Supporting Valid Time: An Historical Algebra

TR87-008

August 1987

Edwin McKenzie
Richard Snodgrass

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175

Abstract

We define an historical algebra for historical relations. This historical algebra, a straightforward extension of the conventional relational algebra, supports valid time, the time when an object or relationship in the enterprise being modeled is valid. Historical versions of the five relational operators union, difference, cartesian product, selection, and projection are defined and a new operator, historical derivation, is introduced. The algebra includes aggregates and is shown to have the expressive power of the temporal query language TQuel. The algebra is consistent with the user-oriented model of historical relations as space-filling objects and satisfies all but one of the associative, commutative, and distributive tautologies involving union, difference, and cartesian product.

Contents

Time is a universal attribute of both events and objects in the real world. Events occur at specific points in time; objects and the relationships among objects exist over time. The ability to model this temporal dimension of the real world is essential to many computer system applications (e.g., econometrics, banking, inventory control, medical records, and airline reservations). Unfortunately, conventional database management systems do not support the time-varying aspects of the real world. Conventional databases can be viewed as *snapshot* databases in that they represent the state of the real world at one particular point in time. As a database is changed to reflect changes in the real world, out-of-date information, representing past states of the real world, is deleted. The need for database support for time-varying information has received increasing attention; in the last five years, more that 80 articles relating time to information processing have been published [McKenzie 1986].

In previous papers, we identified three orthogonal kinds of time that a database management system (DBMS) needs to support: valid time, transaction time, and user-defined time [Snodgrass & Ahn 1985, Snodgrass & Ahn 1986]. *Valid time* concerns modeling time-varying reality. The valid time of, say, an event is the clock time at which the event occurred in the real world, independent of the recording of that event in some database. *Transaction time*, on the other hand, concerns the storage of information in the database. The transaction time of an event is the transaction number (an integer) of the transaction that stored the information about the event in the database. *User-defined time* is an uninterpreted domain for which the DBMS supports the operations of input, output, and perhaps comparison. As its name implies, the semantics of user-defined time is provided by the user or application program. These three types of time are orthogonal in the support required of the DBMS.

In this paper we propose extending the relational algebra [Codd 1970] to enable it to handle valid time. The relational algebra already supports user-defined time in that user-defined time is simply another domain, such as integer or character string, provided by the DBMS [Bontempo 1983, Overmyer & Stonebraker 1982, Tandem 1983]. The relational algebra, however, supports neither valid time nor transaction time. Hence, for clarity, we refer to the relational algebra hereafter as the snapshot algebra and our proposed algebra, which supports valid time, as an *historical* algebra. We do not consider here any extension of the snapshot algebra or our historical algebra to support transaction time. Elsewhere [McKenzie & Snodgrass 1987A] we describe an approach for adding transaction time to the snapshot algebra and show that this approach applies without change to all historical algebras supporting valid time. This approach for adding transaction time to the snapshot algebra and historical algebras also provides for scheme evolution [McKenzie & Snodgrass 1987B]. Because valid time and transaction time are orthogonal, we are able to study each type of time in isolation.

1 Approach

To extend the snapshot algebra to support valid time, we define formally an historical algebra. We provide formal definitions for an historical relation, six algebraic operators, and two historical aggregate functions. We then show that the algebra has the expressive power of the *TQuel* (*T*emporal *QUE*ry *L*anguage) [Snodgrass 1987] facilities that support valid time.

The algebra reflects our basic design goal to define an historical algebra that has as many of the most desirable properties of an historical algebra as possible. For example, we wanted the historical algebra to be a straightforward extension of the snapshot algebra so that relations and algebraic expressions in the snapshot algebra would have equivalent counterparts in the historical algebra. Yet, we also wanted the algebra to support historical queries and adhere to the user-oriented model of historical relations as space-filling objects, where the additional, third dimension is valid time. Hence, we did not restrict historical relations to first-normal form, insist on time-stamping of entire tuples, or require that time-stamps be atomic-valued because each of these restrictions would have prevented the algebra from having other, more highly desirable properties. All design decisions (e.g., to time-stamp attributes rather than tuples) were made so that the resulting algebra would possess a maximal set of desirable properties. In Section 4 we briefly discuss our major design decisions and the importance of those decisions in determining the algebra's properties. A detailed discussion of desirable properties of historical algebras as well as an evaluation of our algebra and the historical algebras proposed by others, using the identified properties as evaluation criteria, can be found elsewhere [McKenzie & Snodgrass 1987C].

Efficient direct implementation of the algebra was not one of our primary design objectives. Rather, our goal was to define an algebra that preserves the associative, commutative, and distributive properties of the snapshot algebra in order that optimization strategies developed for the snapshot algebra can be applied in implementations of the historical algebra. Our formulation of the algebraic operators would be inefficient if mapped directly into an implementation. While we can envision more efficient implementations, incorporating such efficiencies in the semantics would have made it much more complex. Finally, we expect that new optimization strategies, unique to the historical algebra, also will be used in its implementation.

In the next section we define our historical algebra. Then we show that the algebra has the expressive power of the TQuel calculus. We conclude the paper with a discussion of the major design decisions we made in defining the algebra. The notational conventions used in the paper are described in Appendix A.

2 An Historical Algebra for Historical Relations

The algebra presented in this section is an extension of the snapshot algebra. As such, it retains the basic restrictions on attribute values found in the snapshot algebra. Neither set-valued attributes nor tuples with duplicate attribute values are allowed. Valid time is represented by a set-valued time-stamp that is associated with individual attributes. A time-stamp represents possibly disjoint intervals and the time-stamps assigned to two attributes in a given tuple need not be identical.

2.1 Historical Relation

Assume that we are given a relation scheme defined as a finite set of attribute names $\mathcal{N} = \{N_1, \ldots, N_m\}$. Corresponding to each attribute name N_a, $1 \leq a \leq m$, is a domain $\mathcal{D}_a$, an arbitrary, non-empty, finite or denumerable set [Maier 83]. Let the positive integers be the domain $\mathcal{T}$, where each element of $\mathcal{T}$ represents a time quantum [Anderson 82]. Assume that, if t_1 immediately precedes t_2 in the linear ordering of $\mathcal{T}$, then t_1 represents the interval $[t_1, t_2)$. The granularity of time (e.g., nanosecond, month, year) associated with $\mathcal{T}$ is arbitrary. Note that because time is a continuous function, all measures of time can be viewed as measures of intervals. Hence, when we speak of a "point in time," we actually refer to an interval whose duration is determined by the granularity of the measure of time being used to specify that "point in time." Also, let the domain $\wp(\mathcal{T})$ be the power set of $\mathcal{T}$. An element of $\wp(\mathcal{T})$ is then a set of integers, each of which represents an interval of unit duration. Also, any group of consecutive integers $t_1, \ldots, t_n$ appearing in an element of $\wp(\mathcal{T})$, together represent the interval $[t_1, t_n + 1)$.

If we let *value* range over the domain $\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_m$ and *valid* range over the domain $\wp(\mathcal{T})$, we can define an ***historical tuple*** p as a mapping from the set of attribute names to the set of ordered pairs (*value*, *valid*),

$$p : \mathcal{N} \rightarrow (\mathcal{D}_1 \cup \cdots \cup \mathcal{D}_m, \ \wp(\mathcal{T}))$$

with the following restrictions:

- $\forall a, \ 1 \leq a \leq m, \ value(p(N_a)) \in \mathcal{D}_a$ and
- $\exists a, \ 1 \leq a \leq m, \ valid(p(N_a)) \neq \emptyset$.

Hereafter, we will refer to $p(N_a)$ simply as p_a, where a denotes attribute N_a in scheme $\mathcal{N}$, when there is no ambiguity of meaning. Note that it is possible for all but one attribute to have an empty time-stamp.

Let $\mathcal{P}$ be the domain of all tuples over the attribute names of the relation scheme $\mathcal{N}$ and the domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$, and $\wp(\mathcal{T})$. Define two tuples, $p, \ p' \in \mathcal{P}$, to be ***value-equivalent*** if and only if $\forall a, \ 1 \leq a \leq m, \ value(p_a) = value(p'_a)$. An ***historical relation*** h is then defined as a finite set of historical tuples, with the restriction that no two tuples in the relation are value-equivalent. $\mathcal{H}$ represents the domain of all historical relations on the relation scheme.

EXAMPLE. Assume that we are given the relation scheme *Student* = {*Name*, *Course*} and the following set of tuples over this relation scheme. For this and all later examples, assume that the granularity of time is a semester relative to the Fall semester 1980. Hence, 1 represents the Fall semester 1980, 2 represents the Spring semester 1981, etc.

$$\begin{aligned} \mathrm{S} = \{ \ & \langle (\text{Phil}, \ \{1,3\}), \ (\text{English}, \ \{1,3\}) \rangle \ , \\ & \langle (\text{Norman}, \ \{1,2\}), \ (\text{English}, \ \{1,2\}) \rangle \ , \\ & \langle (\text{Norman}, \ \{5,6\}), \ (\text{Calculus}, \ \{5,6\}) \rangle \ , \\ & \langle (\text{Phil}, \ \{4\}), \ (\text{English}, \ \{4\}) \rangle \ \} \end{aligned}$$

For notational convenience we enclose each attribute value in parentheses and each tuple in angular brackets (i.e., $\langle \ \rangle$). We assume the natural mapping between attribute names and attribute values (e.g., *Name* $\rightarrow$ (Phil, $\{1,3\}$), and *Course* $\rightarrow$ (English, $\{1,3\}$)). Note that S is not an historical relation because there are value-equivalent tuples in the set (the first and fourth tuples are value-equivalent). If we replace the two value-equivalent tuples in S with a single tuple, then the new set S_1 is an historical relation.

$$\begin{aligned} S_1 = \{ \; & \langle(\text{Phil}, \{1,3,4\}), (\text{English}, \{1,3,4\})\rangle, \\ & \langle(\text{Norman}, \{1,2\}), (\text{English}, \{1,2\})\rangle, \\ & \langle(\text{Norman}, \{5,6\}), (\text{Calculus}, \{5,6\})\rangle \; \} \end{aligned} \qquad \square$$

2.2 Historical Operators

We present eight operators that serve to define the historical algebra. Five of these operators — union, difference, cartesian product, projection, and selection — are analogous to the five operators that serve to define the snapshot algebra for snapshot relations [Ullman 82]. Each of these five operators on historical relations is represented as $\hat{op}$ to distinguish it from its snapshot algebra counterpart op. Historical derivation is a new operator that replaces the time-stamp of each attribute in a tuple with a new time-stamp, where the new time-stamps are computed from the existing time-stamps of the tuple's attributes. The remaining two operators, aggregation and unique aggregation, compute aggregates. After defining the operators, we show that all eight preserve the value-equivalence property of historical relations.

EXAMPLE. The three relations S_1, S_2, and S_3 are used in the examples that accompany the definitions of the operators. S_2, like S_1, is an historical relation over the relation scheme $Student = \{Name, Course\}$. S_3 is an historical relation over the relation scheme $Home = \{Name, State\}$. While the attributes of a tuple in S_1, S_2, and S_3 have the same time-stamp, in general, attributes within a tuple can have different time-stamps.

$$\begin{aligned} S_2 = \{ \; & \langle(\text{Phil}, \{3,4\}), (\text{English}, \{3,4\})\rangle, \\ & \langle(\text{Norman}, \{7\}), (\text{Calculus}, \{7\})\rangle, \\ & \langle(\text{Tom}, \{5,6\}), (\text{English}, \{5,6\})\rangle \; \} \end{aligned}$$

$$\begin{aligned} S_3 = \{ \; & \langle(\text{Phil}, \{1,2,3\}), (\text{Kansas}, \{1,2,3\})\rangle, \\ & \langle(\text{Phil}, \{4,5,6\}), (\text{Virginia}, \{4,5,6\})\rangle, \\ & \langle(\text{Norman}, \{1,2,5,6\}), (\text{Virginia}, \{1,2,5,6\})\rangle, \\ & \langle(\text{Norman}, \{7,8\}), (\text{Texas}, \{7,8\})\rangle \; \} \end{aligned} \qquad \square$$

2.2.1 Union

Let Q and R be historical relations of m-tuples over the same relation scheme. Then the historical union of Q and R, denoted $Q \hat{\cup} R$, is defined as

$$
\begin{aligned}
Q \hat{\cup} R \triangleq & \{q^m \mid Q(q) \wedge \neg(\exists r,\ r \in R \wedge \forall a,\ 1 \leq a \leq m,\ value(q_a) = value(r_a))\} \\
& \bigcup \{r^m \mid R(r) \wedge \neg(\exists q,\ q \in Q \wedge \forall a,\ 1 \leq a \leq m,\ value(r_a) = value(q_a))\} \\
& \bigcup \{u^m \mid \exists q \exists r,\ q \in Q \wedge r \in R \wedge \forall a,\ 1 \leq a \leq m,\ value(u_a) = value(q_a) = value(r_a) \\
& \qquad \wedge\ valid(u_a) = valid(q_a) \cup valid(r_a)\}
\end{aligned}
$$

$Q \hat{\cup} R$ is the set of tuples that are in Q, R, or both, with the restriction that each pair of value-equivalent tuples is represented by a single tuple. Note that if a tuple in Q and a tuple in R are value-equivalent, then they are represented in $Q \hat{\cup} R$ by a single tuple. The time-stamp associated with each attribute of this tuple in $Q \hat{\cup} R$ is the set union of the time-stamps of the corresponding attribute in the value-equivalent tuples in Q and R.

EXAMPLE.

$$
\begin{aligned}
S_1 \hat{\cup} S_2 = \{ & \langle(\text{Phil}, \{1,3,4\}), (\text{English}, \{1,3,4\})\rangle, \\
& \langle(\text{Norman}, \{1,2\}), (\text{English}, \{1,2\})\rangle, \\
& \langle(\text{Norman}, \{5,6,7\}), (\text{Calculus}, \{5,6,7\})\rangle, \\
& \langle(\text{Tom}, \{5,6\}), (\text{English}, \{5,6\})\rangle \ \}
\end{aligned}
$$

□

2.2.2 Difference

Let Q and R be historical relations of m-tuples over the same relation scheme. Then the historical difference of Q and R, denoted $Q \hat{-} R$, is defined as

$$
\begin{aligned}
Q \hat{-} R \triangleq & \{q^m \mid Q(q) \wedge \neg(\exists r,\ r \in R \wedge \forall a,\ 1 \leq a \leq m,\ value(q_a) = value(r_a))\} \\
& \bigcup \{u^m \mid (\exists q \exists r,\ q \in Q \wedge r \in R \wedge \forall a, 1 \leq a \leq m,\ value(u_a) = value(q_a) = value(r_a) \\
& \qquad \wedge\ valid(u_a) = valid(q_a) - valid(r_a)) \\
& \qquad \wedge\ (\exists a,\ 1 \leq a \leq m \wedge valid(u_a) \neq \emptyset) \\
& \}
\end{aligned}
$$

$Q \hat{-} R$ is the set of all tuples that satisfy three criteria. First, a tuple in $Q \hat{-} R$ must have a value-equivalent counterpart in Q. Second, the time-stamp of each attribute of a tuple in $Q \hat{-} R$ must equal the set difference of the time-stamps of the corresponding attribute in the value-equivalent tuple in Q and the value-equivalent tuple in R, if any. Third, the time-stamp of at least one attribute of each tuple in $Q \hat{-} R$ must be non-empty.

EXAMPLE. $\mathrm{S}_1 \hat{-} \mathrm{S}_2 = \{$ ⟨(Phil, {1}), (English, {1})⟩ ,
⟨(Norman, {1,2}), (English, {1,2})⟩ ,
⟨(Norman, {5,6}), (Calculus, {5,6})⟩ $\}$ □

2.2.3 Cartesian Product

Let Q be an historical relation of m_1-tuples and R be an historical relation of m_2-tuples. Then $Q \hat{\times} R$, the historical cartesian product of Q and R, is defined as

$$Q \hat{\times} R \triangleq$$
$$\{u^{m_1+m_2} \mid \ (\exists q,\ q \in Q \wedge \forall a,\ 1 \le a \le m_1,\ \mathit{value}(u_a) = \mathit{value}(q_a) \wedge \mathit{valid}(u_a) = \mathit{valid}(q_a))$$
$$\wedge (\exists r,\ r \in R \wedge \forall a,\ 1 \le a \le m_2,\ \mathit{value}(u_{m_1+a}) = \mathit{value}(r_a) \wedge \mathit{valid}(u_{m_1+a}) = \mathit{valid}(r_a))$$
$$\}$$

The cartesian product operator for historical relations is identical to the cartesian product operator for snapshot relations. $Q \hat{\times} R$ is the set of $(m_1 + m_2)$-tuples whose components $u_1, \ldots, u_{m_1}$ form a tuple in Q and whose components $u_{m_1+1}, \ldots, u_{m_1+m_2}$ form a tuple in R.

EXAMPLE.

$\mathrm{S}_1 \hat{\times} \mathrm{S}_3 = \{$ ⟨(Phil, {1,3,4}), (English, {1,3,4}), (Phil, {1,2,3}), (Kansas, {1,2,3})⟩ ,
⟨(Phil, {1,3,4}), (English, {1,3,4}), (Phil, {4,5,6}), (Virginia, {4,5,6})⟩ ,
⟨(Phil, {1,3,4}), (English, {1,3,4}), (Norman, {1,2,5,6}), (Virginia, {1,2,5,6})⟩ ,
⟨(Phil, {1,3,4}), (English, {1,3,4}), (Norman, {7,8}), (Texas, {7,8})⟩ ,
⟨(Norman, {1,2}), (English, {1,2}), (Phil, {1,2,3}), (Kansas, {1,2,3})⟩ ,
⟨(Norman, {1,2}), (English, {1,2}), (Phil, {4,5,6}), (Virginia, {4,5,6})⟩ ,
⟨(Norman, {1,2}), (English, {1,2}), (Norman, {1,2,5,6}), (Virginia, {1,2,5,6})⟩ ,
⟨(Norman, {1,2}), (English, {1,2}), (Norman, {7,8}), (Texas, {7,8})⟩ ,
⟨(Norman, {5,6}), (Calculus, {5,6}), (Phil, {1,2,3}), (Kansas, {1,2,3})⟩ ,
⟨(Norman, {5,6}), (Calculus, {5,6}), (Phil, {4,5,6}), (Virginia, {4,5,6})⟩ ,
⟨(Norman, {5,6}), (Calculus, {5,6}), (Norman, {1,2,5,6}), (Virginia, {1,2,5,6})⟩ ,
⟨(Norman, {5,6}), (Calculus, {5,6}), (Norman, {7,8}), (Texas, {7,8})⟩ $\}$

Let this be relation S_4 over the relation scheme {*SName*, *Course*, *HName*, *State*}. □

2.2.4 Selection

Let R be an historical relation of m-tuples. Also, let F be a boolean function involving

- Attribute names $N_1, \ldots, N_m$;
- Constants from the domains $\mathcal{D}_1, \ldots, \mathcal{D}_m$;
- Relational operators $<$, $=$, $>$; and
- Logical operators $\wedge$, $\vee$, and $\neg$

where, to evaluate F for a tuple r, $r \in R$, we substitute the value components of the attributes of r for all occurrences of their corresponding attribute names in F. Then the historical selection of R, denoted by $\hat{\sigma}_F(R)$, is defined as

$$\hat{\sigma}_F(R) \triangleq \{r^m \mid r \in R \wedge F(value(r_1), \ \ldots, \ value(r_m))\}$$

Thus, $\hat{\sigma}$ is identical to σ in the snapshot algebra. $\hat{\sigma}_F(R)$ is simply the set of tuples in R for which F is true.

EXAMPLE.

$$
\begin{aligned}
&\hat{\sigma}_{SName=HName}(\mathrm{S}_4) = \\
&\quad \{ \ \langle(\text{Phil}, \{1,3,4\}), (\text{English}, \{1,3,4\}), (\text{Phil}, \{1,2,3\}), (\text{Kansas}, \{1,2,3\})\rangle, \\
&\qquad \langle(\text{Phil}, \{1,3,4\}), (\text{English}, \{1,3,4\}), (\text{Phil}, \{4,5,6\}), (\text{Virginia}, \{4,5,6\})\rangle, \\
&\qquad \langle(\text{Norman}, \{1,2\}), (\text{English}, \{1,2\}), (\text{Norman}, \{1,2,5,6\}), (\text{Virginia}, \{1,2,5,6\})\rangle, \\
&\qquad \langle(\text{Norman}, \{1,2\}), (\text{English}, \{1,2\}), (\text{Norman}, \{7,8\}), (\text{Texas}, \{7,8\})\rangle, \\
&\qquad \langle(\text{Norman}, \{5,6\}), (\text{Calculus}, \{5,6\}), (\text{Norman}, \{1,2,5,6\}), (\text{Virginia}, \{1,2,5,6\})\rangle, \\
&\qquad \langle(\text{Norman}, \{5,6\}), (\text{Calculus}, \{5,6\}), (\text{Norman}, \{7,8\}), (\text{Texas}, \{7,8\})\rangle \ \}
\end{aligned}
$$

Let this be relation S_5 over the relation scheme $\{$*SName*, *Course*, *HName*, *State*$\}$. □

2.2.5 Projection

Let R be an historical relation of m-tuples and let $a_1, \ldots, a_n$ be distinct integers in the range 1 to m. Then the historical projection of R, denoted by $\hat{\pi}_{N_{a_1}, \ldots, N_{a_n}}(R)$, is defined as

$$
\begin{aligned}
\hat{\pi}_{N_{a_1}, \ldots, N_{a_n}}(R) \triangleq \{u^n \mid \quad & (\forall l,\ 1 \le l \le n,\ \forall t,\ t \in valid(u_l), \\
& \qquad \exists r, (r \in R \\
& \qquad\qquad \wedge\ \forall h,\ 1 \le h \le n,\ value(u_h) = value(r_{a_h}) \\
& \qquad\qquad \wedge\ t \in valid(r_{a_l})) \\
&) \\
& \wedge\ (\forall r, (r \in R \wedge \forall l,\ 1 \le l \le n,\ value(r_{a_l}) = value(u_l)), \\
& \qquad \forall h,\ 1 \le h \le n,\ valid(r_{a_h}) \subseteq valid(u_h) \\
&) \\
& \wedge\ (\exists l,\ 1 \le l \le n \wedge valid(u_l) \ne \emptyset) \\
\} &
\end{aligned}
$$

Like the projection operator for snapshot relation, the projection operator for historical relations retains, for each tuple, only the tuple components that correspond to the attribute names in $\{N_{a_1}, \ldots, N_{a_n}\}$. All other tuple components are removed. Value-equivalent tuples in the resulting set are then combined and tuples that have an empty valid component for all tuple components are removed.

EXAMPLE. $\hat{\pi}_{SName,\ State}(\mathrm{S}_5) = \{$ $\langle$(Phil, $\{1,3,4\}$), (Kansas, $\{1,2,3\}$)$\rangle$,
$\langle$(Phil, $\{1,3,4\}$), (Virginia, $\{4,5,6\}$)$\rangle$,
$\langle$(Norman, $\{1,2,5,6\}$), (Virginia, $\{1,2,5,6\}$)$\rangle$,
$\langle$(Norman, $\{1,2,5,6\}$), (Texas, $\{7,8\}$)$\rangle$ $\}$

Let this be relation S_6 over the relation scheme *Enrollment* = $\{$*Name*, *State*$\}$. Also assume that in this relation the time-stamp associated with the value of the attribute *Name* represents the interval(s) when the specified student was enrolled and that the time-stamp associated with the value of the attribute *State* represents the interval(s) when the student was a resident of the specified state. □

The operator $\hat{\pi}$ also supports projections on expressions. For an arbitrary n, let $Evalue_l$, $1 \le l \le n$, be an arbitrary expression involving the attribute names N_a, $1 \le a \le m$. $Evalue_l$ is evaluated, for a tuple r, $r \in R$, by substituting the value components of the attributes of r for all occurrences of their corresponding attribute names in $Evalue_l$. Also, let $Evalid_l$, $1 \le l \le n$, be an arbitrary expression involving the attribute names N_a, $1 \le a \le m$, where $Evalid_l$ is evaluated for a tuple r, $r \in R$, by substituting the valid components of the attributes of r for all occurrences of their corresponding attribute names in $Evalid_l$. In addition, assume that evaluation of $Evalue_l$ for every tuple r produces an element of the domain $\mathcal{D}_b$, $1 \le b \le m$, and that evaluation of $Evalid_l$ produces an element of the domain $\wp(\mathcal{T})$. Then the definition of $\hat{\pi}$, now denoted by $\hat{\pi}_{(Evalue_1,\ Evalid_1), \ldots, (Evalue_n,\ Evalid_n)}(R)$, is constructed from the definition above simply by substituting $Evalue_h(r)$ for $value(r_{a_h})$, $Evalid_h(r)$ for $valid(r_{a_h})$, $Evalue_l(r)$ for $value(r_{a_l})$, and $Evalid_l(r)$ for $valid(r_{a_l})$. Note that this definition of the $\hat{\pi}$ operator is simply a more general version of the definition presented earlier, where N_{a_l}, $1 \leq l \leq n$, is assumed to be the ordered pair of expressions $(N_{a_l},\ N_{a_l})$.

2.2.6 Historical Derivation

The historical derivation operator δ is a new operator that does not have an analogous snapshot operator. It replaces the time-stamp of each attribute in a tuple with a new time-stamp, where the new time-stamps are computed from the existing time-stamps of the tuple's attributes. δ is effectively a combination of selection and projection on a tuple's attribute time-stamps.

Several functions, defined on the domains $\mathcal{T}$ and $\wp(\mathcal{T})$, are used either directly or indirectly in the definition of the historical derivation operator. Before defining the derivation operator itself, we describe informally these auxiliary functions. Formal definitions appear in Appendix B.

FIRST takes a set of times from the domain $\wp(\mathcal{T})$ and maps it into the earliest time in the set.

LAST takes a set of times from the domain $\wp(\mathcal{T})$ and maps it into the latest time in the set.

PRED is the predecessor function on the domain $\mathcal{T}$. It maps a time into its immediate predecessor in the linear ordering of all times.

SUCC is the successor function on the domain $\mathcal{T}$. It maps a time into its immediate successor in the linear ordering of all times.

EXTEND maps two times into the set of times that represents the interval between the first time and the second time.

INTERVAL maps a set of times into the set of intervals containing the minimum number of non-disjoint intervals represented by the input set. Each time in the input set appears in exactly one interval in the output set and each interval in the output set is itself represented by a set of times.

EXAMPLE. Consider the following tuple taken from the relation S_6 defined previously:

$$r = \langle(\text{Norman},\ \{1,2,5,6\}),\ (\text{Texas},\ \{7,8\})\rangle$$

then
$$\begin{aligned} \mathbf{INTERVAL}(valid(r(Name))) &= \big\{\{1,2\},\ \{5,6\}\big\} \\ \mathbf{INTERVAL}(valid(r(State))) &= \big\{\{7,8\}\big\} \end{aligned}$$ □

Given these auxiliary functions, we can now define the historical derivation operator on historical relations. Let R be an historical relation of m-tuples. Let V_a, $1 \leq a \leq m$, be temporal functions involving

- Attribute names $N_1, \ldots, N_m$;
- Constants from the domain $\mathcal{I}$ of non-disjoint intervals defined in Appendix B;
- Functions **FIRST**, **LAST**, and **EXTEND**; and
- Set operators $\cup$, $\cap$, and $-$;

and let G be a boolean function involving

- Temporal functions, as just described;
- Relational operators $<$, $=$, and $>$; and
- Logical operators $\wedge$, $\vee$, and $\neg$.

The functions G and V_a, $1 \leq a \leq m$, are always evaluated for a specific assignment of non-disjoint intervals to attribute names $N_1, \ldots, N_m$. G evaluates to either true or false and V_a evaluates to an element of $\wp(\mathcal{T})$. For a tuple r, $r \in R$, and intervals I_{N_c}, $1 \leq c \leq m$, $I_{N_c} \in \mathbf{INTERVAL}(valid(r_c))$, we evaluate $G(I_{N_1}, \ldots, I_{N_m})$ by substituting I_{N_c} for all occurrences of N_c in G. Likewise, we evaluate $V_a(I_{N_1}, \ldots, I_{N_m})$ by substituting I_{N_c} for all occurrences of N_c in V_a. If any one of r's attribute values has a disjoint time-stamp, there will be multiple distinct evaluations of G (and V_a) for r, one for each possible assignment of intervals to attribute names, each resulting in a value of true or false for G (and a set of time quanta for V_a).

We can now define the derivation of the historical relation R, denoted $\delta_{G, V_1, \ldots, V_m}(R)$, as

$$
\begin{aligned}
\delta_{G, V_1, \ldots, V_m}(R) \triangleq \{u^m \mid \exists r, (r \in R \\
&\wedge \forall a, 1 \leq a \leq m, \\
&\quad (value(u_a) = value(r_a) \\
&\quad \wedge (\forall t, t \in valid(u_a), \\
&\qquad \exists I_{N_1} \cdots \exists I_{N_m}, (I_{N_1} \in \mathbf{INTERVAL}(valid(r_1)) \wedge \cdots \\
&\qquad\qquad \wedge I_{N_m} \in \mathbf{INTERVAL}(valid(r_m)) \\
&\qquad\qquad \wedge G(I_{N_1}, \ldots, I_{N_m}) \\
&\qquad\qquad \wedge t \in V_a(I_{N_1}, \ldots, I_{N_m}) \\
&\qquad\qquad) \\
&\quad) \\
&\quad \wedge (\forall I_{N_1} \cdots \forall I_{N_m}, (I_{N_1} \in \mathbf{INTERVAL}(valid(r_1)) \wedge \cdots \\
&\qquad\qquad \wedge I_{N_m} \in \mathbf{INTERVAL}(valid(r_m)) \\
&\qquad\qquad \wedge G(I_{N_1}, \ldots, I_{N_m})), \\
&\qquad V_a(I_{N_1}, \ldots, I_{N_m}) \subseteq valid(u_a)
\end{aligned}
$$

$$))$$

$$\wedge\ \exists a,\ 1 \leq a \leq m \ \wedge\ valid(u_a) \neq \emptyset$$

$$)\}$$

For a tuple r, $r \in R$, the historical derivation operator determines new time-stamps for r's attributes. The historical derivation function first determines all possible assignments of intervals to attribute names for which the boolean function G is true. For each assignment of intervals to attribute names for which G is true, the operator evaluates V_a, $1 \leq a \leq m$. The sets of times resulting from the evaluations of V_a are then combined to form a new time-stamp for attribute N_a. For notational convenience, we assume that if only one V-function is provided, it applies to all attributes.

EXAMPLES.

$$\delta_{(Name \cap State)=Name,\ Name}(\mathrm{S}_6) = \{ \ \langle(\text{Phil},\ \{1\}),\ (\text{Kansas}, \{1\})\rangle\ , \ \langle(\text{Norman},\ \{1,2,5,6\}),\ (\text{Virginia},\ \{1,2,5,6\})\rangle\ ,\ \}$$

In this example, G is $(Name \cap State) = Name$ and V_1 and V_2 are both $Name$. A student tuple s, $s \in \mathrm{S}_6$, satisfies condition G if the student had at least one interval of enrollment (i.e., $I_{Name} \in \mathbf{INTERVAL}(valid(s(Name)))$) during which his home state (i.e, $State$) did not change (i.e., $(I_{Name} \cap I_{State}) = I_{Name}$, where $I_{State} \in \mathbf{INTERVAL}(valid(s(State)))$). The new time-stamp for each attribute of a tuple that satisfies G for some assignment of intervals I_{Name} and I_{State} is simply the union of the I_{Name} intervals from each assignment of intervals that satisfy G. In the first tuple in S_6, there are three intervals, two assigned to the attribute $Name$ ($\{1\}$, $\{3,4\}$) and one assigned to the attribute $State$ ($\{1,2,3\}$). From this tuple, we find that Phil was a resident of Kansas during his first interval of enrollment ($G(\{1\},\ \{1,2,3\}) = \{1\} \cap \{1,2,3\} \stackrel{\checkmark}{=} \{1\}$) but was a resident of Kansas during only part of his second interval of enrollment ($G(\{3,4\},\ \{1,2,3\}) = \{3,4\} \cap \{1,2,3\} \neq \{3,4\}$). Hence, this tuple's attributes are assigned a time-stamp of $\{1\}$ in the resulting relation. From the second tuple in S_6 we find that Phil was not a resident of Virginia during his first interval of enrollment ($G(\{1\},\ \{4,5,6\}) = \{1\} \cap \{4,5,6\} \neq \{1\}$) and lived in Virginia during only part of his second interval of enrollment ($G(\{3,4\},\ \{4,5,6\}) = \{3,4\} \cap \{4,5,6\} \neq \{3,4\}$). Hence, the time-stamp for this tuple's attributes would be assigned the empty set in the resulting relation except the definition of the historical derivation operator disallows tuples whose attributes all have an empty time-stamp. This tuple is therefore eliminated and does not appear in the resulting relation. From the third tuple in S_6 we find that Norman was a resident of Virginia during both of his intervals of enrollment ($G(\{1,2\},\ \{1,2\}) = \{1,2\} \cap \{1,2\} \stackrel{\checkmark}{=} \{1,2\}$ and $G(\{5,6\},\ \{5,6\}) = \{5,6\} \cap \{5,6\} \stackrel{\checkmark}{=} \{5,6\}$). Hence, this tuple's attributes are assigned a time-stamp of $\{1,2,5,6\}$ in the resulting relation. From the fourth tuple in S_6 we find that Norman was not a resident of Texas at any time during his enrollment ($G(\{1,2\},\ \{7,8\}) = \{1,2\} \cap \{7,8\} \neq \{1,2\}$ and $G(\{5,6\},\ \{7,8\}) = \{5,6\} \cap \{7,8\} \neq \{5,6\}$); this tuple is therefore eliminated from the resulting relation.

$$\delta_{(Name \cap State) \neq Name \,\wedge\, (Name \cap State) \neq \emptyset,\; Name \cap State}(\mathrm{S}_6) = \{\ \langle(\text{Phil}, \{3\}), (\text{Kansas}, \{3\})\rangle,\ \langle(\text{Phil}, \{4\}), (\text{Virginia}, \{4\})\rangle\ \}$$

A student tuple s, $s \in \mathrm{S}_6$, satisfies condition G if the student had at least one interval of enrollment during which his home state changed. The new time-stamp for each tuple that satisfies G for some assignment of intervals I_{Name} and I_{State} is the union of $I_{Name} \cap I_{State}$ from each assignment of intervals that satisfy G. From the first tuple in S_6 we find that Phil had one interval of enrollment during which his home state changed (i.e., $\{3,4\} \cap \{1,2,3\} \overset{\checkmark}{\neq} \{3,4\}$ and $\{3,4\} \cap \{1,2,3\} \overset{\checkmark}{\neq} \emptyset$). Hence, this tuple's attributes are assigned a time-stamp of $\{3,4\} \cap \{1,2,3\} = \{3\}$ in the resulting relation. From the second tuple in S_6 we find that Phil had one interval of enrollment during which his home state changed. Hence, this tuple's attributes are assigned a time-stamp of $\{4\}$ in the resulting relation. Note that Norman does not satisfy the restriction; his home state was the same during his two periods of enrollment. Hence, the third and fourth tuples are eliminated from the resulting relation. □

Note that the historical derivation operator actually performs two functions. First, it performs a selection function on the valid component of a tuple's attributes. For a tuple r, if G is false when an interval from the valid component of each of r's attributes is substituted for each occurrence of its corresponding attribute name in G, then the temporal information represented by that combination of intervals is not used in the calculation of the new time-stamps for r's attributes. Secondly, the derivation operator calculates a new time-stamp for attribute N_a, $1 \leq a \leq m$, from those combinations of intervals for which G is true, using V_a. If $V_1, \ldots, V_m$ are all the same function, the tuple is effectively converted from attribute time-stamping to tuple time-stamping.

The derivation operator is necessarily complex because we allow set-valued time-stamps; it would have been less complex if we had disallowed set-valued time-stamps. Then the derivation operator could have been replaced by two simpler operators, analogous to the selection and projection operators, that would have performed tuple selection and attribute projection in terms of the valid components, rather than the value components, of attributes. But, as we will see in Section 4, disallowing set-valued time-stamps would have required that the algebra support value-equivalent tuples, which would have prevented the algebra from having several other, more highly desirable properties.

2.3 Aggregates

Aggregates allow users to summarize information contained in a relation. Aggregates are categorized as either scalar aggregates or aggregate functions. Scalar aggregates return a single scalar value that is the result of applying the aggregate to a specified attribute of a snapshot relation. Aggregate functions, however, return a set of scalar values, each value the result of applying the aggregate to a specified attribute of those tuples in a snapshot relation having the same values for certain attributes. Database management systems based on the relational model typically provide several aggregate operators. For example, Ingres [Stonebraker et al. 1976] provides a **count**, **sum**,

average, **minimum**, **maximum**, and any aggregate operator. Ingres also provides two versions of the **count**, **sum**, and **average** operators, one that aggregates over all values of an attribute and one that aggregates over only the unique values of an attribute.

Several researchers have investigated aggregates in time-oriented relational databases [Ben-Zvi 1982, Jones et al. 1979, Navathe & Ahmed 1986, Snodgrass, et al. 1987, Tansel, et al. 1985]. Their work reflects the consensus that aggregates when applied to historical relations should return not a scalar value, but a distribution of scalar values over time. Jones, et al. also introduced the concepts of *instantaneous aggregates* and *cumulative aggregates*. Instantaneous aggregates return, for each time t, a value computed only from the tuples valid at time t. Cumulative aggregates return, for each time t, a value computed from all tuples valid at any time up to and including t, regardless of whether the tuples are still valid at time t. Note that a time t has meaning only when defined in terms of the time granularity. Hence, instantaneous aggregates can be viewed as aggregates over an interval whose duration is determined by the granularity of the measure of time being used. Others have generalized the definition of instantaneous and cumulative aggregates by introducing the concept of *moving aggregation windows* [Navathe & Ahmed 1986]. For an aggregation window function w from the domain $\mathcal{T}$ into the non-negative integers, an aggregate returns, for each time t, a value computed from tuples valid either at time t or at some time in the interval of length $w(t)$ immediately preceding time t. Hence, an instantaneous aggregate is an aggregate with an aggregation window function $w(t) = 0$ and a cumulative aggregate is an aggregate with an aggregation window function $w(t) = \infty$.

Klug introduced an approach to handle aggregates in the snapshot algebra [Klug 1982]. His approach makes it possible to define aggregates in a rigorous way. We use his approach to define two historical aggregate functions for our algebra:

- $\widehat{A}$, that calculates non-unique aggregates, and
- $\widehat{AU}$, that calculates unique aggregates.

These two historical aggregate functions serve as the historical counterpart of both scalar aggregates and aggregate functions.

The historical aggregate functions must contend with a variety of demands that surface as parameters (subscripts) to the functions. First, a specific aggregate (e.g., count) must be specified. Secondly, the attribute over which the aggregate is to be applied must be stated and the aggregation window function must be indicated. Finally, to accommodate partitioning, where the aggregate is applied to partitions of a relation, a set of partitioning attributes must be given. These demands complicate the definitions of $\widehat{A}$ and $\widehat{AU}$, but at the same time ensure some degree of generality to these operators.

For both definitions, let R be an historical relation of m-tuples over the relation scheme $\mathcal{N}_R = \{N_1, \ldots, N_m\}$. Also let a, c_1, ..., c_n be distinct integers in the range 1 to m and Q be an historical relation over the relation scheme $\mathcal{N}_Q$, with the restrictions that $\mathcal{N}_Q \subseteq \mathcal{N}_R$ and $\{N_a, N_{c_1}, \ldots, N_{c_n}\} \subseteq \mathcal{N}_Q$. Finally, let $X = \{N_{c_1}, \ldots, N_{c_n}\}$. If X is empty, our historical aggregate functions simply calculate a single distribution of scalar values over time for an arbitrary aggregate applied to attribute N_a of relation R. If X is not empty, our historical aggregate functions calculate, for each subtuple in Q formed from the attributes X, a distribution of scalar values over time for an arbitrary aggregate applied to attribute N_a of the subset of tuples in R whose values for attributes X match the values for attributes X of the tuple in Q. Hence, X corresponds to the by-list of an aggregate function in conventional database query languages. Assume, as does Klug, that for each aggregate operation (e.g., count) we have a family of scalar aggregates that performs the indicated aggregation on R (e.g., COUNT_{N_1}, COUNT_{N_2}, ..., COUNT_{N_m} where COUNT_{N_a}, $1 \le a \le m$, counts the (possibly duplicate) values of attribute N_a of R). We will define our historical aggregate functions in terms of these scalar aggregates.

2.3.1 Partitioning Function

Before defining the historical aggregate functions $\widehat{A}$ and $\widehat{AU}$, we define a partitioning function that will be used in their definitions.

$$
\begin{aligned}
&\mathbf{PARTITION}(R,\ q,\ t,\ w,\ N_a,\ X) \triangleq \\
&\quad \{u^m \mid (\exists r), (r \in R \ \wedge\ \forall l,\ 1 \le l \le n,\ value(r_{c_l}) = value(q_{c_l}) \\
&\qquad\qquad \wedge\ \forall d,\ 1 \le d \le m,\ value(u_d) = value(r_d) \\
&\qquad\qquad \wedge\ \forall d, 1 \le d \le m, \\
&\qquad\qquad\qquad (\ (\forall t',\ t' \in valid(u_d), \\
&\qquad\qquad\qquad\qquad \exists I_d,\ (I_d \in \mathbf{INTERVAL}(valid(r_d)) \\
&\qquad\qquad\qquad\qquad\quad \wedge\ t - w(t) < 1 \rightarrow (I_d \cap \mathbf{EXTEND}(1,\ t) \ne \emptyset) \\
&\qquad\qquad\qquad\qquad\quad \wedge\ t - w(t) \ge 1 \rightarrow (I_d \cap \mathbf{EXTEND}(t - w(t),\ t) \ne \emptyset) \\
&\qquad\qquad\qquad\qquad\quad \wedge\ t' \in I_d \\
&\qquad\qquad\qquad\qquad\quad) \\
&\qquad\qquad\qquad) \\
&\qquad\qquad\qquad \wedge\ (\forall I_d,\ (I_d \in \mathbf{INTERVAL}(valid(r_d)) \\
&\qquad\qquad\qquad\qquad \wedge\ t - w(t) < 1 \rightarrow (I_d \cap \mathbf{EXTEND}(1,\ t) \ne \emptyset) \\
&\qquad\qquad\qquad\qquad \wedge\ t - w(t) \ge 1 \rightarrow (I_d \cap \mathbf{EXTEND}(t - w(t),\ t) \ne \emptyset)) \\
&\qquad\qquad\qquad\quad I_d \subseteq valid(u_e) \\
&\qquad\qquad\qquad)) \\
&\qquad\qquad \wedge\ valid(u_a) \ne \emptyset \\
&\qquad\qquad \wedge\ \forall l,\ 1 \le l \le n,\ valid(u_{c_l}) \ne \emptyset \\
&\qquad\qquad)\}
\end{aligned}
$$

where $q \in Q$, $t \in \mathcal{T}$, w is an aggregation window function, and $1 \le a \le m$. This function retrieves from R those tuples that have the same value component for attribute N_{c_l}, $1 \le l \le n$, as q and have time t or some time in the interval of length $w(t)$ immediately preceding t in the time-stamp of attributes N_a, $N_{c_1}, \ldots$, and N_{c_n}. Note that the time-stamp of attribute N_d, $1 \leq d \leq m$, in the resulting relation is constructed from those intervals in the time-stamp of attribute N_d in R that contain time t or some time in the interval of length $w(t)$ immediately preceding t. The predicates $t - w(t) < 1 \rightarrow \cdots$ and $t - w(t) \geq 1 \rightarrow \cdots$ are used here to ensure that **PARTITION** is well-defined as **EXTEND** is defined only for elements in the domain $\mathcal{T}$.

EXAMPLES.

$$\mathbf{PARTITION}(\mathrm{S}_6,\ \langle\,\rangle,\ 5,\ 0,\ \mathit{Name},\ \emptyset) = \{\ \langle(\text{Norman},\ \{5,6\}),\ (\text{Virginia},\ \{5,6\})\rangle \\ \langle(\text{Norman},\ \{5,6\}),\ (\text{Texas},\ \emptyset)\rangle\ \}$$

Because time 5 is specified and the aggregation window function, denoted by zero, is the constant function $w(t) = 0$, tuples are selected whose time-stamp for attribute *Name* overlaps time 5. Only the third and fourth tuples in S_6 satisfy this requirement. The partitioning function here effectively returns the tuples for those students who were enrolled in school at time 5. Note that the time-stamp of each attribute in the selected tuples has been restricted to the interval from the attribute's original time-stamp overlapping time 5, if any.

$$\mathbf{PARTITION}(\mathrm{S}_6,\ \langle(\text{Phil},\ \{1,3,4\}),\ (\text{Virginia},\ \{4,5,6\})\rangle,\ 5,\ 0,\ \mathit{Name},\ \{\mathit{State}\}) = \\ \{\ \langle(\text{Norman},\ \{5,6\}),\ (\text{Virginia},\ \{5,6\})\rangle\ \}$$

where Q is here assumed to be S_6. Tuples are selected for those students who were enrolled in school and a resident of Phil's state (Virginia) at time 5. Only the third tuple in S_6 satisfies this requirement. Although Phil was a resident of Virginia at time 5, he was not enrolled in school at time 5. Hence, the second tuple in S_6 is not included in this partition.

$$\mathbf{PARTITION}(\mathrm{S}_6,\ \langle(\text{Phil},\ \{1,3,4\}),\ (\text{Virginia},\ \{4,5,6\})\rangle,\ 5,\ 1,\ \mathit{Name},\ \{\mathit{State}\}) = \\ \{\ \langle(\text{Phil},\ \{3,4\}),\ (\text{Virginia},\ \{4,5,6\})\rangle \\ \langle(\text{Norman},\ \{5,6\}),\ (\text{Virginia},\ \{5,6\})\rangle\ \}$$

Here tuples are selected for those students who were enrolled in school and a resident of Virginia within a year ($w(t) = 1$) of time 5. Both the second and third tuples in S_6 satisfy this requirement. The second tuple in S_6 is now included in the partition because Phil was a resident of Virginia and enrolled in school at time 4. □

2.3.2 Non-unique Aggregates

The historical aggregate function $\widehat{A}$ calculates, for each tuple in Q, a distribution of scalar values over time for an arbitrary aggregate applied to attribute N_a of the subset of tuples in R whose value component for attribute N_{c_l}, $1 \leq l \leq n$, matches the value component for attribute N_{c_l} of the tuple in Q. If X is empty, $\widehat{A}$ simply calculates a single distribution of scalar values over time for the aggregate applied to attribute N_a of R. If we let f represent an arbitrary family of scalar aggregates and w represent an aggregation window function, then we can define $\widehat{A}$ on the historical relations Q and R, denoted by $\widehat{A}_{f,\, w,\, N_a,\, X}(Q,\ R)$, as

$$
\begin{aligned}
&\widehat{A}_{f,\, w,\, N_a,\, X}(Q,\ R) \triangleq \\
&\hat{\bigcup}_{\forall t,\ t \in \mathcal{T}}(\hat{\pi}_{X \cup \{N_{agg}\}}(\{q \,\|\, (y,\ \{t\}) \mid q \in Q \\
&\qquad\qquad\qquad\qquad \wedge\, t - w(t) < 1 \rightarrow (valid(q_a) \cap \mathbf{EXTEND}(1,\ t) \neq \emptyset \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge\, \forall l,\ 1 \leq l \leq n, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad valid(q_{c_l}) \cap \mathbf{EXTEND}(1,\ t) \neq \emptyset) \\
&\qquad\qquad\qquad\qquad \wedge\, t - w(t) \geq 1 \rightarrow (valid(q_a) \cap \mathbf{EXTEND}(t - w(t),\ t) \neq \emptyset \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \wedge\, \forall l,\ 1 \leq l \leq n, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad valid(q_{c_l}) \cap \mathbf{EXTEND}(t - w(t),\ t) \neq \emptyset) \\
&\qquad\qquad\qquad\qquad \wedge\, y = f_{N_a}(q,\ t,\ \mathbf{PARTITION}(R,\ q,\ t,\ w,\ N_a,\ X)) \\
&\qquad\qquad\qquad\qquad \}))
\end{aligned}
$$

where "$\|$" denotes concatenation and N_{agg} is the attribute name assigned the aggregate value $(y,\ \{t\})$. If X is not empty, function $\widehat{A}$ first associates with each time t the partition of relation Q whose tuples have t, or a time in the interval of length $w(t)$ immediately preceding t, in the valid component of attributes N_a, N_{c_1}, ..., and N_{c_n}. For each of these partitions, $\widehat{A}$ then constructs a set of historical tuples. Each tuple in the set contains all the attributes X of a tuple q in the partition and a new attribute. This new attribute's valid component is the time t corresponding to the partition and its value component is the scalar value returned by the aggregate f_{N_a}, when f_{N_a} is applied to the partition of R whose tuples have value components that match q's value components for attributes X and whose valid components for attributes N_a, N_{c_1}, ..., and N_{c_n} overlap either t or the interval of length $w(t)$ immediately preceding t. Then $\widehat{A}$ performs an historical union of the resulting sets of historical tuples to produce a distribution of aggregate values over time for each tuple in Q. If X is empty, $\widehat{A}$ constructs for each time t an historical relation that is either empty or contains a single tuple. If the valid component of attribute N_a of no tuple r in R overlaps t or the interval of length $w(t)$ immediately preceding t, then the historical relation is empty. Otherwise, the historical relation contains a single tuple whose valid component is the time t and whose value component is the scalar value returned by the aggregate f_{N_a}, when f_{N_a} is applied to the partition of R whose tuples have a valid component for attribute N_a that overlaps either t or the interval of length $w(t)$ immediately preceding t. Then $\widehat{A}$ performs an historical union of the resulting sets of historical tuples to produce a single distribution of aggregate values over time.

Note that a tuple and a time are passed as parameters to the scalar aggregate f_{N_a}, along with a partition of R, in the definition of $\widehat{A}$. Although most aggregate operators can be defined in terms of a single parameter, the partition of R, the additional parameters are present because aggregates that evaluate to events or intervals, one of which is defined in Section 3.3, require them.

EXAMPLES.

$$\widehat{A}_{\text{COUNT}, 0, \mathit{State}, \emptyset}(\widehat{\pi}_{\mathit{State}}(\mathrm{S}_6),\ \mathrm{S}_6) = \{\ \langle(1,\ \{3,4,7,8\})\rangle\ ,\ \langle(2,\ \{1,2,5,6\})\rangle\ \}$$

The function $\widehat{A}$ computes the number of states in which enrolled students resided. Because $w(t) = 0$ and the time granularity of S_6 is a semester, the resulting relation represents aggregation by semester. Hence, the aggregate is in effect an instantaneous aggregate. For the interval $\{1,2\}$, there were two states (Kansas in the first tuple and Virginia in the third tuple). For the interval $\{3,4\}$, there was one state (Kansas in the first tuple at time 3 and Virginia in the second tuple at time 4). For the interval $\{5,6\}$, there also was only one state (Virginia), but it appeared in both the second and third tuples. It was counted twice because the scalar aggregates embedded within $\widehat{A}$ aggregate over duplicate values. For the interval $\{7,8\}$, there was only one state (Texas in the fourth tuple).

$$\widehat{A}_{\text{COUNT}, 1, \mathit{State}, \emptyset}(\widehat{\pi}_{\mathit{State}}(\mathrm{S}_6),\ \mathrm{S}_6) = \{\ \langle(1,\ \{8,9\})\rangle\ ,\ \langle(2,\ \{1,2,3,4,5,6\})\rangle\ ,\ \langle(3,\ \{7\})\rangle\ \}$$

Again, $\widehat{A}$ computes the number of states in which enrolled students resided, but now $w(t) = 1$. Hence, the resulting relation now represents aggregation by year (assuming two semesters per year). Although nine does not appear in the time-stamp of attribute *State* in any tuple in S_6, a count of one is recorded at time 9 because a tuple, the fourth tuple in S_6, falls into the aggregation window at time 9.

$$\widehat{A}_{\text{COUNT}, \infty, \mathit{State}, \emptyset}(\widehat{\pi}_{\mathit{State}}(\mathrm{S}_6),\ \mathrm{S}_6) = \{\ \langle(2,\ \{1,2,3\})\rangle\ ,\ \langle(3,\ \{4,5,6\})\rangle\ ,\ \langle(4,\ \{7,8,\ldots\})\rangle\ \}$$

Now, with $w(t) = \infty$, $\widehat{A}$ computes a cumulative aggregate of the number of states in which enrolled students resided.

$$\widehat{A}_{\text{COUNT}, 0, \mathit{Name}, \{\mathit{State}\}}(\mathrm{S}_6,\ \mathrm{S}_6) = \{\ \langle(\text{Kansas},\ \{1,2,3\}),\ (1,\ \{1,2,3\})\rangle\ \langle(\text{Virginia},\ \{1,2,4,5,6\}),\ (1,\ \{1,2,4\})\rangle\ \langle(\text{Virginia},\ \{1,2,4,5,6\}),\ (2,\ \{5,6\})\rangle\ \langle(\text{Texas},\ \{7,8\}),\ (1,\ \{7,8\})\rangle\ \}$$

Here, $\widehat{A}$ computes the instantaneous aggregate of the number of enrolled students who resided in each state. In effect, the aggregate is computed for each subset of tuples in S_6 having the same value for the attribute *State*. For example, the first tuple is computed by selecting all the tuples in S_6 with a state of Kansas and then performing the aggregate on this (smaller) set. □

2.3.3 Unique Aggregates

The function $\widehat{A}$ allows its embedded scalar aggregates to aggregate over duplicate attribute values. We now define an historical aggregate function $\widehat{AU}$, identical to $\widehat{A}$ with one exception; it restricts its embedded scalar aggregates to aggregation over unique attribute values. We define $\widehat{AU}$ on the historical relations Q and R, denoted by $\widehat{AU}_{f,\, w,\, N_a,\, X}(Q,\ R)$, as

$$
\begin{aligned}
&\widehat{AU}_{f,\, w,\, N_a,\, X}(Q,\ R) \triangleq \\
&\hat{\cup}_{\forall t,\, t \in \mathcal{T}}(\hat{\pi}_{X \cup \{N_{agg}\}}(\{q \parallel (y,\ \{t\}) \mid q \in Q \\
&\qquad \wedge\, t - w(t) < 1 \rightarrow (valid(q_a) \cap \mathbf{EXTEND}(1,\ t) \neq \emptyset \\
&\qquad\qquad \wedge\, \forall l,\ 1 \leq l \leq n, \\
&\qquad\qquad\quad valid(q_{c_l}) \cap \mathbf{EXTEND}(1,\ t) \neq \emptyset) \\
&\qquad \wedge\, t - w(t) \geq 1 \rightarrow (valid(q_a) \cap \mathbf{EXTEND}(t - w(t),\ t) \neq \emptyset \\
&\qquad\qquad \wedge\, \forall l,\ 1 \leq l \leq n, \\
&\qquad\qquad\quad valid(q_{c_l}) \cap \mathbf{EXTEND}(t - w(t),\ t) \neq \emptyset) \\
&\qquad \wedge\, y = f_{N_a}(q,\ t,\ \delta_{\text{true},\, t}(\hat{\pi}_{N_a}(\mathbf{PARTITION}(R,\ q,\ t,\ w,\ N_a,\ X)))) \\
&\qquad \}))
\end{aligned}
$$

This definition differs from that of $\widehat{A}$ only in that the historical projection on attribute N_a of **PARTITION(...)** followed by the historical derivation eliminates duplicate values of the aggregated attribute before the scalar aggregation is preformed.

EXAMPLE. $\widehat{AU}_{\text{COUNT},\, 0,\, State,\, \emptyset}(\hat{\pi}_{State}(\text{S}_6),\ \text{S}_6) = \{\ \langle(1,\ \{3,4,5,6,7,8\})\rangle\ ,\ \langle(2,\ \{1,2\})\rangle\ \}$

This relation differs from the non-unique variant only during the interval $\{5,6\}$. Here, Virginia is correctly counted only once, even though there are two tuples valid during this interval with a state of Virginia. □

2.3.4 Expressions in Aggregates

The functions $\widehat{A}$ and $\widehat{AU}$ allow expressions to be aggregated and support aggregation by arbitrary expressions. Let *Eaggregate* be an arbitrary expression involving u historical aggregate functions. Also, assume that the v^{th} historical aggregate function applies the scalar aggregate f_v to attribute N_{a_v} where the aggregation window function is w_v, and the partitioning attributes are X_v. Then the definition of $\widehat{A}$, now denoted by

$$\widehat{A}_{f_1,\, ...,\, f_u,\, w_1,\, ...,\, w_u,\, N_{a_1},\, ...,\, N_{a_u},\, X_1,\, ...,\, X_u,\, Eaggregate}(Q,\ R),$$

is constructed from the definition of $\widehat{A}$ above simply by substituting $y = Eaggregate'$ for $y = f_{N_a}(\ldots)$. $Eaggregate'$ is $Eaggregate$ where each reference to the v^{th} aggregate has been replaced by the expression $f_{v\,N_{a_v}}(q,\ t,\ \textbf{PARTITION}(R,\ q,\ t,\ w_v,\ N_{a_v},\ X_v))$. With these changes, $\widehat{A}$ allows expressions to be aggregated. $\widehat{AU}$ can be modified similarly.

If $\widehat{A}$ and $\widehat{AU}$ are to support aggregation by arbitrary expressions, changes must be made to the definitions of **PARTITION**, $\widehat{A}$, and $\widehat{AU}$ given above. First, let $Evalue_l$, $1 \leq l \leq o$, be an expression involving the attribute names $N_{c_1}, \ldots, N_{c_n}$. $Evalue_l$ is evaluated for a tuple r, $r \in R$, by substituting the value components of the attributes of r for all occurrences of their corresponding attribute names in $Evalue_l$. Secondly, let $X = \{Evalue_1, \ldots, Evalue_o\}$ and $d_1, \ldots, d_p$ be the distinct integers in the range 1 to m such that N_{d_h}, $1 \leq h \leq p$, appears in at least one $Evalue_l, 1 \leq l \leq o$. Then new definitions of **PARTITION**, $\widehat{A}$, and $\widehat{AU}$ are constructed from the definitions above simply by substituting the predicate $\forall l,\ 1 \leq l \leq o,\ Evalue_l(r) = Evalue_l(q)$ for the predicate $\forall l,\ 1 \leq l \leq n,\ value(r_{c_l}) = value(q_{c_l})$ and the predicate $\forall l,\ 1 \leq l \leq p,\ valid(u_{d_l}) \neq \emptyset$ for the predicate $\forall l,\ 1 \leq l \leq n,\ valid(u_{c_l}) \neq \emptyset$ in the definition of **PARTITION** and substituting p for n and $valid(q_{d_l})$ for $valid(q_{c_l})$ in the definitions of $\widehat{A}$ and $\widehat{AU}$. With these changes, $\widehat{A}$ and $\widehat{AU}$ support aggregation by arbitrary expressions.

2.4 Preservation of the Value-equivalence Property

Theorem 1 *The operators $\widehat{\cup}$, $\widehat{-}$, $\widehat{\times}$, $\widehat{\sigma}$, $\widehat{\pi}$, δ, $\widehat{A}$, and $\widehat{AU}$ all preserve the value-equivalence property of historical relations.*

PROOF. For the operators $\widehat{\cup}$, $\widehat{-}$, $\widehat{\times}$, $\widehat{\sigma}$, and δ we show that the contrapositive of the theorem holds, that is, if there are value-equivalent tuples in an operator's output relation, then there are value-equivalent tuples in at least one of its input relations. For the operators $\widehat{\pi}$, $\widehat{A}$, and $\widehat{AU}$, we show by contradiction that there cannot be value-equivalent tuples in their output relations.

Case 1. $\widehat{\cup}$. Assume that $Q \,\widehat{\cup}\, R$ contains at least two value-equivalent tuples. From the definition of $\widehat{\cup}$, each tuple in $Q \,\widehat{\cup}\, R$ has a value-equivalent tuple in Q, R, or both. If two value-equivalent tuples $\hat{u}_1$ and $\hat{u}_2$ in $Q \,\widehat{\cup}\, R$ do not have a value-equivalent tuple in R, then both are tuples in Q. Similarly, if they do not have a value-equivalent tuple in Q, then both are tuples in R. If they have a value-equivalent tuple in both Q and R, then each was constructed from a value-equivalent tuple in Q and a value-equivalent tuple in R. If both $\hat{u}_1$ and $\hat{u}_2$ had been constructed from the same tuple in Q and the same tuple in R, then $\hat{u}_1$ and $\hat{u}_2$ would be, by definition, the same tuple. Hence, they were constructed from different value-equivalent tuples in Q, R, or both.

Case 2. $\widehat{-}$. Assume that $Q \,\widehat{-}\, R$ contains at least two value-equivalent tuples. From the definition of $\widehat{-}$, each tuple in $Q \,\widehat{-}\, R$ has a value-equivalent tuple in Q but not in R or a value-equivalent tuple in both Q and R. If two value-equivalent tuples $\hat{u}_1$ and $\hat{u}_2$ in $Q \,\widehat{-}\, R$ do not have a value-equivalent tuple in R, then both are tuples in Q. If they have a value-equivalent tuple in both Q and R, then each was constructed from a value-equivalent tuple in Q and a value-equivalent tuple in R. If both $\hat{u}_1$ and $\hat{u}_2$ had been constructed from the same tuple in Q and the same tuple in R, then $\hat{u}_1$ and $\hat{u}_2$ would be, by definition, the same tuple. Hence, they were constructed from different value-equivalent tuples in Q, R, or both.

Case 3. $\hat{\times}$. Assume that $Q \hat{\times} R$ contains at least two value-equivalent tuples. From the definition of $\hat{\times}$, each tuple in $Q \hat{\times} R$ is constructed from a tuple in Q and a tuple in R. If two value-equivalent tuples $\hat{u}_1$ and $\hat{u}_2$ in $Q \hat{\times} R$ had been constructed from the same tuple in Q and the same tuple in R, then $\hat{u}_1$ and $\hat{u}_2$ would be, by definition, the same tuple. Hence, they were constructed from different value-equivalent tuples in Q, R, or both.

Case 4. $\hat{\sigma}$. Assume that $\hat{\sigma}_F(R)$ contains at least two value-equivalent tuples. From the definition of $\hat{\sigma}$, each tuple in $\hat{\sigma}_F(R)$ is a tuple in R. Hence, any two value-equivalent tuples in $\hat{\sigma}_F(R)$ are also tuples in R.

Case 5. $\hat{\pi}$. Assume that $\hat{\pi}_{N_{a_1}, \ldots, N_{a_n}}(R)$ contains at least two value-equivalent tuples. For any two such tuples there will be at least one time that appears in the time-stamp of an attribute of one tuple but not the other tuple; otherwise, they would be identical. Hence, let $\hat{u}_1$ and $\hat{u}_2$ be two value-equivalent tuples in $\hat{\pi}_{N_{a_1}, \ldots, N_{a_n}}(R)$ such that there is a time t in the time-stamp of attribute N_{a_l}, $1 \leq l \leq n$, of $\hat{u}_1$ but not $\hat{u}_2$. From the first clause of the definition of $\hat{\pi}$, there is a tuple r, $r \in R$, that has t in the time-stamp of attribute N_{a_l} and the same value for attributes $N_{a_1}, \ldots, N_{a_n}$ as $\hat{u}_1$. But, from the second clause of the definition, the time-stamp of attribute N_{a_l} of tuple r is a subset of the time-stamp of attribute N_{a_l} of $\hat{u}_2$, as r also has the same value for attributes $N_{a_1}, \ldots, N_{a_n}$ as $\hat{u}_2$. Hence, t is in the time-stamp of attribute N_{a_l} of $\hat{u}_2$, contradicting the assumption that t is in the time-stamp of attribute N_{a_l} of $\hat{u}_1$ but not $\hat{u}_2$. Similarly, we arrive at a contradiction if we assume that there is a time t in the time-stamp of attribute N_{a_l}, $1 \leq l \leq n$, of $\hat{u}_2$ but not $\hat{u}_1$. Hence, $\hat{u}_1$ and $\hat{u}_2$ have identical attribute time-stamps, which implies that they are the same tuple, contradicting the assumption that $\hat{\pi}_{N_{a_1}, \ldots, N_{a_n}}(R)$ contains at least two value-equivalent tuples. Note that the output relation of $\hat{\pi}$, unlike the output relations of $\hat{\cup}$, $\hat{-}$, $\hat{\times}$, and $\hat{\sigma}$, would not contain value-equivalent tuples even if there were value-equivalent tuples in its input relation.

Case 6. $\hat{\delta}$. Assume that $\hat{\delta}_{G, V_1, \ldots, V_m}(R)$ contains at least two value-equivalent tuples, $\hat{u}_1$ and $\hat{u}_2$. From the definition of $\hat{\delta}$, each tuple in $\hat{\delta}_{G, V_1, \ldots, V_m}(R)$ is constructed from one value-equivalent tuple in R. If $\hat{u}_1$ and $\hat{u}_2$ were constructed from the same value-equivalent tuple r, $r \in R$, then they would be the same tuple, as $\hat{\delta}$ requires not only that every time t in the time-stamp of attribute N_a, $1 \leq a \leq m$, of either $\hat{u}_1$ or $\hat{u}_2$ be in $V_a(\ldots)$ and satisfy $G(\ldots)$ for some assignment of intervals from the time-stamps of r's attributes to attribute names but that $V_a(\ldots)$ be a subset of the time-stamp of attribute N_a of both $\hat{u}_1$ and $\hat{u}_2$. Hence, $\hat{u}_1$ and $\hat{u}_2$ were constructed from different value-equivalent tuples in R.

Case 7. $\widehat{A}$. Assume that $\widehat{A}_{f, w, N_a, X}(Q, R)$ contains at least two value-equivalent tuples. From Case 1 above, if $\widehat{A}_{f, w, N_a, X}(Q, R)$ contains value-equivalent tuples, then the input relation to $\widehat{A}$'s outermost $\hat{\cup}$ operator contains value-equivalent tuples. But, this relation is the output of $\hat{\pi}$, whose output relation was shown in Case 5 above never to contain value-equivalent tuples. Hence, our assumption that $\widehat{A}_{f, w, N_a, X}(Q, R)$ contains at least two value-equivalent tuples is contradicted.

Case 8. $\widehat{AU}$. Simply replace $\widehat{A}$ with $\widehat{AU}$ in Case 7. ∎

2.5 Summary

We first introduced ***historical relations***, in which attribute values are associated with set-valued time-stamps. We then defined eight historical operators:

- Five operators are analogous to the five standard snapshot operators: union ($\hat{\cup}$), difference ($\hat{-}$), cartesian product ($\hat{\times}$), selection ($\hat{\sigma}$), and projection ($\hat{\pi}$).
- Historical derivation ($\hat{\delta}$) effectively performs selection and projection on the valid-time dimension by replacing the time-stamp of each attribute of selected tuples with a new time-stamp.
- Aggregation ($\hat{A}$) and unique aggregation ($\widehat{AU}$) serve to compute a distribution of single values over time for a collection of tuples.

We should mention several other operators that can exist harmoniously with these eight operators. Intersection ($\hat{\cap}$), quotient ($\hat{\div}$), natural join ($\hat{\bowtie}$), and Θ-join ($\hat{\bowtie}_{\Theta}$) can all be defined in terms of the five basic operators, in an identical fashion to the definition of their snapshot counterparts. Finally, the historical rollback operator ($\hat{\rho}$), defined elsewhere [McKenzie & Snodgrass 1987A], serves to generalize the algebra to handle temporal relations incorporating both valid and transaction time.

3 Equivalence with TQuel

We now show that the historical algebra defined above has the expressive power of the *TQuel* (*T*emporal *QUE*ry *L*anguage) [Snodgrass 1987] facilities that support valid time. TQuel is a version of Quel [Held et al. 1975], the calculus-based query language for the Ingres relational database management system [Stonebraker et al. 1976], augmented to handle both valid time and transaction time. Two new syntactic and semantic constructs are provided to support valid time. The ***valid clause*** is the temporal analogue to Quel's target list; it is used to specify the value of the valid time for tuples in the derived relation. This clause consists of the keywords `valid from to` and two temporal expressions, each consisting of tuple variables, temporal constants, and the temporal constructors `begin of`, `end of`, `overlap`, and `extend`. The ***when clause*** is the temporal analogue to Quel's where clause. This clause consists of the keyword `when` followed by a temporal predicate consisting of temporal expressions, the temporal predicate operators `precede`, `overlap`, and `equal`, and the logical operators `or`, `and`, and `not`. (Note that `overlap` is overloaded; it may be either a temporal constructor or a temporal predicate operator, with context differentiating the uses.) A third new construct, the ***as-of*** clause, is provided to handle transaction time but will not be considered here. We will generally limit our discussion of TQuel to its facilities for handling valid time.

Unlike our historical algebra, which assumes attribute time-stamping, TQuel assumes tuple time-stamping. The formal semantics of TQuel conceptually embeds its temporal relations in snapshot relations; such an embedding is done purely for convenience in developing the semantics. TQuel represents valid time by adding two time values to each tuple to specify the time when the tuple became valid (i.e., *From*) and the time when the tuple became invalid (i.e., *To*). Also unlike our historical algebra, TQuel allows value-equivalent tuples in a relation but assumes that value-equivalent tuples are *coalesced* (i.e., tuples with identical values for the explicit attributes neither overlap nor are adjacent in time). As we will see shortly, it is possible to convert the embedded, coalesced snapshot relations used in TQuel's formal semantics to historical relations.

3.1 TQuel Retrieve Statement

Assume that we are given the k snapshot relations $R'_1, \ldots, R'_k$ whose schemes are respectively,

$$\mathcal{N}_1 = \{N_{1,1}, \ldots, N_{1,m_1}, \mathit{From}_1, \mathit{To}_1\}$$

$$\ldots$$

$$\mathcal{N}_k = \{N_{k,1}, \ldots, N_{k,m_k}, \mathit{From}_k, \mathit{To}_k\}$$

For notational convenience, we associate " $'$ " with TQuel relations, tuple variables, and expressions to differentiate them from their counterparts in the historical algebra and assume that $N_{1,1}, \ldots, N_{k,m_k}$ are unique. Furthermore, let $i_1, i_2, \ldots, i_n$ be integers, not necessarily distinct, in the range 1 to k and a_l, $1 \le l \le n$, be a distinct integer in the range 1 to m_{i_l}. Then, the TQuel retrieve statement has the following syntax

$$\begin{array}{ll}
\texttt{range of } r'_1 \texttt{ is } R'_1 & \\
\ldots & \\
\texttt{range of } r'_k \texttt{ is } R'_k & \\
\texttt{retrieve into } R'_{k+1}(N_{k+1,1} = r'_{i_1}.N_{i_1,a_1}, \ldots, N_{k+1,n} = r'_{i_n}.N_{i_n,a_n}) & (1) \\
\quad \texttt{valid from } \upsilon \texttt{ to } \chi & \\
\quad \texttt{where } \psi & \\
\quad \texttt{when } \tau &
\end{array}$$

This statement computes a new relation R'_{k+1} over the relational scheme

$$\mathcal{N}_{k+1} = \{N_{k+1,1}, \ldots, N_{k+1,n}, \mathit{From}_{k+1}, \mathit{To}_{k+1}\}$$

Its tuple calculus statement has the following form

$$
\begin{aligned}
R'_{k+1} = \{u^{n+2} \mid (\exists r'_1)\cdots(\exists r'_k) & \\
& (r'_1 \in R'_1 \wedge \cdots \wedge r'_k \in R'_k \\
& \wedge\, u(N_{k+1,1}) = r'_{i_1}(N_{i_1,a_1}) \wedge \cdots \wedge u(N_{k+1,n}) = r'_{i_n}(N_{i_n,a_n}) \\
& \wedge\, u(From_{k+1}) = \Phi'_\upsilon((r'_1(From_1),\ r'_1(To_1)), \ldots, (r'_k(From_k),\ r'_k(To_k))) \\
& \wedge\, u(To_{k+1}) = \Phi'_\chi((r'_1(From_1),\ r'_1(To_1)), \ldots, (r'_k(From_k),\ r'_k(To_k))) \qquad (2) \\
& \wedge\, Before(u(From_{k+1}),\ u(To_{k+1})) \\
& \wedge\, \Psi'_\psi(r'_1(N_{1,1}), \ldots, r'_k(N_{k,m_k})) \\
& \wedge\, \Gamma'_\tau((r'_1(From_1),\ r'_1(To_1)), \ldots, (r'_k(From_k),\ r'_k(To_k))) \\
&)\}
\end{aligned}
$$

where *Before* is the "<" predicate on integers, the ordered pair $(r'_i(From_i),\ r'_i(To_i))$, $1 \leq i \leq k$, represents the interval $[r'_i(From_i),\ r'_i(To_i))$, and Ψ'_ψ, Φ'_υ, Φ'_χ, and Γ'_τ are the denotations described below of ψ, υ, χ, and τ respectively.

Ψ'_ψ is obtained by replacing each occurrence of an attribute reference $r'_i.N_{i,a}$, $1 \leq i \leq k$, $1 \leq a \leq m_i$, in ψ with $r'_i(N_{i,a})$ and each occurrence of a logical operator with its corresponding logical predicate. That is,

$r'_i.N_{i,a} \rightarrow r'_i(N_{i,a})$,

`and` $\rightarrow \wedge$,

`or` $\rightarrow \vee$, and

`not` $\rightarrow \neg$.

Φ'_υ and Φ'_χ are obtained by replacing each occurrence of a tuple variable r'_i in υ and χ with the ordered pair $(r'_i(From_i),\ r'_i(To_i))$ and each occurrence of a temporal constructor with a corresponding function. That is,

$r'_i \rightarrow (r'_i(From_i),\ r'_i(To_i))$

`begin of` $I \rightarrow beginof(I)$,

`end of` $I \rightarrow endof(I)$,

I_1 `overlap` $I_2 \rightarrow overlap(I_1,\ I_2)$, and

I_1 `extend` $I_2 \rightarrow extend(I_1,\ I_2)$

where *beginof*, *endof*, *overlap*, and *extend* are functions on the domain I. Formal definitions for these functions are presented elsewhere [Snodgrass 1987].

Γ'_τ is obtained by replacing each occurrence of a logical operator in τ with its corresponding logical predicate according to the rules given for its replacement in ψ, replacing each occurrence of a tuple variable or temporal constructor according to the rules given for their replacement in υ and χ, and replacing each occurrence of a temporal predicate operator with an analogous predicate on intervals. That is,

$$I_1 \texttt{ precede } I_2 \rightarrow \textit{precede}(I_1,\ I_2),$$

$$I_1 \texttt{ overlap } I_2 \rightarrow \textit{overlap}(I_1,\ I_2), \text{ and}$$

$$I_1 \texttt{ equal } I_2 \rightarrow \textit{equal}(I_1,\ I_2)$$

where *precede*, *overlap*, and *equal* are predicates on the domain $\mathcal{I}$. Formal definitions for these predicates are presented elsewhere [Snodgrass 1987].

3.2 Correspondence with the Historical Algebra

To compare the expressive power of TQuel and the historical algebra presented in Section 2, we first relate relations in the two systems, then expressions in the new TQuel clauses, and finally the retrieve statement with algebraic expressions.

Definition 1 *The transformation function* $\mathbf{T}$ *maps a TQuel embedded snapshot relation over the scheme* $\{N_1,\ \ldots,\ N_m,\ \textit{From},\ \textit{To}\}$ *into its equivalent historical relation, valid in our historical algebra over the scheme* $\{N_1,\ \ldots,\ N_m\}$.

$$\begin{aligned}
\mathbf{T}(R') \triangleq \{u^m \mid\ & (\forall a, 1 \le a \le m,\ \forall t,\ t \in \textit{valid}(u(N_a)),\\
&\quad \exists r',\ (r' \in R'\\
&\quad\quad \wedge \forall c,\ 1 \le c \le m,\ \textit{value}(u(N_c)) = r'(N_c)\\
&\quad\quad \wedge\, t \in \mathbf{EXTEND}(r'(\textit{From}),\ \mathbf{SUCC}(r'(\textit{To})))\\
&\quad\quad)\\
&)\\
&\wedge (\forall r',\ (r' \in R' \ \wedge\ \forall a,\ 1 \le a \le m,\ r'(N_a) = \textit{value}(u(N_a))),\\
&\quad \forall c,\ 1 \le c \le m,\ \mathbf{EXTEND}(r'(\textit{From}),\ \mathbf{SUCC}(r'(\textit{To}))) \subseteq \textit{valid}(u(N_c)))\\
&)\}
\end{aligned}$$

The first clause of this definition ensures that each tuple in $\mathbf{T}(R')$ has at least one value-equivalent tuple in R. The second clause in the definition ensures that each subset of value-equivalent tuples in R is represented by a single tuple in $\mathbf{T}(R')$. Note also that the same time-stamp is assigned to each attribute of a tuple in $\mathbf{T}(R')$. This time-stamp is simply the union of the time-stamps of the tuple's value-equivalent tuples in R'. Because TQuel assumes that value-equivalent tuples are coalesced, the time-stamp of each tuple in R' is a distinguishable interval of time in the attribute time-stamps of its value-equivalent counterpart in $\mathbf{T}(R')$, as shown by the following lemma.

Lemma 1 $\forall r,\ r \in \mathbf{T}(R'),\ \forall a,\ 1 \le a \le m,\ \forall I,\ I \in \mathbf{INTERVAL}(valid(r(N_a))),$

$$
\begin{aligned}
&\exists r',\ (r' \in R' \\
&\quad \wedge \forall c,\ 1 \le c \le m,\ value(r(N_c)) = r'(N_c) \\
&\quad \wedge I = \mathbf{EXTEND}(r'(From),\ \mathbf{SUCC}(r'(To))) \\
&\quad)
\end{aligned}
$$

PROOF. Apply the definitions of coalescing and **INTERVAL** to **T** and simplify. ∎

Definition 2 *We define a m+2-tuple TQuel relation R' and a m-tuple relation R in our historical algebra to be* equivalent *if, and only if, $R = \mathbf{T}(R')$. In addition, we define a TQuel query and an expression in our historical algebra to be* equivalent *if, and only if, they evaluate to equivalent relations.*

Let Ψ_ψ, Φ_υ, and Φ_χ be the denotations in our algebra of ψ, υ, and χ respectively. Ψ_ψ is obtained by replacing each occurrence of $r_i'(N_{i,a})$, $1 \le i \le k$, $1 \le a \le m_i$, in Ψ_ψ' with $N_{i,a}$. Φ_υ and Φ_χ are obtained by replacing each occurrence of an ordered pair $(r_i'(From_i),\ r_i'(To_i))$, $1 \le i \le k$, in Φ_υ' and Φ_χ' with $N_{i,1}$ and each occurrence of a TQuel function with its algebraic equivalent. That is,

$$
\begin{aligned}
&(r_i'(From_i),\ r_i'(To_i)) \rightarrow N_{i,1}, \\
&beginof(I) \rightarrow \mathbf{FIRST}(I), \\
&endof(I) \rightarrow \mathbf{LAST}(I), \\
&overlap(I_1,\ I_2) \rightarrow I_1 \cap I_2, \text{ and} \\
&extend(I_1,\ I_2) \rightarrow \mathbf{EXTEND}(\mathbf{FIRST}(I_1), \mathbf{LAST}(I_2)).
\end{aligned}
$$

Also let Γ_τ be the denotation in our algebra of τ. Γ_τ is obtained by replacing each occurrence of an ordered pair $(r_i'(From_i),\ r_i'(To_i))$ and each occurrence of a TQuel function in Γ_τ' with its algebraic equivalent according to the rules above and each occurrence of the predicates ***precede***, ***overlay***, and ***equal*** with its algebraic equivalent. That is,

$$
\begin{aligned}
&precede(I_1,\ I_2) \rightarrow \mathbf{LAST}(I_1) < \mathbf{FIRST}(I_2) \vee \mathbf{LAST}(I_1) = \mathbf{FIRST}(I_2), \\
&overlap(I_1,\ I_2) \rightarrow I_1 \cap I_2 \ne \emptyset, \text{ and} \\
&equal(I_1,\ I_2) \rightarrow I_1 = I_2.
\end{aligned}
$$

Note from the definition of $\mathbf{T}(R')$ that a tuple in $\mathbf{T}(R')$ has the same time-stamp for each of its attributes. Hence, although we require that each occurrences of an ordered pair $(r_i'(From_i),\ r_i'(To_i))$ in Φ_υ', Φ_χ', and Γ_τ' be replaced with the same attribute name (i.e., $N_{i,1}$), we could have specified any attribute of relation R_i.

We will need the following two lemmas in the equivalence proof to be presented shortly.

Lemma 2 *Φ_v, Φ_χ, and Γ_τ are semantically equivalent to Φ'_v, Φ'_χ, and Γ'_τ respectively. That is, the result of evaluating Φ'_v, Φ'_χ, and Γ'_τ for tuples r'_i, $r'_i \in R'_i$, $1 \leq i \leq k$, is the same as the result of evaluating Φ_v, Φ_χ, and Γ_τ for the intervals I_i, $I_i = \mathbf{EXTEND}(r'_i(From_i),\ \mathbf{SUCC}(r'_i(To_i)))$ substituted for the attribute name $N_{i,1}$.*

PROOF. The semantic equivalence follows directly from the definitions of the functions used in Φ'_v, Φ'_χ, and Γ'_τ [Snodgrass 1987]. ▮

Lemma 3 $t \in \mathbf{EXTEND}(\Phi'_v(\ldots),\ \mathbf{SUCC}(\Phi'_\chi(\ldots))) \rightarrow Before(\Phi'_v(\ldots),\ \Phi'_\chi(\ldots))$.

PROOF. It follows directly from the definition of **EXTEND**, given in Appendix B, that $t \in \mathbf{EXTEND}(\Phi'_v(\ldots),\ \mathbf{SUCC}(\Phi'_\chi(\ldots)))$ implies $\Phi'_v(\ldots) \leq t < \Phi'_\chi(\ldots))$, which in turn implies $Before(\Phi'_v(\ldots),\ \Phi'_\chi(\ldots))$ ▮

Having defined the algebraic equivalents of TQuel relations and expressions in the new TQuel clauses, we can now define the algebraic equivalent of a TQuel retrieve statement. Every Quel retrieve statement (a target list and where clause) is equivalent to an algebraic expression that represents cartesian product of the relations associated with tuple variables, followed by selection by the where-clause predicate, and then projection on the attributes in the target list. Similarly, every TQuel retrieve statement is equivalent to an algebraic expression that represents cartesian product of the referenced relations, followed by selection by the where-clause predicate, historical derivation as specified by the when and valid clauses, and then projection on the attributes in the target list.

Theorem 2 *Every TQuel retrieve statement of the form of* (1) *found on page 22 is equivalent to an expression in our historical algebra of the form*

$$R = \hat{\pi}_{N_{i_1,a_1},\ldots,N_{i_n,a_n}}(\delta_{\Gamma_\tau,\,\mathbf{EXTEND}(\Phi_v,\mathbf{SUCC}(\Phi_\chi))}(\hat{\sigma}_{\Psi_\psi}(\mathbf{T}(R'_1)\hat{\times}\ldots\hat{\times}\mathbf{T}(R'_k)))). \quad (3)$$

PROOF. To prove that R and R'_{k+1} are temporally equivalent, we must show that $R = \mathbf{T}(R'_{k+1})$. From set theory and the definition of $\mathbf{T}$, it follows that R and $\mathbf{T}(R'_{k+1})$ are equal if, and only if, the following holds.

$$\begin{aligned}
&(\forall r,\ r \in R,\ \forall a, 1 \leq a \leq n,\ \forall t,\ t \in valid(r(N_a)),\\
&\qquad \exists r'_{k+1}, (r'_{k+1} \in R'_{k+1}\\
&\qquad\qquad \wedge \forall c,\ 1 \leq c \leq n,\ value(r(N_c)) = r'_{k+1}(N_{k+1,c})\\
&\qquad\qquad \wedge t \in \mathbf{EXTEND}(r'_{k+1}(From_{k+1}),\ \mathbf{SUCC}(r'_{k+1}(To_{k+1})))\\
&\qquad\qquad) \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (4)\\
&)\\
&\wedge (\forall r,\ r \in R,\ \forall r'_{k+1}, (r'_{k+1} \in R'_{k+1}\ \wedge\ \forall a,\ 1 \leq a \leq n,\ r'_{k+1}(N_{k+1,a}) = value(r(N_a))),
\end{aligned}$$

$$\forall c,\ 1 \le c \le n,$$
$$\mathbf{EXTEND}(r'_{k+1}(\mathit{From}_{k+1}),\ \mathbf{SUCC}(r'_{k+1}(\mathit{To}_{k+1}))) \subseteq \mathit{valid}(r(N_c))$$
$$)$$

To prove the validity of (4), we show that the tuple calculus for R reduces to (4). First, construct the tuple calculus statement for R from the definitions of the historical operators $\hat{\times}$, $\hat{\sigma}$, δ, and $\hat{\pi}$, using straightforward substitution, change of variable, and simplification (i.e., the definition of $\mathbf{T}(R'_1)\hat{\times}\ldots\hat{\times}\mathbf{T}(R'_k)$ obtained from the $\hat{\times}$ operator is substituted for references to the historical relation in the definition of $\hat{\sigma}$, etc.).

$$\hat{\pi}_{N_{i_1,a_1},\ldots,N_{i_n,a_n}}(\delta_{\Gamma_\tau,\mathbf{EXTEND}(\Phi_v,\mathbf{SUCC}(\Phi_\chi))}(\hat{\sigma}_{\Psi_\psi}(\mathbf{T}(R'_1)\hat{\times}\ldots\hat{\times}\mathbf{T}(R'_k)))) \triangleq$$

$$\begin{array}{l}
\{r^n \mid \quad (\forall c,\ 1 \le c \le n,\ \forall t,\ t \in \mathit{valid}(r(N_c)),\\
\qquad\qquad (\exists r_1)\cdots(\exists r_k)(\exists I_1)\cdots(\exists I_k),\\
\qquad\qquad\quad (r_1 \in \mathbf{T}(R'_1) \wedge \cdots \wedge r_k \in \mathbf{T}(R'_k)\\
\qquad\qquad\quad \wedge\ I_1 \in \mathbf{INTERVAL}(\mathit{valid}(r_1(N_{1,1}))) \wedge \cdots\\
\qquad\qquad\qquad\qquad\qquad\qquad \wedge\ I_k \in \mathbf{INTERVAL}(\mathit{valid}(r_k(N_{k,1})))\\
\qquad\qquad\quad \wedge\ \forall l,\ 1 \le l \le n,\ \mathit{value}(r(N_l)) = \mathit{value}(r_{i_l}(N_{i_l,a_l}))\\
\qquad\qquad\quad \wedge\ \Psi_\psi(\mathit{value}(r_1(N_{1,1})),\ \ldots,\ \mathit{value}(r_k(N_{k,m_k})))\\
\qquad\qquad\quad \wedge\ \Gamma_\tau(I_1,\ \ldots,\ I_k)\\
\qquad\qquad\quad \wedge\ t \in \mathbf{EXTEND}(\Phi_v(I_1,\ \ldots,\ I_k),\ \mathbf{SUCC}(\Phi_\chi(I_1,\ \ldots,\ I_k)))\\
\qquad\qquad\quad))\\
\qquad \wedge\ ((\forall r_1)\cdots(\forall r_k)(\forall I_1)\cdots(\forall I_k)\\
\qquad\qquad (r_1 \in \mathbf{T}(R'_1) \wedge \cdots \wedge r_k \in \mathbf{T}(R'_k)\\
\qquad\qquad \wedge\ I_1 \in \mathbf{INTERVAL}(\mathit{valid}(r_1(N_{1,1}))) \wedge \cdots\\
\qquad\qquad\qquad\qquad\qquad \wedge\ I_k \in \mathbf{INTERVAL}(\mathit{valid}(r_k(N_{k,1})))\\
\qquad\qquad \wedge\ \forall l,\ 1 \le l \le n,\ \mathit{value}(r_{i_l}(N_{i_l,a_l})) = \mathit{value}(r(N_l))\\
\qquad\qquad \wedge\ \Psi_\psi(\mathit{value}(r_1(N_{1,1})),\ \ldots,\ \mathit{value}(r_k(N_{k,m_k})))\\
\qquad\qquad \wedge\ \Gamma_\tau(I_1,\ \ldots,\ I_k)\\
\qquad\qquad),\\
\qquad\quad \forall c,\ 1 \le c \le n,\\
\qquad\qquad \mathbf{EXTEND}(\Phi_v(I_1,\ \ldots,\ I_k),\ \mathbf{SUCC}(\Phi_\chi(I_1,\ \ldots,\ I_k))) \subseteq \mathit{valid}(r(N_c))\\
\qquad)\\
\qquad \wedge\ (\exists c,\ 1 \le c \le n \wedge \mathit{valid}(r(N_c)) \neq \emptyset)
\end{array} \tag{5}$$

```
23        }
```

The three main clauses in the above calculus statement correspond to the three clauses in the definition of $\hat{\pi}$, which appears on page 8. The $\hat{\times}$ operator contributes the phrase $r_1 \in \mathbf{T}(R'_1) \wedge \cdots \wedge r_k \in \mathbf{T}(R'_k)$ that appears in lines 3 and 12 of the calculus statement. The $\hat{\sigma}$ operator contributes the predicate found on lines 7 and 16 and the δ operator contributes the predicates found on lines 4-5, 8-9, 13-14, and 17-20.

We now use the definitions and lemmas presented earlier, along with set theory, to reduce the tuple calculus for R to (4). The first clause in (5), along with Lemma 1, implies that

$$
\begin{aligned}
&\forall r,\, r \in R,\ \forall c,\ 1 \leq c \leq n,\ \forall t,\ t \in \mathit{valid}(r(N_c)),\\
&\quad (\exists r'_1)\cdots(\exists r'_k),\\
&\qquad (r'_1 \in R'_1 \wedge \cdots \wedge r'_k \in R'_k\\
&\qquad \wedge \forall l,\ 1 \leq l \leq n,\ \mathit{value}(r(N_l)) = r'_{i_l}(N_{i_l,a_l})\\
&\qquad \wedge \Psi_\psi(r'_1(N_{1,1}), \ldots, r'_k(N_{k,m_k}))\\
&\qquad \wedge \Gamma_\tau(\mathbf{EXTEND}(r'_1(\mathit{From}_1), \mathbf{SUCC}(r'_1(\mathit{To}_1))), \ldots,\\
&\qquad\qquad\qquad\qquad \mathbf{EXTEND}(r'_k(\mathit{From}_k), \mathbf{SUCC}(r'_k(\mathit{To}_k))))\\
&\qquad \wedge t \in \mathbf{EXTEND}(\Phi_\upsilon(\mathbf{EXTEND}(r'_1(\mathit{From}_1), \mathbf{SUCC}(r'_1(\mathit{To}_1))), \ldots,\\
&\qquad\qquad\qquad\qquad\qquad \mathbf{EXTEND}(r'_k(\mathit{From}_k), \mathbf{SUCC}(r'_k(\mathit{To}_k)))),\\
&\qquad\qquad\qquad \mathbf{SUCC}(\Phi_\chi(\mathbf{EXTEND}(r'_1(\mathit{From}_1), \mathbf{SUCC}(r'_1(\mathit{To}_1))), \ldots,\\
&\qquad\qquad\qquad\qquad\qquad \mathbf{EXTEND}(r'_k(\mathit{From}_k), \mathbf{SUCC}(r'_k(\mathit{To}_k)))))\\
&\qquad\qquad\qquad)))
\end{aligned}
\tag{6}
$$

Applying Lemma 2 to (6) results in

$$
\begin{aligned}
&\forall r,\, r \in R,\ \forall c,\ 1 \leq c \leq n,\ \forall t,\ t \in \mathit{valid}(r(N_c)),\\
&\quad (\exists r'_1)\cdots(\exists r'_k),\\
&\qquad (r'_1 \in R'_1 \wedge \cdots \wedge r'_k \in R'_k\\
&\qquad \wedge \forall l,\ 1 \leq l \leq n,\ \mathit{value}(r(N_l)) = r'_{i_l}(N_{i_l,a_l})\\
&\qquad \wedge \Psi'_\psi(r'_1(N_{1,1}), \ldots, r'_k(N_{k,m_k}))\\
&\qquad \wedge \Gamma'_\tau((r'_1(\mathit{From}_1), r'_1(\mathit{To}_1)), \ldots, (r'_k(\mathit{From}_k), r'_k(\mathit{To}_k)))\\
&\qquad \wedge t \in \mathbf{EXTEND}(\Phi'_\upsilon((r'_1(\mathit{From}_1), r'_1(\mathit{To}_1)), \ldots, (r'_k(\mathit{From}_k), r'_k(\mathit{To}_k))),\\
&\qquad\qquad\qquad \mathbf{SUCC}(\Phi'_\chi((r'_1(\mathit{From}_1), r'_1(\mathit{To}_1)), \ldots, (r'_k(\mathit{From}_k), r'_k(\mathit{To}_k))))\\
&\qquad\qquad\qquad)))
\end{aligned}
\tag{7}
$$

The third clause of (5) on page 27 implies that $\forall r,\ r \in R,\ (\exists c)(\exists t),\ 1 \le c \le n,\ t \in valid(r(N_c))$. Hence, applying Lemma 3 and the tuple calculus statement for R'_{k+1} in (2) on page 23 to (7) results in

$$
\begin{aligned}
&\forall r, r \in R,\ \forall c,\ 1 \le c \le n,\ \forall t,\ t \in valid(r(N_c)),\\
&\quad \exists r'_{k+1},\ (r'_{k+1} \in R'_{k+1}\\
&\qquad\qquad \wedge \forall l,\ 1 \le l \le n,\ value(r(N_l)) = r'_{k+1}(N_{k+1,l})\\
&\qquad\qquad \wedge\ t \in \mathbf{EXTEND}(r'_{k+1}(From),\ \mathbf{SUCC}(r'_{k+1}(To)))\\
&\qquad\qquad)
\end{aligned}
$$

Thus, the first clause of (4) is shown to hold. A similar argument can be made, starting with the second main clause of (5), to show that the second clause of (4) holds. Since (4) holds, R and R'_{k+1} are equivalent and the historical algebra expression is equivalent to the indicated TQuel retrieve statement. ∎

3.3 TQuel Aggregates

TQuel aggregates [Snodgrass, et al. 1987] are a superset of the Quel aggregates. Hence, each of Quel's six non-unique aggregates (i.e., `count`, `any`, `sum`, `avg`, `min`, and `max`) and three unique aggregates (i.e., `countU`, `sumU`, and `avgU`) has a TQuel counterpart. The TQuel version of each of these aggregates performs the same fundamental operation as its Quel counterpart, with one significant difference. Because an historical relation represents the changing value of its attributes and aggregates are computed from the entire relation, aggregates in TQuel return a distribution of values over time. Hence, while in Quel an aggregate with no by-list returns a single value, in TQuel the same aggregate returns a sequence of values, each assigned its valid times. When there is a by-list, an aggregate in TQuel returns a sequence of values for each value of the attributes in the by-list.

Several aggregates are only found in TQuel: standard deviation (`stdev` and `stdevU`), average time increment (`avgti`), the variability of time spacing (`varts`), oldest value (`first`), newest value (`last`), *From*-*To* interval with the earliest *From* time (`earliest`), and *From*-*To* interval with the latest *From* time (`latest`).

Each TQuel aggregate has a counterpart in our historical algebra. The algebraic equivalents of TQuel aggregates are defined in terms of the historical aggregate functions $\widehat{A}$ and $\widehat{AU}$, which were defined in Section 2.3. Before defining the algebraic equivalents of TQuel aggregates in the context of a TQuel retrieve statement however, we consider the families of scalar aggregates that appear as parameters to $\widehat{A}$ and $\widehat{AU}$ in the algebraic equivalents of TQuel aggregates. Each aggregate in one of these families of scalar aggregates returns, for a partition of historical relation R at time t, the same value returned by its analogous TQuel scalar aggregate for a partition of relation R' at time t, where $R = \mathrm{T}(R')$.

We define here the families of scalar aggregates that appear as parameters to $\widehat{A}$ and $\widehat{AU}$ in the algebraic equivalents of the TQuel aggregates `count`, `countU`, `first`, and `earliest`. We present these definitions to illustrate our approach for defining the families of scalar aggregates that appear in the algebraic equivalents of TQuel aggregates. The approach can be used to define the families of scalar aggregates found in the algebraic equivalents of the other TQuel aggregates as well. The aggregates `count` and `countU` illustrate how conventional aggregate operators, now applied to historical relations, can be handled. The aggregate `first` is an example of an aggregate that evaluates to a non-temporal domain such as character but uses an attribute's valid time in a way different from the conventional aggregate operators. Finally, `earliest` illustrates an aggregate that evaluates to an interval.

For the definitions that follow, let R be an historical relation of m-tuples over the relation scheme $\mathcal{N} = \{N_1, \ldots, N_m\}$ and Q be an historical relation over an arbitrary subscheme of $\mathcal{N}$.

Although the scalar aggregate COUNT, introduced on page 14, is sufficient to define the algebraic equivalent of the TQuel aggregates `count` and `countU` for an aggregation window of length zero (i.e., an instantaneous aggregate), it is not sufficient to define the algebraic equivalent of `count` and `countU` for an aggregation window of any other length. Hence, we define another family of scalar aggregates COUNTINT_{N_a}, $1 \leq a \leq m$, that accommodates aggregation windows of arbitrary length by counting intervals rather than values.

$$\text{COUNTINT}_{N_a}(q,\ t,\ R) = \sum_{r \in R} |\mathbf{INTERVAL}(valid(r_a))|$$

where N_a is an attribute of both Q and R, $q \in Q$, and $t \in \mathcal{T}$. Recall that **INTERVAL**, formally defined in Appendix B, returns the set of intervals contained in its argument. Hence, COUNTINT simply sums the number of intervals in the time-stamp of attribute N_a of each tuple in R.

Next, we consider the TQuel aggregate `first`. This aggregate requires a family of scalar aggregate functions FIRSTVALUE_{N_a}, $1 \leq a \leq m$, where FIRSTVALUE_{N_a} produces the oldest value of attribute N_a. That is,

$$\begin{aligned}
\text{FIRSTVALUE}_{N_a}(q,\ t,\ R) \in \{u \mid R \neq \emptyset \rightarrow \exists r, (r \in R \\
&\wedge \forall r',\ r' \in R,\ \mathbf{FIRST}(r(N_a)) \leq \mathbf{FIRST}(r'(N_a)) \\
&\wedge u = value(r(N_a)) \\
&) \\
\wedge\ R = \emptyset \rightarrow u = \mathbf{NULLVALUE}(N_a) \\
\}
\end{aligned}$$

where **NULLVALUE** is an auxiliary function that returns a special null value for the domain associated with its argument. Note that the set $\{u \mid \ldots\}$ need not be a singleton set. If there are two or more elements in the set, FIRSTVALUE returns only one element, that element being selected arbitrarily. This procedure is the same as that used by the TQuel aggregate `first` to select the oldest value of an attribute when there are multiple values that satisfy the selection criteria. If R is empty, FIRSTVALUE returns a special null value for the domain associated with attribute N_a.

Finally, we define the algebraic equivalent of the TQuel aggregate `earliest`. Unlike other TQuel aggregates, which produce a distribution of scalar values over time, `earliest` produces a distribution of intervals over time. Defining an algebraic equivalent for this aggregate is slightly more complicated owing to this distinction. We first introduce a family of auxiliary functions $\mathbf{ORDERINT}_{N_a}$, $1 \leq a \leq m$, which orders chronologically all distinguishable intervals in the time-stamp of attribute N_a for tuples of historical relation R.

$$
\begin{aligned}
S \triangleq \mathbf{ORDERINT}_{N_a}(R) \leftrightarrow (\forall r)(\forall I), &(r \in R \wedge I \in \mathbf{INTERVAL}(\mathit{valid}(r(N_a)))),\\
&\quad \exists v,\ 1 \leq v \leq |S| \wedge S_v = I\\
&\wedge \forall v, 1 \leq v \leq |S|,\\
&\quad (\exists r)(\exists I), (r \in R \wedge I \in \mathbf{INTERVAL}(\mathit{valid}(r(N_a))) \wedge S_v = I)\\
&\wedge \forall v, 2 \leq v \leq |S|,\\
&\quad (\mathbf{FIRST}(S_{v-1}) < \mathbf{FIRST}(S_v)\\
&\quad \vee (\mathbf{FIRST}(S_{v-1}) = \mathbf{FIRST}(S_v) \wedge \mathbf{LAST}(S_{v-1}) < \mathbf{LAST}(S_v)))
\end{aligned}
$$

where S is a sequence of length $|S|$ and S_v is the v^{th} element of S. Evaluating $\mathbf{ORDERINT}_{N_a}(R)$ results in a sequence of the intervals appearing in the time-stamp of attribute N_a of tuples in R. The intervals are ordered from earliest starting time to latest starting time. When two or more intervals have the same starting time, they are ordered from the earliest stopping time to the latest stopping time. The first clause states that each interval in the time-stamp of attribute N_a of a tuple in R appears in S, the second clause states that no additional intervals are present, and the third clause provides the ordering conditions.

Now, we can define a family of scalar aggregate functions POSITION_{N_a}, $1 \leq a \leq m$, where POSITION_{N_a} first identifies, for a tuple q and time t, the interval in the valid component of attribute N_a in q that overlaps t and then calculates the position of that interval in $\mathbf{ORDERINT}_{N_a}(R)$, for an historical relation R. If no interval in the valid component of attribute N_a overlaps t or the interval is not in $\mathbf{ORDERINT}_{N_a}(R)$, POSITION_{N_a} returns zero.

$$
\begin{aligned}
\text{POSITION}_{N_a}(q,\ t,\ R) = u \leftrightarrow \quad &((\exists I)(\exists S_v), (I \in \mathbf{INTERVAL}(\mathit{valid}(q(N_a)))\\
&\qquad \wedge 1 \leq v \leq |\mathbf{ORDERINT}_{N_a}(R)|\\
&\qquad \wedge S_v \in \mathbf{ORDERINT}_{N_a}(R)\\
&\qquad \wedge t \in I \wedge I = S_v)\\
&) \rightarrow u = v\\
&\wedge ((\forall I)(\forall S_v), (I \in \mathbf{INTERVAL}(\mathit{valid}(q(N_a)))\\
&\qquad \wedge 1 \leq v \leq |\mathbf{ORDERINT}_{N_a}(R)|
\end{aligned}
$$

$$\wedge\ S_v \in \mathbf{ORDERINT}_{N_a}(R)$$
$$),\ t \notin I\ \vee\ I \neq S_v$$
$$) \rightarrow u = 0$$

Note that POSITION, unlike COUNTINT and FIRSTVALUE, requires parameters q and t, as well as R.

Now assume that we are given a family of scalar aggregate functions SMALLEST_{N_a}, $1 \leq a \leq m$, where SMALLEST_{N_a} produces the smallest value of numeric attribute N_a. That is,

$$\text{SMALLEST}_{N_a}(q,\ t,\ R) = u \leftrightarrow R \neq \emptyset \rightarrow \exists r,\ (r \in R$$
$$\wedge\ \forall r',\ r' \in R,\ value(r(N_a)) \leq value(r'(N_a))$$
$$\wedge\ u = value(r(N_a))$$
$$)$$
$$\wedge\ R = \emptyset \rightarrow u = 0$$

The families of scalar aggregates POSITION and SMALLEST are both needed to define the algebraic equivalent of the TQuel aggregate `earliest` for attribute N_a of relation R'. First, POSITION is used to assign each interval in the time-stamp of attribute N_a of a tuple in $\mathbf{T}(R')$ to an integer representing the interval's relative position in the chronological ordering of intervals. Then, SMALLEST is used to determine, from this assignment of intervals to integers, the times, if any, when each interval was the earliest interval. If we assume an aggregation window function $w(t) = 0$ and an empty set of by-clause attributes, the algebraic equivalent of the TQuel aggregate `earliest` for attribute N_a of relation R' is

$$\hat{\sigma}_{N_{earliest,1}=N_{earliest,2}}(\hat{A}_{\text{SMALLEST},\,0,\,N_{position},\,\emptyset}(R_{position},\ R_{position})\ \hat{\times}\ R_{position}) \tag{8}$$

over the scheme $\mathcal{N}_{earliest} = \{N_{earliest,1},\ N_{\text{earliest},2}\}$ where

$$R_{position} = \hat{\sigma}_{N_{position} \neq 0}(\hat{A}_{\text{POSITION},\,\infty,\,N_a,\,\emptyset}(R,\ R)) \tag{9}$$

over the scheme $\mathcal{N}_{position} = \{N_{position}\}$.

EXAMPLE. If we assume an aggregation window function $w(t) = 0$ and an empty set of by-clause attributes, then `earliest` for attribute *State* of relation S_6 is

$$\hat{\sigma}_{N_{earliest,1}=N_{earliest,2}}(\hat{A}_{\text{SMALLEST},\,0,\,N_{position},\,\emptyset}(R_{position},\ R_{position})\ \hat{\times}\ R_{position}) =$$
$$\begin{aligned}\{\ &\langle(1,\ \{1,2\}),\ (1,\ \{1,2\})\rangle\ ,\\ &\langle(2,\ \{3\}),\ (2,\ \{1,2,3\})\rangle\ ,\\ &\langle(3,\ \{4,5,6\}),\ (3,\ \{4,5,6\})\rangle\ ,\\ &\langle(5,\ \{7,8\}),\ (5,\ \{7,8\})\rangle\ \}\end{aligned}$$

where $R_{position}$ is

$$\hat{\sigma}_{N_{position}\neq 0}(\hat{A}_{\text{POSITION},\,\infty,\,State,\,\emptyset}(\text{S}_6,\ \text{S}_6)) =$$
$$\begin{aligned}\{\ &\langle(1,\ \{1,2\})\rangle\ ,\\ &\langle(2,\ \{1,2,3\})\rangle\ ,\\ &\langle(3,\ \{4,5,6\})\rangle\ ,\\ &\langle(4,\ \{5,6\})\rangle\ ,\\ &\langle(5,\ \{7,8\})\rangle\ \}\end{aligned}$$ □

As illustrated in this example, the algebraic equivalent of `earliest` is a two-attribute historical relation. The valid component of the first attribute is the time when the valid component of the second attribute was the earliest interval. Also note that the value component of both attributes is the position of the valid component of the second attribute in $\mathbf{ORDERINT}_{N_a}(R)$.

3.3.1 TQuel Aggregates in the Target List

In Section 3.2 we showed the algebraic equivalent of the TQuel retrieve statement without aggregates. We now show the algebraic equivalent of a TQuel retrieve statement with aggregates in its target list. We consider changes to the algebraic expression to support one non-unique aggregate in the target list only; similar changes would be needed for each additional aggregate in the target list.

Once again assume that we are given the k snapshot relations $R'_1,\ \ldots,\ R'_k$ whose schemes are respectively,

$$\mathcal{N}_1 = \{N_{1,1},\ \ldots,\ N_{1,m_1},\ From_1,\ To_1\}$$
$$\ldots$$
$$\mathcal{N}_k = \{N_{k,1},\ \ldots,\ N_{k,m_k},\ From_k,\ To_k\}$$

where, for notational convenience, we assume that $N_{11},\ \ldots,\ N_{k,m_k}$ are unique. Also, let

i_1, i_2, …, i_n and j_1, j_2, …, j_p be integers, not necessarily distinct, in the range 1 to k, indicating the tuple variables (possibly repeated) appearing in the target list and aggregate, respectively;

a_l, $1 \le l \le n$, be an integer in the range 1 to m_{i_l}, indicating the attribute names appearing in the target list where $(\forall u)(\forall v)$, $(1 \le u \le n \wedge 1 \le v \le n \wedge u \ne v \wedge i_u = i_v)$, $a_u \ne a_v$;

c_h, $1 \le h \le p$, be an integer in the range 1 to m_{j_h}, indicating the attribute names appearing in the aggregate where $(\forall u)(\forall v)$, $(1 \le u \le p \wedge 1 \le v \le p \wedge u \ne v \wedge j_u = j_v)$, $c_u \ne c_v$; and

$\bar{j}_1$, $\bar{j}_2$, …, $\bar{j}_x$ be the distinct integers in j_1, j_2, …, j_p where $\bar{j}_1 = j_1$, indicating the x (non-repeated) tuple variables appearing in the aggregate.

Then, the TQuel retrieve statement with the aggregate f'_1 in the target list has the following syntax

$$
\begin{array}{l}
\texttt{range of } r'_1 \texttt{ is } R'_1 \\
\ldots \\
\texttt{range of } r'_k \texttt{ is } R'_k \\
\texttt{retrieve into } R'_{k+1}(N_{k+1,1} = r'_{i_1}.N_{i_1,a_1},\ \ldots,\ N_{k+1,n} = r'_{i_n}.N_{i_n,a_n}, \\
\qquad\qquad N_{k+1,n+1} = f'_1(r'_{j_1}.N_{j_1,c_1} \texttt{ by } r'_{j_2}.N_{j_2,c_2},\ \ldots,\ r'_{j_p}.N_{j_p,c_p} \\
\qquad\qquad\qquad \texttt{for } \omega_1 \\
\qquad\qquad\qquad \texttt{where } \psi_1 \\
\qquad\qquad\qquad \texttt{when } \tau_1)) \\
\quad \texttt{valid from } v \texttt{ to } \chi \\
\quad \texttt{where } \psi \\
\quad \texttt{when } \tau
\end{array}
\tag{10}
$$

This statement computes a new relation R'_{k+1} over the relational scheme

$$\mathcal{N}_{k+1} = \{N_{k+1,1},\ \ldots,\ N_{k+1,n},\ N_{k+1,n+1},\ From_{k+1},\ To_{k+1}\}$$

The for clause specifies an aggregation window function for the aggregate f'_1. ω_1 contains one or more keywords that determine, along with the time granularity of R'_1, …, R'_k, the length of the aggregation window at each time t. The keywords `each instant` represent the aggregation window function $w(t) = 0$ (i.e., an instantaneous aggregate) and the keyword `ever` represents the aggregation window function $w(t) = \infty$ (i.e., a cumulative aggregate). The length of the aggregation window specified by other keywords (e.g., `each day`, `each week`, `each year`) is a function of the underlying time granularity of the database. For example, if the time granularity is a day, then $\omega =$ `each week` translates to the aggregation window function $w(t) = 6$. Also, the aggregation window function need not be a constant function. For example, if the time granularity is a day, then $\omega =$ `each month` translates to the aggregation window function w, where $w(t) = 31$ if t corresponds to January 31 and $w(t) = 28$ if t corresponds to February 28. We let Ω_{ω_1} be the function denoted by ω_1 and the time granularity of $R'_1, \ldots, R'_k$.

Every TQuel retrieve statement of the form of (10) is equivalent to an expression in our historical algebra of the form

$$
\begin{aligned}
R = \hat{\pi}_{N_{i_1,a_1}, \ldots, N_{i_n,a_n}, N_{agg_1,p}}(\delta_{\Gamma_\tau,\, \mathbf{EXTEND}(\Phi_v, \mathbf{SUCC}(\Phi_\chi)) \cap N_{\bar{\jmath}_1,1} \cap \cdots \cap N_{\bar{\jmath}_x,1} \cap N_{agg_1,p}}(\\
\hat{\sigma}_{\Psi_\psi \wedge N_{j_2,c_2}=N_{agg_1,1} \wedge \cdots \wedge N_{j_p,c_p}=N_{agg_1,p-1}}(\mathbf{T}(R'_1) \hat{\times} \cdots \hat{\times} \mathbf{T}(R'_k) \hat{\times} R_{agg_1}))) \qquad (11)
\end{aligned}
$$

where

$$
\begin{aligned}
R_{agg_1} = \widehat{A}_{f_1,\, \Omega_{\omega_1},\, N_{j_1,c_1},\, \{N_{j_2,c_2}, \ldots, N_{j_p,c_p}\}}(\hat{\pi}_{N_{j_1,c_1}, \ldots, N_{j_p,c_p}}(\mathbf{T}(R'_{\bar{\jmath}_1}) \hat{\times} \cdots \hat{\times} \mathbf{T}(R'_{\bar{\jmath}_x})), \\
\delta_{\Gamma_{\tau_1},\, N_{\bar{\jmath}_1,1}, \ldots, N_{\bar{\jmath}_x,m_{\bar{\jmath}_x}}}(\hat{\sigma}_{\Psi_{\psi_1}}(\mathbf{T}(R'_{\bar{\jmath}_1}) \hat{\times} \cdots \hat{\times} \mathbf{T}(R'_{\bar{\jmath}_x})))) \qquad (12)
\end{aligned}
$$

over the scheme $\mathcal{N}_{agg_1} = \{N_{agg_1,1}, \ldots, N_{agg_1,p}\}$, where $\forall u, 1 \leq u \leq p-1,\ N_{agg_1,u} = N_{j_{u+1},c_{u+1}}$ and $N_{agg_1,p}$ is the attribute name associated with the aggregate value. Here we assume that f_1 is the family of scalar aggregates (e.g., COUNTINT) corresponding to the family of TQuel aggregates f'_1 (e.g., count). Expression (12) applies the where and when predicates to the cartesian product of the relations associated with tuples variables appearing in the aggregate, and applies the aggregate operator to the result. Expression (11) differs only slightly from the expression (3) on page 26 for a retrieve statement without aggregates. The expanded selection operator provides the necessary linkage between the attributes in the aggregate's by-list and corresponding attributes in the base relations. The expanded derivation operator imposes the TQuel restriction that the valid time of tuples in the derived relation be the intersection of the valid time specified in the valid clause, the valid times of the tuples in the base relations participating in the aggregation, and the valid time of the aggregate itself. Of course, if f'_1 is a unique aggregate, then $\widehat{AU}$ should be used instead of $\widehat{A}$ in (12).

Two changes to (11) are required to handle special cases. First, if a tuple variable $\bar{\jmath}_u$, $1 \leq u \leq x$, does not appear outside the aggregate f'_1 in (10), then $N_{\bar{\jmath}_u,1}$ does not appear in the second subscript of the δ operator. Also, if $\bar{\jmath}_1$ appears neither outside the aggregate f'_1 in (10) nor in its by clause, then R_{agg_1} is replaced by

$$
R_{agg_1} \hat{\cup} \{ \langle (\mathbf{NULLVALUE}(N_{\bar{\jmath}_1,1}), \{t \mid \forall r,\ r \in R_{agg_1},\ r \notin valid(r(N_{agg_1,p}))\}) \rangle \}
$$

The first change removes the restriction that the valid time of a tuple in the derived relation must intersect the valid time of at least one tuple in the base relation associated with tuple variable $\bar{\jmath}_u$. The second change, ensures that a value (possibly a distinguished null value) for the aggregate is specified at each time t, $t \in \mathcal{T}$.

3.3.2 TQuel Aggregates in the Inner Where Clause

Aggregates may also appear in the where, when, and valid clauses of a TQuel retrieve statement. We now show the algebraic equivalents of TQuel retrieve statements with aggregates in these clauses, first presenting the algebraic equivalent of a TQuel retrieve statement with an aggregate in an inner where clause. Assume that a TQuel aggregate f'_2 appears in ψ_1 in (10) and let

$g_1, g_2, \ldots, g_y$ be integers, not necessarily distinct, in the range 1 to k, indicating the (possibly repeated) tuple variables appearing in the nested aggregate where $\forall g_u,\ 1 \le u \le y,\ \exists j_v,\ 1 \le v \le p,\ g_u = j_v$;

d_l, $1 \le l \le y$, be an integer in the range 1 to m_{g_l}, indicating the attribute names appearing in the nested aggregate where $(\forall u)(\forall v)$, $(1 \le u \le y \wedge 1 \le v \le y \wedge u \ne v \wedge g_u = g_v)$, $d_u \ne d_v$; and

$\bar{g}_1, \bar{g}_2, \ldots, \bar{g}_z$ be the distinct integers in $g_1, g_2, \ldots, g_y$ where $\bar{g}_1 = g_1$, indicating the z (non-repeated) tuple variables in the aggregate.

Then, f'_2 in ψ_1 has the following syntax

$$
\begin{aligned}
&f'_2(r'_{g_1}.N_{g_1,d_1}\ \texttt{by}\ r'_{g_2}.N_{g_2,d_2}, \ldots, r'_{g_y}.N_{g_y,d_y}\\
&\qquad\qquad \texttt{for}\ \omega_2\\
&\qquad\qquad \texttt{where}\ \psi_2\\
&\qquad\qquad \texttt{when}\ \tau_2)
\end{aligned}
$$

As this TQuel retrieve statement is complicated, containing a nested aggregate with a full complement of by, for, where, and when clauses, we should expect a somewhat complicated algebraic equivalent.

When modified to account for f'_2 in ψ_1, the algebraic equivalent of f'_1, given in (12), becomes,

$$
\begin{aligned}
R_{agg_1} = {} & \hat{\pi}_{N_{j_2,c_2}, \ldots, N_{j_p,c_p}, N_{agg_1}}(\widehat{A}_{f_1,\, \Omega_{\omega_1},\, N_{j_1,c_1},\, \{N_{j_1,m_{j_1}+1}, N_{j_2,c_2}, \ldots, N_{j_p,c_p}\}}(\\
& \quad \hat{\pi}_{N_{j_1,c_1}, N_{j_1,m_{j_1}+1}, N_{j_2,c_2}, \ldots, N_{j_p,c_p}}(\mathrm{T}(R'_{\bar{\jmath}_1}) \hat{\times} \{\langle(1, \mathcal{T})\rangle\} \hat{\times} \cdots \hat{\times} \mathrm{T}(R'_{\bar{\jmath}_z})),\\
& \quad \hat{\pi}_{N_{\bar{\jmath}_1,1}, \ldots, N_{\bar{\jmath}_1,m_{\bar{\jmath}}+1}, N_{\bar{\jmath}_2,1}, \ldots, N_{\bar{\jmath}_z,m_{\bar{\jmath}_z}}}(\\
& \qquad \delta_{\Gamma_{\tau_1},\, N_{\bar{\jmath}_1,1}, \ldots, N_{\bar{\jmath}_1,m_{\bar{\jmath}_1}},\, N_{\bar{\jmath}_1,m_{\bar{\jmath}_1}+1} \cap N_{agg_2,y},\, N_{\bar{\jmath}_2,1}, \ldots, N_{\bar{\jmath}_z,m_{\bar{\jmath}_z}},\, N_{agg_2,1}, \ldots, N_{agg_2,y}}(\\
& \qquad\qquad \hat{\sigma}_{\Psi_{\psi_1} \wedge N_{g_2,d_2}=N_{agg_2,1} \wedge \cdots \wedge N_{g_y,d_y}=N_{agg_2,y-1}}(\\
& \qquad\qquad\quad \mathrm{T}(R'_{\bar{\jmath}_1}) \hat{\times} \{\langle(1, \mathcal{T})\rangle\} \hat{\times} \cdots \hat{\times} \mathrm{T}(R'_{\bar{\jmath}_z}) \hat{\times} R_{agg_2})))))
\end{aligned}
\tag{13}
$$

where the attribute name N_{agg_1} here refers to the aggregate produced in $\hat{A}$ by f_1, the reference to the aggregate f_2' in ψ_1 is replaced by a reference to $N_{agg_2,y}$, and

$$R_{agg_2} = \hat{A}_{f_2,\, \Omega_{\omega_2},\, N_{g_1,d_1},\, \{N_{g_2,d_2},\, \ldots,\, N_{g_y,d_y}\}}(\hat{\pi}_{N_{g_1,d_1},\, \ldots,\, N_{g_y,d_y}}(\mathbf{T}(R'_{\jmath_1})\hat{\times}\cdots\hat{\times}\mathbf{T}(R'_{\jmath_x})),$$
$$\delta_{\Gamma_{\tau_2},\, N_{\jmath_1,1},\, \ldots,\, N_{\jmath_x,m_{\jmath_x}}}(\hat{\sigma}_{\Psi_{\psi_2}}(\mathbf{T}(R'_{\jmath_1})\hat{\times}\cdots\hat{\times}\mathbf{T}(R'_{\jmath_x}))))$$

over the scheme $\mathcal{N}_{agg_2} = \{N_{agg_2,1},\ \ldots,\ N_{agg_2,y}\}$, and f_2 is the family of scalar aggregates corresponding to the family of TQuel aggregates f_2'.

$\{\langle(1,\ \mathcal{T})\rangle\}$ is a constant relation containing a single tuple whose value component may be an arbitrary value from an arbitrary domain. Here, we effectively add an additional attribute to $R_{\jmath_1}$ and then use the attribute as an implicit by-list attribute to restrict tuples in the partition of $\mathbf{T}(R'_{\jmath_1})\hat{\times}\cdots\hat{\times}\mathbf{T}(R'_{\jmath_x})$ at time t to only those tuples that satisfy the predicate in ψ_1 involving the aggregate f_2' at time t.

3.3.3 TQuel Aggregates in the Inner When Clause

Assume now that the aggregate f_2' appears in τ_1 in (11) rather than in ψ_1. The only aggregates that can appear in τ_1 are `earliest` and `latest`. Therefore, if we let R_{agg_2} be the two-attribute algebraic equivalent of f_2', then the algebraic equivalent of f_1' would be the same as that given in (13) for an aggregate in the inner where clause, with one exception. The reference to f_2' in τ_1 is replaced by a reference to $N_{agg_2,y+1}$, not $N_{agg_2,y}$. The valid component of $N_{agg_2,y}$ is the time when the valid component of $N_{agg_2,y+1}$ was the oldest interval, hence $N_{agg_2,y+1}$ is used in evaluating τ_1.

If we assume that f_2' is `earliest`, then R_{agg_2} is

$$R_{agg_2} = \hat{\sigma}_{N_{agg_2,y}=N_{agg_2,y+1}}(\hat{A}_{\mathrm{SMALLEST}_{N_{g_1,d_1}},\, \Omega_{\omega_2},\, N_{position},\, \{N_{g_2,d_2},\, \ldots,\, N_{g_y,d_y}\}}($$
$$\hat{\pi}_{N_{position},\, N_{g_2,d_2},\, \ldots,\, N_{g_y,d_y}}(R_{position}\hat{\times}\mathbf{T}(R'_{\jmath_1})\hat{\times}\cdots\hat{\times}\mathbf{T}(R'_{\jmath_x})), \quad (14)$$
$$\delta_{\Gamma_{\tau_2}\wedge N_{g_1,d_1}=N_{position},\, N_{position},\, N_{\jmath_1,1},\, \ldots,\, N_{\jmath_x,m_{\jmath_x}}}($$
$$\hat{\sigma}_{\Psi_{\psi_2}}(R_{position}\hat{\times}\mathbf{T}(R'_{\jmath_1})\hat{\times}\cdots\hat{\times}\mathbf{T}(R'_{\jmath_x}))))$$
$$\hat{\times}\ (R_{position}\hat{\cup}\{\langle(0,\ \mathcal{T})\rangle\}))$$

over the scheme $\mathcal{N}_{agg_2} = \{N_{agg_2,1},\ \ldots,\ N_{agg_2,y+1}\}$ where

$$R_{position} = \hat{\sigma}_{N_{position}\neq 0}(\hat{A}_{\mathrm{POSITION},\, \infty,\, N_{g_1,d_1},\, \emptyset}(\mathbf{T}(R'_{\jmath_1}),\ \mathbf{T}(R'_{\jmath_1}))) \quad (15)$$

Expression (14), while structurally equivalent to expression (8) on page 32, is considerably more complex because of the presence of by, when, and where clauses in the nested aggregate.

The attributes of $\widehat{A}$'s first argument now include the attributes appearing in the by clause and the attributes of $\widehat{A}$'s second argument include the attributes of relations associated with tuple variables appearing in the aggregate. Also, tuples in the second argument are now required to satisfy the where predicate and, for some interval in the time-stamp of attribute N_{g_1,d_1}, the when predicate. Finally, because TQuel assumes `earliest` and `latest` return $\mathcal{T}$ for an empty partition of R', the tuple $\langle (0, \mathcal{T}) \rangle$ is added to $R_{position}$ so that $\mathcal{T}$ will be considered the earliest interval at those times when the partition of $\widehat{A}$'s second argument is empty. Recall that SMALLEST, defined on page 32, returns zero when passed an empty relation.

3.3.4 TQuel Aggregates in the Outer Where Clause

Assume that the TQuel aggregate f_1' appears in ψ in (10) rather than in the target list. Then, the algebraic equivalent of the TQuel retrieve statement is

$$R = \hat{\pi}_{N_{i_1,a_1}, \ldots, N_{i_n,a_n}}(\delta_{\Gamma_\tau}, \text{EXTEND}(\Phi_v, \text{SUCC}(\Phi_\chi)) \cap N_{j_1,1} \cap \cdots \cap N_{j_z,1} \cap N_{agg_1,p}(\hat{\sigma}_{\Psi_\psi \wedge N_{j_2,e_2}=N_{agg_1,1} \wedge \cdots \wedge N_{j_p,e_p}=N_{agg_1,p-1}}(\mathbf{T}(R_1')\hat{\times}\cdots\hat{\times}\mathbf{T}(R_k')\hat{\times}R_{agg_1})))$$

where the reference to f_1' in ψ is replaced by a reference to $N_{agg_1,p}$. Note that the only other change from expression (11) is the elimination of attribute $N_{agg_1,p}$ from the projection, since the aggregate does not appear in the target list.

3.3.5 TQuel Aggregates in the Outer When Clause

Assume now that the aggregate f_1' appears in τ in (10). Then, the algebraic equivalent of the TQuel retrieve statement is

$$R = \hat{\pi}_{N_{i_1,a_1}, \ldots, N_{i_n,a_n}}(\delta_{\Gamma_\tau}, \text{EXTEND}(\Phi_v, \text{SUCC}(\Phi_\chi)) \cap N_{j_1,1} \cap \cdots \cap N_{j_z,1} \cap N_{agg_1,p}(\hat{\sigma}_{\Psi_\psi \wedge N_{j_2,e_2}=N_{agg_1,1} \wedge \cdots \wedge N_{j_p,e_p}=N_{agg_1,p-1}}(\mathbf{T}(R_1')\hat{\times}\cdots\hat{\times}\mathbf{T}(R_k')\hat{\times}R_{agg_1})))$$

where the reference to f_1' in τ is replaced by a reference to $N_{agg_1,p+1}$. If the aggregate f_1' is in v or χ rather than τ, analogous changes would be required.

3.3.6 Multiply-nested Aggregation

The approach described above for handling aggregates in the inner where and when clauses can be used to handle aggregates in a qualifying where or when clause of an aggregate in the outer where, when, or valid clauses. This method of converting TQuel aggregates to their algebraic equivalents, when there is an aggregate in a qualifying clause, can also handle an arbitrary level of nesting of aggregates.

3.4 Correspondence Theorems

Now that all possible locations for aggregates in a TQuel retrieve statement have been examined, we can assert that

Theorem 3 *Every TQuel retrieve statement has an equivalent expression in our historical algebra.*

PROOF. Induct on the number of aggregates appearing in the statement to arrive at an equivalent algebraic expression, applying the replacements discussed above in Sections 3.3.1 through 3.3.5, as appropriate. Incorporate the handling of transaction time via the rollback operator ($\hat{\rho}$) as discussed elsewhere [McKenzie & Snodgrass 1987A]. Construct a tuple calculus expression for the retrieve statement and the algebraic expression, then prove equivalence using the technique used in the proof of Theorem 2. While the proof is aided by the presence of auxiliary relations in the tuple calculus semantics for aggregates [Snodgrass 1987], it is still cumbersome and offers little additional insight. ∎

In a similar fashion, by also using the `modify_state` and `modify_scheme` commands described elsewhere [McKenzie & Snodgrass 1987B], one can construct equivalent algebraic statements for the TQuel create, delete, append, replace, and destroy statements.

Theorem 4 *The historical algebra defined here is strictly more powerful than TQuel.*

PROOF. The previous theorem shows that the expressive power of the algebra is as great as that of TQuel. Now, for two TQuel relations R_1' and R_2', consider the algebraic expression $\mathrm{T}(R_1') \hat{\times} \mathrm{T}(R_2')$. Because the semantics of TQuel requires that tuples rather than attributes be time-stamped, this algebraic expression has no counterpart in TQuel. Hence, the algebra is strictly more powerful than TQuel. ∎

4 Review of Design Decisions

In defining the historical algebra presented in Section 2, we were faced with three major design decisions: whether to time-stamp tuples or attributes, whether to allow single-valued or set-valued time-stamps, and whether to allow single-valued or set-valued attributes. We discuss here our choices and the importance of those choices in determining the properties of the algebra. We also mention the choices to these design decisions made by the developers of seven other historical algebras: Ben-Zvi's Time Relational Model [Ben-Zvi 1982], Clifford's proposed extension to the snapshot algebra [Clifford & Croker 1987], Gadia's homogeneous and multihomogeneous historical algebras [Gadia 1984, Gadia 1986], Jones' extension to the snapshot algebra to support time-oriented operations for LEGOL [Jones et al. 1979], Tansel's historical algebra [Tansel 1986], and Navathe's historical algebra [Navathe & Ahmed 1986]. A detailed review and evaluation of historical algebras, using desirable properties as evaluation criteria, can be found elsewhere [McKenzie & Snodgrass 1987C].

4.1 Time-stamped Attributes

We decided to time-stamp attributes rather than tuples to support historical queries. We wanted the algebra to allow for the derivation of information valid at a time t from information in underlying relations valid at other times, much as the snapshot algebra allows for the derivation of information about entities or relationships from information in underlying relations about other entities or relationships. This requirement implies that the algebra allow units of related information, possibly valid at disjoint times, to be combined into a single related unit of information possibly valid at some other times. Support for such a capability required that we define a cartesian product operator that concatenates tuples, independent of their valid times, and preserves, in the resulting tuple, the valid-time information for each of the underlying tuples. Only by time-stamping attributes could we define a cartesian product operator with this property and maintain closure under cartesian product.

Tansel and Gadia also time-stamp attributes. Only Tansel's algebra and Gadia's multihomogeneous model, however, allow tuples with disjoint attribute time-stamps; Gadia's homongeneous model requires that a tuple's attribute time-stamps be identical. Clifford assigns a time-stamp, termed a ***lifespan***, to each tuple in a relation and to each attribute in the relation's scheme. The lifespan of each attribute of a tuple is then computed as the intersection of the tuple's lifespan and the attribute's lifespan, as specified in the relation's scheme. Ben-Zvi, Jones, and Navathe all time-stamp tuples only.

4.2 Set-valued Time-stamps

We decided to allow set-valued attribute time-stamps for several reasons. First, we wanted the algebra to support the user-oriented conceptual view of historical relations as 3-dimensional objects [Ariav 1986, Clifford & Tansel 1985] and each historical operator to have an interpretation, consistent with its semantics, in accordance with this conceptual framework. That is, we wanted the definitions of the algebraic operations to be consistent with the conceptual view that historical operators manipulate space-filling objects. For example, the difference operator should take two space-filling objects (i.e., historical relations) and produce a object that represents the mass (i.e., total historical information) present in the first object but not present in the second object. Note that this description of operations on historical relations as "volume" operations on 3-dimensional objects is consistent not only with the conceptual view of historical relations as space-filling objects but also with the semantics of the individual snapshot algebraic operations as operations on 2-dimensional tables, extended to account for the additional dimension represented by valid time. Secondly, we wanted the algebra to satisfy the following commutative, associative, and distributive tautologies involving union, difference, and cartesian product that are defined in set theory [Enderton 1977] as well as the non-conditional commutative laws involving selection and projection presented by Ullman [Ullman 1982], while supporting the definition of historical intersection in terms of historical difference.

$$Q \hat{\cup} R = R \hat{\cup} Q$$

$$Q \hat{\times} R = R \hat{\times} Q$$

$\hat{\sigma}_{F_1}(\hat{\sigma}_{F_2}(R)) = \hat{\sigma}_{F_2}(\hat{\sigma}_{F_1}(R))$

$Q \hat{\cup} (R \hat{\cup} S) = (Q \hat{\cup} R) \hat{\cup} S$

$Q \hat{\times} (R \hat{\times} S) = (Q \hat{\times} R) \hat{\times} S$

$Q \hat{\times} (R \hat{\cup} S) = (Q \hat{\times} R) \hat{\cup} (Q \hat{\times} S)$

$\hat{\sigma}_F(Q \hat{\cup} R) = \hat{\sigma}_F(Q) \hat{\cup} \hat{\sigma}_F(R)$

$\hat{\sigma}_F(Q \hat{-} R) = \hat{\sigma}_F(Q) - \hat{\sigma}_F(R)$

$\hat{\pi}_X(Q \hat{\cup} R) = \hat{\pi}_X(Q) \hat{\cup} \hat{\pi}_X(R)$

$Q \hat{\cap} R = Q \hat{-} (Q \hat{-} R)$

We specifically did not include one tautology, the distributive property of cartesian product over difference, in this list because it is inconsistent with the conceptual view of operations on historical relations as "volume" operations on space-filling objects [McKenzie & Snodgrass 1987C]. Finally, we wanted there to be a unique representation for each historical relation to keep the semantics of the algebra as simple as possible.

If we had decided to disallow set-valued attribute time-stamps, then we would had to have premitted value-equivalent tuples to model accurately real-world temporal relationships. Yet, value-equivalent tuples, because they spread temporal relationships among attributes across tuples, would have caused problems in defining an algebra with the above properties. If value-equivalent tuples had been allowed (and set-valued attribute time-stamps disallowed), a unique representation for each historical relation could not have been specified without imposing inter-tuple restrictions on the attribute time-stamps of value-equivalent tuples. Also, historical operators, in particular the difference operator, that would have satisfied both the conceptual view of historical operations as "volume" operations on space-filling objects and the above tautologies, while preventing loss of information about temporal relationships as an operator side-effect, could not have been defined.

By allowing set-valued attribute time-stamps (and disallowing value-equivalent tuples), we were able to define an algebra that has the desired properties. Because value-equivalent tuples are disallowed, each historical relation is guaranteed to have a unique representation. In addition, the definitions of historical operators given in Section 2 are consistent with the conceptual view of historical operations as "volume" operations on space-filling objects, and the algebra satisfies the ten tautologies listed above.

The decision to allow set-valued attribute time-stamps unfortunately prevented the algebra from having other less desirable, but nonetheless desirable, properties. If we had not allowed set-valued attribute time-stamps, we could have retained the first-normal-form property of the snapshot algebra. Also, we could have replaced the single complex historical derivation operator with two simple operators, one performing historical selection and the other performing historical projection.

Clifford and Gadia also allow set-valued time-stamps. Ben-Zvi, Jones, Navathe, and Tansel all allow only single-valued time-stamps.

4.3 Single-valued Attributes

We decided to restrict attributes to single values to retain in our algebra the commutative properties of the selection operator found in the snapshot algebra. If we had allowed set-valued attributes, without imposing intra-tuple restrictions on attribute time-stamps, then we would had to have combined the functions of the selection and historical derivation operators into a single, more powerful operator. This consolidation would have been necessary to ensure that the temporal predicate in the current historical derivation operator was considered to be true for an assignment of intervals to attribute names only when the predicate in the current selection operator held for the attribute values associated with those intervals. This new operator would have satisfied the commutative properties of the current selection operator only in restricted cases. Hence we would have limited the usefulness of key optimization strategies in future implementations of our algebra.

Ben-Zvi, Jones, and Navathe also restrict attributes to single values. Clifford, Gadia, and Tansel, however, allow set-valued attribute values.

5 Summary and Future Work

This paper makes two contributions. First, an historical algebra is defined as a straightforward extension of the conventional relational algebra. Secondly, the algebra is shown to have the expressive power of the temporal query language TQuel.

The design of an historical algebra is a surprisingly difficult task. Although defining an algebra that has a given property is easy, it is much more difficult to define an algebra that has many desirable properties. We found that many subtle issues arise when attempting to define an algebra that satisfies several design goals. Also, all desirable properties of historical algebras are not compatible [McKenzie & Snodgrass 1987C]. Hence, the best that can be hoped for is not an algebra with all possible desirable properties but an algebra with a maximal subset of the most desirable properties.

The historical algebra defined in Section 2 has what we consider to be the most desirable properties of an historical algebra. First, the algebra is a straightforward extension of the snapshot algebra. Each relation and algebraic expression in the snapshot algebra has an equivalent counterpart in the historical algebra. Expressions in the snapshot algebra can be converted to their historical equivalent simply by replacing each snapshot operator with its corresponding historical operator and converting the referenced snapshot relations to historical relations by assigning all attributes the same time-stamp. The historical operators $\hat{\cup}$, $\hat{-}$, $\hat{\times}$, $\hat{\sigma}$, and $\hat{\pi}$ all reduce to their snapshot counterparts when all attribute time-stamps are the same. The algebra is also consistent with the conceptual view of historical relations as 3-dimensional, space-filling objects and the view of operations on historical relations as "volume" operations. In addition, the algebra supports historical queries, has the expressive power of a non-procedural temporal query language, includes aggregates, does not exhibit temporal data loss as an operator side-effect, and has a unique representation for each historical relation. Finally, the algebra satisfies all but one of the commutative, associative, and distributive tautologies involving union, difference, and cartesian product as well as the non-conditional commutative laws involving selection and projection. No other historical algebra to our knowledge has all these properties.

The obvious future work is an implementation of the algebra as defined in Section 2 and development of optimization strategies. At this point, we feel that the formal definition of temporal databases and their query languages has yielded many results (c.f., [McKenzie 1986]), while implementation issues such as access methods, physical storage structures, and novel storage devices remain largely unexplored.

6 Acknowledgements

Research by the first author was sponsored in part by the United States Air Force. Research by the second author was sponsored in part by an IBM Faculty Development Award and in part by the Office of Naval Research under contract N00014-86-K-0680. The work was also supported by NSF grant DCR-8402339.

7 Bibliography

[Ariav 1986] Ariav, G. *A Temporally Oriented Data Model. ACM Transactions on Database Systems,* 11, No. 4, Dec. 1986, pp. 499-527.

[Ben-Zvi 1982] Ben-Zvi, J. *The Time Relational Model.* PhD. Diss. Computer Science Department, UCLA, 1982.

[Bontempo 1983] Bontempo, C. J. *Feature Analysis of Query-By-Example,* in Relational Database Systems. New York: Springer-Verlag, 1983. pp. 409-433.

[Clifford & Tansel 1985] Clifford, J. and A.U. Tansel. *On an Algebra for Historical Relational Databases: Two Views,* in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 247-265.

[Clifford & Croker 1987] Clifford, J. and A. Croker. *The Historical Data Model (HRDM) and Algebra Based on Lifespans,* in *Proceedings of the International Conference on Data Engineering,* IEEE Computer Society. Los Angeles, CA: Feb. 1987.

[Codd 1970] Codd, E.F. *A Relational Model of Data for Large Shared Data Bank. Communications of the Association of Computing Machinery,* 13, No. 6, June 1970, pp. 377-387.

[Enderton 1977] Enderton, H.B. *Elements of Set Theory.* New York, N.Y.: Academic Press, Inc., 1977.

[Gadia 1984] Gadia, S.K. *A Homogeneous Relational Model and Query Languages for Temporal*

Databases. 1984. (Unpublished paper.)

[Gadia 1986] Gadia, S.K. *Toward a Multihomogeneous Model for a Temporal Database,* in *Proceedings of the International Conference on Data Engineering,* IEEE Computer Society. Los Angeles, CA: IEEE Computer Society Press, Feb. 1986, pp. 390-397.

[Held et al. 1975] Held, G.D., M. Stonebraker and E. Wong. *INGRES–A Relational Data Base Management System. Proceedings of the AFIPS 1975 National Computer Conference,* 44, May 1975, pp. 409-416.

[Jones et al. 1979] Jones, S., P. Mason and R. Stamper. *LEGOL 2.0: A Relational Specification Language for Complex Rules. Information Systems,* 4, No. 4, Nov. 1979, pp. 293-305.

[Klug 1982] Klug, A. *Equivalence of Relational Algebra and Relational Calculus Query Languages Having Aggregate Functions. Journal of the Association of Computing Machinery,* 29, No. 3, July 1982, pp. 699-717.

[McKenzie 1986] McKenzie, E. *Bibliography: Temporal Databases. ACM SIGMOD Record,* 15, No. 4, Dec. 1986, pp. 40-52.

[McKenzie & Snodgrass 1987A] McKenzie, E. and R. Snodgrass. *Extending the Relational Algebra to Support Transaction Time,* in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. U. Dayal and I. Traiger. Association for Computing Machinery. San Francisco, CA: May 1987, pp. 467-478.

[McKenzie & Snodgrass 1987B] McKenzie, E. and R. Snodgrass. *Scheme Evolution and the Relational Algebra.* Technical Report TR87-003. Computer Science Department, University of North Carolina at Chapel Hill. May 1987.

[McKenzie & Snodgrass 1987C] McKenzie, E. and R. Snodgrass. *A Survey of Historical Algebras.* Technical Report TR87-020. Computer Science Department, University of North Carolina at Chapel Hill. Sep. 1987.

[Navathe & Ahmed 1986] Navathe, S.B. and R. Ahmed. *A Temporal Relational Model and a Query Language.* UF-CIS Technical Report TR-85-16. Computer and Information Sciences Department, University of Florida. Apr. 1986.

[Overmyer & Stonebraker 1982] Overmyer, R. and M. Stonebraker. *Implementation of a Time Expert in a Database System. ACM SIGMOD Record,* 12, No. 3, Apr. 1982, pp. 51-59.

[Snodgrass & Ahn 1985] Snodgrass, R. and I. Ahn. *A Taxonomy of Time in Databases,* in *Proceedings of ACM SIGMOD International Conference on Management of Data,* Ed. S. Navathe. Association for Computing Machinery. Austin, TX: May 1985, pp. 236-246.

[Snodgrass & Ahn 1986] Snodgrass, R. and I. Ahn. *Temporal Databases. IEEE Computer,* 19, No. 9, Sep. 1986, pp. 35-42.

[Snodgrass 1987] Snodgrass, R. *The Temporal Query Language TQuel. ACM Transactions on*

Database Systems, 12, No. 2, June 1987, pp. 247-298.

[Snodgrass, et al. 1987] Snodgrass, R., S. Gomez and E. McKenzie. *Aggregates in the Temporal Query Language TQuel.* TempIS Technical Report 16. Computer Science Department, University of North Carolina at Chapel Hill. July 1987.

[Stonebraker et al. 1976] Stonebraker, M., E. Wong, P. Kreps and G. Held. *The Design and Implementation of INGRES. ACM Transactions on Database Systems,* 1, No. 3, Sep. 1976, pp. 189-222.

[Tandem 1983] Tandem Computers, Inc. *ENFORM Reference Manual.* Cupertino, CA, 1983.

[Tansel, et al. 1985] Tansel, A.U., M.E. Arkun and G. Ozsoyoglu. *Time-By-Example Query Language for Historical Databases.* Technical Report. Bernard M. Baruch College, CUNY. 1985.

[Tansel 1986] Tansel, A.U. *Adding Time Dimension to Relational Model and Extending Relational Algebra. Information Systems,* 11, No. 4 (1986), pp. 343-355.

[Ullman 1982] Ullman, J.D. *Principles of Database Systems, Second Edition.* Potomac, Maryland: Computer Science Press, 1982.

A Notational Conventions

This appendix describes the notational conventions used in this paper.

Notation	Usage
$\hat{\cup}$	Historical union operator
$\hat{-}$	Historical difference operator
$\hat{\times}$	Historical cartesian product operator
$\hat{\sigma}$	Historical selection operator
$\hat{\pi}$	Historical projection operator
$\hat{\delta}$	Historical derivation operator
$\hat{A}$	Historical aggregation function for non-unique aggregates
$\widehat{AU}$	Historical aggregation function for unique aggregates
a, b, c, d	Attribute variables
$\mathcal{D}_a$	Arbitrary flat domain associated with attribute N_a
F	Predicate in the historical selection operator
f	Scalar aggregate
G	Predicate in the historical derivation operator
g, i, j	Relation variables
h, l	Variables ranging over attributes in target list, by-list, or aggregate
$\mathcal{I}$	Domain of intervals
I	Interval
I_{N_a}	Interval from the time-stamp of attribute N_a
I_a	Shorthand for I_{N_a}
k	Number of relations
m, m_i	Number of attributes in relation schemes $\mathcal{N}$, $\mathcal{N}_i$
$\mathcal{N}$, $\mathcal{N}_i$	Relation schemes
N_a, $N_{i,a}$	Attribute names
n	Length of target list or by-list
$\wp(\mathcal{I})$	Power set of $\mathcal{I}$
$\wp(\mathcal{T})$	Power set of $\mathcal{T}$
p, y	Number of attributes appearing in an aggregate
Q, R, R_i	Historical relations

q, r, r_i	Historical tuple variables
Q', R', R'_i	TQuel relations
q', r', r'_i	TQuel tuple variables
$\mathcal{T}$	Time Domain
T	Subset of $\mathcal{T}$
t	Element of $\mathcal{T}$
u, v	Temporary variables
V_a	Temporal function in the historical derivation operator
$valid(r(N_a))$	Time-stamp of attribute N_a of tuple r
$valid(r_a)$	Shorthand for $valid(r(N_a))$
$value(r(N_a))$	Value component of attribute N_a of tuple r
$value(r_a)$	Shorthand for $value(r(N_a))$
w	Aggregation window function
X	Set of by-list attributes in an aggregate
x, z	Number of tuple variables appearing in an aggregate

B Auxiliary Functions

We used several auxiliary functions in the definition of the historical derivation operator. We present here formal definitions for each of those auxiliary functions.

FIRST takes a set of times from the domain $\wp(\mathcal{T})$ and maps it into the earliest time in the set.

$$\mathbf{FIRST} : \wp(\mathcal{T}) \rightarrow \mathcal{T} \cup \perp$$

$$\mathbf{FIRST}(T) \triangleq \begin{cases} \perp & T = \emptyset \\ t,\ t \in T \wedge \forall t',\ t' \in T,\ t \leq t' & \text{otherwise} \end{cases}$$

LAST takes a set of times from the domain $\wp(\mathcal{T})$ and maps it into the latest time in the set.

$$\mathbf{LAST} : \wp(\mathcal{T}) \rightarrow \mathcal{T} \cup \perp$$

$$\mathbf{LAST}(T) \triangleq \begin{cases} \perp & T = \emptyset \\ t,\ t \in T \wedge \forall t',\ t' \in T,\ t \geq t' & \text{otherwise} \end{cases}$$

PRED is the predecessor function on the domain $\mathcal{T}$. It maps a time into its immediate predecessor in the linear ordering of all times.

$$\mathbf{PRED} : \mathcal{T} \rightarrow \mathcal{T} \cup \perp$$

$$\mathbf{PRED}(t) \triangleq \begin{cases} \perp & t = \mathbf{FIRST}(\mathcal{T}) \\ t_P,\ t_P \in \mathcal{T} \wedge t_P < t \wedge \forall t',\ t' \in \mathcal{T} \wedge t' < t,\ t' \leq t_P & \text{otherwise} \end{cases}$$

SUCC is the successor function on the domain $\mathcal{T}$. It maps a time into its immediate successor in the linear ordering of all times.

$$\mathbf{SUCC} : \mathcal{T} \rightarrow \mathcal{T}$$

$$\mathbf{SUCC}(t) \triangleq t_S,\ t_S \in \mathcal{T} \wedge t_S > t \wedge \forall t',\ t' \in \mathcal{T} \wedge t' > t,\ t' \geq t_S$$

Let the domain $\mathcal{I}$ be the subset of $\wp(\mathcal{T})$ that represents all possible non-disjoint intervals of time.

$$\mathcal{I} \triangleq \{I \mid I \in \wp(\mathcal{T}) \wedge \forall t,\ t \in I \rightarrow \mathbf{FIRST}(I) \leq t \leq \mathbf{LAST}(I)\}$$

Note that $\mathcal{I}$ includes intervals of length 1. Also let $\wp(\mathcal{I})$ be the power set of $\mathcal{I}$. While $\mathcal{I} \subset \wp(\mathcal{T})$, each element of $\wp(\mathcal{I})$ is a set, each of whose elements are also elements of $\wp(\mathcal{T})$.

EXTEND maps two times into the set of times that represents the interval between the first time and the second time.

$$\textbf{EXTEND} : \mathcal{T} \times \mathcal{T} \rightarrow \mathcal{I} \cup \perp$$

$$\textbf{EXTEND}(t_1,\, t_2) \triangleq \begin{cases} \perp & t_1 > t_2 \\ \{t \mid t_1 \leq t \leq t_2\} & \text{otherwise} \end{cases}$$

INTERVAL maps a set of times into the set of intervals containing the minimum number of non-disjoint intervals represented by the input set. Each time in the input set appears in exactly one interval in the output set and each interval in the output set is itself represented by a set of times.

INTERVAL partitions a set of times into its corresponding set of intervals where each interval is itself represented by a set of times.

$$\textbf{INTERVAL} : \wp(\mathcal{T}) \rightarrow \wp(\mathcal{I}) \cup \emptyset$$

$$\textbf{INTERVAL}(T) \triangleq \begin{cases} \emptyset & T = \emptyset \\ \{I \mid \forall t,\ t \in I, t \in T \\ \qquad \wedge\, \textbf{PRED}(t) \in T \rightarrow \textbf{PRED}(t) \in I & \text{otherwise} \\ \qquad \wedge\, \textbf{SUCC}(t) \in T \rightarrow \textbf{SUCC}(t) \in I\} & \end{cases}$$

Note that **INTERVAL** partitions a set of times into the minimum number of non-disjoint intervals represented by the set; each time in T appears in exactly one interval.