\mu_1 \cup \mu_2) = \mu_1$$
$$\mu_1 \cup -\mu_1 = [0, \text{NOW}]$$
$$\mu_1 \cap -\mu_1 = \varnothing$$
$$\mu_1 \cup (\mu_2 \cap \mu_3) = (\mu_1 \cup \mu_2) \cap (\mu_1 \cup \mu_3)$$
$$\mu_1 \cap (\mu_2 \cup \mu_3) = (\mu_1 \cap \mu_2) \cup (\mu_1 \cap \mu_3)$$

2.2 Attribute Values

To capture the changing value of an attribute a *temporal value* of an attribute A is defined to be a function from a temporal element into the domain of A. An example of a temporal value of the attribute COLOR is ([25, 32] red, [33, NOW] blue). If ξ is a temporal value, $[\![\xi]\!]$ denotes its domain. Thus $[\![$([25, 32] red, [33, NOW] blue)$]\!]$ = [25, NOW]. $\xi \downarrow \mu$ denotes the restriction of ξ to the temporal element μ. A temporal value is also called an *attribute value* or simply a *value*.

2.3 Associative Navigation

The counterpart of the construct $A\,\theta\,B$ of the relational model is $[\![A\,\theta\,B]\!]$, which captures the time when A is in θ relationship to B. This is introduced through $[\![\xi_1\,\theta\,\xi_2]\!]$ = {t: ξ_1 and ξ_2 are defined at t, and $\xi_1(t)\,\theta\,\xi_2(t)$ is TRUE}. For example $[\![$([25, 32] red, [33, NOW] blue) = ([0, NOW] blue)$]\!]$ = [33, NOW]. The construct $[\![bA\,\theta\,b]\!]$ is also allowed, where b is a constant, which is evaluated by identifying the constant b with the value ([0, NOW] b).

2.4 Homogeneity

A *homogeneous tuple* τ over a scheme R is a function from R such that for every attribute A in R, $\tau(A)$ is a temporal value of A and all the temporal values in the tuple have the same domain. Informally, a tuple is a concatenation of temporal values whose temporal domains are the same. The assumption that all temporal values in a tuple have the same domain makes the tuples homogeneous.

Suppose τ is a tuple. Then the temporal domain of τ is the temporal domain of any attribute and is denoted by $[\![\tau]\!]$. The tuple is said to be *void* if its domain is empty. A void tuple represents absence of information and such a tuple will not be allowed to occur in a relation. If μ is a temporal element, $\tau \downarrow \mu$ is obtained by restricting each value in τ to the temporal element μ.

Suppose r is a set of homogeneous nonvoid tuples over a scheme R. Then $[\![r]\!]$ is defined to be the union of domains of all tuples in r, i.e., $[\![r]\!] = \cup_{\tau \in r}[\![\tau]\!]$. The *restriction* of r to temporal element μ, denoted $r \downarrow \mu$, is defined in a natural manner. The *temporal snapshot* of r at an instant t, denoted r(t), is defined to be $r \downarrow \{t\}$. All values in a snapshot have the same timestamp {t}. An ordinary snapshot $|r \downarrow \{t\}|$ is obtained from the temporal snapshot by omitting the timestamps. Because of the homogeneity assumption, the ordinary snapshot of a temporal relation is a classical relation without nulls.

Note that every set of tuples over a scheme R is not considered a relation. The relations in the parametric model are required to have keys, which will be introduced next.

2.5 Relations and Designation of Keys

A *relation* r over a scheme R, with $K \subseteq R$ as the *key* of r, is a finite set of nonvoid tuples such that no key attribute value in a tuple changes with time, and every pair of tuples assume different values on at least one key attribute. Note that a key in the parametric model is not required to be minimal or nonempty.[2] Sometimes, the key attributes will be underlined for emphasis.

EXAMPLE 1. Fig. 1 shows a database with a relation emp(NAME SALARY DEPT) with NAME as its key, and a relation management (DEPT MANAGER) with DEPT as its key. Note that $[\![$management$]\!] = [11, 60] \cup [71, \text{NOW}]$.

The temporal snapshot emp(50) of the emp relation at t = 50 is shown in Fig. 2a. The ordinary snapshot |emp(50)| of the emp relation at t = 50 is shown in Fig. 2b.

Keys play a critical role in the parametric model. A key provides a persistent identity to an object and it helps in incorporating a real world object as a single tuple in the parametric model (perhaps as best as possible in a relational system). In every snapshot of the management relation in Fig. 1, DEPT and MANAGER functionally determine each other. However, viewing MANAGER as the key would require the management relation to be restructured as the management_1 relation shown in Fig. 3.

2.6 Algebra for Homogeneous Relations

The algebra includes three types of expressions:

1) *relational expressions*, which evaluate to relations;
2) *domain expressions*, which evaluate to temporal elements; and

2. The parametric model extends to spatial data, where an empty key is of special interest. An empty key guarantees that the relation contains only one tuple; such a spatial relation is a tabular representation of a map.

NAME	SALARY	DEPT
[11,60] John	[11,49] 15K [50,54] 20K [55,60] 25K	[11,44] Toys [45,60] Shoes
[0,20] ∪ [41,51] Tom	[0,20] 20K [41,51] 30K	[0,20] Hardware [41,51] Clothing
[71,NOW] Inga	[71,NOW] 25K	[71,NOW] Clothing
[31,NOW] Leu	[31,NOW] 15K	[31,NOW] Toys
[0,44] ∪ [50,NOW] Mary	[0,44] ∪ [50,NOW] 25K	[0,44] ∪ [50,NOW] Credit

(a)

DEPT	MANAGER
[11,49] Toys	[11,44] John [45,49] Leu
[41,47] ∪ [71,NOW]Clothing	[41,47] Tom [71,NOW] Inga
[45,60] Shoes	[45,60] John

(b)

Fig. 1. A database: (a) the emp relation; (b) the management relation.

NAME	SALARY	DEPT
{50} John	{50} 20K	{50} Shoes
{50} Tom	{50} 30K	{50} Clothing
{50} Leu	{50} 15K	{50} Toys
{50} Mary	{50} 25K	{50} Credit

Temporal snapshot emp(50)

NAME	SALARY	DEPT
John	20K	Shoes
Tom	30K	Clothing
Leu	15K	Toys
Mary	25K	Credit

Ordinary snapshot |emp(50)|

Fig. 2. Snapshots of emp relation at t = 50.

3) *Boolean expressions*, which evaluate to Boolean values (TRUE or FALSE).

These three types of expressions are mutually recursive.

Domain Expressions

Domain expressions are the syntactic counterpart of temporal elements. They are formed using temporal elements (e.g., $[11, 20] \cup [31, 40]$), $⟦A⟧$, $⟦A\ \theta\ B⟧$, $⟦A\ \theta\ b⟧$, $⟦E⟧$, $\cup$, $\cap$, and $-$, where A and B are attributes, b is a constant, and E is a relational expression. If μ is a domain expression and τ is a tuple then $\mu(\tau)$, resulting from the substitution of τ in μ, is a temporal element, and such substitution is defined in a natural way. Following is an example of tuple substitution.

DEPT	MANAGER
[11,44] Toys [45,60] Shoes	[11,60] John
[45,49] Toys	[45,49] Leu
[41,47] Clothing	[41,47] Tom
[71,NOW] Clothing	[71,NOW] Inga

Fig. 3. management$_1$ with MANAGER as the key.

EXAMPLE 2. Consider the domain expression ⟦SALARY = 20K⟧. For a given tuple, this expression retrieves the time domain where salary is 20K. Suppose τ is John's tuple in Fig. 1. Then ⟦SALARY = 20K⟧ (τ) evaluates to [50, 54]. As another example, consider the domain expression ⟦SALARY = 20K⟧ $\cap$ (⟦DEPT = Toys⟧ $\cup$ ⟦ DEPT = Shoes⟧). For a given employee, this expression retrieves the time domain consisting of instants where salary is 20K and the department is other than Toys or Shoes. For John's tuple, it evaluates to the empty set $\emptyset$.

Boolean Expressions

Boolean expressions are syntactic counterparts of Boolean values TRUE and FALSE. They are formed using $\mu \subseteq \nu$, where μ and ν are domain expressions. More complex expressions are formed using $\wedge$, $\vee$, and $\neg$. Note that expressions of the form $\mu = \nu$, $\mu \neq \nu$, etc., can be derived using the above constructs. If t is an instant of time, $\{t\} \subseteq \nu$ is written as $t \in \nu$.

Relational expressions are the syntactic counterpart of temporal relations. Before the relational operators are described, it is necessary to introduce the concept of weak equality.

Weakly Equal Relations and Weak Operators

Two relations r and s are said to be *weakly equal* if r and s have the same snapshots, i.e., for all instants t in [0, NOW], r(t) = s(t) [7]. For example, the management$_1$ relation in Fig. 3 is weakly equal to the management relation in Fig. 1. Because the tuples are require to be nonvoid, the following lemma holds:

LEMMA 1. *Suppose* r *and* s *are weakly equal relations (i.e.,* $\forall t \in [0, now]\ r(t) = s(t)$*). If* r *and* s *have the same key, then* r = s.

A unary relational operator O is said to be *weak* if O(r$_1$) is weakly equal to O(r$_2$) whenever r$_1$ is weakly equal to r$_2$. Similarly, a binary relational operator O is said to be *weak* if O(r$_1$, s$_1$) is weakly equal to O(r$_2$, s$_2$) whenever r$_1$ is weakly equal to r_2 and s_1 is weakly equal to s_2. The relational operators are described next. If an operator is not weak, it is said to be *strong*.

Restructuring Operator

The restructuring operator allows the user to change the key of a relation. Suppose r is a relation over R with key K. Further suppose $K' \subseteq R$ is such that for each instant t in [0, NOW] the functional dependency $K' \rightarrow R$ holds in every snapshot $|r(t)|$ of r. Then there is a unique relation weakly equal to r but with key K′. The uniqueness follows from Lemma 1. This unique relation is denoted as $I_{K'}(r)$.[3] Clearly, the restructuring operator is a weak operator.

In the classical case, the restructuring operator is simply the identity operator: I(r) = r for every relation r. In other words, the restructuring operator does nothing useful in the classical case, and therefore, it is invisible. In the temporal case, the restructuring operator is needed to change the designated key of a relation.

EXAMPLE 3. $I_{MANAGER}$(management) = management_1. (See Fig. 1 and Fig. 3.)

Union and Difference

If r and s are relations over the scheme R and with the key K, then $r \cup s$ and $r - s$ also have the same scheme and key. In the case of union, the tuples of r and s that agree on all key attributes are collapsed together; other tuples of r and s remain unaltered in $r \cup s$. In the case of difference, if a tuple τ of r and a tuple τ' of s agree on all their key attributes, then instants where t and τ' agree on all attributes are removed from the domain of τ; and other tuples of r remain unchanged in r – s. As in [7], it can be proved that for every instant t, $(r \cup s)(t) = r(t) \cup s(t)$ and $(r - s)(t) = r(t) - s(t)$; therefore, union and difference are weak operators (See [7]).

Projection

In the parametric model a user thinks in terms of relations that have keys. The operator $\pi_X(r)$ defined by $\pi_X(r) = \{\tau[X]: t \in r\}$ is not a user operator because $\{\tau[X]: \tau \in r\}$, the set of tuples yielded by $\pi_X(r)$, lacks a key. However, $\pi_X(r)$, called the *internal projection*, viewed as an operator on arbitrary sets of tuples, is interesting on its own merit. On one hand, the internal projection helps in understanding the nature of the projection suitable for users, and on the other, the internal projection can be used to enhance optimization of relational expressions.

Now let's introduce the projection operator suitable for the users of the parametric model. Suppose r is a relation over the scheme R, K is the key of R, and X is a subset of R. There are two cases: either the user has a key in mind for the result of the projection or the user wishes to rely on the system to designate a key.

- If the user has a key K′ in mind (the functional dependency $K' \rightarrow X$ must be satisfied by $\pi_X(r)$), then the syntactic form $\Pi_{X:K'}(r)$ can be used, which evaluates to the relation $I_{K'}(\pi_X(r))$, with K′ as its key.

3. In our other papers, $I_{K'}(r)$ is sometimes written as r:K′. The former alternative is more appropriate for this paper.

- Alternatively, the determination of a key can be left to the system. In the simple case when the key K of the input relation is a subset of the projected attributes X, $\Pi_X(r)$ is defined to be the relation $\pi_X(r)$ with K as its key. If K is not a subset of X, then $\Pi_X(r)$ is defined to be the relation $I_X(\pi_X(r))$, with X as its key.

The user projection and the internal projection enjoy the same weak identities. The internal projection is inexpensive compared to the user projection. Therefore, during evaluation of expressions the user projection can be substituted by the internal projection.

Selection

Selection is a powerful operator in temporal databases. If f is a Boolean expression and μ is a domain expression, then the selection $\sigma(r, f, \mu)$ evaluates to $\{\tau \downarrow \mu(\tau): \tau \in r \wedge f(\tau) \wedge \tau \downarrow \mu(\tau) \text{ is not void}\}$. If f evaluates to TRUE for a tuple, σ allows a user to select only a relevant part of it, which is specified by μ. The key of $\sigma(r, f, \mu)$ is the same as the key of r.

EXAMPLE 4. The query *give information about employees while they were in Toys or Shoes if they are currently employed* can be expressed as follows:

$$\sigma(\text{emp}, \text{NOW} \subseteq [\![\text{NAME}]\!], [\![\text{DEPT} = \text{Toys}]\!] \cup [\![\text{DEPT} = \text{Shoes}]\!])$$

Note that in the emp relation the condition NOW $\subseteq$ $[\![$NAME$]\!]$ only holds for Inga, Leu, and Mary. Among these employees the domain $[\![$DEPT = Toys$]\!]$ $\cup$ $[\![$DEPT = Shoes$]\!]$ is empty for Inga and Mary. The empty domain amounts to absence of information. Therefore, tuples of Inga and Mary will be filtered out. Thus, the domain expression μ in the selection $\sigma(r, f, \mu)$ plays a role generally reserved for Boolean expressions in the classical model. Only Leu will meet the criteria in the given query.

The selection operator involves all three types of expressions as operands. Now it is appropriate to make some remarks about how the selection operator in the parametric model relates to the classical selection. Clearly, one difference is that whereas the classical selection involves two operands, the selection operator in the parametric model involves three. Consider $\sigma(r, f,)$ and $\sigma(r, , \mu)$, two less general form of selection in the parametric model obtained by omitting one of the operands:

- When f is omitted in $\sigma(r, f, \mu)$, f defaults to TRUE. In other words $\sigma(r, , \mu)$ is shorthand for $\sigma(r, \text{TRUE}, \mu)$. This form of selection is a weak operator, and it is the one that corresponds to the classical selection.
- When μ is omitted in $\sigma(r, f, \mu)$, μ defaults to [0, NOW]. Thus, $\sigma(r, f,)$ is a shorthand for $\sigma(r, f, [0, \text{NOW}])$. This allows a user to select the whole information in a tuple if it satisfies the condition f. This form of selection is a strong operator. In fact this is the only strong operator form in the algebra for the parametric model.

It turns out that $\sigma(r, f, \mu)$ can be broken into a strong selection followed by a weak selection: $\sigma(r, f, \mu) = \sigma(\sigma(r, f,), \mu)$. It is not an exaggeration to say that the subexpression

σ(r, f,) is the main source of additional expressive power in temporal databases beyond the classical model. In other words, σ(r, f,) marks the main point of departure of temporal databases from classical databases.

LEMMA 2. *The selection operator forms* $\sigma(r, \mathrm{f}, \mu)$ *and* $\sigma(r, \mathrm{f},)$ *are strong.*

PROOF. The lemma can be proved by an example. Consider the management relation in Fig. 1 and the management$_1$ relation in Fig. 3. The management relation is weakly equal to the management$_1$ relation. Consider the following two relational expressions:

$$\sigma(\text{management}, [11,45] \subseteq [\![\text{DEPT} = \text{Toys}]\!],)$$
$$\sigma(\text{management}_1, [11,45] \subseteq [\![\text{DEPT} = \text{Toys}]\!],)$$

The first expression yields the relation with the Toys tuple from the management relation, while the second expression yields the empty relation. Hence, the two expressions are not weakly equal. □

EXAMPLE 5. To illustrate the use of the restructuring operator and the selection operator, the query *give information about managers who were managers at least during* [*11, 45*] can be expressed as follows:

$$\sigma(\text{I}_{\text{MANAGER}}(\text{management}), [11,45] \subseteq [\![\text{MANAGER}]\!],)$$

Natural Join

Suppose r and s are relations with schemes R and S, respectively. A tuple in the natural join r ◊ s of r and s is obtained by concatenating a tuple in r and a tuple in s and only preserving the instants where both the tuples are defined and agree on their common attributes. Formally, suppose τ_1 is a tuple in r and τ_2 is a tuple in s. Then $\tau_1 \lozenge \tau_2$ may be defined as the largest homogeneous tuple τ over RS such that τ agrees with τ_1 on R and with τ_2 on S. Now r ◊ s is defined as $\{\tau_1 \lozenge \tau_2 : \tau_1 \in R, \tau_2 \in S \text{ and } \tau_1 \lozenge \tau_2 \text{ is not null}\}$. The key of r ◊ s is the concatenation of the keys of r and s. Note that (r ◊ s)(t) = r(t) ◊ s(t). Therefore, the natural join is a weak operator.

Unihomogeneous Cross Product

The unihomogeneous cross product r × s is simply defined as a special case of the natural join r ◊ s when the schemes of r and s are disjoint.

The literal cross product of unihomogeneous relations is not necessarily unihomogeneous. The closure of the parametric model under literal cross product requires additional machinery and it is beyond the scope of this paper. The unihomogeneous cross product is a limited form of cross product. The behavior of the unihomogeneous and literal cross product operators is very similar: they satisfy several identities that are similar and have similar heuristics for algebraic optimization. Therefore the unihomogeneous cross product serves as a placeholder for the literal cross product in the algorithm for algebraic optimization to be presented in this paper.

LEMMA 3. *The union, difference, projection, restructuring, natural join, unihomogeneous cross product, and selection operator of the form* $\sigma(r, , \mu)$ *are weak. In addition,* π*, the internal projection, is a weak operator on a set of tuples.*

3 ALGEBRAIC IDENTITIES FOR THE MODEL

This section presents the identities in the parametric model, which are divided in two groups:

1) *domain identities* (the identities among domain expressions) and
2) *relational identities* (the identities among relational expressions).

The identities will be established in Section 4.

3.1 The Domain Identities

With respect to tuple substitution, an important difference exists between the behavior of the atomic domain expressions of the form $[\![A]\!]$, $[\![A\ \theta\ B]\!]$, and $[\![A\ \theta\ b]\!]$ on one hand, and $[\![r]\!]$ and a constant temporal element on the other. The evaluation of an expression in the former group varies from tuple to tuple; but the evaluation of the latter is independent of tuple substitution. Therefore, the expressions appearing in the first group are said to be *tuple dependent*, whereas the expressions appearing in the second group are said to be *tuple independent*. The domain identities are listed below. Note that the prefix "D" in their numbering comes from "domain."

D1 $[\![r \cup s]\!] = [\![r]\!] \cup [\![s]\!]$
D2 $[\![\Pi_X(r)]\!] = [\![r]\!]$
D3 If μ is a tuple independent domain expression, then $[\![\sigma(r, f, \mu)]\!] = [\![\sigma(r, f,)]\!] \cap \mu$
D4 $[\![I_K(r)]\!] = [\![r]\!]$

3.2 The Relational Identities

The relational identities are listed below. Note that the prefix "R" in their numbering comes from "relational." In these identities the projection operator Π_X can also be replaced by π_X, and in that case the condition that the key of the operand is a subset of X is not needed.

Commutativity and Associativity of Natural Join

R1 r ◊ s = s ◊ r
R2 q ◊ (r ◊ s) = (q ◊ r) ◊ s

Cascade of Projections

R3 $\Pi_{A1\cdots An}(\Pi_{B1\cdots Bm}(r)) = \Pi_{A1\cdots An}(r)$, where $A_1 \ldots A_n \subseteq B_1 \ldots B_m$

Cascades of Selections

R4 $\sigma(\sigma(r, f_1,), f_2,) = \sigma(r, f_1 \wedge f_2,)$
R5 $\sigma(\sigma(r, f_1,), f_2,) = \sigma(\sigma(r, f_2,), f_1,)$
R6 $\sigma(\sigma(r, , \mu_1), , \mu_2) = \sigma(r, , \mu_1 \cap \mu_2)$
R7 $\sigma(\sigma(r, , \mu_1), , \mu_2) = \sigma(\sigma(r, , \mu_2), , \mu_1)$
R8 $\sigma(\sigma(r, f,), , \mu) = \sigma(r, f, \mu)$

Cascades of Projections and Selections

R9 If f involves only attributes in $A_1 \ldots A_n$ and $A_1 \ldots A_n \supseteq$ key of r, then

$$\sigma(\Pi_{A1\cdots An}(r), f,) = \Pi_{A1\cdots An}(\sigma(r, f,))$$

R10 If μ involves only attributes in $A_1 \ldots A_n$, then

$$\sigma(\Pi_{A1\cdots An}(r), , \mu) = \Pi_{A1\cdots An}(\sigma(r, , \mu))$$

R11 If f and μ involve only attributes in $A_1 \ldots A_n$ and $A_1 \ldots A_n \supseteq$ key of r, then
$\sigma(\Pi_{A1\cdots An}(r), f, \mu) = \Pi_{A1\cdots An}(\sigma(r, f, \mu))$

This identity is interesting in itself, and its generalization is given in R12.

R12 If f and μ involve attributes $A_1 \ldots A_n$ plus some attributes $B_1 \ldots B_m$, then

$$\Pi_{A1\cdots An}\sigma(r, f, \mu) = \Pi_{A1\cdots An}\sigma(\pi_{A1\cdots AnB1\cdots Bm}(r), f, \mu)$$

This identity is used from left to right. The use of the internal projection π has the advantage that restructuring may be avoided.

Commutativity of Natural Join and Selection

R13 If all the attributes of μ are in R (the scheme of r), then

$$\sigma(r \lozenge s, , \mu) = \sigma(r, , \mu) \lozenge s$$

R14 If all the attributes of μ are in R as well as in S, then

$$\sigma(r \lozenge s, , \mu) = \sigma(r, , \mu) \lozenge \sigma(s, , \mu)$$

R15 If A is an attribute in r, B is an attribute in s, and μ is a tuple independent domain expression, then

$$\sigma (r \lozenge s, \mu \subseteq [\![A\ \theta\ B]\!], \nu) = \sigma (\sigma (r, \mu \subseteq [\![C]\!],) \lozenge s, \mu \subseteq [\![A\ \theta\ B]\!], \nu)$$

where C is any attribute in r

R16 If μ has attributes only in R, then

$$\sigma(r \lozenge s, f, \mu) = \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$$

In fact $\sigma(r \lozenge s, f, \mu) = \sigma(\sigma(r, \mu \cap [\![A]\!] \neq \phi,) \lozenge s, f, \mu)$, where A is any attribute in R since it can be easily shown that $\sigma(r \lozenge s, f, \mu) = \sigma(r \lozenge s, f, \mu \cap [\![A]\!])$.

Commutativity of Selection with Union and Difference

R17 If r and s have the same scheme and the same key, then

$$\sigma(r \cup s, , \mu) = \sigma(r, , \mu) \cup \sigma(s, , \mu)$$

R18 If r and s have the same scheme and same key, then

$$\sigma(r - s, , \mu) = \sigma(r, , \mu) - \sigma(s, , \mu)$$

Commutativity of Projection with Natural Join and Union

R19 If $A_1 \ldots A_n$ is a list of attributes of which $B_1 \ldots B_m$ are attributes of r, $C_1 \ldots C_k$ are attributes of s, and all the common attributes of r and s are in $A_1 \ldots A_n$, then

$$\Pi_{A1\cdots An}(r \lozenge s) = \Pi_{B1\cdots Bm} (r) \lozenge \Pi_{C1\cdots Ck}(s)$$

R20 If r and s have the same scheme and the same key, then

$$\Pi_{A1\cdots An}(r \cup s) = \Pi_{A1\cdots An}(r) \cup \Pi_{A1\cdots An}(s)$$

Commutativity Properties of Restructuring

R21 $I_K(r \cup s) = I_K(r) \cup I_K(s)$
R22 $I_K(r - s) = I_K(r) - I_K(s)$
R23 If $K_1 \rightarrow R$ in r, $K_2 \rightarrow S$ in s, and $K = K_1K_2$ where K_1 is the part of K in R and K_2 is the part of K in S, then

$$I_K(r \lozenge s) = I_{K1}(r) \lozenge I_{K2}(s)$$

R24 If $K \subseteq A_1 \ldots A_n$ and $K \rightarrow R$ holds in r, then

$I_K\Pi_{A1\cdots An}(r) = \Pi_{A1\cdots An}I_K(r)$, and
$I_K\pi_{A1\cdots An}(r) = \Pi_{A1\cdots An}I_K(r)$

R25 If $K \rightarrow R$ (the scheme of r) holds in r, then

$$I_K(\sigma(r, , \mu)) = \sigma(I_K(r), , \mu)$$

Cascade of Restructuring Operators

R26 If $K1 \rightarrow R$ and $K2 \rightarrow R$ hold in r, then

$$I_{K1}(I_{K2}(r)) = I_{K1}(r)$$

Properties of Cross Product

R27 $r \times s = s \times r$
R28 $q \times (r \times s) = (q \times r) \times s$
R29 If A is an attribute in r, and B is an attribute in s, and μ is a tuple independent domain expression, then

$$\sigma(r \times s, \mu \subseteq [\![A\ \theta\ B]\!], \nu) = \sigma(\sigma (r, \mu \subseteq [\![C]\!],) \times s, \mu \subseteq [\![A\ \theta\ B]\!], \nu)$$

where C is any attribute in r

R30 If $A_1 \ldots A_n$ is a list of attributes of which $B_1 \ldots B_m$ are attributes of r and $C_1 \ldots C_k$ are attributes of s, then

$$\Pi_{A1\cdots An}(r \times s) = \Pi_{B1\cdots Bm}(r) \times \Pi_{C1\cdots Ck}(s)$$

R31 If $K_1 \rightarrow R$ in r, $K_2 \rightarrow S$ in s, and $K = K_1K_2$, where K_1 is the part of K in R and K_2 is the part of K in S, then

$$I_K(r \times s) = I_{K1}(r) \times I_{K2}(s)$$

R32 If μ has attributes only in R, then

$$\sigma(r \times s, f, \mu) = \sigma(\sigma(r, \mu \neq \phi,) \times s, f, \mu)$$

This identity is similar to R16. In addition, $\sigma(r \times s, f, \mu) = \sigma(\sigma(r, \mu \cap [\![A]\!] \neq \phi,) \times s, f, \mu)$, where A is any attribute in R.

Commutativity of Selections and Domain Expressions

R33 $\sigma(r, , \mu \cup \nu) = \sigma(r, , \mu) \cup \sigma(r, , \nu)$
R34 $\sigma(r, , \mu \cap \nu) = \sigma(r, , \mu) \cap \sigma(r, , \nu)$
R35 $\sigma(r, , \mu - \nu) = \sigma(r, , \mu) - \sigma(r, , \nu)$
R36 $\sigma(r, , - \mu) = r - \sigma(r, , \mu)$

3.3 Counterexamples

This section ends with a discussion of some identity forms that seem similar to the ones presented above but do not hold in the parametric model. These identity forms are collected in the following proposition. The proof of the proposition consists of a series of counterexamples.

PROPOSITION 1. *The following equalities do not hold*:

$\sigma(\sigma(r, f_1, \mu_1), f_2, \mu_2) = \sigma(r, f_1 \wedge f_2, \mu_1 \cap \mu_2)$
$\sigma(r, f, \mu) = \sigma(\sigma(r, , \mu), f,)$
$\sigma(\Pi_X(r), f,) = \Pi_X(\sigma(r, f,))$
$\sigma(r \lozenge s, f,) = \sigma(r, f,) \lozenge s$ even if all attributes of f are in R
$\sigma(r \cup s, f,) = \sigma(r, f,) \cup \sigma(s, f,)$
$\sigma(r - s, f,) = \sigma(r, f,) - \sigma(s, f,)$

PROOF. The proof consists of a series of counterexamples, each of which is covered by a separate counterexample.

The first counterexample shows that the equality $\sigma(\sigma(r, f_1, \mu_1), f_2, \mu_2) = \sigma(r, f_1 \wedge f_2, \mu_1 \cap \mu_2)$ does not hold. Let r(AB) be a relation with the single tuple ([0, 10] a, [0, 10] b), let $f_1 = f_2 = [0, 10] \subseteq [\![B = b]\!]$, and let $\mu_1 = \mu_2 = [0, 5]$. Then the left-hand side of the equality evaluates to the empty relation while the right-hand side evaluates to the relation with the single tuple ([0,5] a, [0,5] b).

Now it is shown that $\sigma(r, f, \mu) = \sigma(\sigma(r, , \mu), f,)$ does not hold. Let r(AB) be a relation with the single tuple ([0,10] a , [0,10] b), let $f = [0,10] \subseteq [\![B = b]\!]$, and let $\mu = [0, 5]$. Then the right-hand side of the equality evaluates to the empty relation while the left-hand side evaluates to the relation with the single tuple ([0,5]a, [0,5] b) .

The next counterexample shows that the equality $\sigma(\Pi_X(r), f,) = \Pi_X(\sigma(r, f,))$ does not hold. Suppose r is a relation over AB with A as its key having the single tuple ([0,10]a, [0,5]b_1[6,10]b_2) . Let f be $[0,5] \subseteq [\![B = b_1]\!]$ and X be the attribute B. In this case, the key of r, which is A, is not a subset of X, which is B. The expression $\sigma(\Pi_B(r), f,)$ evaluates to the tuple ([0,5]b_1). However, $\Pi_B(\sigma(r, f,))$ evaluates to the relation with two tuples {([0, 5]b_1), ([6, 10]b_2)}. Hence, the equation $\sigma(\Pi_X(r), f,) = \Pi X(\sigma(r, f,))$ does not hold in general.

Now it is shown that the equality $\sigma(r \lozenge s, f,) = \sigma(r, f,) \lozenge s$ does not hold even if all the attributes in f are attributes in R. Let r(AB) consist of the single tuple ([0,5]a, [0,5]b) and let s(CD) consist of the single tuple ([0,3]c, [0,3]d) . Let f be the formula $[0,5] \subseteq [\![B = b]\!]$. Then the expression $\sigma(r \lozenge s, f,)$ evaluates to the empty relation while $\sigma(r, f,) \lozenge s$ evaluates to the relation with the single tuple ([0,3]a, [0,3]b, [0, 3]c, [0,3]d). This example also shows that the equality $\sigma(r \lozenge s, f, \mu) = \sigma(r, f, \mu) \lozenge s$ does not hold even if μ has attributes only in R.

Next consider the equality $\sigma(r \cup s, f,) = \sigma(r, f,) \cup \sigma(s, f,)$. The equality $\sigma(r - s, f,) = \sigma(r, f,) - \sigma(s, f,)$ is treated in a similar manner. Let r(AB) and s(AB) have key A. Let r(AB) consist of the single tuple ([0, 5]a, [0, 5]b) and let s(AB) consist of the single tuple ([6, 10]a, [6, 10]b). Let f be the formula $[0, 5] \subseteq [\![B = b]\!]$. Then $\sigma(r \cup s, f,)$ evaluates to the single tuple relation ([0, 10]a, [0, 10]b), while $\sigma(r, f,) \cup \sigma(s, f,)$ evaluates to the relation consisting of the single tuple ([0,5]a, [0,5]b).

This counterexample shows that the equality $I_K(\sigma(r, f,)) = \sigma(I_K(r), f,)$ does not hold. Let r(AB) be the relation with key A having the single tuple ([0, 10]a, [0, 5]b_1, [6, 10]b_2). Let f be $[0, 5] \subseteq [\![B = b_1]\!]$. Then $I_B(\sigma(r, f,))$ has the tuples ([0, 5]a, [0, 5]b_1) and ([6, 10]a, [6, 10]b). However, $\sigma(I_B(r), f,)$ has only the tuple ([0,5]a, [0,5]b_1). □

4 THE INFERENCE THEOREM FOR WEAK IDENTITIES AND ITS APPLICATION

This section establishes the identities presented in the previous section. The concept of weak identities is formally introduced. The Inference Theorem to induce the weak identities from their classical counterparts is stated and proved. Although the Inference Theorem is adequate to cover all weak identities from Section 3, a generalization of the Inference Theorem to widen its applicability to additional weak identities is also presented. Strong identities are proved from scratch.

In this section the variables e and E are used to denote a temporal or a classical relational expression. The variables μ and M denote a (temporal) domain expression. Note that the counterpart of the Boolean expressions in the classical model are the domain expressions in the parametric model and not the Boolean expression in the parametric model. The Boolean expressions in the parametric model do not have a counterpart in the classical model. Therefore, the semantics of the Boolean expressions in the parametric and classical models are quite different. To avoid confusion, the variable f denotes a Boolean expressions in the parametric model and the variable ϕ denotes a Boolean expressions in the classical model. Subscripts are used whenever necessary. Note that the inference theorem has to consider only weak expressions and such expressions do not involve the Boolean expressions in the parametric model.

The identities used for algebraic optimization typically ignore domain-specific relationships among constants. For example, the fact that $A > 10 \vee A \leq 10$ is TRUE is not taken into account, and an identity such as $\sigma(r, A > 10) \cup \sigma(r, A \leq 10) = r$ goes undetected by the algebraic optimizer. The same is not true of structural identities arising from the algebraic operators; e.g., the identity $\sigma(r, A > 10) \cup \sigma(r, A \leq 10) = \sigma(r, A \leq 10) \cup \sigma(r, A > 10)$ may be detected by the optimizer. The latter is a particular case of the identity $\sigma(e, \phi_1) \cup \sigma(e, \phi_2) = \sigma(e, \phi_2) \cup \sigma(e, \phi_1)$, where e is a variable over relational expressions and ϕ_1 and ϕ_2 are variables over classical Boolean expressions. Note that this expression makes sense only if the attributes in ϕ_1 and ϕ_2 are subsets of the attributes in e. Thus a variable hides all details about a subexpression except the attributes of the subexpression. There is no need to be concerned about the detection of every possible identity and it is enough to concentrate only on the identities that are to be used in the optimizer. Such identities have already listed in the previous section. The list is not meant to be exhaustive, and it is left open to future exploration. However, it must be emphasized that the identities do not take facts such as $[\![A > 10]\!] \cup [\![A \leq 10]\!] = [\![A]\!]$ into account. Similarly, the identities ignore relationships among constant temporal elements, e.g., the fact that the union of [11, 20] and [21, 30] is [11, 30] will not taken into account. Note that the identity $\sigma(r, , [11, 20]) \cup \sigma(r, , [21, 30]) = \sigma(r, [11, 30])$ will be ignored by the optimizer, but the identity $\sigma(r, , [11, 20]) \cup \sigma(r, , [21, 30]) = \sigma(r, , [21, 30]) \cup \sigma(r, , [11, 20])$ will be taken into account.

4.1 Weak Identities and Their Translation

Suppose $E_1(e_1, e_2, ..., e_n)$ and $E_2(e_1, e_2, ..., e_n)$ are relational expressions involving the relational expressions $e_1, e_2, ..., e_n$. Then the identity $E_1(e_1, e_2, ..., e_n) = E_2(e_1, e_2, ..., e_n)$ is said to be *weak* if both E_1 and E_2 consist of only weak operations on $e_1, e_2, ..., e_n$.

Fig. 4 gives rules of translation T that transform a weak relational expression E into classical expressions T(E). The translation of $[\![E]\!]$ necessitates the introduction of a Boolean expression IsNonEmpty(·) [2]. For a classical relation r, IsNonEmpty(r) checks if r is nonempty, and returns TRUE or FALSE.

EXAMPLE 6. Consider the temporal identity $\sigma(\sigma(r, , \mu_1), , \mu_2) = \sigma(r, , \mu_1 \cap \mu_2)$. Its translation is the classical identity $\sigma(\sigma(r, \phi_1), \phi_2) = \sigma(r, \phi_1 \wedge \phi_2)$. Note that among other things, the variables μ_1 and μ_2 for domain expressions are translated to the variables ϕ_1 and ϕ_2 over Boolean expressions in the classical algebra. As another example, consider the weak identity $I_K(\sigma(r, , \mu)) = \sigma(I_K(r), , \mu)$. Its translation is the trivial classical identity $\sigma(r, \phi) = \sigma(r, \phi)$.

Now consider the identity $\sigma(\sigma(r, f_1,), f_2,) = \sigma(\sigma(r, f_2,), f_1,)$ in the parametric model. This identity is not a weak identity. Therefore the rules of translation do not apply to this identity. Next, consider the following identities:

$$\sigma(r, , [11,20]) \cup \sigma(r, , [\![A \le B]\!]) = \sigma(r, , [\![A \le B]\!]) \cup \sigma(r, , [11,20]) \text{ and}$$

$$\sigma(r, , [11,20]) \cup \sigma(r, , [21,30]) = \sigma(r, [11,30]).$$

Before these identities are translated into classical identities they should be preprocessed by a variable substitution. The substitutions $r \to e$, $[11,20] \to \mu_1$, and $[\![A \le B]\!] \to \mu_2$ change the first identity to $\sigma(e, , \mu_1) \cup \sigma(e, , \mu_2) = \sigma(e, , \mu_2) \cup \sigma(e, , \mu_1)$. The translation of this weak identity is the classical identity $\sigma(e, \phi_1) \cup \sigma(e, \phi_2) = \sigma(e, \phi_2) \cup \sigma(e, \phi_1)$. Note that the attributes of μ_2 as well as ϕ_2 are A and B, whereas μ_1 and ϕ_1 have no attributes.

The second identity is preprocessed as $\sigma(r, , \mu_1) \cup \sigma(r, , \mu_2) = \sigma(r, \mu_3)$, and then translated as $\sigma(r, \phi_1) \cup \sigma(r, \phi_2) = \sigma(r, \phi_3)$. It should be clear that the validity of the first identity is detected by the system, but the validity of the second identity goes undetected.

4.2 The Inference Theorem for Weak Identities

To state and prove the Inference Theorem for Weak Identities, a lemma that considers all relational expressions exhaustively with the exception of constant temporal elements must be presented. Recall from Section 2.4 that if r is a snapshot relation, $|r|$ denotes the ordinary classical relation after removing the timestamps from r. Similarly $|\tau|$ denotes the classical snapshot tuple after removing the timestamps from a temporal snapshot tuple τ.

Weak temporal expression	Classical expression T(·)
Variable $e, e_1, e_2, \cdots$ over relational expressions	No change
Variables $\mu, \mu_1, \mu_2, \cdots$ over domain expressions	$\phi, \phi_1, \phi_2, \cdots$, respectively
$E_1 \cup E_2$ $E_1 - E_2$, $E_1 \lozenge E_2$ $E_1 \times E_2$	$T(E_1) \cup T(E_2)$ $T(E_1) - T(E_2)$ $T(E_1) \lozenge T(E_2)$ $T(E_1) \times T(E_2)$
$\sigma(E, , \mu)$	$\sigma(T(E), T(\mu))$
$\Pi_{X:K}(E)$, $\pi_{X:K}(E)$	$\Pi_X(T(E))$
$I_K(E)$	$T(E)$
$[\![A\theta B]\!]$	$A\theta B$
$[\![E]\!]$	IsNonEmpty(T(E))
$-M$	$\neg T(M)$
$M_1 \cup M_2$	$T(M_1) \vee T(M_2)$
$M_1 \cap M_2$	$T(M_1) \wedge T(M_2)$

Fig. 4. Translation rules for weak expressions.

LEMMA 4. *Suppose* $r_1, ..., r_n$ *are temporal snapshot relations defined over a fixed but arbitrary instant and* $E_1(r_1, ..., r_n)$ *is a relational expression consisting of weak operators and not involving constant temporal elements. Then the following holds*:

$$\tau \in E(r_1, ..., r_n) \textit{ if and only if}$$
$$|\tau| \in T(E)(|r_1|, ..., |r_n|)$$

PROOF. Assume that the domain of the snapshot relations is {t}. The proof is by induction on the complexity of E.

- Suppose E is r_1. Then $T(r_1) = r_1$ and $\tau \in r_1$ if and only if $|\tau| \in |r_1|$.
- Suppose E is $E_1 \cup E_2$. (The cases $E_1 - E_2$, $E_1 \times E_2$, $\Pi_{X:K}(E_1)$, $\pi_X(E_1)$, and $I_K(E_1)$ are similar.)

$\tau \in (E_1 \cup E_2)(r_1, ..., r_n)$

$\Leftrightarrow \tau \in E_1(r_1, ..., r_n)$ or $\tau \in E_2(r_1, ..., r_n)$

$\Leftrightarrow |\tau| \in T(E_1)(|r_1|, ..., |r_n|)$ or $|\tau| \in T(E_2)(|r_1|, ..., |r_n|)$, by the induction hypothesis

$\Leftrightarrow |\tau| \in (T(E_1) \cup T(E_2))(|r_1|, ..., |r_n|)$

$\Leftrightarrow |\tau| \in T(E_1 \cup E_2)(|r_1|, ..., |r_n|)$, by the rule of translation for $T(E_1 \cup E_2)$.

- Suppose E is $\sigma(E_1, , M)$. The proof is by induction on the complexity of M.
  - M is $[\![A\ \theta\ B]\!]$
    $\tau \in \sigma(E_1, , [\![A\ \theta\ B]\!])\ (r_1, ..., r_n)$
    $\Leftrightarrow \tau \in E_1(r_1, ..., r_n)$ and $[\![A\ \theta\ B]\!](\tau) = \{t\}$
    $\Leftrightarrow |\tau| \in T(E_1)(|r_1|, ..., |r_n|)$
    and $\tau[A](t)\ \ \theta\ \tau[B](t)$ holds
    $\Leftrightarrow |\tau| \in T(E_1)(|r_1|, ..., |r_n|)$
    and $|\tau|[A]\ \theta\ |\tau|[B]$ holds
    $\Leftrightarrow |\tau| \in \sigma(T(E_1), A\ \theta\ B)(|r_1|, ..., |r_n|)$
    $\Leftrightarrow |\tau| \in T(\sigma(E_1, , [\![A\ \theta\ B]\!]))(|r_1|, ..., |r_n|)$
  - Suppose M is $[\![E_2]\!]$
    $\tau \in \sigma(E_1, , [\![E_2]\!])(r_1, ..., r_n)$
    $\Leftrightarrow \tau \in E_1(r_1, ..., r_n)$ and $[\![E_2]\!](r_1, ..., r_n)(\tau) = \{t\}$
    $\Leftrightarrow \tau \in E_1(r_1, ..., r_n)$ and $t \in [\![E_2]\!](r_1, ..., r_n)$ holds
    $\Leftrightarrow \tau \in E_1(r_1, ..., r_n)$ and $([\![E_2]\!])(r_1, ..., r_n)$ is not empty
    $\Leftrightarrow |\tau| \in T(E_1)(|r_1|, ..., |r_n|)$ and IsNonEmpty$(E_2)(|r_1|, ..., |r_n|)$ holds
    $\Leftrightarrow |\tau| \in \sigma(T(E_1)$, IsNonEmpty$(E_2))$ $(|r_1|, ..., |r_n|)$
    $\Leftrightarrow |\tau| \in T(\sigma(E_1, , [\![E_2]\!]))(|r_1|, ..., |r_n|)$
  - Suppose M is $M_1 \cup M_2$. (The cases $M_1 \cap M_2$ and $-M_1$ are similar.)
    $\tau \in \sigma(E_1, , M_1 \cup M_2)(r_1, ..., r_n)$
    $\Leftrightarrow \tau \in \sigma(E_1, , M_1)(r_1, ..., r_n)$
    or $\tau \in \sigma(E_1, , M_1)(r_1, ..., r_n)$
    $\Leftrightarrow |\tau| \in \sigma(T(E_1), T(M_1)(|r_1|, ..., |r_n|)$
    or $|\tau| \in \sigma(T(E_1), T(M_2)(|r_1|, ..., |r_n|)$
    $\Leftrightarrow |\tau| \in \sigma(T(E_1), T(M_1)$
    $\cup\ T(M_2)(|r_1|,..., |r_n|)$
    $\Leftrightarrow |\tau| \in \sigma(T(E_1), T(M_1 \cup M_2))$
    $(|r_1|, ..., |r_n|)$
    $\Leftrightarrow |\tau| \in T(\sigma(T(E_1), , M_1 \cup M_2)$
    $(|r_1|, |r_n|)$ □

Now the Inference Theorem for Weak Identities is ready to be stated and proved. The theorem makes an exception for constant temporal elements and follows directly from Lemma 4.

THEOREM 1 (Inference Theorem for Weak Identities). *Suppose* $E_1(e_1, ..., e_n)$ *and* $E_2(e_1, ..., e_n)$ *are weak relational expressions involving* $e_1, ..., e_n$, *such that* E_1 and E_2 *do not involve constant temporal elements, and the key of* E_1 = *key of* E_2. *Then* $E_1 = E_2$ *holds as a temporal identity if and only if* $T(E_1) = T(E_2)$ *holds as a classical identity.*

PROOF. Suppose $T(E_1) = T(E_2)$ holds as a classical identity. If t is an arbitrary instant, then

$\tau \in E_1(e_1, ..., e_n)(t)$
$\Leftrightarrow \tau \in E_1(e_1(t), ..., e_n(t))$,
because E_1 is weak in $e_1, ..., e_n$
$\Leftrightarrow |\tau| \in T(E_1)(|e_1(t)|, ..., |e_n(t)|)$, by Lemma 4
$\Leftrightarrow |\tau| \in T(E_2)(|e_1(t)|, ..., |e_n(t)|)$,
by hypothesis $T(E_1) = T(E_2)$
$\Leftrightarrow \tau \in E_2(e_1(t), ..., e_n(t))$, by Lemma 4
$\Leftrightarrow \tau \in E_2(e_1, ..., e_n)(t)$, because E_2 is weak in $e_1, ..., e_n$

This proves that $E_1(t) = E_2(t)$ for every instant t. Because E_1 and E_2 have the same key, from Lemma 1 it follows that the temporal identity $E_1 = E_2$ holds. The converse is proved in a similar manner. □

4.3 The Proof of Identities

Now let's proceed to prove the identities presented in Section 3. The weak identities follow from the Inference Theorem. The strong identities are covered by the following lemma:

LEMMA 5. *The strong identities* R4, R5, R8, R9, R11, R12, R15, R16, R29, *and* R32 *hold.*

PROOF. Proof of the identities R12 and R15 is easy. The identities R4, R5, R8, R9, R11, and R16 are considered below. Identities R29 and R32 are similar to R15 and R16, respectively.

Consider Identity R4: $\sigma(\sigma(r, f_1,), f_2,) = \sigma(r, f_1 \wedge f_2,)$

$\tau \in \sigma(\sigma(r, f_1,), f_2,)$
$\Leftrightarrow \tau \in \sigma(r, f_1,) \wedge f_2(\tau)$
$\Leftrightarrow \tau \in r \wedge f_1(\tau) \wedge f_2(\tau)$
$\Leftrightarrow \tau \in r \wedge (f_1 \wedge f_2)(\tau)$
$\Leftrightarrow \tau \in \sigma(r, f_1 \wedge f_2,)$

Identity R5: $\sigma(\sigma(r, f_1,), f_2,) = \sigma(\sigma(r, f_2,), f_1,)$ follows from the commutativity of the Boolean expressions $f_1 \wedge f_2 = f_2 \wedge f_1$.

Consider Identity R8: $\sigma(\sigma(r, f,), , \mu) = \sigma(r, f, \mu)$.
$\tau \in \sigma(r, f, \mu)$
$\Leftrightarrow \exists\tau' \in r$ such that $f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau$
and $\tau' \downarrow \mu(\tau')$ is not empty
$\Leftrightarrow \exists\tau' \in \sigma(r, f,)$ such that $\tau' \downarrow \mu(\tau') = \tau$
and $\tau' \downarrow \mu(\tau')$ is not empty
$\Leftrightarrow \tau \in \sigma(\sigma(r, f,), , \mu)$

Now consider Identity R9. Suppose f involves only attributes in $A_1 ... A_n$ and $A_1 ... A_n \supseteq$ key of r. The proof of $\sigma(\Pi_{A1 \cdots An}(r), f,) = \Pi_{A1 \cdots An}(\sigma(r, f,))$ is as follows:

$\tau \in \sigma(\Pi_{A1 \cdots An}(r), f,)$
$\Leftrightarrow \tau \in \Pi_{A1 \cdots An}(r) \wedge f(\tau)$
$\Leftrightarrow \exists\tau' \in r$ such that $\tau'[A_1 ... A_n] = \tau \wedge f(\tau)$
$\Leftrightarrow \exists\tau' \in r$ such that $\tau'[A_1 ... A_n] = \tau \wedge f(\tau')$
(since f only involves attributes in $A_1 ... A_n$)
$\Leftrightarrow \exists\tau' \in \sigma(r, f,)$ such that $\tau'[A_1 ... A_n] = \tau$
$\Leftrightarrow \tau \in \Pi_{A1 \cdots An}(\sigma(r, f,))$

Next consider Identity R11. Suppose f and μ involve only attributes in $A_1 ... A_n$ and $A_1 ... A_n \supseteq$ key of r. The proof of $\sigma(\Pi_{A1 \cdots An}(r), f, \mu) = \Pi_{A1 \cdots An}(\sigma(r, f, \mu))$ is as follows:

$\sigma(\Pi_{A1\cdots An}(r), f, \mu)$
$= \sigma(\sigma(\Pi_{A1\cdots An}(r), f,), , \mu)$ (by R8)
$= \sigma(\Pi_{A1\cdots An}\sigma(r, f,), , \mu)$ (by R9)
$= \Pi_{A1\cdots An}(\sigma(\sigma(r, f,), , \mu))$ (by R10)
$= \Pi_{A1\cdots An}(\sigma(r, f, \mu))$ (by R8)

The proof of Identity R16, $\sigma(r \lozenge s, f, \mu) = \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$, is as follows:

$\tau \in \sigma(r \lozenge s, f, \mu)$
$\Rightarrow \exists\tau' \in r \lozenge s\, [f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau$
$\wedge\, \tau' \downarrow \mu(\tau')$ is not empty]
$\Rightarrow \exists\tau_r \in r\, \exists\tau_s \in s$[for some $\tau' \downarrow f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau$
$\wedge\, \tau' \downarrow \mu(\tau')$ is not empty
$\wedge\, \tau_r \downarrow [\![\tau_s]\!] = \tau'[R]$ and $\tau_s \downarrow [\![\tau_r]\!] = \tau'[S]$
and τ' is over RS and τ' is not empty]
$\Rightarrow \exists\tau_r \in \sigma(r, \mu \neq \phi,), \exists\tau_s \in s$[for some $\tau'[f(\tau')$
$\wedge\, \tau' \downarrow \mu(\tau') = \tau \;\wedge\; \tau' \downarrow \mu(\tau')$ is not empty
$\wedge\, \tau_r \downarrow [\![\tau_s]\!] = \tau'[R]$ and $\tau_s \downarrow [\![\tau_r]\!] = \tau'[S]$
and τ' is over RS and τ' is not empty)]
(since μ has attributes only in R hence
$\mu(\tau) \neq \phi \Rightarrow \mu(\tau_r) \neq \phi$
$\Rightarrow \exists\tau' \in \sigma(r, \mu \neq \phi,) \lozenge s[f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau$
$\wedge\, \tau' \downarrow \mu(\tau')$ is not empty]
$\Rightarrow \tau \in \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$

Hence, $\sigma(r \lozenge s, f, \mu) \subseteq \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$. (α)

$\tau \in \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$
$\Rightarrow \exists\tau' \in \sigma(r, \mu \neq \phi,) \lozenge s\, [f(\tau') \wedge \tau' \downarrow \mu(\tau')$
$= \tau \;\wedge\; \tau' \downarrow \mu(\tau')$ is not empty]
$\Rightarrow \exists\tau_r \in \sigma(r, \mu \neq \phi,)\, \exists\tau_s \in s$[for some $\tau'[f(\tau')$
$\wedge\, \tau' \downarrow \mu(\tau') = \tau \;\wedge \tau' \downarrow \mu(\tau')$ is not empty
$\wedge\, \tau_r \downarrow [\![\tau_s]\!] = \tau'[R]$ and $\tau_s \downarrow [\![\tau_r]\!] = \tau'[S]$
and τ' is over RS and τ' is not empty]
$\Rightarrow \exists\tau_r \in r, \exists\tau_s \in s$[for some $\tau'[f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau$
$\wedge\, \tau' \downarrow \mu(\tau')$ is not empty
$\wedge\, \tau_r \downarrow [\![\tau_s]\!] = \tau'[R]$ and $\tau_s \downarrow [\![\tau_r]\!] = \tau'[S]$
and τ' is over RS and τ' is not empty]
$\Rightarrow \exists\tau' \in r \lozenge s[f(\tau') \wedge \tau' \downarrow \mu(\tau') = \tau \;\wedge \tau' \downarrow \mu(\tau')$
is not empty]
$\Rightarrow \tau \in \sigma(r \lozenge s, f, \mu)$

Hence $\sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu) \subseteq \sigma(r \lozenge s, f, \mu)$ (β)

The result follows from (α) and (β).

Recall that (R16) states that if μ has attributes only in R then $\sigma(r \lozenge s, f, \mu) = \sigma(\sigma(r, \mu \neq \phi,) \lozenge s, f, \mu)$. In fact, $\sigma(r \lozenge s, f, \mu) = \sigma(\sigma(r, \mu \cap [\![A]\!] \neq \phi,) \lozenge s, f, \mu)$ where A is any attribute in R, since it is easily shown that $\sigma(r \lozenge s, f, \mu) = \sigma(r \lozenge s, f, \mu \cap [\![A]\!])$. □

THEOREM 2. *The identities* D1 *to* D4 *and* R1 *to* R36 *hold.*

PROOF. The weak identities R1, R2, R3, R6, R7, R10, R13, R14, R17 to R28, R30, R31, and R33 to R36 are inferred from the Inference Theorem by reducing them to their classical counterparts. (See [34] for the classical counterparts.) The identities D1 to D4 follow directly from the definition of the $[\![\cdot]\!]$ operator. Only left are the strong identities R4, R5, R8, R9, R11, R12, R16, R15, R29, and R32, which have been proved in Lemma 5. □

Weak temporal expression	Translation T(·)(t)
A constant temporal element μ	$t \in \mu$

Fig. 5. Translation rule for a constant temporal element.

4.4 The Generalized Inference Theorem for Weak Identities

The weak expressions covered by the Inference Theorem do not include the constant temporal elements. The translation of an expression E that does not involve constant temporal elements is uniform for all instants t. The translation of a constant temporal element μ is "$t \in \mu$" (see Fig. 5). The phrase "$t \in \mu$" has no classical counterpart and it makes explicit use of a variable t that ranges over instants. This leads to the addition of the parameter t to the concept of translation: the translation now is T (·)(t) instead of simply T(·). Note that T(·)(t) is independent of t for all expressions in Fig. 4. Now two lemmas are stated that lead us to generalize the Inference Theorem to include constant temporal elements.

LEMMA 6. *Suppose* r *is a temporal snapshot relation defined at a fixed but arbitrary instant* t, *and* μ *is a constant temporal element. Then* $\tau \in \sigma(r, , \mu)$ *if and only if* $|\tau|$ *is in* $|r|$ *and the condition* $t \in \mu$ *is satisfied.*

LEMMA 7. *Suppose* $r_1, \ldots, r_n$ *are temporal relations and* $E_1(r_1, \ldots, r_n)$ *is a relational expression using weak operators. Further suppose that the rules of translation in Fig. 4 are supplemented with the rule in Fig. 5. Then* $\tau \in E(r_1(t), \ldots, r_n(t))$ *if and only if* $|\tau| \in T(E)(t)(|r_1(t)|, \ldots, |r_n(t)|)$.

THEOREM 3 (Generalized Inference Theorem for Weak Identities). *Suppose* $E_1(e_1, \ldots, e_n)$ *and* $E_2(e_1, \ldots, e_n)$ *are weak relational expressions involving* $e_1, \ldots, e_n$, *such that the key of* E_1 = *key of* E_2. *Then* $E_1 = E_2$ *holds as a temporal identity if* $T(E_1)(t) = T(E_2)(t)$ *holds as a classical identity for all instants* t.

EXAMPLE 7. Now reconsider the identity $\sigma(r, , [11, 20]) \cup \sigma(r, , [21, 30]) = \sigma(r, [11, 30])$ from Example 6. With $\sigma(r, , [11,20]) \cup \sigma(r, , [21,30])$ as E_1 and $\sigma(r, [11,30])$ as E_2, and for an arbitrary instant t, $T(E_1)(t) = T(E_2)(t)$ becomes

$$(\sigma(r, , [11,20]) \cup \sigma(r, , [21,30]))(t) = \sigma(r, [11,30])(t)$$

This is proved as follows:

$\tau \in (\sigma(r, , [11,20]) \cup \sigma(r, , [21,30]))(t)$
$\Leftrightarrow (\tau \in r$ and $t \in [11,20])$
or $(\tau \in r$ and $t \in [21,30])$
$\Leftrightarrow \tau \in r$ and $(t \in [11,20]$ or $t \in [21,30])$
$\Leftrightarrow \tau \in r$ and $(t \in [11,30])$
$\Leftrightarrow \tau \in \sigma(r, , [11,30])$

Note that even though the Generalized Inference Theorem for Weak Identities requires the proof of $T(E_1)(t) = T(E_2)(t)$ for all instants t, in reality it amounts to verifying simple set theoretic facts such as $[11, 20] \cup [21, 30] =$

[11, 30]. It also turns out that it is not necessary to verify $T(E_1)(t) = T(E_2)(t)$ at all instants, but only for instants in the spectrum of the identity $E_1 = E_2$. Along the lines of [9], the *spectrum* of an identity $E_1 = E_2$ can be defined as the union of {0, NOW} and the set of end points of all intervals of the form [a, b) that are mentioned in the identity. Instead of formalizing the concept of a spectrum, it is illustrated in the following example.

EXAMPLE 8. Again consider the identity $\sigma(r, , [11, 20]) \cup \sigma(r, , [21, 30]) = \sigma(r, [11, 30])$. To prove it, first note that it uses intervals [11, 20], [21, 30], and [11, 30]. Rewrite these intervals in the form [a, b) to obtain the intervals [11, 21), [21, 31), and [11, 31). Thus the spectrum of the identity $E_1 = E_2$ is {0, 11, 21, 31, NOW}.

To prove $T(E_1)(t) = T(E_2)(t)$ at all instants t, it is sufficient to verify it for the instants 0, 11, 21, 31, and NOW in the spectrum. For example at t = 11, $T(E_1)(t) = T(E_2)(t)$ becomes $(\sigma(r, , [11, 20]) \cup \sigma(r, , [21, 30]))(11) = \sigma(r, [11, 30])(11)$, which is same as $11 \in [11, 20]$ or $11 \in [11, 20]$ if and only if $11 \in [11, 30]$, which holds.

5 Algorithm for Algebraic Optimization

This section sketches an algorithm that uses the identities from the previous section. The algorithm transforms a given temporal query to a more optimal query based upon certain heuristics. Since the join and the cross product are the most expensive operators, the algorithm tries to reduce the size of the operands of these operators. Similarly, the restructuring operator is a fairly expensive operator in temporal databases, and the algorithm attempts to reduce the size of its operand. These and other ideas in the algorithm are listed below:

- *Perform selections as early as possible.* A selection reduces the size of the relation. Since the result of the selection may be an operand in a larger expression, this could reduce the cost of execution of the query.
- *Perform projections as early as possible.* A projection also reduces the size of the relation by making the tuples smaller. In some cases, the number of tuples may also decrease due to duplicate removal.
- *Reduce the size of the operands in a join.* The join is the most expensive relational operation. Reducing the size of the operands in a join makes a substantial reduction in the cost of the operation. The size of the operands is usually reduced by using the two heuristics above.
- *Reduce the size of the operands in the restructuring operation.* The restructuring operation is also a fairly expensive operation. The size of the operand to the operation can be reduced by using the first two heuristics mentioned above. The size of the operand can also be reduced by pushing the restructuring operation ahead of the join or cross product to yield two restructuring operations but with much smaller relations (see Identities R23 and R31 in Section 3). On the average the cost of the join stays the same.
- *Combine the cascades of unary operations.* It may sometimes be possible to transform two selections into a single selection operation. Similarly, in a cascade of projection operations, all but the last projection can be eliminated. Also, in a cascade of restructuring operations only the last one needs to be executed.
- *Remove redundant operations.* If R is the scheme of an expression E, then $\Pi_R(E) = E$ and the redundant projection can be removed. Similarly, if K is the key of an expression E, then $I_K(E) = E$ and the restructuring operation is redundant.
- *Combine operations that can be executed simultaneously.* Sometimes two or more operations can be executed simultaneously in a single step. For instance, in the expression $\Pi_X\sigma(r \Diamond s, f, \mu)$ the selection and possibly the projection can be done with the join. Similarly, in $\Pi_X\sigma(I_K(r)\ f, \mu)$, the selection and possibly the projection can be performed with the restructuring.

The algorithm for algebraic optimization is shown in Fig. 6. At the outer level, this algorithm simply calls one of the two algorithms R or D:

- the algorithm R is called if the given expression is a relational expression, and
- algorithm is D is called if the expression is a domain expression.

Algorithms R and D are mutually recursive and do most of the work.

Here are some examples to illustrate the optimization algorithm.

EXAMPLE 9. Suppose r($\underline{A}$ B C) and s($\underline{C}$ D E F) are two relations. Consider the query

$$\Pi_{ACDF}(\sigma(r \Diamond s, [0, 5] \subseteq [\![A]\!], \text{NOW}))$$

In Step R-1.2, the generalized version in R16 is applied. In $\sigma(r \Diamond s, [0, 5] \subseteq [\![A]\!], \text{NOW})$, the domain expression μ is NOW. Because μ does not contain any attributes, the attributes of μ are vacuously a subset of ABC, the scheme of r, as well as CDEF, the scheme of s. Therefore, the rule R16 is applied twice, adding selections to r as well as to s and obtaining

$$\Pi_{ACDF}\sigma(\sigma(r, [\![A]\!] \cap \text{NOW} \neq \phi,) \Diamond \sigma(s, [\![D]\!] \cap \text{NOW} \neq \phi,), [0, 5] \subseteq [\![A]\!], \text{NOW})$$

In Step R-1.3, the identities R11 followed by R19 are applied.

When R11 is applied, the projection is pushed inside the outer selection, and the following is obtained:

$$\sigma(\Pi_{ACDF}(\sigma(r, [\![A]\!] \cap \text{NOW} \neq \phi,) \Diamond \sigma(s , [\![D]\!] \cap \text{NOW} \neq \phi,), [0, 5] \subseteq [\![A]\!], \text{NOW}))$$

When R19 is applied, the projection is commuted with the natural join, and the following is obtained:

$$\sigma(\Pi_{AC}\sigma(r, [\![A]\!] \cap \text{NOW} \neq \phi,) \Diamond \Pi_{CDF}\sigma(s , [\![D]\!] \cap \text{NOW} \neq \phi,), [0, 5] \subseteq [\![A]\!], \text{NOW})$$

Algorithm Algebraic Optimization

Input: An algebraic expression E

Output: An equivalent expression E′, which is more efficient than E

Procedure:

If E is a relational expression call Algorithm R
else if E is a domain expression call Algorithm D.

Algorithm R: Optimization of a relational subexpression

Input: A relational expression E

Output: An equivalent expression E′, which is more efficient than E

Procedure:

R-1. Apply the following sequence of steps:

R-1.1. Apply identities R4 to R8 to transform a single selection into a cascade of selections because it may be easier to push a simpler selection further down the expression tree.

R-1.2. Apply identities R4 to R18, and R25 (including the generalization in R16) to move selections as far down the tree as possible.

R-1.3. Apply identities R3, R9 to R11, R12, R19, R24 and R30 to move projections as far down the expression tree as possible. Eliminate redundant projections where possible.

R-1.4. Apply identities R3 to R12 to transform a cascade of selections and projections into a single selection, a single projection or a selection followed by a projection.

R-1.5. Use identities R23, R24, R26, and R31 to move restructuring operations down the tree. Use identities R23 and R24 to move restructuring operations before the join and cross product. Identity R26 eliminates all but the last restructuring operation in a cascade of restructuring operations. Also, eliminate a restructuring operation if it restructures an expression on its own key, i.e., eliminate redundant restructuring.

R-1.6. Eliminate redundant selections, i.e., replace selections of the form $\sigma(r, ,)$ by r.

R-1.7. Apply identities R33 to R36 to reduce the number of relational operators $\cup$, $\cap$, and $-$.

R-1.8. Identify sequences of operations that can be executed simultaneously. For instance, in $\Pi_X(\sigma(I_K(r), f, \mu)$ the selection and projection can be performed while doing the restructuring I_K if Π_X involves no further restructuring.

R-2. From the resulting relational expression, consider each domain expression not nested within another domain expression. Optimize it using algorithm D.

Algorithm D: Optimization of a domain expression

Input: A domain expression E

Output: An equivalent expression E′, which is more efficient than E

Procedure:

D-1. Apply identities D1 to D4 from the left-hand side to the right-hand side. In this way, an expression $[\![E]\!]$ reduces to the union/intersection of $[\![E_1]\!]$, $[\![E_2]\!]$, ⋯, $[\![E_n]\!]$, where E_1, E_2, ⋯, E_n are simpler than E. Furthermore, if any E_i is a stored relation r, then it is possible that $[\![r]\!]$ is stored with the relation.

D-2. From the resulting domain expression, consider each relational expression not nested within another relational expression. Optimize it using algorithm R.

Fig. 6. Algorithm for algebraic optimization.

This can be computed in the following steps. Compute $s_1 = \Pi_{AC}\sigma(r, [\![A]\!] \cap \text{NOW} \neq \phi,)$ in a single step. The selection restricts r to those tuples that have information about the current instant. Compute $s_2 = \Pi_{CDF}\sigma(s, [\![D]\!] \cap \text{NOW} \neq \phi,)$ in a single step. The selection restricts s to those tuples that have information about the current instant. Compute $\sigma(s_1 \lozenge s_2, [0, 5] \subseteq [\![A]\!], \text{NOW})$ in a single step. The join is performed on relations smaller than in the original query, thus improving efficiency.

EXAMPLE 10. Suppose r($\underline{A}$ B C) and s($\underline{D}$ E F G) are relations and suppose it is known that the functional dependency C → ABC holds in r. Consider the query

$$\Pi_{CDE}(\sigma(I_{CD}(r \lozenge s), , [\![B = 5]\!] \cap [\![E = F]\!])).$$

In Step R–1.1, identity R6 is applied to change the selection to a cascade of selections. This gives

$$\Pi_{CDE}(\sigma(\,\sigma(I_{CD}(r \lozenge s), , [\![B = 5]\!]), , [\![E = F]\!]))$$

In Step R–1.2, Identities R25 followed by R13 (twice) are applied to push the selections inside the restructuring operator and the join to get

$$\Pi_{CDE}(I_{CD}(\sigma(r, , [\![B = 5]\!]) \Diamond \sigma(s, , [\![E = F]\!])))$$

In step R-1.3, identities R24 followed by R19 are applied to push the projection inside the restructuring operator and the natural join to get

$$I_{CD}(\pi_C(\sigma(r, , [\![B = 5]\!]))) \Diamond \Pi_{DE}\sigma(s, , [\![E = F]\!])$$

Step R-1.4 has no effect on the expression.

In Step R-1.5 identity R23 is applied followed by R24 to push the restructuring operation inside the join and projection. The resulting redundant restructuring operator on the second part of the join is removed, resulting in the following:

$$\Pi_C I_C\sigma(r, , [\![B = 5]\!]) \Diamond \Pi_{DE}\sigma(s, , [\![E = F]\!])$$

This query can be evaluated in the following steps. Compute $s_1 = \sigma(r, , [\![B = 5]\!])$. Compute $s_2 = \Pi_C(I_C(s_1 1))$ in a single step. Compute $s_3 = \Pi_{DE}\sigma(s, , [\![E = F]\!])$ in a single step. Compute $s_2 \Diamond s_3$.

EXAMPLE 11. The following example illustrates the mutual recursion between algorithms R and D of the Optimization Algorithm. Consider the following query:

$$[\![\sigma(\text{management}, , [0, 20])]\!].$$

Apply identity D3 to transform the given expression into $[\![\sigma(\text{management}, ,)]\!] \cap [0, 20]$. Now Algorithm R is called for $\sigma(\text{management}, ,)$, returning management, and therefore the expression $[\![\text{management}]\!] \cap [0, 20]$ is obtained. It is clear that the resulting expression is more efficient to evaluate than the original one. The efficiency would be further enhanced if the domain $[\![\text{management}]\!]$ is stored with the management relation.

6 CONCLUSIONS

This paper has presented algebraic identities and optimization for the parametric model for temporal databases. The injunction of domain expressions allows users to express many natural language selections as a selections in the algebra for the model. Therefore, the need for optimization in the parametric model is not artificially inflated as in some temporal models that necessitate additional joins to express queries [12], [32].

The Inference Theorem for Weak Identities establishes a direct correspondence between classical relational identities and weak identities in the parametric model. The theorem is an interesting principle that draws a boundary between classical databases and temporal databases. On the classical side of this boundary, the theorem was applied to avoid redundancy in the treatment of weak temporal identities. Strong identities, the identities on the truly temporal side of the boundary, are new and they were established from scratch.

The concept of a key makes the parametric model object driven, the identities more natural, and the algebra more declarative. For example, $\sigma(r, , \mu \cup \nu) = \sigma(r, , \mu) \cup \sigma(r, , \nu)$ and other identities in the parametric model allow a user the flexibility to formulate a natural language query in different ways without having to worry about a unique evaluation. For a given strong query form, different weakly equal representations of a relation *do* yield results that are not even weakly equal. This, leads one to postulate that in a given context only one representation of a relation is the most natural. It is our conjecture that the most natural representations for relations in temporal databases are those that are obtained through the concept of a key.

The exploitation of identities in the presence of index structures is a possible area of future work. Another direction is to extend optimization to aggregates, subqueries, and other SQL-like constructs. In the case of classical data, promising optimization techniques for such syntactic constructs have recently appeared in [3], [4]. It would be interesting to extend those techniques to the parametric model.

ACKNOWLEDGMENTS

The authors thank the referees and the editor for their suggestions in improving the presentation of this paper.

REFERENCES

[1] G. Bhargava and S.K. Gadia, “The Concept of an Error in a Database: An Application of Temporal Databases,” *Proc. INSDOC COMAD ’90 Int’l Conf. Management of Data*, Dec. 1990; also available as Technical Report No. TR97-15, Computer Science Dept., Iowa State Univ., Ames, 1997.

[2] G. Bhargava and S.K. Gadia, “Relational Database Systems with Zero Information Loss,” *IEEE Trans. Knowledge and Data Eng.*, vol. 5, pp. 76-87, 1993.

[3] G. Bhargava, P. Goel, and B. Iyer, “Hypergraph Based Reorderings of Outerjoin Queries with Complex Predicates,” *Proc. ACM SIGMOD Int’l Conf. Management of Data*, 1995.

[4] G. Bhargava, P. Goel, and B. Iyer, “Efficient Processing of Outer Joins and Aggregate Functions,” *Proc. Third IEEE Int’l Conf. Data Eng.*, 1995.

[5] J. Clifford and A. Croker, “The Historical Data Model (HRDM) and Algebra Based on Lifespans,” *Proc. 11th IEEE Int’l Conf. Data Eng.*, 1995.

[6] R. Elmasri, G.T.J. Wu, and Y.-J. Kim, “The Time Index: An Access Structure for Temporal Data,” *Proc. 16th VLDB Conf.*, 1990.

[7] S.K. Gadia, “Weak Temporal Relations,” *Proc. Fifth Ann. ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, 1986.

[8] S.K. Gadia, “Toward A Multi-Homogeneous Model for a Temporal Database,” *Proc. IEEE Int’l Conf. Data Eng.*, 1986.

[9] S.K. Gadia, “A Homogeneous Relational Model and Query Languages for Temporal Databases,” *ACM Trans. Database Systems*, vol. 13, no. 4, 1988.

[10] S.K. Gadia, “A Bibliography and Index of Our Works on Belief Data,” Technical Report No. TR97-13, Computer Science Dept., Iowa State Univ., Ames, 1997.

[11] S.K. Gadia, and G. Bhargava, “A Formal Treatment of Errors and Updates in a Relational Database,” 1988-89, unpublished manuscript available as Technical Report No. TR97-14, Computer Science Dept., Iowa State Univ., Ames, 1997.

[12] S.K. Gadia and S. Nair, “Temporal Databases: A Prelude to Parametric Data,” chapter in *Temporal Databases: Theory, Design, and Implementation*. Benjamin-Cummings, pp. 28-66, 1993.

[13] S.K. Gadia, S. Nair, and Y. Poor, “Incomplete Information in Relational Temporal Databases,” *Proc. 18th Int’l Conf. Very Large Data Bases*, pp. 395-406, 1992.

[14] H. Gunadhi and A. Segev, “Efficient Indexing Methods for Temporal Relations,” *IEEE Trans. Knowledge and Data Eng.*, 1991.

[15] A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching,” *Proc. ACM SIGMOD Int’l Conf. Management of Data*, 1984.

[16] S.K. Gadia and J.H. Vaishnav, "A Query Language for a Homogeneous Temporal Database," *Proc. Fourth Ann. ACM SIGACT-SIGMOD Symp. Principles of Database Systems*, pp. 51-56, 1985.
[17] S.K. Gadia and C.-S. Yeung, "A Generalized Model for a Relational Temporal Database," *Proc. ACM SIGMOD Int'l Conf. Management of Data*, 1988.
[18] S.K. Gadia and C.-S. Yeung, "Inadequacy of Interval Timestamps in Temporal Databases," *Information Sciences*, vol. 54, pp. 1-22, 1991.
[19] C. Kolovson and M. Stonebraker, "Indexing Techniques for Historical Databases," *Proc. Fifth Int'l Conf. Data Eng.*, 1989.
[20] N. Lorentzos and R.G. Johnson, "Extending Relational Algebra to Extend Temporal Data," *Information Systems*, vol. 13, pp. 289-296, 1988.
[21] T.Y. Leung and R.R. Muntz, "Query Processing for Temporal Databases," *Proc. IEEE Int'l Conf. Data Eng.*, 1990.
[22] D. Lomet and B. Salzberg, "Transaction-Time Databases," chapter in *Temporal Databases: Theory, Design, and Implementation*, A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, eds., pp. 388-417, 1993.
[23] E. McKenzie and R. Snodgrass, "Evaluation of Relational Algebras Incorporating the Time Dimension in Databases," *ACM Computing Surveys*, vol. 23, 1991.
[24] S. Navathe and R. Ahmed, "A Temporal Relational Model and Query Language," *Information Systems*, vol. 49, 1989.
[25] S. Nair and S.K. Gadia, "Algebraic Optimization in a Relational Model for Temporal Databases," *Proc. First Int'l Conf. Information and Knowledge Management*, pp. 169-176, 1992.
[26] M.J. Smith and P.Y.T. Chang, "Optimizing the Performance of a Relational Algebra Database Interface," *Comm. ACM*, 1975.
[27] N.L. Sarda, "Algebra and Query Language for a Historical Data Model," *Computer J.*, vol. 33, pp. 11-18, 1988.
[28] M.H. Scholl, *Theoretical Foundation of Algebraic Optimization Using Unnormalized Relations*, Lecture Notes in Computer Science 243, 1986.
[29] A. Segev, "Join Processing and Optimization in Temporal Relational Databases," chapter in *Temporal Databases: Theory, Design, and Implementation*. Benjamin-Cummings, pp. 356-387, 1993.
[30] R. Snodgrass, "The Temporal Query Language TQuel," *ACM Trans. Database Systems*, vol. 12, pp. 247-298, 1987.
[31] A.U. Tansel, "Adding Time Dimension to Relational Model and Extending Relational Algebra," *Information Systems*, vol. 11, no. 4, 1986.
[32] A.U. Tansel, J. Clifford, S.K. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, *Temporal Databases: Theory, Design, and Implementation*. Benjamin-Cummings, 1993.
[33] J.P. Tremblay and R. Manohar, *Discrete Math. Structures with Applications to Computer Science*. McGraw Hill, Computer Science Series, 1975.
[34] J.D. Ullman, *Principles of Database and Knowledge-Base Systems*, vol. II: The New Technologies. Computer Science Press, 1988.

Shashi K. Gadia obtained the BS degree (Hons.) and MS degree in mathematics, both from the Birla Institute of Technology and Science, Pilani, India; and the PD degree in mathematics from the University of Illinois, Urbana, in 1977. He has been a member of the faculty of the Computer Science Department at Iowa State University since 1986. His research interests include temporal-, spatial-, and multilevel-security data; optimization; incomplete information; query languages; user interfaces; pattern matching; and relational- and object-oriented databases. He is a member of the IEEE Computer Society.

Sunil S. Nair received the MS degree in physics from the University of Bombay, India, in 1984; and the MS and PhD degrees in 1988 and 1993, respectively, in computer science from Iowa State University. He is currently with Sybase Inc., Emeryville, California. His research interests include temporal databases, spatial databases, incomplete information, and query optimization, and he has published a number of papers in these areas in conference proceedings.