Point- Versus Interval-based Temporal Data Models

M. H. Böhlen R. Busatto C. S. Jensen

Department of Computer Science
Aalborg University
Fredrik Bajers Vej 7E, DK–9220 Aalborg Øst, Denmark
{boehlen, busatto, csj}@cs.auc.dk

Abstract

The association of timestamps with various data items such as tuples or attribute values is fundamental to the management of time-varying information. Using intervals in timestamps, as do most data models, leaves a data model with a variety of choices for giving a meaning to timestamps. Specifically, some such data models claim to be point-based while other data models claim to be interval-based. The meaning chosen for timestamps is important—it has a pervasive effect on most aspects of a data model, including database design, a variety of query language properties, and query processing techniques, e.g., the availability of query optimization opportunities.

This paper precisely defines the notions of point-based and interval-based temporal data models, thus providing a new, formal basis for characterizing temporal data models and obtaining new insights into the properties of their query languages. Queries in point-based models treat snapshot equivalent argument relations identically. This renders point-based models insensitive to coalescing. In contrast, queries in interval-based models give significance to the actual intervals used in the timestamps, thus generally treating non-identical, but possibly snapshot equivalent, relations differently. The paper identifies the notion of time-fragment preservation as the essential defining property of an interval-based data model.

1 Introduction

Temporal data models include timestamp attributes in their relation schemas and give special semantics to the values of these attributes in their query languages. Virtually all data models intended for practical use employ some form of intervals for their timestamp values. With a fine-granularity time domain such as, e.g., the default TIMESTAMP domain of SQL, it is generally impractical to record individually all the time points when some database fact was, is, or will be true.

Intervals may simply be employed for reasons of practicality, i.e., as syntactical shorthands for time points, as has been done in some data models (e.g., [12], [14, ch. 1]). Thus, terming a data model interval-based simply if it employs interval timestamps bears little real significance—it says little about the qualities of the data model. Rather the notions of point- and interval-based data models must be defined on a semantic level. The question then is what the real defining properties of point- and interval-based data models are—this paper provides an answer to this question.

To get a feel for the range of possible semantics of data models, related to points versus intervals, it is instructive to consider a simple example. We assume that the two tuple-timestamped relations r_1 and r_2, below, are given and consider possible definitions of the temporal difference of these two relations, $r_1 -^{\mathrm{T}} r_2$.

r_1:

A	TS
a	[1,10]
a	[11,20]
a	[21,30]

r_2:

A	TS
a	[5,15]

The results, R_1 through R_4, of four possible definitions of the difference operator are given next, and are discussed in turn.

R_1:

A	TS
a	[1,4]
a	[16,30]

R_3:

A	TS
a	[21,25]
a	[26,30]

R_2:

A	TS
a	[1,10]
a	[11,20]
a	[21,30]

R_4:

A	TS
a	[1,4]
a	[16,20]
a	[21,30]

The first result contains the times associated with value a in r_1 that are not associated with value a in r_2. This result is consistent with the perception that intervals are abbreviations for time points, and nothing more. Thus, the first definition has a point-based feel to it.

The first result may also be characterized as being *coalesced*. In coalescing, value-equivalent tuples (tuples with identical non-temporal attribute values) with adjacent or overlapping intervals are replaced by a single tuple with the same non-temporal attribute values and an interval that is the union of the intervals of the original tuples.

In contrast, result R_2 is far from being point-based in nature. This result contains all tuples in r_1 not in r_2. This definition of difference simply considers intervals as atomic values. Thus, it may be said to "respect" the actual intervals given to the tuples, and it is devoid of any flavor of point-based-ness. It may even be questioned whether this operator is temporal at all—it is simply the standard set-theoretical difference operator. Result R_3 returns tuples from r_1 with intervals that do not overlap with intervals of tuples in r_2; the intervals of qualifying tuples are nonetheless split into two. The utility of a temporal difference operator of this kind appears questionable.

The last result is similar to the first one: it also contains the times associated with value a in r_1 that are not associated with value a in r_2. Put precisely, R_1 and R_4 are snapshot equivalent [12]. The difference is that the second tuple in R_1 is "represented" by two tuples in R_4. In other words, R_1 is the coalesced version of R_4. The idea behind this definition is to be point-based while also trying to respect the intervals associated with the tuples in the argument relations.

It is our contention that R_1 and R_4 are results of point-based operations and that R_2 and R_4 are results of interval-based operations. The operation yielding result R_3 is thus neither point- nor interval-based.

So far, attempting to capture in general the defining properties of point-based and, in particular, interval-based data models has proven notoriously difficult and has led to much confusion. This paper gives meaningful and general definitions of what point- and interval-based data models are and thus contributes to clearing away the confusion. The definitions provide a foundation for deciding whether a model is point- or interval-based.

To the knowledge of the authors, no papers have previously been devoted to the issues addressed here. Rather, some papers that either define new data models or describe existing data models have made brief statements concerning the point- versus interval-based nature of data models.

For example, Snodgrass states that "A temporal query language should have a canonical model, in which relations are identical if and only if all of their snapshots are identical" [12, p. 288]. Chomicki states that "It is important to see that the data model of TQuel is point-based, not interval-based. Intervals serve only as a representational device. The truth values of facts are associated with points, not intervals" [5, p. 521] and that "…a model is point-based if facts are associated with single time points, interval-based – if they are associated with intervals (represented as pairs of points)" [6, p. x+7]. Finally, Toman states that "In this paper we establish the correspondence between the point- and interval-based views of temporal databases and the corresponding first-order temporal languages. This correspondence shows that all first-order queries can be conveniently asked using the point-based query language (…) and then mechanically translated to an interval-based query language…" [15, p. 59].

The literature reveals that different researchers perceive the notions of point-based and interval-based data models quite differently. In particular, the notion of interval-based data model remains to be given a formal definition, alongside a rigorous semantics.

In the next section, we further motivate the topic and explore the problem space. Section 3 introduces the notions of temporal data models and time domains, providing the basis for formally defining the notions of point-based and interval-based temporal data models in Sections 4 and 5. Section 6 informally discusses some of the aspects of the definitions, and Section 7 concludes and points to directions for future research.

2 Motivation and Problem Space

Before giving formal definitions of point- and interval-based models, this section explores the properties of the two kinds of models, showing that there are significant differences between them.

2.1 Semantics and Expressive Power

When asking queries on a temporal database, the results may vary depending on whether or not argument relations are coalesced. For example, this is the case for selections and projections that involve the argument timestamps. To see this, consider the two relations in Figure 1. The relation at the top is uncoalesced whereas the one at the bottom is the corresponding coalesced relation.

Employment

Name	Position	TS
Lars	programmer	[92/01/01,94/12/31]
Lars	programmer	[95/01/01,96/12/31]
Niels	accountant	[92/01/01,96/12/31]

Employment

Name	Position	TS
Lars	programmer	[92/01/01,96/12/31]
Niels	accountant	[92/01/01,96/12/31]

Figure 1: Uncoalesced and Corresponding Coalesced Relation Instance

Consider the uncoalesced Employment relation, which models job contracts in a company with temporary positions only. The query $\pi_{\mathrm{Name,\ START(TS)}}(\mathrm{Employment})$ returns three tuples because three contracts were signed (function START returns the start time of an interval). If the exact same query is evaluated over the coalesced instance, only two tuples are returned.

The example illustrates that there exist queries that can be asked over the uncoalesced instances, but not over the coalesced ones. For example, the coalesced instance of the employment relation does not reveal that Lars signed two contracts, let alone when he signed the second one.

On the other hand, it is impossible to come up with a query that can be answered over the coalesced, but not over the uncoalesced, instance. The reason for this is that it is possible to derive the coalesced relation instance from an uncoalesced one, e.g., using a regular SQL statement [4].

These considerations indicate that a model that is able to tell coalesced and uncoalesced relation instances apart, and in this sense is interval-based, is in some sense more powerful than a model that cannot tell them apart, i.e., a point-based model. Next, we explore this difference further.

2.2 Data Modeling

The relative expressiveness of data models that do or do not differentiate between coalesced and uncoalesced relation instances may also involve data modeling. It may be argued that if the database schema is designed appropriately, it is possible to answer the same queries using a coalesced model as can be answered by an uncoalesced model.

For example, if individual contracts are important, which is not unlikely, we can record their unique numbers in the Employment relation, as shown in Figure 2.

Employment

Name	Position	ContrId	TS
Lars	programmer	1091	[92/01/01,94/12/31]
Lars	programmer	2154	[95/01/01,96/12/31]
Niels	accountant	1095	[92/01/01,96/12/31]

Figure 2: Alternative Modeling of Employment

This way, it may be possible to “compensate” for the lack on uncoalesced relations in a point-based data model. It may be argued that it is quite natural that certain queries cannot be answered if they were not anticipated when the database was designed—this is true for any database.

Still, introducing additional attributes may sometimes have subtle drawbacks not experienced if the attributes could be omitted because the data model allowed uncoalesced relations. For example, we might introduce dependencies (contract numbers increase over time) or we might not be able to faithfully represent our mini-world (“new” follow-up contracts with the same contract number).

2.3 Query Processing and Query Optimization

The point- versus interval-basis of a query language also affects *query processing* and *query optimization*. For an interval-based language, care has to be taken that processing and optimization strategies respect the interval-based semantics, which can be quite complex. This severely restricts the possibilities to manipulate and transform intervals. In contrast, specific timestamps may be modified (as long as snapshot equivalence is preserved) in a point-based language, allowing the database system a choice of timestamps among several alternatives. This indicates that an interval-based language leaves less possibilities for query optimization and, thus, efficient evaluation strategies.

In favor of an interval-based language, it can be said that a point-based database system *must* guarantee that the result of queries do not depend on the specific choice of timestamp values. This guarantee is met by performing coalescing operations, which can be expensive [4]. While it is possible to sometimes eliminate coalescing during query optimization, there remain situations where coalescing has to be performed [13, ch. 27].

3 Temporal Data Models and Time Domains

A *relational data model* $\mathcal{M} = \langle \mathcal{D}, \mathcal{A} \rangle$ is composed of a set of data structures, $\mathcal{D}$, and a set $\mathcal{A}$ of algebraic operations defined on the data structures. A *temporal relational data model* is a relational data model that has *temporal relations* as the underlying data structure, and whose operators are all temporal.

Temporal relations include a temporal attribute. The exact denotation of this attribute is not important for the topic of this paper, but for simplicity we assume that it denotes the tuple’s valid time, i.e., when the information recorded by the remaining attributes of a tuple is true in the modeled reality. Tuples of temporal relations may therefore be put under the form $\langle x_1, \ldots, x_n \| ts \rangle$ or as $\langle \boldsymbol{x} \| ts \rangle$ when the number of attributes is immaterial. We term $x_1, \ldots, x_n$ the *non-temporal* (or *explicit*) attribute values, and ts is the (*tuple*) *timestamp*. A finite set of such relations may be referred to as the *timestamp representation* of a temporal database [1].

An operator is *temporal* if and only if it generates a temporal relation whenever applied to temporal relations.

When designing a temporal data model, an important and central aspect is the choice of appropriate timestamps of the database facts. *Time points* and *time intervals*, defined below, provide the most common choices. Intervals may be built from time points [2], [14, ch. 21].

Definition 3.1 (Time-point and Time-interval Domains) Let T be an infinite set.

1. $\mathcal{T}^p = \langle T, < \rangle$ is a *time point domain* over T iff $<$ defines a total order on T. Each element of T corresponds to a *time point* of $\mathcal{T}^p$.
2. A *time interval* I of $\mathcal{T}^p$ is any connected subset of $\mathcal{T}^p$, i.e., $(p_1 \in I \wedge p_2 \in I \wedge p_3 \in T \wedge p_1 < p_3 < p_2) \Rightarrow p_3 \in I$
3. $\mathcal{T}^i = \langle \mathcal{I}, \subset \rangle$ is the *time interval domain* over $\mathcal{T}^p$ iff $\mathcal{I}$ is the set of all time intervals of $\mathcal{T}^p$. Both $\mathcal{T}^p$ and $\mathcal{T}^i$ are *time* (or *temporal*) *domains* over T.

4. A *timestamp* over $\mathcal{T}^p$ is either a time point or a time interval of $\mathcal{T}^p$.

A temporal relation r whose tuples are all timestamped with either time points or time intervals of a time point domain $\mathcal{T}^p$ represents a *temporal relation over* $\mathcal{T}^p$. When the timestamps are points [intervals], r may be referred to as *point- [interval-] timestamped (temporal) relation* over $\mathcal{T}^p$. If $\mathcal{M} = \langle \mathcal{D}, \mathcal{A} \rangle$ is a temporal data model such that $\mathcal{D}$ is a set of temporal relations over $\mathcal{T}^p$, then $\mathcal{T}^p$ is the *time point domain* of $\mathcal{M}$.

In general, since time intervals are sets of time points, it is not always clear in what sense the usage of intervals as timestamps differs from the usage of points. To exemplify this, assume that the integers with the $<$ order is our time point domain. Then it seems reasonable to claim that the relations $r_1 = \{\langle a\|2\rangle, \langle a\|3\rangle, \langle a\|4\rangle\}$ and $r_2 = \{\langle a\|[2,4]\rangle\}$ have the same information contents, i.e., that $\langle a\rangle$ is valid at instants 2, 3, and 4, and nothing more. This assumption, nonetheless, is incorrect for r_2 because intervals in addition to being points also are uniquely delimited by *start* and *end points* (which may or may not be part of the interval). Hence, we would timestamp a tuple such as $\langle a\rangle$ with intervals if the end points bear some meaning, and use time points as timestamps if the notion of end points is meaningless.

Predicates and operations for points and intervals are described in almost all definitions of temporal data models [14] [13, ch. 10]. Some interval predicates and operators apply just to interval data models—their properties would make them meaningless in a point-based framework; for example, the operators *start* and *end* that retrieve the *initial* and *final* instants, respectively, of an interval could not be defined for a point-based database in the same way as described above (cf. Section 4).

The point timestamp representation of a temporal database is infeasible from the storage viewpoint for all but the simplest temporal relations, so intervals are used as an abbreviation for sets of points for practical reasons. For example, relation r_1 above may be represented by r_2. Whether an interval is an abbreviation for a set of points or not depends on the operators of the data model. Only if the point contents of the output of a temporal operator remain invariant for sets of argument relations with the same point contents, it is possible to consider intervals as abbreviations for sets of points. This property is more formally explored next.

4 Point-based Data Models

It would be easy to decide whether or not a data model is point- or interval-based if this could always be determined by inspecting the data type of the timestamps used. However, syntactic criteria are available only for simple point-timestamped relations[1]. The major difficulty concerns relations involving intervals as timestamps. This section defines the notion of a point-based data model.

In a point-based data model, two interval-timestamped relations that correspond to the same point-timestamped relation are considered equivalent, in the sense that they record the same information. The notion of *snapshot equivalence* [7, 9] formalizes this.

Definition 4.1 (Snapshot Equivalence) Let $\mathcal{T}^p = \langle T, < \rangle$ be a time point domain.

1. The *timeslice* operator τ_p for a time point $p \in T$ maps an interval-timestamped relation over $\mathcal{T}^p$ to a non-temporal one, and is defined as
$$\tau_p(r) = r' \text{ iff } \\ \forall \boldsymbol{x}(\exists I(\langle \boldsymbol{x}\|I\rangle \in r \wedge p \in I) \Leftrightarrow \langle \boldsymbol{x}\rangle \in r')$$

2. Two interval-timestamped relations over $\mathcal{T}^p$, r_1 and r_2, are *snapshot equivalent,* i.e., $r_1 \overset{p}{=} r_2$, iff
$$\forall p\,(p \in T \Rightarrow \tau_p(r_1) = \tau_p(r_2))$$

The notion of snapshot equivalence allows us to characterize operators that, when applied to snapshot equivalent relations, yield results that are also snapshot equivalent [9]. Such operators are faithful to the point-based nature of the timestamps of their argument relations, and we will use them to define point-based data models.

Definition 4.2 (Point-based Operator) Let $\mathcal{O}$ be a n-ary operator on interval-timestamped relations, and $\{r_1, \ldots, r_n\}$ and $\{r'_1, \ldots, r'_n\}$ be sets of interval-timestamped relations that satisfy the preconditions of $\mathcal{O}$. $\mathcal{O}$ is *point-based* iff it preserves snapshot equivalence, i.e., iff it satisfies the following property
$$r_1 \overset{p}{=} r'_1 \wedge \ldots \wedge r_n \overset{p}{=} r'_n \Rightarrow \mathcal{O}(r_1, \ldots, r_n) \overset{p}{=} \mathcal{O}(r'_1, \ldots, r'_n)$$

Example 4.1 The *temporal intersection natural join* ($\bowtie^t$) is a binary operator. Two argument tuples with identical explicit join attribute values contribute to the result if their timestamps overlap. Timestamps of result tuples are the intersections of the timestamps of argument tuples. Thus, if $r_1 = \{\langle a\|[2,5]\rangle, \langle a\|[7,11]\rangle\}$ and $r_2 = \{\langle a\|[3,9]\rangle\}$ then $r_1 \bowtie^t r_2 = \{\langle a\|[3,5]\rangle, \langle a\|[7,9]\rangle\}$. It can be shown that this operator preserves snapshot equivalence, hence it is point-based.

Example 4.2 The *coalescing operator* (coal) is a unary operator that merges value-equivalent tuples (tuples with mutually identical explicit attribute values) if the union of their timestamps is an interval. The merged tuple then

[1] In this sense, temporal logic can be also said to be point-based, as the temporal domain consists of points [5].

has this union as its new timestamp. Thus, if $r_1 = \{\langle a\|[2,5]\rangle, \langle a\|[6,11]\rangle\}$ then $\text{coal}(r_1) = \{\langle a\|[2,11]\rangle\}$. Like temporal intersection natural join, this operation is point-based because snapshot equivalent arguments will yield snapshot equivalent results: for snapshot-equivalent arguments, the result will always be the exact same, which is a trivial case of snapshot equivalence.

With the definition above, we are in a position to define point-based data models.

Definition 4.3 (Point-based Temporal Data Model) A temporal data model $\mathcal{M} = \langle \mathcal{D}, \mathcal{A}\rangle$ with time point domain $\mathcal{T}^p$ is *point-based* iff the following conditions are met.

1. $\mathcal{D}$ is entirely composed of interval-timestamped relations over $\mathcal{T}^p$, and
2. the operators of $\mathcal{A}$ are all point-based.

Lemma 4.1 A temporal data model $\mathcal{M} = \langle \mathcal{D}, \mathcal{A}\rangle$ is point-based iff, for every operator $\mathcal{O}$ of $\mathcal{A}$,

$$\mathcal{O}(r_1, \ldots, r_n) \overset{p}{=} \mathcal{O}(\text{coal}(r_1), \ldots, \text{coal}(r_n))$$

where $r_1, \ldots, r_n$ are relations of $\mathcal{D}$ that satisfy the preconditions of $\mathcal{O}$.

The lemma illustrates why the *start* and *end* functions mentioned in Section 3 cannot be defined in a point-based data model by considering individual intervals in isolation. The presence of, e.g., the tuple $\langle \boldsymbol{x}\|[a,b]\rangle$ in a point-based relation does not mean that a is truly a start point for $\boldsymbol{x}$, since the relation may contain other value-equivalent tuples that overlap with this interval. As a result, the computation of the above functions in a point-based data model requires that the argument relation first be coalesced. The definition of *start* could then be expressed as follows.

$$\langle \boldsymbol{x}\|I\rangle \in r \wedge \langle \boldsymbol{x}\|I'\rangle \in \text{coal}(r) \wedge I \subseteq I' \Rightarrow \\ \textit{start}(\langle \boldsymbol{x}\|I\rangle, r) = \textit{start}(I')$$

Finally, in a point-based data model, it holds true that intervals are nothing but abbreviations for sets of points. Hence, it is always possible to translate any interval-timestamped relation r into a corresponding point-timestamped relation r^p. The following relationship holds between the two relations.

$$\langle \boldsymbol{x}\|y\rangle \in r^p \quad \text{iff} \quad \exists I(y \in I \wedge \langle \boldsymbol{x}\|I\rangle \in r)$$

Algorithmically, r^p can be generated by simply replacing each tuple $\langle \boldsymbol{x}\|I\rangle$ in r by a tuple $\langle \boldsymbol{x}\|y\rangle$ for exactly each time point $y \in I$.

5 Interval-based Data Models

To the best of our knowledge, no good definition of an interval-based data model exists. Purely syntactical definitions are inappropriate, and defining any data model that is not point-based as interval-based is also unsatisfactory.

The distinction between point-based and non-point-based models is orthogonal to what distinguishes interval-based data models from those that are not interval-based. For example, an operator of an interval-based data model needs not be point-based, but there are operators of such models that are point-based.

To define the notion of an interval-based data model, we distinguish between the algebraic operators that are *timestamp-preserving* and those that are *timestamp-transforming*. The former operators are unproblematic and easily qualify for the interval-based status. The latter operators must satisfy additional properties to qualify for the interval-based status.

Specifically, when intervals are adopted as timestamps, there will normally be several ways of timestamping result relations. In such cases, the argument interval timestamps should be preserved *as intactly as possible*, i.e., maximally respected, if an operator is to be regarded as interval-based. This means that, whenever an operation requires the modification of an argument interval timestamp, the resulting interval should be the one (or one of the choices) that maximally takes the argument interval into consideration. Alternatively, this property could be stated as *the largest possible fragments of the argument interval timestamps should be preserved in the result*. The objective of the remainder of this section is to formalize the above notions.

5.1 Interval-based Requirements

The first step is to define the notion of *minimum requirements* for an algebraic temporal operator. Informally, the minimum requirements define the set of time points that the timestamps of the result of a temporal operator must include. Explanations follow the formal definition.

Definition 5.1 (Minimum Requirements) Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{A}\rangle$ be a temporal data model with time point domain $\mathcal{T}^p$, where $\mathcal{D}$ is a set of interval-timestamped relations. The *minimum requirements* for a n-ary temporal operator is a formula of the form $\phi(r_1, \ldots, r_n, \boldsymbol{x}, A)$ where

1. the timestamp A associated with a result tuple that satisfies the requirements for the argument relations $r_1, \ldots, r_n \in \mathcal{D}$ is a (not necessarily connected) set of time points of $\mathcal{T}^p$, and
2. $(\phi(r_1, \ldots, r_n, \boldsymbol{x}, A_1) \wedge \phi(r_1, \ldots, r_n, \boldsymbol{x}, A_2)) \Rightarrow A_1 = A_2$.

Clearly, ϕ must also include the preconditions for the specified operator. From the second condition of the definition, it follows that, for each sequence of explicit attributes values $\boldsymbol{x}$ of the result, there is one and only one associated set of instants A, since the minimum requirements do not impose any partition on this set of instants (i.e., ϕ defines a partial, parameterized function $f_{r_1,\ldots,r_n}$ such that $f_{r_1,\ldots,r_n}(\boldsymbol{x}) = A$). Thus, formula ϕ specifies a family of operators, in the sense that A may be (usually) split into a list of intervals in several distinct ways. We use A to emphasize that we are dealing with generic sets of instants, i.e., temporal elements, rather than with intervals only.

The next step towards defining interval-based data models is to characterize the set of *relevant argument tuples* for each particular result tuple, as defined by the minimum requirements ϕ for an operator. A set of argument tuples S is relevant for a particular result tuple $\langle \boldsymbol{x}, A\rangle$ iff both $\boldsymbol{x}$ and A can be entirely determined from S, but not from any proper subset of S.

Definition 5.2 (Relevant Argument Tuples) Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{A}\rangle$ be a temporal data model with time point domain $\mathcal{T}^p$, where $\mathcal{D}$ is a set of interval-timestamped relations. Let ϕ denote the minimum requirements for a n-ary temporal operator, $r_1, \ldots, r_n$ be temporal relations of $\mathcal{D}$, A be a set of time points of $\mathcal{T}^p$, and $\boldsymbol{x}$ be a finite sequence of attribute values. S is a set of *relevant argument tuples* w.r.t. ϕ for the argument relations $r_1, \ldots, r_n$ and the result tuple $\langle \boldsymbol{x}, A\rangle$, i.e., relevant$(\boldsymbol{x}, A, S, \phi, r_1, \ldots, r_n)$ iff

$$\begin{array}{l}
\phi(r_1, \ldots, r_n, \boldsymbol{x}, A) \wedge \\
\exists r_1', \ldots, r_n'(\\
\quad r_1' \subseteq r_1 \wedge \ldots \wedge r_n' \subseteq r_n \wedge \\
\quad S = \bigcup_{i=1}^{n} r_i' \wedge \phi(r_1', \ldots, r_n', \boldsymbol{x}, A) \wedge \\
\quad \forall r_1'', \ldots, r_n''((r_1'' \subseteq r_1' \wedge \ldots \wedge r_n'' \subseteq r_n' \wedge \\
\qquad \bigcup_{i=1}^{n} r_i'' \subset S) \Rightarrow \neg\phi(r_1'', \ldots, r_n'', \boldsymbol{x}, A)).
\end{array}$$

Note that S does not necessarily correspond to a relation, since $r_1, \ldots, r_n$ may not be union compatible. Also, it is necessary to require that the result $\langle \boldsymbol{x}, A\rangle$ satisfy the minimum requirements for both the original argument relations and their restricted forms because the one does not imply the other.

Example 5.1 Assume a temporal difference operator. Let the integers with the $<$ order be the underlying time point domain, and ϕ^D denote the minimum requirements for this operator. Assume $r_1 = \{\langle a\|[2,10]\rangle\}$ and $r_2 = \{\langle a\|[1,4]\rangle, \langle a\|[8,11]\rangle, \langle a\|[12,17]\rangle, \langle b\|[2,6]\rangle\}$. If $r_1'' = r_1' = r_1$, $r_2' = \{\langle a\|[1,4]\rangle\}$, and $r_2'' = \{\langle a\|[1,4]\rangle, \langle a\|[8,11]\rangle\}$, then

$$\begin{array}{l}
\phi^D(r_1, r_2, \langle a\rangle, \{5,6,7\}) \\
\phi^D(r_1', r_2', \langle a\rangle, \{5,6,7,8,9,10\}) \\
\phi^D(r_1'', r_2'', \langle a\rangle, \{5,6,7\})
\end{array}$$

The set of relevant argument tuples for the result tuple is $S'' = r_1'' \cup r_2''$. No proper subset of S'' satisfies the minimum requirements.

The next example illustrates that set of relevant argument tuples is not uniquely defined.

Example 5.2 The minimum requirements for temporal difference are given by the formula ϕ,

$$\begin{array}{l}
\textit{union\_compatible}(r_1, r_2) \wedge \\
\forall p(\exists I_1(\langle \boldsymbol{x}\|I_1\rangle \in r_1 \wedge p \in I_1 \wedge \\
\qquad \forall I_2(\langle \boldsymbol{x}\|I_2\rangle \in r_2 \Rightarrow p \notin I_2)) \Leftrightarrow p \in A)
\end{array}$$

Assume $r_1 = \{\langle a\|[4,8]\rangle, \langle a\|[1,6]\rangle, \langle a\|[7,10]\rangle\}$ and $r_2 = \{\langle a\|[1,3]\rangle, \langle a\|[9,10]\rangle\}$. Then $A = \{4,5,6,7,8\}$ satisfies the minimum requirements for temporal difference for the explicit attribute $\langle a\rangle$ and the input relations r_1 and r_2. Concerning the relevant argument tuples of r_1 and r_2 for $\langle a\rangle$, there are two sets that satisfy the definition, $S_1 = \{\langle a\|[4,8]\rangle\} \cup \{\ \}$ and $S_2 = \{\langle a\|[1,6]\rangle, \langle a\|[7,10]\rangle\} \cup \{\langle a\|[1,3]\rangle, \langle a\|[9,10]\rangle\}$. Note that $r_1 \cup r_2$ does not qualify as a set of relevant argument tuples, since there are subrelations of r_1 and r_2 whose union also satisfies ϕ.

A final auxiliary concept concerns the notion of *maximal interval partition*, which determines the decomposition of a set of time points into the least possible number of non-overlapping intervals:

Definition 5.3 (Maximal Interval Partition) Let A be a (doubly bounded) set of time points. The *maximal interval partition* for A is a sequence of intervals $I_1, \ldots, I_n$ such that

1. $I_1 \cup \ldots \cup I_n = A$,
2. $I_i \cap I_j = \emptyset$, with $i \neq j$, $1 \leq i \leq n, 1 \leq j \leq n$, and
3. any other interval partition $I_1', \ldots, I_p'$ for A satisfying the above two conditions is such that $p > n$.

Using the concepts developed so far, we can now define the notion of *interval-based operator*.

Definition 5.4 (Interval-based Operator) Let $\mathcal{M} = \langle \mathcal{D}, \mathcal{A}\rangle$ be a temporal data model with time point domain $\mathcal{T}^p$, where $\mathcal{D}$ is a set of interval-timestamped relations. Let ϕ denote the minimum requirements for a n-ary temporal operator. A temporal operator $\mathcal{O} \in \mathcal{A}$ that satisfies ϕ is *interval-based* iff for any argument temporal relations $r_1, \ldots, r_n \in \mathcal{D}$, the following holds.

1. $\exists I(\langle \boldsymbol{x}\|I\rangle \in \mathcal{O}(r_1, \ldots, r_n)) \Rightarrow$
   $\phi(r_1, \ldots, r_n, \boldsymbol{x}, \bigcup \pi_{ts}(\sigma_{expl=\boldsymbol{x}}(\mathcal{O}(r_1, \ldots, r_n))))$

2. If (a) $\phi(r_1, \ldots, r_n, \boldsymbol{x}, A)$, (b) $\boldsymbol{S} = \bigcup_{i=1}^{p} S_i$, where *relevant*$(\boldsymbol{x}, A, S_i, \phi, r_1, \ldots, r_n)$, for all i, $1 \leq i \leq p$, (c) $\langle \boldsymbol{y} || I \rangle \in \boldsymbol{S}$, (d) $A \cap I \neq \emptyset$, and (e) $I_1, \ldots, I_m$ correspond to the maximal interval partition for $A \cap I$, then $\langle \boldsymbol{x} || I_1 \rangle, \ldots, \langle \boldsymbol{x} || I_m \rangle \in \mathcal{O}(r_1, \ldots, r_n)$.

The first condition of Definition 5.4 ensures that, for each group of tuples of $\mathcal{O}(r_1, \ldots, r_n)$ whose explicit attributes values are $\boldsymbol{x}$, the union of all timestamps of such tuples is identical to the set of time points identified by the minimum requirements for the same argument, i.e., specified and resulting timestamps must be extensionally identical for $\boldsymbol{x}$. The second condition ensures the preservation of the relevant input intervals in the result, whenever possible, under the form of overlapping fragments[2].

The preservation of argument timestamp fragments in an interval-based operator is illustrated in Figure 3. For the relevant argument intervals and the corresponding hypothetical set of output time points A given in Figure 3, two sets of output intervals are given. The first one is built on top of a minimal decomposition strategy, where each interval of the result must be contained in one of the relevant input intervals, but no output intervals may overlap, even when there is overlapping at the input level. The second solution is the only one that satisfies all conditions of Definition 5.4: for each relevant input interval, its intersection with A, represented under the form of (maximal) intervals, is included in the output. In particular, note that Definition 5.4 does not allow the result intervals to be chopped or merged.

Example 5.3 The minimum requirements formula $\phi(r_1, r_2, \boldsymbol{x}, A)$ for *temporal union* is

$$\text{union\_compatible}(r_1, r_2) \wedge$$
$$\forall p (\exists I ((\langle \boldsymbol{x} || I \rangle \in r_1 \vee \langle \boldsymbol{x} || I \rangle \in r_2) \wedge p \in I) \Leftrightarrow p \in A) \; .$$

The set S of relevant argument tuples for a particular result $\langle \boldsymbol{x}, A \rangle$ can be computed from the definition of S and the minimum requirements for temporal union; for the above case, it can be shown that S is made of the tuples of r_1 and r_2 that are value-equivalent to $\langle \boldsymbol{x}, A \rangle$.

Figure 4 shows several alternative definitions for a temporal union operator that satisfy ϕ. Alternatives 1, 2, and 3 represent operators that do not preserve argument fragments, since the relevant argument timestamps are not properly represented in any of them. Alternative 4 is the only one that contains all the required (fragments of) intervals. All four alternatives represent satisfactory solutions with respect to snapshot-equivalence preservation. As a fragment-preserving operator, the temporal union operator is superfluous—it amounts to its standard set-theoretical counterpart.

[2]The usage of *temporal elements*, defined e.g., in [8], would lead to a single result tuple of the form $\langle \boldsymbol{x} || I_1 \cup \ldots \cup I_m \rangle$.

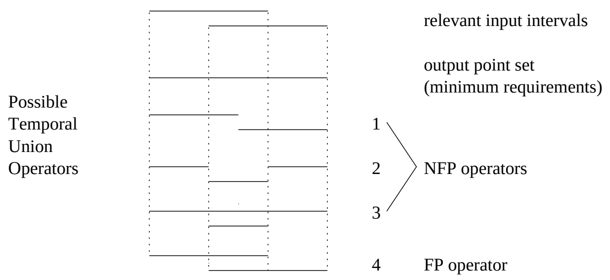

Figure 4: Alternative Solutions for Temporal Union

Example 5.4 The temporal join operator from Example 4.1 is not only point-based, but also interval-based, because its intersecting of argument intervals satisfies fragment preservation.

Example 5.5 A selection operator that constrains the timestamp and returns a temporal relation is interval-based, but not point-based. For example, consider $\sigma_{I \text{ BEFORE } [7,9]}(r)$ with $r = \{\langle a || [2,5] \rangle, \langle a || [6,11] \rangle\}$. The result is $\{\langle a || [2,5] \rangle\}$. Using $r' = \{\langle a || [2,11] \rangle\}$, which is snapshot equivalent to r, as the argument would yield an empty result. Since selection does not preserve snapshot-equivalence, it is not point-based. On the other hand, it is easy to see that selection is interval-based. All result intervals are identical to argument intervals and, therefore, fragments are trivially preserved.

The definition of an interval-based temporal data model follows.

Definition 5.5 (Interval-based Temporal Data Model) A temporal data model $\mathcal{M} = \langle \mathcal{D}, \mathcal{A} \rangle$ with time point domain $\mathcal{T}^p$ is *interval-based* iff the following conditions are met.

1. $\mathcal{D}$ is entirely composed of interval-timestamped relations over $\mathcal{T}^p$, and
2. the operators of $\mathcal{A}$ are all interval-based.

6 Discussion

In this section, we discuss properties of point- and interval-based data models. We start by discussing the scope of our approach. Then we look at mixed data models, i.e., models that are neither point- nor interval-based, and finally we evaluate representative temporal data models.

6.1 Scope of our Approach

The scope of the definitions of point- and interval-based operators are temporal extensions of relational algebra operators, i.e., temporal variants of $\sigma, \pi, \cup, \backslash, \times$, and their derivatives. These are the basic operators of a temporal algebra, and they have been investigated in almost all temporal data models. Our definitions can be used to evaluate and classify

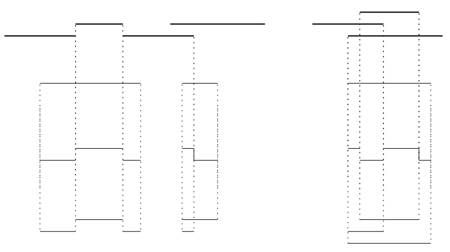

Figure 3: Hypothetical Interval-based Operator

these operators and, thus, models. However, the definitions are applicable to all possible temporal operators. For example, we have illustrated the application to coalescing. To further illustrate and delimit our definitions, we discuss possible extensions.

Enlarging the set of interval-based operators. There exist candidate operators that do not preserve fragments, but that we still might want to classify as interval-based. Two such operators are described in Examples 6.1 and 6.2, below.

Example 6.1 Assume a *regular time-shift operator* o^{rts} that returns all tuples of the argument relation with the timestamps being shifted one time unit to the right. Assume $r_1 = \{\langle a||[2,5]\rangle, \langle a||[6,11]\rangle\}$ and $r_2 = \{\langle a||[2,11]\rangle\}$. Then $o^{rts}(r_1) = \{\langle a||[3,6]\rangle, \langle a||[7,12]\rangle\}$ and $o^{rts}(r_2) = \{\langle a||[3,12]\rangle\}$.

Example 6.2 Assume an *irregular time-shift operator* o^{its} that, for any tuple of the argument relation, shifts the start time by two units to the right and the end time by one unit to the right. Assume $r_1 = \{\langle a||[2,5]\rangle, \langle a||[6,11]\rangle\}$ and $r_2 = \{\langle a||[2,11]\rangle\}$. Then $o^{its}(r_1) = \{\langle a||[4,6]\rangle, \langle a||[8,12]\rangle\}$ and $o^{its}(r_2) = \{\langle a||[4,12]\rangle\}$.

The time-shift operators are faithful to their argument intervals in the sense that they dislocate each single interval present in the input relation. Therefore, they could be classified as interval-based. Our definition is stricter in this respect and classifies both as non-interval-based because argument interval fragments are not preserved.

Note that the regular time-shift operator is point-based whereas the irregular one is not. Specifically, a regular time-shift moves all intervals of tuples in an argument relation. All basic properties of the argument, e.g., the length and the relative positions of the tuples' time intervals, are preserved. The irregular time-shift operator, on the other hand, changes some basic properties of the input set. Because start and end times are shifted differently, it may be that intervals that meet or overlap in the argument do not meet or overlap in the result ($o^{its}(r_1)$ illustrates this). This clearly leads to violations of snapshot-equivalence preservation.

Narrowing the set of interval-based operators. It can make sense to make the definition of interval-based more restrictive. Our current definition of interval-based comprises all operators and models that do not interpret timestamps in any special way. Examples are relational operators in SQL-92 [11] (extended with a period data type), relational operations in IXSQL [10], and non-sequenced operations in ATSQL [3]. In all these cases, standard relational operators are applied to timestamps. Such an approach is trivially faithful to argument intervals—the operators have no special temporal semantics. One can argue that such operators do not provide (enhanced) temporal support. This is an argument in favor of not classifying such operators as interval-based, or even temporal.

6.2 Mixed Data Models

With point- and interval-based being orthogonal properties we get four classes of operators. Specifically, coalescing is point- but not interval-based, temporal selection is interval- but not point-based, temporal intersection join is point- and interval-based, and the irregular time-shift operator is neither point- nor interval-based.

From Definitions 4.3 and 5.5, it follows that there exist temporal data models that are neither point- nor interval-based. In practice, we expect many models to have point-based and interval-based operations. For example, both IXSQL and ATSQL have a core set of interval-based operations. However, both models also include a point-based coalescing operation. We term such models *mixed*.

6.3 An Evaluation of Temporal Data Models

In this section we touch upon a few popular temporal data models and evaluate them according to our criteria. Note that we only consider proper temporal algebraic operators, i.e., operators that take temporal relations as arguments and return a temporal relation (cf. Section 3).

SQL-92 [11] SQL-92 (extended with an interval data type) is based on the relational algebra and treats intervals as atomic values without any special temporal semantics. This means that all operators are time-fragment preserving. Therefore, SQL-92 is an interval-based data model. It also follows that SQL-92 is not point-based.

IXSQL [10] IXSQL operators are timestamp-preserving because they inherit the standard SQL-92 semantics. In addition, IXSQL provides normalize and unnormalize operations in order to convert between time points and intervals. These special operations are point-based, but not interval-based: snapshot equivalence is preserved, but interval fragments are not. Thus, IXSQL is a mixed data model.

TSQL2 [13] Unlike the two previous models, TSQL2 employs a temporal algebra that gives a special meaning to timestamps. It was one of the design goals of TSQL2 to make the format of timestamps irrelevant. This is achieved by enforcing a canonical representation based on temporal elements. Thus, TSQL2 is clearly not interval-based. On the other hand, all operators preserve snapshot equivalence because they are defined over the canonical representation of a database. This makes TSQL2 a point-based data model.

ATSQL [3] ATSQL introduces sequenced and nonsequenced statements together with corresponding algebras. Nonsequenced statements provide the power of regular SQL-92 statements and are, like SQL-92 and IXSQL statements, interval-based. Sequenced statements are also interval-based. In addition, most sequenced statements are point-based. Coalescing is available to enforce a canonical representation of snapshot-equivalent relations. Thus, while clearly interval-based in nature ATSQL has also a non-interval-based operation (coalescing) which makes it a mixed data model.

7 Conclusions and Research Directions

We have provided definitions for point- and interval-based operators and data models. Point-based operators are defined by employing the notion of snapshot equivalence. The notion of an interval-based operator is much more elusive. The essence is to define what it means for an operator to maximally preserve, or respect, the timestamps of argument tuples when timestamping result tuples. Based on the notion of fragment preservation, we have provided a definition of interval-based operators. Throughout the paper, we have explored the properties of point-based and interval-based data models.

Several promising directions for further research may be identified. First, the mapping of instances in one temporal data model to instances in another has already been explored in a point-based framework [9], but this mapping has not been explored in the context of interval-based data models.

Next, we have argued that interval-based data models are in some sense more expressive than point-based data models. The added expressiveness comes at the cost of more complicated operators that are harder to define and, more importantly, understand and use. A continued exploration of the relative merits of the two kinds of models is in order, as are studies of possible refinements of the definition of interval-based data models.

Finally, we have illustrated that interval-based operators can be quite different in nature. In particular, some of them are timestamp preserving while others are timestamp transforming. It would be interesting to exploit these two notions to obtain a more detailed classification of data models.

Acknowledgements

This research was supported in part by the Danish Technical Research Council through grant 9700780 and by the CHOROCHRONOS project, funded by the European Commission DG XII Science, Research and Development, as a Networks Activity of the Training and Mobility of Researchers Programme, contract no. FMRX-CT96-0056.

References

[1] S. Abiteboul, L. Herr, and J. Van den Bussche. Temporal Connectives Versus Explicit Timestamps in Temporal Query Languages. *Recent Advances in Temporal Databases*, pages 43–57. Springer-Verlag, 1995.

[2] J. van Benthem. *The Logic of Time – A Model-Theoretic Investigation into the Varieties of Temporal Ontology and Temporal Discourse*. Kluwer, 1991.

[3] M. H. Böhlen and C. S. Jensen. Seamless Integration of Time into SQL. TR R-96-2049, Aalborg University, Department of Computer, Fredrik Bajers Vej 7, DK-9220 Aalborg, Dec 1996.

[4] M. H. Böhlen, R. T. Snodgrass, and M. D. Soo. Coalescing in Temporal Databases. *Proc. of the 22nd VLDB Conf.*, pages 180–191. Morgan Kaufmann, Sep 1996.

[5] J. Chomicki. Temporal Query Languages: a Survey. *Proc. of the First Intern. Conf. on Temporal Logic*, pages 506–534, 1994.

[6] J. Chomicki. Temporal Query Languages: a Survey. 1995. Submitted to IEEE TKDE (available via URL `http://www.cis.ksu.edu/~chomicki`).

[7] S. K. Gadia. Weak Temporal Relations. *Proc. of the 5th PODS Symposium*, 1986.

[8] S. K. Gadia. A Homogeneous Relational Model and Query Languages for Temporal Databases. *ACM TODS*, 13 (4):418–448, 1988.

[9] C. Jensen, M. Soo, and R. T. Snodgrass. Unifying Temporal Models via a Conceptual Model. *Information Systems*, 19(7):513–547, 1994.

[10] N. A. Lorentzos and Y. G. Mitsopoulos. SQL Extension for Interval Data. *IEEE TKDE*, 9(3):480–499, May 1997.

[11] J. Melton and A. R. Simon. *Understanding the new SQL: A Complete Guide*. Morgan Kaufmann, 1993.

[12] R. T. Snodgrass. The Temporal Query Language TQuel. *ACM TODS*, 12(2):247–298, Jun 1987.

[13] R. T. Snodgrass. *The TSQL2 Temporal Query Language*. Kluwer Academic, 1995.

[14] A. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. T. Snodgrass. *Temporal Databases: Theory, Design, and Implementation*. Benjamin/Cummings, 1993.

[15] D. Toman. Point-based vs Interval-based Temporal Query Languages. *Proc. of the 15th ACM PODS Symposium*, pages 58–67, Jun 1996.