An algebraic framework for temporal attribute characteristics

Michael Böhlen [a], Johann Gamper [a] and Christian S. Jensen [b]

[a] *Faculty of Computer Science, Free University of Bozen-Bolzano, Dominikanerplatz 3, 39100 Bolzano, Italy*
E-mail: {boehlen; gamper}@inf.unibz.it
[b] *Aalborg University, Aalborg, Denmark*
E-mail: csj@cs.aau.dk

Most real-world database applications manage temporal data, i.e., data with associated time references that capture a temporal aspect of the data, typically either when the data is valid or when the data is known. Such applications abound in, e.g., the financial, medical, and scientific domains. In contrast to this, current database management systems offer preciously little built-in query language support for temporal data management. This situation persists although an active temporal database research community has demonstrated that application development can be simplified substantially by built-in temporal support. This paper's contribution is motivated by the observation that existing temporal data models and query languages generally make the same rigid assumption about the semantics of the association of data and time, namely that if a subset of the time domain is associated with some data then this implies the association of any further subset with the data. This paper offers a comprehensive, general framework where alternative semantics may co-exist. It supports so-called malleable and atomic temporal associations, in addition to the conventional ones mentioned above, which are termed constant. To demonstrate the utility of the framework, the paper defines a characteristics-enabled temporal algebra, termed CETA, which defines the traditional relational operators in the new framework. This contribution demonstrates that it is possible to provide built-in temporal support while making less rigid assumptions about the data and without jeopardizing the degree of the support. This moves temporal support closer to practical applications.

Keywords: temporal databases, temporal algebra, attribute characteristics, malleable attributes, temporal data semantics

AMS subject classification: 68P15

1.   Introduction

Time-referenced, or temporal, data abounds in contemporary database management applications. Due perhaps to the ubiquity of temporal data and evident difficulties in managing such data using presently known concepts and techniques,

several research communities have studied the management of temporal data quite extensively, for the last several decades [1, 4, 7, 24, 25, 29].

The artificial intelligence research community has devoted considerable efforts to the study of the modeling of reality using different types of temporal entities or propositions, and this community distinguishes among such concepts as, e.g., properties, events, and processes [5]. However, it has not been a main concern to integrate these concepts into the data models and query languages of the contemporary database management systems.

In contrast, the database research community has adopted a more systems-oriented approach and has favored quite rigid assumptions about the semantics of temporal data [12, 18, 21, 23]. In particular, strict point-based and snapshot-equivalence-based semantics have been favored. With these semantics, two temporal relations are identical if their pairs of snapshots at all points in time are identical. It is typically assumed that if some data is associated with a subset of the time domain then it is also associated with any subset of this subset. For example, if an employee is assigned to a project during some week, it is assumed to also be true that the employee is assigned to the project during, say, the Tuesday of that week. This assumption enables the design of relatively simple temporal data models and query languages that offer advanced built-in support for managing temporal data. However, this assumption also limits the generality and applicability of the resulting models and query languages.

This paper aims to generalize temporal data models and query languages, thus broadening their applicability; and it aims to do so without sacrificing the degree of built-in support. In particular, this paper proposes a framework where different semantics may be assigned to the association of data with time values. For example, assume that our employee works for 40 h during some week. This does not mean that it is true that the employee worked for 40 h during Tuesday of that week. Rather, we would expect that the employee worked for perhaps 8 h during each workday. The framework proposed here accommodates a range of semantics such as this one. We will refer to these different semantics by saying that the attributes have different characteristics.

The framework is based on the relational model and the relational algebra. An algebraic framework is chosen to avoid issues to do with syntax and language design, instead enabling a focus on semantics. The framework satisfies basic properties and permits various temporal semantics. In the framework, where the time domain is assumed to be discrete, timestamps in the form of time intervals, i.e., convex sets of time points, are used for capturing the temporal aspects of data.

The framework enables operations on data to respect, in a precise technical sense, the grouping of time points into timestamps. More precisely, the grouping of time points in the timestamps of the tuples that result from algebraic operations is unique: Timestamps are maximal, while preserving the original grouping of time points in the argument tuples.

With our framework, the definition of a temporal algebra is carried out in three steps. The first step is to define the *minimal requirements*. The minimal requirements

are defined for each algebraic operator; they specify the output tuples together with associated sets of time points, but they do not group the time points into timestamps. As an example, we define the minimal requirements for the algebraic operators of a temporal algebra, CETA.

The second step groups the time points of the output tuples into timestamps and generates for each possible timestamp a candidate result tuple. For each candidate result tuple we determine its lineage, i.e., the minimal set of input tuples that are required to derive it. These input tuples are called *required argument tuples*.

The third step considers the lineage information and selects those candidate result tuples, called *maximal output tuples*, that minimize the required argument tuples and respect the original grouping of time points as much as possible, i.e., form maximal intervals. These intervals are the timestamps of the result tuples. We also adjust the attribute values of the result tuples according to the characteristics of their attributes to the new timestamp. This is done independently of the specific operators.

This three-step process limits the complexity of the definitions of algebraic operators, since the minimal requirements, which are defined for each algebraic operator, do not specify the grouping of time points into timestamps. The framework generalizes the scope of applicability of built-in support for temporal data in existing data models and query languages. It brings temporal data management concepts closer to being supported in contemporary database management systems, which currently offer only very limited built-in support for managing temporal data.

The remainder of the paper is structured as follows: section 2 covers related work. Section 3 motivates the proposed framework and also introduces an example to be used throughout the paper. Section 4 then proceeds to provide basic definitions and notational conventions. Section 5 covers the adjustment of attribute values according to the characteristics of their attributes. The algebraic aspects of the framework are then presented in section 6, and section 7 presents key properties of the framework. Section 8 applies the framework, defining the characteristics-enabled temporal algebra, CETA. The paper ends with conclusions and an outlook about future work.

2. Related work

Different research communities have investigated various aspects of time-varying information [6, 8, 15, 27, 31]. This paper falls within the area of temporal database systems that strives to provide support for the management of temporal data [22, 28, 34]. Most temporal database systems assume a point-based semantics that permits attributes with a constant characteristics only. By also taking into consideration the grouping of time points, we significantly broaden the scope of applications of temporal relational algebras [19, 26, 30, 33] and, by implication, temporal database management systems. This work extends our initial work on algebras that combine point and interval properties [10, 13]. The framework presented in this paper can be used to extend any point-based algebra to provide a unique

grouping of time points that is based on the original grouping of time points in the argument relations. We discuss how to add support for malleable and atomic attribute characteristics to a temporal algebra.

Terenziani and Snodgrass [31] compare the point-based database approach to related work in linguistics, philosophy, and artificial intelligence, where the different semantics that may be tied to time interval associations with data have received more attention. They argue that it is necessary to distinguish between telic and atelic facts. An atelic relation is used if truth values are associated with time points, and a telic relation is used if truth values are associated with time intervals. They emphasize that the distinction between telic and atelic facts is not rigid: Both are needed, and it must be possible to convert between the two. They propose a three-sorted temporal relational model, together with coercion functions to map between atelic and telic relations. Thus, they have three different types of relations (atelic, telic, and nontemporal) together with corresponding algebras. Their work focuses on the semantics of temporal relations and statements. In contrast, we are interested in providing a foundation for the flexible and non-constraining support that temporal database systems should offer, to enable them to facilitate applications with different needs.

Assuming a fixed granularity, the malleable characteristic with the uniformity assumption resembles the constant characteristic. The uniformity assumption makes it possible to normalize malleable values to the length of a chronon, and then treat them as constant values. However, a normalization renders the data unsuitable for direct use by applications. For example, it would be rather cumbersome if applications had to deal with normalized grants or salaries (funding or salary per second). Further, the kind of normalization that would be needed impacts many aspects of a database management system, e.g., indexing, and it is unclear whether normalization is practical. Also, if the malleable characteristic is refined further (e.g., to use non-uniform distributions), normalization is no longer applicable. Finally, even if all data were normalized, it might still be relevant for some applications to preserve the original grouping of time points [10, 11].

The lineage of tuples has previously been defined and used for the maintenance of data warehouse views [16, 17]. In our framework, the required argument tuples capture the lineage for temporal algebraic operators, and this lineage is used for the grouping of time points into timestamps of result tuples according to their original grouping in the argument relations.

3.   Motivation

To motivate the proposed framework, we proceed to consider concrete examples of different temporal semantics. This section's coverage is informal – the remainder of the paper offers a formal coverage.

As suggested briefly in the introduction, it is appropriate to apply the different temporal semantics of data at the level of attributes in our relational context. In

particular, the associations of the values of different attributes in the same relation with time values may have different semantics. To illustrate these different semantics, or attribute characteristics, we consider constant, malleable, and atomic attributes.

An attribute is *constant* if a value of this attribute, associated with some time value, termed a timestamp, may also be associated, unchanged, with a modified timestamp. The literature also uses the terms atelic [31] and homogeneous [5] for such attributes. It is instructive to consider an example.

Consider a project database where the relation EMPL stores information about employments, including the name of an employee (N), a contract identifier (CI), the name of a project (P), the hours an employee is assigned to a project (H), and the valid time (T) over which a tuple holds true. See figure 1.

We assign the constant characteristic to attributes Name and Project. This means that the values of these attributes also apply when their associated timestamps are transformed. Suppose that the timestamp $[1, 3]$ of the first tuple is split into $[1, 2]$ and $[3, 3]$. Then values Jan and P1 also hold for the new timestamps, i.e., Jan works for P1 during both intervals.

Next, an attribute is *malleable* if its values change if the associated timestamps change. In the example, we assign the malleable characteristic to attribute Hours. This means that a value of this attribute does not also hold for proper subsets of its timestamp. In general, additional information is needed in order to determine an appropriate value of a malleable attribute when a timestamp is modified. In particular, information is needed on how to inter- or extrapolate a malleable attribute value. For simplicity, we assume a uniform distribution as a default. Thus, if a timestamp is transformed, the attribute value is transformed proportionally. Using the first tuple in the example, the Hours value is adjusted to 200 for the interval $[1, 2]$ and to 100 for the interval $[3, 3]$.

Finally, an attribute is *atomic* if its values cannot be associated with modified timestamps. An atomic attribute roughly corresponds to Allen's notion of an event [5]. For example, chemotherapies used in cancer treatment often prescribe a specific amount of drugs to be taken over a specific time period. Neither the amount of drugs nor the time period can be modified to prevent wrong prescriptions.

EMPL

N	CI	P	H	T
Jan	137	P1	300	[1,3]
Tom	138	P1	400	[1,4]
Tom	147	P1	400	[5,6]
Ann	139	P1	300	[1,6]
Tom	140	P2	200	[2,3]
Jan	142	P2	600	[2,5]

{N/Jan,CI/137,P/P1,H/300,T/[1,3]}
{N/Tom,CI/138,P/P1,H/400,T/[1,4]}
{N/Tom,CI/147,P/P1,H/400,T/[5,6]}
{N/Ann,CI/139,P/P1,H/300,T/[1,6]}
{N/Tom,CI/140,P/P2,H/200,T/[2,3]}
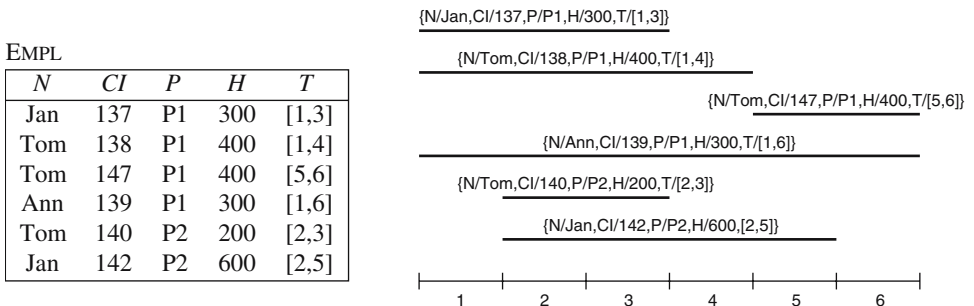{N/Jan,CI/142,P/P2,H/600,[2,5]}

Figure 1. A project database.

Note that almost all temporal data models proposed by the temporal database community assume that all attributes have the constant characteristic. This applies to the so-called point-based data models [13, 32] and to models where snapshot equivalent [20, 23] relations are treated as being identical.

A framework that encompasses malleable and atomic attributes must support the following functionality:

- *Malleable attributes require additional consideration when timestamps are modified.* Additional information is needed as to how to inter- or extrapolate malleable attribute values. We will assume a uniform distribution as a default. Thus, if the timestamp is transformed, the attribute value has to be transformed proportionally.

  Note also that malleable attributes cannot be coalesced in the traditional sense [11]. Thus, two tuples with pairwise identical attribute values and adjacent timestamps cannot be consolidated into a single tuple with the union of the adjacent timestamps as its new timestamp. For example, the projection of relation EMPL on attributes Name, Project, and Hours yields two tuples that, apart from attribute T, have pairwise identical attribute values, i.e., $\{N/Tom, P/P1, H/400, T/[1,4]\}$ and $\{N/\text{Tom}, P/P1, H/400, T/[5,6]\}$. Coalescing these two tuples leads to the wrong result that Tom worked for 400 h in the interval $[1,6]$. The reason is that the Hours attribute is malleable.

- *Timestamps associated with atomic attribute values cannot be modified at all.* Atomic attributes represent facts or events that are accomplished over a specific time interval. Hence, the time interval cannot be changed as we have seen in the chemotherapy example.

- *The characteristic of an attribute is subject to change.* We do not make any a priori assumptions about the characteristics of attributes. Rather, the characteristic of an attribute may vary depending on the context or the specific operation being performed on the data. For instance, in the chemotherapy example, an atomic characteristic is useful as a conservative default. At the same time, a medical doctor should have the possibility of applying the constant or malleable characteristics when querying or updating treatments.

We will use constant, atomic, and malleable characteristics with the uniformity assumption as representative attribute characteristics as we discuss the handling of different attribute characteristics.

4. Preliminaries

Next, we introduce basic definitions and notational conventions to be used in the paper. Specifically, we define a time domain, (temporal) relation schemas, (temporal) tuples, and (temporal) relations in turn.

Definition 4.1 (*Time Domain*). A *time domain*, D^T, is a discrete, totally ordered domain. The elements are termed chronons (or time points), and the ordering relation is given as $<^T$.

The natural numbers with the relation $<$ satisfy these requirements and we use them as our time domain throughout. A timestamp is a convex set with respect to $<$, i.e., a time interval over this domain, and is represented by two chronons, denoting its inclusive starting and ending points, respectively. For example, $[1, 7]$ represents the timestamp which starts with time point 1 and ends with time point 7. For timestamps, we introduce membership and subset relations: $t \in T$ means that chronon t is included in timestamp T. For two timestamps T and T', $T' \subseteq T$ iff all chronons in T' are also included in T.

A *relation schema* is a three-tuple $S = (\Omega, \Delta, dom)$, where Ω is a non-empty, finite set of attributes, Δ is a finite set of domains, and $dom : \Omega \to \Delta$ is a function that associates a domain with each attribute. A *temporal relation schema* is a relation schema with at least one time attribute, i.e., an attribute over domain D^T, where $D^T \in \Delta$. For the purpose of this paper, we define the temporal relation schema $R = \{A_1, \ldots, A_k, T\}$. Note that the assumption that each relation has a timestamp attribute T is just for convenience. There is no implicit timestamp attribute, and all definitions are parameterized with a timestamp attribute. The rename operator ρ can be used to adjust schemas as appropriate. For example, in binary operations a timestamp attribute in one of the argument relations can be renamed so that the argument relations have a common timestamp attribute.

A *tuple over schema* $S = (\Omega, \Delta, dom)$ is a function $x : \Omega \to \cup_{\delta \in \Delta} \delta$, such that for every attribute A of Ω, $x(A) \in dom(A)$. A tuple is temporal iff its schema is temporal. We will represent a tuple as a set of attribute/value pairs:

$$x = \{A_1/v_1, \ldots, A_k/v_k, T/t\}$$

A *relation over schema* S is a finite set of tuples over S, denoted as **r**. If S is a temporal schema, **r** is called a temporal relation.

To simplify notation and improve readability, we introduce a few abbreviations. For a tuple x and a set of attributes **X**, we write $x[\mathbf{X}]$ for the projection of the tuple on the attributes in **X**, i.e., $x[\mathbf{X}] = \{A/v \mid A \in \mathbf{X} \wedge A/v \in x\}$. For the tuple $x = \{N/\text{Jan}, CI/137, P/\text{P1}, H/300, T/[1, 3]\}$, the projection $x[\{N, H\}]$ yields $\{N/\text{Jan}, H/300\}$. For a tuple x and an attribute A we write $x.A$ to denote the value of the attribute A in x, i.e., $x.A = v$ iff $A/v \in x$. In the previous example we have $x.N = \text{Jan}$. A vector of relations $\langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ will be denoted as $\vec{\mathbf{r}}$. We define the subset relation $\vec{\mathbf{s}} \subseteq \vec{\mathbf{r}}$ as follows: $\vec{\mathbf{s}} \subseteq \vec{\mathbf{r}}$ iff $\mathbf{s}_1 \subseteq \mathbf{r}_1, \ldots, \mathbf{s}_l \subseteq \mathbf{r}_l$. Similarly, $\vec{\mathbf{s}} \subset \vec{\mathbf{r}}$ iff $\mathbf{s}_1 \subseteq \mathbf{r}_1, \ldots, \mathbf{s}_l \subseteq \mathbf{r}_l$, and $\mathbf{s}_i \subset \mathbf{r}_i$ for at least one i.

5. Support for multiple attribute characteristics

To be able to support multiple characteristics, we first augment the definition of a temporal relation to also capture attribute characteristics.

Definition 5.1 (*Attribute Characteristics*). The *attribute characteristics* of a temporal relation are given by a function $C : \Omega \setminus \{T\} \to \{a, c, m\}$.

The attribute characteristics of a relation with schema $\{A_1, \ldots, A_k, T\}$ are then given as $C = \{A_1{:}c_1, \ldots, A_k{:}c_k\}$, where c_i are the attribute characteristics. We use m, a, and c to denote malleable, atomic, and constant characteristics, respectively. For example, $H{:}m$ means that the Hours attribute has the malleable characteristic.

The tuples in the result of a query,[1] termed *output tuples*, are derived from the tuples in the argument relations, termed the *input tuples*. Each output tuple is generally shaped by just a subset of the input tuples. We denote the subset of input tuples that affect the output tuples the *required argument tuples* (to be defined formally in section 6).

Exactly the required argument tuples have to be considered when calculating an output tuple of an algebraic operator. For example, in the case of a temporal selection, the result are all those input tuples that satisfy the selection predicate. Each output tuple is determined by one required argument tuple – the one it is identical to.

For other operators, the situation is more complex. Consider a binary temporal join: Here, an output tuple is produced for each pair of input tuples with timestamps that intersect. Output tuples have one timestamp attribute, whose timestamp is equal to the intersection of the timestamps of the input tuples (cf. Definition 8.1). In the case of the join, the output timestamp is different from the timestamps of the input tuples, and the non-timestamp attribute values may have to be adjusted.

We proceed to define how the attribute values of a tuple are adjusted according the characteristics of the attributes. Intuitively, we start out with a temporal tuple $\{A_1/v_1, \ldots, A_k/v_k, T/I\}$. The definition then adjusts each non-timestamp attribute value of the tuple in turn and returns an adjusted tuple. Explanation and examples follow the definition. Another instance from the literature of adjusting attribute values is the coercion function introduced by Camossi et al. [14].

Definition 5.2 (*Adjustment of Attribute Values*). Let $\tilde{z} = \{A_1/v_1, \ldots, A_k/v_k\}$ be a set of attribute/value pairs, let I be a timestamp, let C be the attribute characteristics, let f be an aggregate function, and let $\vec{s} = \langle s_1, \ldots, s_l \rangle$ be the required argument tuples of $\tilde{z} \cup \{T/I\}$. The *adjustment of attribute values* is defined as follows:

$adj(\tilde{z}, I, \vec{s}, C)$
$\quad = \{eval(A_1/v_1, I, S, C), \ldots, eval(A_k/v_k, I, S, C)\}, where\ S = s_1 \uplus \ldots \uplus s_l$
$eval(w, I, S, C)$

$$
= \begin{cases}
A/v & \text{iff} & w = A/v \wedge A{:}c \in C \\
A/(v * |I|/|I'|) & \text{iff} & w = A/v \wedge A{:}m \in C \wedge \exists y \in S(A/v \in y \wedge I' = y.T) \\
A/v & \text{iff} & w = A/v \wedge A{:}a \in C \wedge \exists y \in S(A/v \in y \wedge I = y.T) \\
A/\textit{UNDEF} & \text{iff} & w = A/v \wedge A{:}a \in C \wedge \exists y \in S(A/v \in y \wedge I \neq y.T) \\
A'/f(\{\!\{v \mid y \in S \wedge A/v = eval(y[A], I, y, C)\}\!\}) \\
& \text{iff} & w = A'/f(\{\!\{A/v_1, \ldots, A/v_q\}\!\}) \wedge S = \{\!\{y_1, \ldots, y_q\}\!\} \; .
\end{cases}
$$

---

[1] Queries are relational algebra expression, e.g., Example 8.1.

To simplify the notation in the evaluation function, we replace $\vec{\mathbf{s}} = \langle \mathbf{s}_1, \ldots, \mathbf{s}_l \rangle$ by a (heterogeneous) bag S.[2] The value *UNDEF* belongs to each domain in Δ and means that the function is not defined.

The first argument of *eval* is a single attribute/value pair. The function *eval* then uses timestamp I, the required argument tuples S, and the attribute characteristic of the attribute/value pair for adjusting the attribute value. In the case of a constant attribute, the attribute's value is not changed at all. For a malleable attribute, the attribute's value is adjusted proportionally according to the new interval I. For an atomic attribute, the intervals I and I' must be identical, and the attribute value is not changed. Otherwise, the value of an atomic attribute is undefined over I, represented by the unique constant *UNDEF*.

For aggregations we have to adjust and aggregate q values. The attribute/value pairs are collected in the bag of the attribute values to be adjusted. Additionally, the attribute/value pairs are also included in the required argument tuples, where they are available together with the associated timestamps. Since the timestamps are needed to adjust the values, the attribute/value pairs are extracted from the required argument tuples and adjusted before the aggregation function is evaluated.

Example 5.1 Consider the relation in figure 1. We want to sum up the working hours per project. With attribute Hours being malleable and the remaining attributes being constant, we expect the following tuple to belong to the result set:

$$\{P/\text{P1}, H/750, T/[1,3]\}$$

This result tuple depends on three input tuples (no. 1, 2, and 4) of relation EMPL. These tuples must be adjusted before they are used in the aggregate function. We call the *adj* function as follows:

$$adj(\{P/\text{P1}, SH/sum\{\!\!\{H/300, H/400, H/300\}\!\!\}\},$$
$$[1,3],$$
$$\langle\{\!\!\{ \{N/\text{Jan}, P/\text{P1}, H/300, T/[1,3]\},$$
$$\{N/\text{Tom}, P/\text{P1}, H/400, T/[1,4]\},$$
$$\{N/\text{Ann}, P/\text{P1}, H/300, T/[1,6]\}\}\!\!\}\rangle,$$
$$\{P{:}c, H{:}m\})$$
$$= \{P/\text{P1}, SH/750\}$$

In the first step the *eval* function is pushed into the tuple which is evaluated. Since P is constant, we get $eval(P/\text{P1}, \ldots) = P/\text{P1}$ for the first attribute. For the evaluation of

---

[2] To avoid attribute name clashes in S, we use qualified attribute names, $R.A$, and rename attributes as appropriate.

the second attribute, $eval(SH/sum(\{\!\{H/300, H/400, H/300\}\!\}), \ldots)$, the *eval* function is applied to the aggregate attribute of each required argument tuple:

$$eval(H/300, [1, 3], \{N/\text{Jan}, P/\text{P1}, H/300, T/[1, 3]\}, \{P{:}c, H{:}m\}) = H/300$$

$$eval(H/400, [1, 3], \{N/\text{Jan}, P/\text{P1}, H/400, T/[1, 4]\}, \{P{:}c, H{:}m\}) = H/300$$

$$eval(H/300, [1, 3], \{N/\text{Ann}, P/\text{P1}, H/300, T/[1, 6]\}, \{P{:}c, H{:}m\}) = H/150$$

The bag of adjusted attribute values is finally evaluated by the aggregation function, i.e., $sum(\{\!\{300, 300, 150\}\!\}) = 750$, yielding $SH/750$ as the final result of the *adj* function.

Example 5.2 Assume the chemotherapy example and the tuple $\{N/\text{Jan}, D/310, T/[1, 6]\}$ that prescribes a dosage of 310 over a six-day period to Jan. Assume a query that wants to retrieve information about the treatment during day 1.

$$adj(\{N/\text{Jan}, D/310\},$$
$$[1, 1],$$
$$\langle\{\{N/\text{Jan}, D/310, T/[1, 6]\}\}\rangle,$$
$$\{N{:}c, D{:}a\})$$
$$= \{N/\text{Jan}, D/\textit{UNDEF}\}$$

Since D has the atomic characteristic and the input and output intervals are different, undefined is returned for the dosage attribute.

The definition of the evaluation function is flexible and allows the modification of attribute values if the timestamp is reduced or if the timestamp is extended. For example, if we want to estimate the hours Jan is working in a specific project based on his hours in a shorter period, we have to increase the value proportionally to the increase of the timestamp. It is not necessary that the new timestamp overlaps the old one. An example of a completely disjoint timestamp is the extrapolation of quarterly numbers of a company from one year to another.

6.   An algebraic framework for temporal attribute characteristics

We proceed to describe how attribute characteristics are to be taken into account in algebraic query languages on temporal relations with multiple attribute characteristics. Particular attention is given to the description of how the grouping of time points in the timestamps of input tuples affects the grouping of time points in the timestamps of the output tuples.

The overall objective of the algebraic framework is to provide built-in support for handling multiple attribute characteristics in a temporal setting. Specifically, the framework supports the adjustment of attribute values if timestamps change and it preserves the grouping of time points as defined by the input tuples. The definition of the framework proceeds in the following steps:

1. For each possible (unadjusted) result tuple, determine the set of time points that should be associated with that tuple ($\rightarrow$ minimal requirements).

2. Determine all minimal sets of argument tuples that completely determine a convex subset of time points of each such result tuple ($\rightarrow$ required argument tuples).

3. Timestamp result tuples with the same required argument tuples with maximal subsets of convex time points. Use the timestamps together with the associated argument tuples and the attribute characteristics to adjust the attribute values of the result tuples by applying the *adj* function from the previous section ($\rightarrow$ maximal output tuples).

These steps will be formalized in the following definitions.

6.1. Minimal requirements

Informally, the minimal requirements determine, for each (unadjusted) output tuple of an algebraic operator, the set of time points that must be included in one or more timestamps to be associated with the output tuple.

Definition 6.1 (*Minimal Requirements, MR*). Let $\vec{\mathbf{r}} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be argument relations with timestamp attribute T, $\tilde{z}$ be a set of attribute/value pairs of an output tuple modulo the adjustment of the attribute values, and E be a set of (not necessarily connected) time points associated with $\tilde{z}$. The *minimal requirements* for an *l*-ary temporal operator $\mathcal{O}$ is a formula of the form

$$\mathcal{O}^{MR}[T](\vec{\mathbf{r}}, \tilde{z}, E)$$

such that $\mathcal{O}^{MR}[T](\vec{\mathbf{r}}, \tilde{z}, E_1) \wedge \mathcal{O}^{MR}[T](\vec{\mathbf{r}}, \tilde{z}, E_2) \Rightarrow E_1 = E_2$, i.e., for each possible output tuple there is only one minimal requirement.

This is a general scheme – the minimal requirements must be defined for each operator individually. Hence, if we instantiate the above definition for concrete operators, such as the standard relational operators, we have $\mathcal{O}^{MR} \in \{\sigma^{MR}[P], \pi^{MR}[\mathbf{X}], \cup^{MR}, -^{MR}, \times^{MR}, \mathcal{G}^{MR}[\mathbf{G}][\mathbf{F}]\}$, where P is a selection predicate, $\mathbf{X}$ is a subset of attributes on which to project, $\mathbf{G}$ is a subset of attributes on which to group, and $\mathbf{F} = \{f_1(A_1)/A'_1, \ldots, f_p(A_p)/A'_p\}$ is a list of aggregate functions f_i on attributes A_i along with new

attribute names A_i' for the resulting values. For instance, the minimal requirements for temporal projection would be defined as follows:

$$\pi^{MR}[\mathbf{X}][T](\langle\mathbf{r}\rangle,\tilde{z},E) \text{ iff } \forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge \tilde{z} = x[\mathbf{X}]))$$

where $\tilde{z}$ is the unadjusted output tuple and E is the set of all time points for which $\tilde{z}$ holds true.

Recall that the purpose is to specify the time points that must be associated with result tuples. The minimal requirements of a temporal operator specify for each prospective, unadjusted result tuple the set of time points E that the timestamp(s) of that tuple must include.

Below we exemplify the minimal requirements by defining them for temporal aggregation. The aggregation example is used throughout the paper. We consider the calculation of the sums of hours, *SH*, spent on projects as a running example. The complete definition of the minimal requirements for CETA is given in section 8.

Example 6.1 (Temporal Aggregation, MR). The minimal requirements for the temporal aggregation are defined as follows:

$$\mathcal{G}^{MR}[\mathbf{G}][\mathbf{F}][T](\langle\mathbf{r}\rangle,\tilde{z},E) \text{ iff}$$

$$\forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge$$

$$av_1 = \{\!\{y[A_1] \mid y \in \mathbf{r} \wedge t \in y.T \wedge x[\mathbf{G}] = y[\mathbf{G}]\}\!\} \wedge \ldots \wedge$$

$$av_p = \{\!\{y[A_p] \mid y \in \mathbf{r} \wedge t \in y.T \wedge x[\mathbf{G}] = y[\mathbf{G}]\}\!\} \wedge$$

$$\tilde{z} = x[\mathbf{G}] \cup \{A_1'/f_1(av_1),\ldots,A_p'/f_p(av_p)\}))$$

where $\mathbf{G}$ is a subset of the attributes on which to group the result, and $\mathbf{F} = \{f_1(A_1)/A_1',\ldots,f_p(A_p)/A_p'\}$ as described above.

Applying this definition to the table EMPL we get the following minimal requirements (shown graphically for project $P1$ in figure 2):

$$\mathcal{G}^{MR}[P][sum(H)/SH][T](\langle\text{EMPL}\rangle,\{P/\text{P1}, SH/sum(\{\!\{H/300, H/400, H/300\}\!\})\},\{1,2,3\})$$

$$\mathcal{G}^{MR}[P][sum(H)/SH][T](\langle\text{EMPL}\rangle,\{P/\text{P1}, SH/sum(\{\!\{H/400, H/300\}\!\})\},\{4,5,6\})$$

$$\mathcal{G}^{MR}[P][sum(H)/SH][T](\langle\text{EMPL}\rangle,\{P/\text{P2}, SH/sum(\{\!\{H/200, H/600\}\!\})\},\{2,3\})$$

$$\mathcal{G}^{MR}[P][sum(H)/SH][T](\langle\text{EMPL}\rangle,\{P/\text{P2}, SH/sum(\{\!\{H/600\}\!\})\},\{4,5\})$$

The second minimal requirement actually covers what will turn out to be two output tuples. This is so because the two output tuples have identical unadjusted non-timestamp attribute values. In particular, the non-timestamp attribute values of these preliminary output tuples are taken directly from the input tuples, without any modifications. When we adjust the attribute values based on the relevant argument tuples and the attribute characteristics, we shall see that two output tuples result.

{N/Jan,...,P/P1,H/300,T/[1,3]}

{N/Tom,...,P/P1,H/400,T/[1,4]}

{N/Tom,...,P/P1,H/400,T/[5,6]}

Input tuples from EMPL

{N/Ann,...,P/P1,H/300,T/[1,6]}

1   2   3

{P/P1,SH/sum({{H/300,H/400,H/300}})} 4   5   6

Output time points E

and output tuples $\tilde{z}$
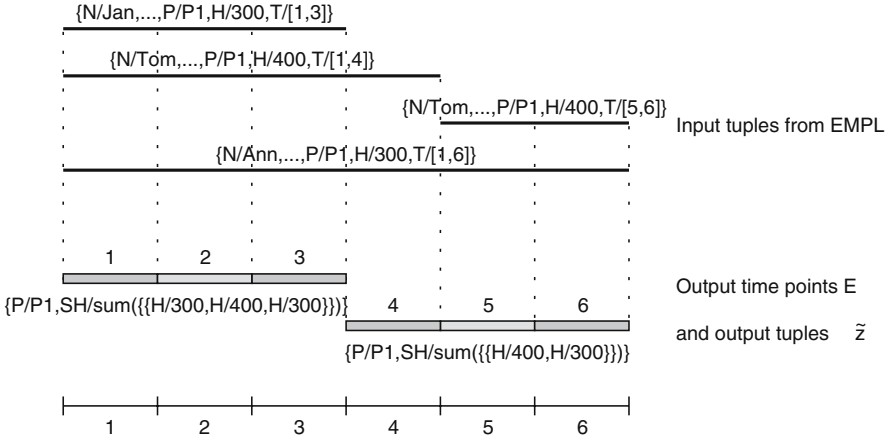
{P/P1,SH/sum({{H/400,H/300}})}

1   2   3   4   5   6

Figure 2. Graphical representation of the minimal requirements.

The minimal requirements specify the output tuples modulo the grouping of time points into interval timestamps and the transformation of malleable attribute values according to the timestamps that will be associated with the output tuples.

6.2. Required argument tuples

Having associated output tuples and time points, the next step is to trace output tuples back to input tuples. In general, an output tuple is completely determined by a small subset of tuples from the input relations, e.g., a result tuple of a binary join is completely determined by two input tuples, one from each of the two argument relations. This provides the information how to group time points into timestamps.

Definition 6.2 (*Required Argument Tuples, RAT*). Let $\vec{r} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be argument relations with timestamp attribute T, let $\mathcal{O}$ be an l-ary temporal operator, let I be a time interval, and let $\tilde{z}$ be a set of unadjusted attribute/value pairs (of a result tuple). The *required argument tuples*, $\vec{s}$, for $\tilde{z}$ and I are a minimal subset of the argument relations, $\vec{s} \subseteq \vec{r}$, that completely determine $\tilde{z}$:

$$\mathcal{O}^{RAT}[T](\vec{r}, \tilde{z}, I, \vec{s}) \text{ iff}$$
$$\exists E(\mathcal{O}^{MR}[T](\vec{r}, \tilde{z}, E) \wedge I \subseteq E) \wedge$$
$$\exists E'(\mathcal{O}^{MR}[T](\vec{s}, \tilde{z}, E') \wedge I \subseteq E' \wedge \vec{s} \subseteq \vec{r}) \wedge$$
$$\forall \vec{s}'', E''(\vec{s}'' \subset \vec{s} \wedge \mathcal{O}^{MR}[T](\vec{s}'', \tilde{z}, E'') \Rightarrow I \not\subseteq E'')$$

This definition splits the timestamp E into all possible (convex) sets, i.e., time intervals $I \subseteq E$, and it isolates for each such time interval and corresponding output

tuple $\tilde{z}$ the minimal set of input tuples, $\vec{\mathbf{s}} \subseteq \vec{\mathbf{r}}$, that completely determine this specific output. In other words, the result tuple $\tilde{z} \cup \{T/I\}$ can be determined entirely from the required argument tuples $\vec{\mathbf{s}}$, but not from any proper subset of them.

Example 6.2 (Temporal Aggregation, RAT). We continue our example by determining the required argument tuples for the aggregation query. We consider each MR and generate for each timestamp included in the temporal extent of the MR a corresponding required argument tuple. From the minimal requirement with the associated set of times $\{4, 5, 6\}$, we generate six required argument tuples, including the following ones for project P1:

$$\mathcal{G}^{RAT}[P][sum(H)/SH][T](\langle \text{EMPL} \rangle,$$
$$\{P/\text{P1}, SH/sum(\{\!\{H/400, H/300\}\!\})\},$$
$$[4, 6],$$
$$\langle\{\{N/\text{Tom}, \dots, P/\text{P1}, H/400, T/[1, 4]\},$$
$$\{N/\text{Tom}, \dots, P/\text{P1}, H/400, T/[5, 6]\},$$
$$\{N/\text{Ann}, \dots, P/\text{P1}, H/300, T/[1, 6]\}\}\rangle)$$

$$\mathcal{G}^{RAT}[P][sum(H)/SH][T](\langle \text{EMPL} \rangle,$$
$$\{P/\text{P1}, SH/sum(\{\!\{H/400, H/300\}\!\})\},$$
$$[4, 4],$$
$$\langle\{\{N/\text{Tom}, \dots, P/\text{P1}, H/400, T/[1, 4]\},$$
$$\{N/\text{Ann}, \dots, P/\text{P1}, H/300, T/[1, 6]\}\}\rangle)$$

$$\mathcal{G}^{RAT}[P][sum(H)/SH][T](\langle \text{EMPL} \rangle,$$
$$\{P/\text{P1}, SH/sum(\{\!\{H/400, H/300\}\!\})\},$$
$$[5, 6],$$
$$\langle\{\{N/\text{Tom}, \dots, P/\text{P1}, H/400, T/[5, 6]\},$$
$$\{N/\text{Ann}, \dots, P/\text{P1}, H/300, T/[1, 6]\}\}\rangle)$$

Figure 3 shows a graphical representation of these RATs. The first RAT states that in order to get output tuple $\{P/\text{P1}, SH/sum(\{\!\{H/400, H/300\}\!\})\}$ over time interval $[4, 6]$, precisely the three input tuples specified above are required. For the same MR, the RATs for the time intervals $[5, 5]$ and $[6, 6]$ are the same as for $[5, 6]$, and the RAT for $[4, 5]$ is the same as for $[4, 6]$.
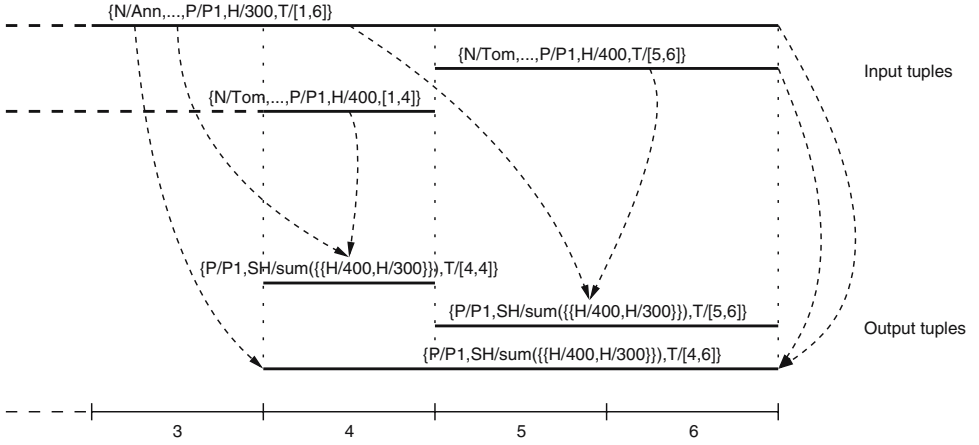
{N/Ann,...,P/P1,H/300,T/[1,6]}

{N/Tom,...,P/P1,H/400,T/[5,6]}

Input tuples

{N/Tom,...,P/P1,H/400,[1,4]}

{P/P1,SH/sum({{H/400,H/300}}),T/[4,4]}

{P/P1,SH/sum({{H/400,H/300}}),T/[5,6]}

Output tuples

{P/P1,SH/sum({{H/400,H/300}}),T/[4,6]}

3   4   5   6

Figure 3. Graphical representation of selected RATs.

6.3. Maximal output tuples

The final step is to identify those required argument tuples that are maximal in terms of temporal extent and at the same time respect the grouping of time points in the timestamps of the input tuples. Along with the grouping of time points into timestamps, we adjust the attribute values of the output tuples to fit their timestamps.

We use the following abbreviations for vectors to keep the definition of the maximal output tuples syntactically simple:

$$\overline{I''} = \langle I''_1, \ldots, I''_q \rangle$$

$$\overline{I''} \subset I = I''_1 \subset I \wedge \ldots \wedge I''_q \subset I$$

$$\cup \overline{I''} = I''_1 \cup \ldots \cup I''_q$$

$$\overline{\vec{\mathbf{s}}''} = \langle \vec{\mathbf{s}}''_1, \ldots, \vec{\mathbf{s}}''_q \rangle$$

$$\overline{\vec{\mathbf{s}}''} \subset \vec{\mathbf{s}} = \vec{\mathbf{s}}''_1 \subset \vec{\mathbf{s}} \wedge \ldots \wedge \vec{\mathbf{s}}''_q \subset \vec{\mathbf{s}}$$

$$\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, \overline{I''}, \overline{\vec{\mathbf{s}}''}) = \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I''_1, \vec{\mathbf{s}}''_1) \wedge \ldots \wedge \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I''_q, \vec{\mathbf{s}}''_q)$$

Definition 6.3 (*Maximal Output Tuples*). Let $\vec{\mathbf{r}} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be argument relations with timestamp attribute T, let $\vec{\mathbf{s}} = \langle \mathbf{s}_1, \ldots, \mathbf{s}_l \rangle$ with $\vec{\mathbf{s}} \subseteq \vec{\mathbf{r}}$, let $\mathcal{O}$ be an l-ary temporal operator, let z be a set of attribute/value pairs, let I be a time interval, and let C be attribute characteristics. Then $z \cup \{T/I\}$ is a *maximal output tuple* if I cannot be

enlarged without changing the required argument tuples and if no proper subsets of $\vec{s}$ qualify as required argument tuples for $\tilde{z}$ over sub-intervals of I:

$\mathcal{O}^{MOT}[T][C](\vec{\mathbf{r}}, z, I, \vec{s})$ iff

$$\exists \tilde{z}(\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I, \vec{s}) \wedge z = adj(\tilde{z}, I, \vec{s}, C) \wedge \tag{1}$$

$$\neg\exists I'(I' \supset I \wedge \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I', \vec{s})) \wedge \tag{2}$$

$$\neg\exists \overline{I''}, \overline{\vec{s}''}(\overline{I''} \subset I \wedge I = \cup\overline{I''} \wedge \overline{\vec{s}''} \subset \vec{s} \wedge \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, \overline{I''}, \overline{\vec{s}''}))) \tag{3}$$

Condition 1 selects a specific required argument tuple and adjusts the attribute values in $\tilde{z}$ according to time interval I. Condition 2 ensures that I is a maximal time interval for the specific RAT, $\vec{s}$. Condition 3 avoids coalescing of snapshot-equivalent tuples. In other words, it respects the grouping of time points in the input tuples by propagating this grouping to the output tuples.

Example 6.3 (Temporal aggregation, MOT). For the calculation of the MOTs we need the characteristics of the attributes. Assume the malleable characteristic for attribute Hours and the constant characteristic for all other attributes. We then get the following MOTs:

$$\mathcal{G}^{MOT}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle \text{EMPL} \rangle, \{P/P1, SH/750\}, [1,3], \langle \ldots \rangle)$$

$$\mathcal{G}^{MOT}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle \text{EMPL} \rangle, \{P/P1, SH/150\}, [4,4], \langle \ldots \rangle)$$

$$\mathcal{G}^{MOT}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle \text{EMPL} \rangle, \{P/P1, SH/300\}, [5,6], \langle \ldots \rangle)$$

$$\mathcal{G}^{MOT}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle \text{EMPL} \rangle, \{P/P2, SH/500\}, [2,3], \langle \ldots \rangle)$$

$$\mathcal{G}^{MOT}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle \text{EMPL} \rangle, \{P/P2, SH/300\}, [4,5], \langle \ldots \rangle)$$

The RAT for project P1 over time interval $[4, 6]$ (resulting from the first MR) is not a MOT because the third condition of the definition is violated: There exist two RATs for the same output values for sub-intervals of $[4, 6]$, namely one for $[4, 4]$ and one for $[5, 6]$. And these two RATs qualify as MOTs. Similarly, we can show that the RAT that yields $\{P/P1, SH/750\}$ is the only MOT to cover the interval $[1, 3]$. No other required argument tuple over any sub-interval of $[1, 3]$ qualifies as a MOT due to the second condition of the above definition.

Definition 6.4 (*Characteristics-Enabled Temporal Operator*). Let $\vec{\mathbf{r}} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be argument relations with timestamp attribute T, let x and z be sets of attribute/value pairs, and let C be attribute characteristics. Operator $\mathcal{O}$ is a *characteristics-enabled*

temporal operator iff the result tuples coincide with the maximal output tuples, i.e., iff for all temporal argument relations, $\vec{r}$, the following holds:

$$x \in \mathcal{O}[T][C](\vec{r}) \Leftrightarrow \exists z, I, \vec{s}(\mathcal{O}^{MOT}[T][C](\vec{r}, z, I, \vec{s}) \wedge x = z \cup \{T/I\})$$

Example 6.4 (Characteristics-Enabled Temporal Operator). The result tuples of the temporal aggregation query are the following:

$$\begin{aligned}
\mathcal{G}[P][sum(H)/SH][T][P{:}c, H{:}m](\langle\text{EMPL}\rangle) = \{&\{P/\text{P1}, SH/750, T/[1,3]\}, \\
&\{P/\text{P1}, SH/150, T/[4,4]\}, \\
&\{P/\text{P1}, SH/300, T/[5,6]\}, \\
&\{P/\text{P2}, SH/500, T/[2,3]\}, \\
&\{P/\text{P2}, SH/300, T/[4,5]\}\}
\end{aligned}$$

7. Properties of the algebraic framework

In this section we prove the key properties of our algebraic framework, which carry over to concrete algebraic operators defined in terms of their minimal requirements. Our algebraic framework is an extension of point-based temporal algebras. It extends the point-based semantics by propagating the grouping of time points from the input tuples to the output tuples. In the following two theorems we show that this grouping of the time points from the minimal requirements into time intervals neither introduces nor drops time points from the output tuples.

Theorem 7.1 (*Soundness*). Let $\mathcal{O}$ be an l-ary temporal operator in our framework, $\vec{r} = \langle r_1, \ldots, r_l \rangle$ be the argument relations with timestamp attribute T, z and $\tilde{z}$ be output tuples, E be a set of time points, C be attribute characteristics, and I be a time interval. The maximal output tuples are *sound* with respect to the minimal requirements, i.e., the maximal output tuples include only time points that are included in the minimum requirements:

$$\forall t, \vec{r}, \vec{s}, C, z, I(\mathcal{O}^{MOT}[T][C](\vec{r}, z, I, \vec{s}) \wedge t \in I \Rightarrow \exists E, \tilde{z}(\mathcal{O}^{MR}[T](\vec{r}, \tilde{z}, E) \wedge t \in E))$$

Proof. The proof traces the generation of maximal output tuples back to the minimal requirements. By definition of maximal output tuples, a MOT, $\mathcal{O}^{MOT}[T][C](\vec{r}, z, I, \vec{s})$, is derived from a RAT with the same time interval I, i.e., $\mathcal{O}^{RAT}[T](\vec{r}, \tilde{z}, I, \vec{s})$. By definition of required argument tuples, RATs group time points from the minimal requirements into time intervals, which does not introduce new time points. The two steps together prove the theorem. $\square$

Theorem 7.2 (*Completeness*). Let $\mathcal{O}$ be an *l*-ary temporal operator in our framework, $\vec{\mathbf{r}} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be the argument relations with timestamp attribute T, z and $\tilde{z}$ be output tuples, E be a set of time points, C be attribute characteristics, and I be a time interval. The maximal output tuples are *complete* with respect to the minimal requirements, i.e., the maximal output tuples cover all time points from the minimal requirements:

$$\forall t, \vec{\mathbf{r}}, C, \tilde{z}, E(\mathcal{O}^{MR}[T](\vec{\mathbf{r}}, \tilde{z}, E) \wedge t \in E \Rightarrow \exists I, z, \vec{\mathbf{s}}(\mathcal{O}^{MOT}[T][C](\vec{\mathbf{r}}, z, I, \vec{\mathbf{s}}) \wedge t \in I))$$

Proof.   The definition of required argument tuples splits the temporal extension E of MRs into all possible sub-intervals (groupings of convex sets of time points) and generates a corresponding RAT. For each MR we therefore get one or more RATs, $\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I_1, \vec{\mathbf{s}}_1), \ldots, \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I_p, \vec{\mathbf{s}}_p)$. This is graphically shown in figure 4. Since $I_1, \ldots, I_p$ are all possible sub-intervals of E, all time points of E are covered by the RATs.

To show that the RATs that qualify as MOTs cover all time points we start with the RATs over the maximal time intervals. If each of these RATs is also a MOT, all time points are covered, and we are done. Otherwise, at least one of these RATs, say $\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I_j, \vec{\mathbf{s}}_j)$, does not qualify as MOT, and we will show that the time points of I_j are covered by other MOTs. Since in this case condition 3 of the MOT definition is violated, there exist RATs, $\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I_{j_1}, \vec{\mathbf{s}}_{j_1}), \ldots, \mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I_{j_q}, \vec{\mathbf{s}}_{j_q})$ such that $I_{j_1} \subset I_j, \ldots, I_{j_q} \subset I_j$ and $\vec{\mathbf{s}}_{j_1} \subset \vec{\mathbf{s}}_j, \ldots, \vec{\mathbf{s}}_{j_q} \subset \vec{\mathbf{s}}_j$ and all I_{j_i} together cover all time points of I_j, i.e., $I_j = I_{j_1} \cup \ldots \cup I_{j_q}$. If each of these RATs qualifies as MOT, all time points, $t \in I_j$, are covered, and we are done. Otherwise, either condition 3 or condition 2 of the MOT definition is violated. In the former case, we can again split at least one of the RATs into several ones, and we proceed as above. In the latter case, at least one of the RATs can with the same input tuples, $\vec{\mathbf{s}}_{j_i}$, be extended to a larger time interval, I'_{j_i}, i.e., there exists a RAT $\mathcal{O}^{RAT}[T](\vec{\mathbf{r}}, \tilde{z}, I'_{j_i}, \vec{\mathbf{s}}_{j_i})$ with $I'_{j_i} \supset I_{j_i}$. We show that all time points in I'_{j_i} are covered, and consequently all time points in I_{j_i} are covered. In this process, the cardinality of the $\vec{\mathbf{s}}_j$ is strictly decreasing, while the union of the intervals
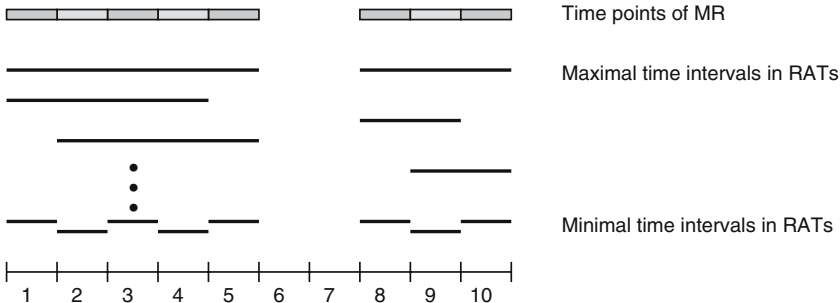

Figure 4. Grouping of time points from the minimal requirements.

I_{j_i} covers all time points. Eventually, this process terminates, when no further subsets of the input tuples $\vec{s}_j$ can be generated and the intervals I_{j_i} cannot be further extended. Then the corresponding RATs qualify as MOTs. □

These two theorems together show that all time points and only those time points from the minimal requirements are covered by the time intervals of the maximal output tuples. The next theorem shows that the time intervals of the maximal output tuples are unique and maximal.

Theorem 7.3 (*Uniqueness of Maximal Output Tuples*). Let $\mathcal{O}$ be an *l*-ary temporal operator in our framework, $\vec{r}^{min} = \langle \mathbf{r}_1, \ldots, \mathbf{r}_l \rangle$ be a minimal vector of input relations with timestamp attribute T to produce z over I (that is if one tuple is removed from $\vec{r}^{min}$ then the result is empty), z be an output tuple, I and I' be time intervals, C be attribute characteristics, and $Int(D^T)$ be the set of all intervals over the time domain D^T. The timestamp of the maximal output tuples is unique: It groups as many time points as possible, while preserving the grouping of the time points in the input tuples.

$$\forall \vec{r}^{min}, z, I, C(\mathcal{O}^{MOT}[T][C](\vec{r}^{min}, z, I, \vec{r}^{min}) \Rightarrow$$
$$\neg \exists I'(\mathcal{O}^{MOT}[T][C](\vec{r}^{min}, z, I', \vec{r}^{min}) \wedge I \neq I' \wedge I \cup I' \in Int(D^T)))$$

Proof. We have to show that, given a specific output tuple z and a set of input tuples, $\vec{r}^{min}$, for z, the time intervals I in the maximal output tuples are maximal.

Since $\vec{r}^{min}$ is minimal, there exists a RAT of the form $\mathcal{O}^{RAT}[T](\vec{r}^{min}, \tilde{z}, I, \vec{r}^{min})$ for the maximal output tuple on the left-hand side. From the definition of RAT we get also that for all strict subsets $\vec{r}^{min}$ the minimal requirements do not cover I.

Now we do a proof by contradiction. We assume that there exists a MOT of the form $\mathcal{O}^{MOT}[T][C](\vec{r}^{min}, z, I', \vec{r}^{min})$, where $I \neq I'$ and $I \cup I' \in Int(D^T)$. Then by the definition of MOTs there exists a RAT of the form $\mathcal{O}^{RAT}[T](\vec{r}^{min}, \tilde{z}, I', \vec{r}^{min})$. Next we do a case analysis for the interval I'. If $I' \supset I$ or $I' \subset I$, condition 2 of the MOT definition leads to a contradiction. Otherwise, I and I' are overlapping or adjacent intervals, and there exists a RAT of the form $\mathcal{O}^{RAT}[T](\vec{r}^{min}, \tilde{z}, I'', \vec{s}'')$, where $I'' = I \cup I'$ and $\vec{s}'' \subseteq \vec{r}^{min}$. If $\vec{s}'' = \vec{r}^{min}$, condition 2 of the MOT definition leads to a contradiction. Otherwise, $\vec{s}'' \subset \vec{r}^{min}$, and condition 3 of the RAT definition leads to a contradiction, i.e., since I is not covered by any of the subsets of $\vec{r}^{min}$, I'' cannot be covered as well.

8. A characteristics-enabled temporal algebra

In this section we instantiate the framework described in section 6 and define CETA, a *characteristics-enabled temporal algebra*. To define a new algebra all we have to do is to specify the minimal requirements of each operator in the algebra. In the following definition of CETA operators we adapt the traditional relational algebra operators selection, projection, union, difference, Cartesian product, and aggregation.

Definition 8.1 (*Minimal Requirements CETA Operators*). Let $\mathbf{r}, \mathbf{s}$ be temporal input relations with schemas $\{A_1, \ldots, A_k, T\}$ and $\{B_1, \ldots, B_l, T\}$, respectively, and timestamp attribute T, let z be a set of attribute/value pairs of an output tuple modulo the adjustment of the attribute values, E be a set of (not necessarily connected) time points, P be a predicate for the selection operator, $\mathbf{A}$ and $\mathbf{B}$ be abbreviations for the attributes $\{A_1, \ldots, A_k\}$ and $\{B_1, \ldots, B_l\}$, respectively, $\mathbf{X}$ be a subset of the attributes for the projection, $\mathbf{G}$ be a subset of the attributes on which to group the result, and $\mathbf{F} = \{f_1(A_1)/A_1', \ldots, f_p(A_p)/A_p'\}$ be aggregate functions f_i applied to attributes A_i, with the result values stored as attributes A_i'. The minimal requirements for the standard, temporal relational algebra operators selection, projection, union, difference, Cartesian product, and aggregation are defined as follows:

$\sigma^{MR}[P][T](\langle \mathbf{r} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge P(x) \wedge \tilde{z} = x[\mathbf{A}]))$

$\pi^{MR}[\mathbf{X}][T](\langle \mathbf{r} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge \tilde{z} = x[\mathbf{X}]))$
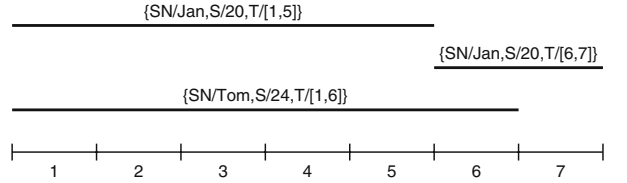
$\cup^{MR}[T](\langle \mathbf{r}, \mathbf{s} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x((x \in \mathbf{r} \vee x \in \mathbf{s}) \wedge t \in x.T \wedge \tilde{z} = x[\mathbf{A}]))$

$-^{MR}[T](\langle \mathbf{r}, \mathbf{s} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge \tilde{z} = x[\mathbf{A}] \wedge$

$\quad\quad\quad\quad \forall y(y \in \mathbf{s} \wedge y[\mathbf{B}] = x[\mathbf{A}] \Rightarrow t \notin y.T)))$

$\times^{MR}[T](\langle \mathbf{r}, \mathbf{s} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x, y(x \in \mathbf{r} \wedge y \in \mathbf{s} \wedge t \in x.T \wedge t \in y.T \wedge \tilde{z} = x[\mathbf{A}] \cup y[\mathbf{B}]))$

$\mathcal{G}^{MR}[\mathbf{G}][\mathbf{F}][T](\langle \mathbf{r} \rangle, \tilde{z}, E)$ iff

$\quad \forall t(t \in E \Leftrightarrow \exists x(x \in \mathbf{r} \wedge t \in x.T \wedge$

$\quad\quad\quad\quad av_1 = \{\!\{y[A_1] \mid y \in \mathbf{r} \wedge t \in y.T \wedge x[\mathbf{G}] = y[\mathbf{G}]\}\!\} \wedge \ldots \wedge$

$\quad\quad\quad\quad av_p = \{\!\{y[A_p] \mid y \in \mathbf{r} \wedge t \in y.T \wedge x[\mathbf{G}] = y[\mathbf{G}]\}\!\} \wedge$

$\quad\quad\quad\quad \tilde{z} = x[\mathbf{G}] \cup \{A_1'/f_1(av_1), \ldots, A_p'/f_p(av_p)\}))$

For the selection, projection, and union operators, the output tuples inherit the timestamps of the input tuples unchanged, no attribute values are modified, and only the grouping of time points has to be considered. For the difference, Cartesian product, and aggregation operators, the timestamps of the output tuples are usually different

SAL

SN	S	T
Jan	20	[1,5]
Jan	20	[6,7]
Tom	24	[1,6]

{SN/Jan,S/20,T/[1,5]}

{SN/Jan,S/20,T/[6,7]}

{SN/Tom,S/24,T/[1,6]}

BON

BN	B	T
Jan	12	[4,7]
Tom	4	[2,3]

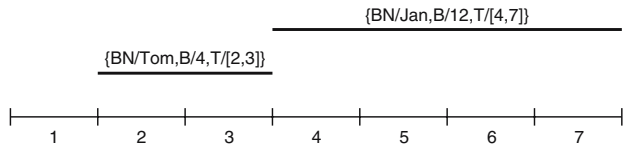{BN/Jan,B/12,T/[4,7]}

{BN/Tom,B/4,T/[2,3]}

Figure 5. The relations EMPL and SAL of the project database.

from those of the input tuples. Hence, in addition to the grouping of time points, the values of malleable attributes have to be adjusted accordingly.

For the following example we introduce two additional relations in the project database. The relation SAL stores name (*SN*) and salary (*S*) of employees, and the relation *Bon* stores name (*BN*) and bonus (*B*) of employees. The Salary and Bonus attributes have a *malleable* characteristics and are stored in KEuro. The other attributes are constant. Figure 5 shows these two relations.

Example 8.1 (Temporal Cartesian product). Consider the retrieval of all employees who in a certain period got a bonus that is equal to or more than half of their salary. If we are looking at the attribute values as they are stored in the database and consider them as constant values, we might believe that this is the case only for Jan during the time interval $[4, 7]$. Since Bonus and Salary attributes are malleable, however, we have to adjust these attribute values before applying the selection condition. Then we get a different result: Jan's bonus is greater than half of his salary in the time interval $[4, 5]$

{SN/Jan,S/20,T/[1,5]}

{SN/Jan,S/20,T/[6,7]}   Input tuples from SAL

{BN/Jan,B/12,T/[4,7]}   Input tuple from BON

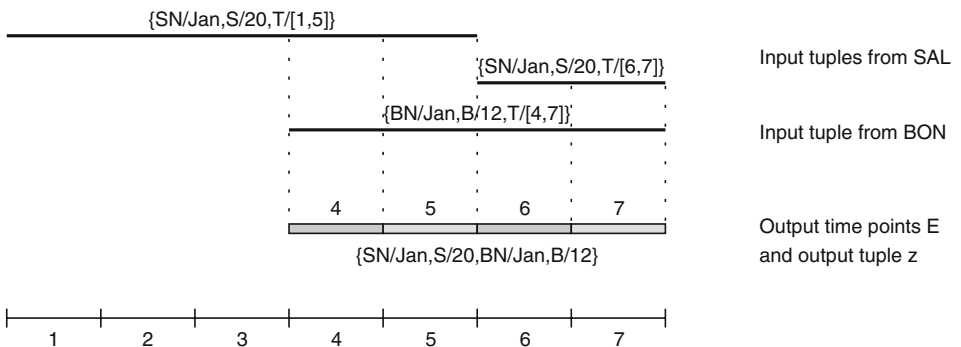{SN/Jan,S/20,BN/Jan,B/12}   Output time points E and output tuple z

Figure 6. Graphical representation of selected MRs.

(his salary is 8, and his bonus is 6), and Tom's bonus is half of his salary in the time interval $[2, 3]$ (his salary is 8, and his bonus is 4).

Formally, this query can be answered by computing first the temporal Cartesian product of the two relations and then applying the selection operator:

$$\sigma[P][T][C](\langle \times [T][C](\langle \text{SAL}, \text{BON} \rangle) \rangle)$$

The selection condition is $P = (SN = BN \wedge B \geq S/2)$, and the attribute characteristics is $C = \{SN:c, S:m, BN:c, CI:m\}$.

The minimal requirements for the temporal Cartesian product of *SAL* and *BON* are then given as follows:

$$\times^{MR}[T](\langle \text{SAL}, \text{BON} \rangle, \{SN/\text{Jan}, S/20, BN/\text{Jan}, B/12\}, \{4, 5, 6, 7\})$$

$$\times^{MR}[T](\langle \text{SAL}, \text{BON} \rangle, \{SN/\text{Jan}, S/20, BN/\text{Tom}, B/4\}, \{2, 3\})$$

$$\times^{MR}[T](\langle \text{SAL}, \text{BON} \rangle, \{SN/\text{Tom}, S/24, BN/\text{Jan}, B/12\}, \{4, 5, 6\})$$

$$\times^{MR}[T](\langle \text{SAL}, \text{BON} \rangle, \{SN/\text{Tom}, S/24, BN/\text{Tom}, B/4\}, \{2, 3\})$$

Note that the first minimal requirement combines two pairs of input tuples and coalesces the corresponding output tuples, since they are equivalent on the non-timestamp attributes values. This situation is graphically shown in figure 6.

The next step is to calculate the required argument tuples. For the first minimal requirement above we get a total of 11 RATs, one for each sub-interval of the
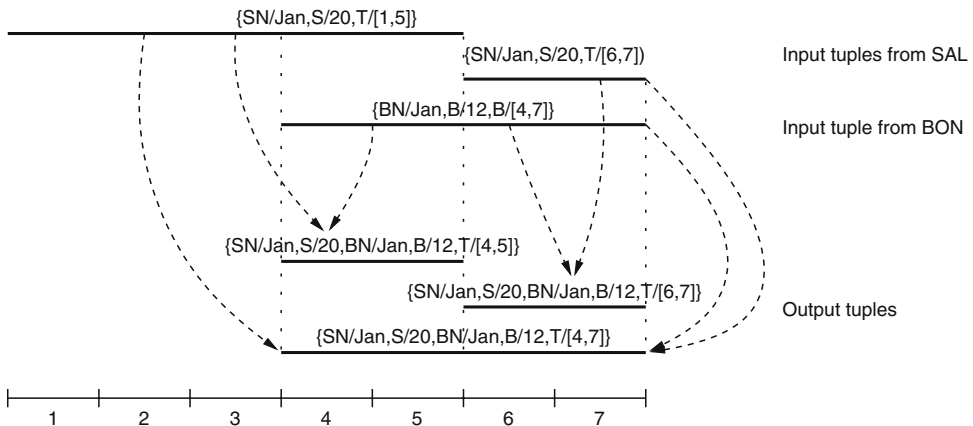

Figure 7. Graphical representation of selected RATs.

temporal extent $\{4, 5, 6, 7\}$. Three of these RATs are graphically illustrated in figure 7 and given below:

$\times^{RAT}[T](\langle\text{SAL}, \text{BON}\rangle,$

$\qquad \{SN/\text{Jan}, S/20, BN/\text{Jan}, B/12\},$

$\qquad [4, 7],$

$\qquad \langle\{\{SN/\text{Jan}, S/20, T/[1, 5]\}, \{SN/\text{Jan}, S/20, T/[6, 7]\}, \{BN/\text{Jan}, B/12, T/[4, 7]\}\}\rangle)$

$\times^{RAT}[T](\langle\text{SAL}, \text{BON}\rangle,$

$\qquad \{SN/\text{Jan}, S/20, BN/\text{Jan}, B/12\},$

$\qquad [4, 5],$

$\qquad \langle\{\{SN/\text{Jan}, S/20, T/[1, 5]\}, \{BN/\text{Jan}, B/12, T/[4, 7]\}\}\rangle)$

$\times^{RAT}[T](\langle\text{SAL}, \text{BON}\rangle,$

$\qquad \{SN/\text{Jan}, S/20, BN/\text{Jan}, B/12\},$

$\qquad [6, 7],$

$\qquad \langle\{\{SN/\text{Jan}, S/20, T/[6, 7]\}, \{BN/\text{Jan}, B/12, T/[4, 7]\}\}\rangle)$

The last step is to select those RATs that group as many time points as possible and at the same time respect the grouping of time points in the argument tuples. With Salary and Bonus attributes being malleable and the other attributes being constant, we get the following MOTs for the temporal Cartesian product:

$\times^{MOT}[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle\text{SAL}, \text{BON}\rangle, \{SN/\text{Jan}, S/8, BN/\text{Jan}, B/6\}, [4, 5], \langle\ldots\rangle)$

$\times^{MOT}[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle\text{SAL}, \text{BON}\rangle, \{SN/\text{Jan}, S/20, BN/\text{Jan}, B/6\}, [6, 7], \langle\ldots\rangle)$

$\times^{MOT}[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle\text{SAL}, \text{BON}\rangle, \{SN/\text{Jan}, S/8, BN/\text{Tom}, B/4\}, [2, 3], \langle\ldots\rangle)$

$\times^{MOT}[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle\text{SAL}, \text{BON}\rangle, \{SN/\text{Tom}, S/12, BN/\text{Jan}, B/9\}, [4, 6], \langle\ldots\rangle)$

$\times^{MOT}[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle\text{SAL}, \text{BON}\rangle, \{SN/\text{Tom}, S/8, BN/\text{Tom}, B/4\}, [2, 3], \langle\ldots\rangle)$

The RAT with the output tuple $\{SN/\text{Jan}, S/20, BN/\text{Jan}, B/6\}$ and time interval $[4, 7]$ is not a MOT, since condition 3 in the definition is violated, i.e., there exist RATs for the same output tuple over sub-intervals of $[4, 7]$ and they are produced by a subset of the input tuples. In fact, we have one RAT over the time interval $[4, 5]$ and one over the time interval $[6, 7]$. Both of them qualify as MOT and completely cover the timestamp of the first minimal requirement.

The result of the temporal Cartesian product is then given as follows:

$$\times[T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle \text{SAL}, \text{BON}\rangle) = \{\{SN/\text{Jan}, S/8, BN/\text{Jan}, B/6, T/[4,5]\},$$
$$\{SN/\text{Jan}, S/20, BN/\text{Jan}, B/6, T/[6,7]\},$$
$$\{SN/\text{Jan}, S/8, BN/\text{Tom}, B/4, T/[2,3]\},$$
$$\{SN/\text{Tom}, S/12, BN/\text{Jan}, B/9, T/[4,6]\},$$
$$\{SN/\text{Tom}, S/8, BN/\text{Tom}, B/4, T/[2,3]\}\}$$

We denote this intermediate result relation as SALBON. It is the input for the following temporal selection, which retrieves all tuples with corresponding names and where the bonus is equal or greater than half of the salary, i.e., $SN{=}BN \wedge B \geq S/2$. The minimal requirements for the temporal selection are given as

$$\sigma^{MR}[T](\langle \text{SALBON}\rangle, \{xSN/\text{Jan}, S/8, BN/\text{Jan}, B/6\}, \{4,5\})$$
$$\sigma^{MR}[T](\langle \text{SALBON}\rangle, \{SN/\text{Tom}, S/8, BN/\text{Tom}, B/4\}, \{2,3\})$$

We calculate the required argument tuples and the maximal output tuples in turn and get as a final result the expected output tuples

$$\sigma[SN = BN \wedge B \geq S/2][T][SN{:}c, S{:}m, BN{:}c, B{:}m](\langle \text{SALBON}\rangle) =$$
$$\{\{SN/\text{Jan}, S/8, BN/\text{Jan}, B/6, T/[4,5]\},$$
$$\{SN/\text{Tom}, S/8, BN/\text{Jan}, B/4, T/[2,3]\}\}$$

9. Summary

This paper offers a generalized algebraic framework that supports a variety of temporal semantics. Specifically, we show that it is possible to extend a strictly point-based semantics to support malleable and atomic attributes. This can be achieved without jeopardizing the degree of temporal support and without adding complexity to the basic definitions of algebraic operators. The framework may be carried over to SQL. For instance, the query in Example 8.1 could be expressed by the following SQL-query: SELECT * FROM SAL[$SN{:}c, S{:}m$], BON[$BN{:}c, CI{:}m$])WHERE $SN = BN$ AND $CI \geq S/2$.

In order to support malleable and atomic attribute characteristics, the grouping of time points into timestamps must be controlled precisely. A natural choice is to preserve the original groupings of time points. We use the lineage of result tuples to group time points according to their original grouping.

In future work, we aim to investigate efficient implementation in detail. Based on the MD join [2, 3], a general-purpose algebraic aggregate/join operator for multi-

dimensional data, we are currently developing an algorithm for the temporal aggregation defined in this paper.

Another promising direction for future research is to incorporate the algebraic framework proposed in this paper into one of the several temporal extensions to SQL that have been proposed [9]. The resulting, more flexible support for time-referenced data may constitute an important step towards built-in support for temporal database management in database management systems.

Acknowledgements

References

[1] T. Abraham and J.F. Roddick, Survey of spatio-temporal databases, Geoinformatica 3(1) (March 1999) 61–99.

[2] M.O. Akinde and M.H. Böhlen, The efficient computation of subqueries in complex OLAP queries, in: *Proceedings of the 19th International Conference on Data Engineering*, Bangalore, India (March 2003) pp. 163–174.

[3] M.O. Akinde, M.H. Böhlen, T. Johnson, L.V.S. Lakshmanan and D. Srivastava, Efficient OLAP query processing in distributed data warehouses, in: *Proceedings of the Eighth Conference on Extending Database Technology*, Prague, Czech Republic, (March 2002) pp. 336–353.

[4] J.F. Allen, Maintaining knowledge about temporal intervals, Communications of the ACM 16(11) (November 1983) 832–843.

[5] J.F. Allen, Towards a general theory of action and time, Artificial Intelligence 23(2) (July 1984) 123–154.

[6] I. Androutsopoulos, *Exploring Time, Tense and Aspect in Natural Language Database Interfaces* (Benjamins, 2002).

[7] J. Ben–Zvi, *The Time Relational Model*, PhD thesis, Computer Science Department, UCLA, 1982.

[8] M. Böhlen, J. Chomicki, R. Snodgrass and D. Toman, Querying TSQL2 databases with temporal logic, in: *Proceedings of the Fifth International Conference on Extending Database Technology*, Avignon, France (March 1996) pp. 325–341.

[9] M.H. Böhlen and C.S. Jensen, Temporal data model and query language concepts, in: *Encyclopedia of Information Systems*, Vol. 4 (Academic, 2003) pp. 437–453.

[10] M.H. Böhlen, C.S. Jensen and R.T. Snodgrass, Temporal statement modifiers, ACM Transactions on Database Systems 25(4) (December 2000) 407–456.

[11] M.H. Böhlen, R.T. Snodgrass and M.D. Soo, Coalescing in temporal databases, in: *Proceedings of the 22nd International Conference on Very Large Data Bases*, Mumbai (Bombay), India (February 1996) pp. 180–191.

[12] I.T. Bowman and D. Toman, Optimizing temporal queries: Efficient handling of duplicates, Data and Knowledge Engineering 44(2) (February 2003) 143–164.

[13] M.H. Böhlen, R. Busatto and C.S. Jensen, Point- versus interval-based temporal data models, in: *Proceedings of the 14th International Conference on Data Engineering*, Orlando, Florida (February 1998) pp. 192–200.

[14] E. Camossi, E. Bertino, M. Mesiti and G. Guerrini, Handling expiration of multigranular temporal objects, Journal of Logic and Computation 14(1) (February 2004) 23–50.

[15] J. Chomicki, D. Toman and M.H. Böhlen, Querying ATSQL databases with temporal logic, ACM Transactions on Database Systems 26(2) (June 2001) 145–178.

[16] Y. Cui, J. Widom and J.L. Wiener, Tracing the lineage of view data in a warehousing environment, ACM Transactions on Database Systems 25(2) (June 2000) 179–227.

[17] Y. Cui and J. Widom, Lineage tracing for general data warehouse transformations, in: *Proceedings of the 27th International Conference on Very Large Databases*, Rome, Italy (September 2001) pp. 471–480.

[18] O. Etzion, S. Jajodia and S. Sripada, eds., *Temporal Databases: Research and Practice*, Lecture Notes in Computer Science, Vol. 1399 (Springer, 1998).

[19] D. Gabbay and P. McBrien, Temporal logic & historical databases, in: *Proceedings of the 17th International Conference on Very Large Databases*, Barcelona, Catalonia, Spain (September 1991) pp. 423–430.

[20] S.K. Gadia, Weak temporal relations, in: *Proceedings of the Fifth ACM Symposium on Principles of Database Systems*, Cambridge, Massachusetts, USA (March 1986) pp. 70–77.

[21] S.K. Gadia, A homogeneous relational model and query languages for temporal databases, ACM Transactions on Database Systems 13(4) (December 1988) 418–448.

[22] C.S. Jensen and C.E. Dyreson, A consensus glossary of temporal database concepts – February 1998 Version, in: [O. Etzion et al., 1998] (1998) pp. 367–405.

[23] C.S. Jensen, M.D. Soo and R.T. Snodgrass, Unifying temporal models via a conceptual model, Information Systems 19(7) (1994) 513–547.

[24] R. Kowalski and M. Sergot, A logic-based calculus of events, New Generation Computing 4(1) (1986) 67–95.

[25] J. McCarthy and P.J. Hayes, Some philosophical problems from the standpoint of artificial intelligence, Machine Intelligence 4 (1969) 463–502.

[26] L.E. McKenzie and R.T. Snodgrass, Evaluation of relational algebras incorporating the time dimension in databases, ACM Computing Surveys 23(4) (December 1991) 501–543.

[27] R. Nelken, *Questions, Time, and Natural Language Interfaces to Temporal Databases*, PhD thesis, The Technion – Israel Institute of Technology, 2001.

[28] R.T. Snodgrass, *Developing Time-Oriented Database Applications in SQL*, (Morgan Kaufmann, San Francisco, California, 2000).

[29] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev and R. Snodgrass, eds., *Temporal Databases: Theory, Design, and Implementation* (Benjamin/Cummings, Redwood City, California, 1993).

[30] A.U. Tansel, Temporal relational data model, IEEE Transactions on Knowledge and Data Engineering 9(3) (May/June 1997) 464–479.

[31] P. Terenziani and R.T. Snodgrass, Reconciling point-based and interval-based semantics in temporal relational databases: A treatment of the telic/atelic distinction, IEEE Transactions on Knowledge and Data Engineering 16(5) (May 2004) 540–551.

[32] D. Toman, Point-based vs interval-based temporal query languages, in: *Proceedings of the 15th ACM Symposium on Principles of Database Systems*, Montreal, Canada (June 1996) pp. 58–67.

[33] A. Tuzhilin and J. Clifford, A temporal relational algebra as a basis for temporal relational completeness, in: *Proceedings of the 16th International Conference on Very Large Databases*, Brisbane, QLD, Australia (August 1990) pp. 13–23.

[34] Y. Wu, S. Jajodia and X.S. Wang, Temporal database bibliography update, in: [O. Etzion et al., 1998] (1998) pp. 338–366.