From Data Streams to Fields: Extending Stream Data Models with Field Data Types

Qinghan Liang, Silvia Nittel (✉), and Torsten Hahmann

Spatial Informatics, School of Computing and Information Science
University of Maine, Orono, USA
qinghan.liang@maine.edu, nittel@spatial.maine.edu,
torsten.hahmann@maine.edu

Abstract. With ubiquitous live sensors and sensor networks, increasingly large numbers of individual sensors are deployed in physical space. Sensor data *streams* are a fundamentally novel mechanism to create and deliver observations to information systems, enabling us to represent spatio-temporal continuous phenomena such as radiation accidents, pollen distributions, or toxic plumes almost as instantaneously as they happen in the real world. While data stream engines (DSE) are available to process high-throughput updates, DSE support for phenomena that are continuous in both space and time is not available. This places the burden of handling any tasks related to the integration of potentially very large sets of concurrent sensor streams into higher-level abstractions on the user. In this paper, we propose a formal extension to stream data model languages based on the concept of fields to support high-level abstractions of continuous ST phenomena that are known to the DSE, and therefore, can be supported through queries and processing optimization. The proposed field data types are formalized in a data model language independent way using second order signatures. We formalize both the set of supported field types are as well as the embedding into stream data model languages.

Keywords: data streams, sensor data streams, data stream engines, fields, field data types

1 Introduction

Motivation With ubiquitous live sensors and wireless sensor networks, increasingly large numbers of individual sensors are deployed in physical space such as urban environments [25], forests [10], for earthquake monitoring [12], or precision agriculture. Such large numbers of live streaming sensors enable us to collect observations that are sufficiently *dense* in both space and time to now represent *continuous* change in space *and* time in near real-time. Examples for spatio-temporal continuous phenomena are, for instance, pollen distributions, toxic plumes, radiation accidents, or soil moisture distributions.

Sensor data *streams* are a fundamentally novel mechanism to create and deliver observations to information systems, enabling us to represent entities, processes and events in information systems almost instantaneously as they happen in the real world [5, 24]. While a stream seems similar to a time series, that is, it consists of an ordered set of time-stamp records, a stream is a significantly different *programming* abstraction: it is an *unbounded* multiset of elements, continuously producing records as time advances. New updates are constantly pushed into queries during query execution, resulting in real-time query answers. *Data stream engines* (DSEs) have been designed as a high-throughput alternative to database management systems (DBMS). Today, open source and commercial DSE such as Apache Spark [1] achieve query performance over streams with a throughput of >1 Million updates/s; in comparison, DBSs are limited to 500 updates/s [26]. Similar to DBSs, DSEs provide data model and query languages, which make it easier for users to program applications, enabling them to define data schemata and SQL-type queries over data streams. The core concept of stream data models is that of a *stream* (as opposed to a relation). Stream query languages contain *continuous queries* as well as *query windows* as evaluation contexts over streams. While DBS and DSE are separate technologies, stream data models and query languages are formally integrated with the relational algebra to guarantee compatibility between both technologies [4].

DSEs make it feasible to monitor and analyze phenomena that are continuous in space and time while delivering real-time answers to queries. However, today's stream data model languages provide concepts to represent individual sensor data streams. For instance, point geometry types are available to create stream tuples with sensor location attributes [3]. Such low-level support enables only modeling individual sensor data streams, and it is the responsibility of the user and application code to handle any tasks related to the *integration* of potentially very large sets of concurrent sensor streams into *higher-level abstractions* such as spatio-temporally continuous phenomena. Currently, the bulk of programming and understanding of continuous phenomena is pushed into the application code, and this code has to be re-implemented for each application again and again. We believe that DSEs should provide a generic, flexible and high-level abstraction for continuous spatio-temporal (ST) phenomena. The complexity of integrating individual sensor streams into a higher-level representation on-the-fly should be hidden from users while still allowing them to configure the mapping between sensor streams and a continuous phenomenon. Spatio-temporal continuous phenomena need to be supported on the level of both data model language and query execution support in DSEs.

Transforming large sets of individual sensor data streams into the high-level representation of a ST continuous phenomenon is not an easy feat. A representation of a continuous phenomenon must always be an approximation based on captured samples. When dealing with up to 1,000 concurrent streams, it is unlikely that their updates and sampling rates are synchronized. Instead, each stream might have its own sampling frequency. To create a snapshot of a spatially continuous phenomenon at a desired time stamp across all sensors requires resampling or interpolating existing streams. Further, the types of analyses of such phenomena will vary but still run concurrently. Thus, locking the representation of a continuous phenomenon into a single, fixed resolution of cells, triangulations or contour lines is severely limiting.

Contributions In this paper, we propose an extension to stream data models that is based on *field* data types, which directly tap into streamed data rather than accessing stored data. Fields have long been proposed as a unifying information system abstraction for continuous phenomena [8, 21] and are well understood on an ontological level [15, 16], but are still uncommon as actual information system interfaces and implementations [22, 13, 7]. We believe that a field data type is the most promising approach to handling the complexities of sensor stream processing for continuous phenomena.

While the extension of DSEs with fields on the data model, query language and processing level is a complex task, we have investigated feasibility aspects of processing fields on-the-fly based on massive sensor data streams in previous work [28, 27]. In this paper, we focus solely on introducing the *formal framework* of extending data stream models with field types. We introduce our proposed field data types for stream data models on an abstract level using second order signatures [17], which allow us to define the field types independently from specific data models and programming languages. The *abstract model* is continuous in space and time and used to formalize the universe of field types that can be constructed. The abstract model is complemented by a *discrete model* that is amendable to direct implementation. It grounds the purely continuous view of the abstract model in the realities of discrete computer systems. It does so by relating the abstract field types to computational concepts such as tuples, streams, windows, and interpolation functions. Our field stream data model with its dynamic spatial, temporal, and spatio-temporal fields serves as the foundation for sophisticated spatio-temporal data analyses, bridging the gap between raw point-based sensor streams and the detection of trends and events.

The following two Sections 2 and 3 present the background of our work and the related work. The abstract model for the field types is presented in Sec. 4, while Sec. 5 describes the discrete model suitable for embedding in a data model language. Sec. 6 offers our conclusions and identifies future work.

2 Background

2.1 Data Streams

The core concept of relational database systems (DBS) is the *relation*, which is a set of persistently stored data tuples. Each relational query is performed over a stored relation in its *entirety*. On the other hand, data stream engines (DSEs) are concerned with frequently updated data and, thus, have *streams* as their core concept. A stream is an unbounded sequence of tuples that arrive in some temporal order, with multiple tuples likely arriving out of order or simultaneously. More formally, streams are defined as “unbounded, append-only multisets of time-stamped tuples” [4]. Unbounded means in this context that we cannot predict how many tuples will arrive at any point of time that is in the future; at any time point in the past up until now, this multiset is finite. Tuples have either explicit timestamps, which are added at the data source and denote the real-world event time, or implicit timestamps, which are added to tuples when they arrive at the DSE [5]. Stream data models support linear, dense and discrete time models [26].

While each tuple in a stream is assigned a timestamp, not every tuple necessarily has a spatial attribute. For that reason, we introduce a special type of stream – *a spatio-temporal stream* – as a stream wherein every tuple also has a spatial attribute. It is based on the following concept of a spatio-temporal relation.

Definition 1. *Let S be a discrete set of point locations, T be a discrete bounded set of timestamps, and V_1,...,V_n be value domains. Then a ST-Relation is a relation* $\Re_{ST} \subseteq$

$S \times T \times V_1 \times \cdots \times V_n$ such that for every (s_i,t_i) $\in S \times T$ there exists at most one $(v_1 \times \cdots \times v_n) \in V_1 \times \cdots \times V_n$ such that $(s_i, t_i, v_1, \ldots, v_n) \in \Re_{ST}$.

Definition 2. *Let S be a discrete set of point locations, T be a discrete bounded set of timestamps, and V_1,...,V_n be value domains (e.g., values of measurements). Let further T_S be a set of timestamps with initial and last timepoints $t_{S1}, t_{Sf} \in T_S$ such that for every $t_{Si} \in T_S$, $t_{S1} \leq t_{Si} \leq t_{Sf}$. Then, an ST-Stream is a function $S_{ST} : T_S \rightarrow \Re$ such that for a fixed spatial domain S, a fixed temporal domain T, and a fixed value domain $V_1 \times \cdots \times V_n$, all $\mathcal{R} \in \Re$ are ST-Relations with $\mathcal{R} \subseteq S \times T \times V_1 \times \cdots \times V_n$.*

A ST-Stream can be the time series of observations from a single sensor or from an entire geosensor network. Viewing an ST-Stream as all streamed updates of geographically and thematically related sensors is a powerful abstraction that allows us to reason about complex spatio-temporal events that take place within the space and time observed by the streams. In our model, ST-streams encompass streams that consists of the updates from only a single sensor as well as the aggregation of streams that form continuous dynamic ST-Fields.

2.2 Fields as Formal Foundation for Streaming ST Continuous Phenomena

The term *field* (more precisely *geo-spatial fields*) is widely used to describe entities in physical space that are continuous in space and time and lack boundaries. A field implies a continuous quality of an observed phenomenon in the real-world, such as temperature, that is present at every point in time and space on Earth (and beyond). A plethora of different computer representations have been developed for continuous phenomena, and it is common practice to pick a representation that matches best the data capture method in order to represent a particular phenomenon. For instance, temperature may be represented by measurements at irregularly distributed sample points, foliage as regular grid cells, or pollen density as isolines, with more complex representations possible. While implementations of these computer representations enable many specialized analytical operations in geographic information systems, the diversity and lack of coherence of representation seriously impedes integration and cross-cutting analyses [8, 21]. This problem is exacerbated if spatio-temporal fields are considered, and multi-faceted integration over space, time, multiple parameters, and various sensor platforms is the objective.

Over the last two decades, geo-spatial fields have been mathematically formalized, e.g., in [16, 9]. In the context of our field stream data model, we are mainly concerned with fields that are approximated based on observations, so-called *sampled fields* (in contrast to equation fields). We present a mathematical definition of the following types of fields that are the underlying basic components of the field model: a *spatial field*, which addresses continuity over space, a *temporal field*, representing continuity over time, and a *spatio-temporal field*, representing continuity over both space and time. We are particularly interested in the important notion of a *sampled field* because it addresses the spatial domain of fields, that is, it represents either sensor locations or the continuous quality of the phenomenon.

A **spatial field** is defined as follows: Given a spatial domain S and an attribute domain V, a spatial field F_S over S is a computable, possibly partial, function f :

$S \rightarrow V$ from spatial locations in S to attribute values in V. The spatial locations in S are points, and a subset of S are sensor locations. More details on spatial domains for fields are discussed in [15]. The attribute domain V can be finite or infinite, discrete or continuous, numeric or symbolic.

A **temporal field** F_t is a function $f : T \rightarrow V$ from the time points in T to attribute values in V. A temporal field represents the change of an attribute over time. The attribute could be a location (e.g., a trajectory of a moving object), a sensor measurement, or a stock price. For observed temporal fields, the time domain is both linear and dense, that is, time advances linearly. For a temporal field based on a sensor stream, the time domain is discrete and isomorphic to $\mathbb{N}$; each natural number corresponds to a non-decomposable unit of time which is the sampling time. For a continuous temporal field, the time domain is isomorphic to $\mathbb{R}$ since the real-world phenomenon exists without temporal 'gaps'.

Spatio-temporal fields have both spatial dimensions as well as a temporal dimension. For example, in a spatio-temporal temperature field over a lake, each value $f(s, t)$ identifies the temperature at location s and time t. Galton [16] defines a spatio-temporal field as a function $f : S \times T \rightarrow V$ that assigns each pair of a spatial location in S and a time point in T an attribute value in V. Again, the temporal domain T is linear and dense, and can be discrete or continuous. Choosing a snapshot point of view and a discrete temporal domain, the spatio-temporal field is equivalently defined as a function $f : T \rightarrow (S \rightarrow V)$ mapping time points to a spatial field $S \rightarrow V$.

In the next section, we discuss current support for continuous ST phenomena based on observation streams in information systems.

3 Related Work

Fields as data types are not commonly available in information systems, mostly due wide-spread use of established software, implementations and tools that are based on different types of representations. The OGC coverage interface specification [6] assumes a field-type representation for space-time varying phenomena, and has been an industry standard for more than a decade. However, its emphasis is on standardizing coverage operators, not data representations. Also, observation streams can not be accommodated directly with this interface specification. Work more closely related to ours is [7], which proposes a field data type as a generic data type to represent time series, trajectories and coverages. The idea of a single, generic data type to express different specialized fields is similar to our objectives. In this work, the field types have been prototypically implemented on top of an array database system, but, it is unclear if a formal, generic and reusable formal embedding has been attempted. [7] focuses on generic data types that are used via a library in a programming language, and the implemented data types use DBS technology for storage. Our work is different in two regards: first, our fields types are designed to extend stream data models (instead of subsuming them). Secondly, our work focuses on streams while the work in Camara et al. [7] addresses persistently stored and long-term collected data.

Similar to us, Ferreira et al. [13] are motivated by the increasing sampling density, and recognize the need to support how objects and fields evolve over time in a more flexible way so that integrated spatial analysis is simplified. An algebra for spatio-temporal data is proposed. This approach is less generic in its data types than [7] and our work since concrete types for time series, trajectories, coverages, as well as objects and events are proposed. Further, this work focuses on both fields and objects and their respective relationships, while our work aims at providing a flexible type systems for generic, composable and potentially complex continuous ST fields. Also, an embedding into a data model or query language is not addressed in [13].

The existing extensions to DSE data models and query languages for spatio-temporal streams are limited today. Therefore, fields can only be support by processing individual sensor data streams and integrating them in application code. Beside naively supporting points in stream data models, the work in [20, 14] focuses on spatio-temporal objects such as moving points and moving regions, and extend Gueting's work on spatio-temporal objects [18] for streams. Our work is complementary in addressing fields and streams. [2] introduces Nile-PDT, which correlates multiple concurrent streams; Nile-PDT is unaware of the spatial dimension, and is useful for extracting features rather than representing fields. GeoStreams [19] explores using DSE for processing large raster data streams, i.e the input of DSE queries are entire rasters, not point observation streams.

In summary, extensions to stream data model language and query language to support spatio-temporal continuous phenomena that are based on point-based observation streams do not exist today.

4 Abstract Model of Continuous Spatio-Temporal Field Data Types

Our objective is to design a data model and integrate the types *streams*, *relations* and *fields* that we defined above seamlessly. When designing the new data model, we distinguish the *abstract (data) model* and the *discrete (data) model* as different levels of abstraction for the sensor data stream model [11]. In our terminology, the abstract model defines data types in an implementation-independent, high-level and formal way. Its definition is driven by semantic understanding of the concepts. The abstract model allows us to use infinite sets in the concept definitions, without worrying about the finite (computer) representations of these sets. Thus, we can define our field data types with an infinite time domain as well as an infinite space domain regardless of the finite data structures and corresponding algorithms at the stage of the discrete data model design. For example, there is no need to worry about whether a trajectory of a moving object shall be represented as a curve or as a polyline in a two-dimensional space, while it is defined as an infinite set of points in the plane. The discrete model is defined as a data model that serves as the basis for implementing the abstract data model; this model addresses the issue of finite computer representations in the context of handling and processing sensor data streams in DSE.

4.1 Second Order Signatures

Second order signatures [17], introduced by Güting in 1993, have been widely used in the database literature to formalize relations, spatial and spatio-temporal data types [18, 20, 14]. Second order signatures allow formalizing both the syntax and semantics of data types and defining operators for those data types. Furthermore, defining our proposed field types in second order signature provides a natural interface with the spatio-temporal type hierarchies mentioned above.

The basic idea of a second-order signature is using two coupled extended signatures to describe a data model: the first signature defines a *type system* and the second signature uses the types of the first signature as sorts and defines *operators* over these types. Since we only focus on defining data types in this paper, we primarily utilize the first signature as a tool. A *signature* is a pair (S, Σ), where S is a set whose elements are called *sorts* and Σ is a set whose elements are called *operators* (note, that this operators are not the same operators defined for the data types but type constructors). In addition, a signature has an associated set of *terms* defined. In a multi-sorted signature, if $t_1, \ldots, t_n$ are terms of sorts $s_1, \ldots, s_n$ and $\omega : s_1 \times \cdots \times s_n \to s$ is n-ary operator, then $\omega(t_1, \ldots, t_n)$ is a term of sort s. In a single-sorted signature, n is equal to 1. A 0-ary operator is called a *constant*.

The basic concept of a signature for a given set of sorts is then extended to introduce automatically list sorts, product sorts, union sorts, and function sorts [17]. Based on this concept extension, the first signature of a second-order signature defines a *type system*; the sorts of the signature describe so-called *kinds* and its operators are the *type constructors*. The *terms* of this signature introduce the available types of this type system.

4.2 Abstract Model: Continuous Spatio-Temporal Field Data Types

As said a (first-order) signature consists of two sets of symbols, i.e. sorts/kinds and operators, and defines a type system. First, we introduce the kinds of our data model in Table 1. The **type constructors** column lists the operators of a signature. Using the **argument sorts**, the *terms* of the result sort (**kind**) are the possible *new data types* we can construct in the proposed data model. Each kind describes a certain set of types; for example, BASE stands for the types int, float, string, and bool. The type constructors show the signature for constructing terms of each type. In this terminology, the symbol $(.)^+$ denotes a list of one or more operands of certain sorts.

The kinds BASE, SPATIAL and TIME are similar to the spatio-temporal data types as defined in [18]. In our model, SPATIAL emphasizes a particular spatial type, i.e. a point location, which is used to model geographic locations of a phenomenon; a subset of the point location are the locations of sensor devices. With regard to TIME, we only consider the type instant. In general, the data types in kind BASE refer to the measurements taken by sensor nodes, which can be represented as integer, float, string, and bool values. Abstract semantics of BASE, SPATIAL, TIME have been defined in [18].

The kinds SPATIALFIELD, TEMPORALFIELD and SPATIALTEMPORALFIELD represent definitions of field types that are a mapping from S, T or $S \times T$ to a single attribute (e.g. temperature). These kinds represent spatial field data types, e.g., the temperature distribution over a specific geographic region (*simpleSfield* data type), temporal field data types, e.g., temperature readings from a single sensor over a time range (*simpleTfield*), and spatio-temporal field data types, e.g., temperature distribution over a specific area and time range (*simpleSTfield*), respectively. The fields have been described in more detail in Section 2. In addition, we introduce the new field data types *simpleSfieldvector*, *simpleTfieldvector* and *simpleSTfieldvector*, correspondingly, to support each space-time location being mapped to a vector of values (of potentially different types). For example, an instance of data type *simpleSTfieldvector* is a mapping from a

Table 1. Abstract model of continuous spatio-temporal field types.

Argument sorts	Kind	Type Constructor
	$\rightarrow$ BASE	integer float string bool
	$\rightarrow$ SPATIAL	point
	$\rightarrow$ TIME	instant
BASE$^+$	$\rightarrow$ SPATIALFIELD	simpleSfield simpleSfieldvector
BASE$^+$	$\rightarrow$ TEMPORALFIELD	simpleTfield simpleTfieldvector
BASE$^+$	$\rightarrow$ SPATIALTEMPORALFIELD	simpleSTfield simpleSTfieldvector
(BASE $\cup$ SPATIAL $\cup$ TIME $\cup$ SPATIALFIELD $\cup$ TEMPORALFIELD $\cup$ SPATIALTEMPORALFIELD)$^+$	$\rightarrow$ CSTFIELD	complexSfield, complexTfield complexSTfield complexSfieldvector complexTfieldvector complexSTfieldvector

spatio-temporal location to a vector consisting of a temperature value, wind speed measurement and so on. This extension provides the capability that multiple measurements that are associated with a single spatio-temporal location can be queried and analyzed at the same time.

The data types introduced so far are the basis for our further extensions. We introduce a new kind CSTFIELD that denotes Complex Spatio-Temporal Fields. Taking the *complexSfield* (complex spatial field) data type as an example, the value domain is not limited to basic measurements as in all other types defined before; instead, the value domain can be a combination of any of the data types defined in the data model so far. Therefore, an instance of *complexSfield* can be a spatial field, in which each location is mapped to a spatial object (e.g., each spatial location is mapped to a view shed region), which is similar to the idea of an object field as defined in [9]. Furthermore, a simple spatial/temporal field can also be a valid domain value for complex spatial/ temporal fields. For example, a *complexTfield* (complex temporal field) can be a mapping from a temporal instant to a spatial field (as we discussed before, a snapshot point of view $f : T \rightarrow (S \rightarrow V)$ which the spatio-temporal field can be equivalently defined). Similarly, we introduce *complexSfieldvector*, *complexTfieldvector*, *complexSTfieldvector* data types for supporting the representation of multiple values/objects/fields under one spatio-temporal framework. If end users need multiple spatio-temporal fields to be correlated, the *complexSTfieldvector* data type is necessary. The data types of kind CST-

FIELD are designed to add more query capability and representational flexibility for end users.

We use the notation A_α to denote the carrier set for each data type, where α is the data type. Each carrier set is extended with the null value $\perp$ that denotes a missing or undefined value. For convenience, we define $\bar{A}_\alpha = A_\alpha \setminus \{\perp\}$.

4.3 Semantics of the type system

The type *simpleSfield* is similar to the commonly used spatial fields in a GIS.

Definition 3. *A simpleSfield*(α) *is a data type with a carrier set*

$$A_{\textit{simpleSfield}(\alpha)} \equiv \big\{f \mid f : \bar{A}_{point} \rightarrow A_\alpha \cup \{\perp\}\big\}$$

where α is a data type (integer, float, string, or boolean) applicable to the type constructor simpleSfield and the carrier set A_α denoting any possible data type of BASE kind.

Next, we define a set *ValueVector* of BASE data types and then use that to define the data type *simpleSfieldvector*.

Definition 4. *A ValueVector is a list of basic data types BASE*$^+$ *with a carrier set*

$$A_{\textit{ValueVector}} \equiv \big\{\{s_1, \ldots, s_n\} \mid \forall i\ [s_i \in A_{\textit{integer}} \cup A_{\textit{real}} \cup A_{\textit{string}} \cup A_{\textit{bool}} \cup \{\perp\}]\big\}$$

for some $n \geq 2$.

Definition 5. *A simpleSfieldvector*(α) *is a data type with carrier set*

$$A_{\textit{simpleSfieldvector}(\alpha)} \equiv \{f \mid f : \bar{A}_{\textit{point}} \rightarrow A_\alpha\} \cup \{\perp\}$$

where α is a data type in sort $BASE^+$, with carrier set A_α denoting ValueVector.

Similarly, *simpleTfieldvector* and *simpleSTfieldvector* can be defined as follows.

Definition 6. *simpleTfieldvector*(α) *and simpleSTfieldvector*(α) *are data types with respective carrier sets*

$$A_{\textit{simpleTfieldvector}(\alpha)} \equiv \{f \mid f : \bar{A}_{\textit{instant}} \rightarrow A_\alpha\} \cup \{\perp\},$$
$$A_{\textit{simpleSTfieldvector}(\alpha)} \equiv \{f \mid f : \bar{A}_{\textit{point}} \times \bar{A}_{\textit{instant}} \rightarrow A_\alpha\} \cup \{\perp\}$$

where α is a data type applicable to the type constructor simpleTfieldvector and simpleSTfieldvector, respectively, with the carrier set A_α denoting ValueVector.

Next, we define *simpleTfield* and *simpleSTfield* analogous to the earlier definition of *simpleSfield*.

Definition 7. *simpleTfield*(α) *and simpleSTfield*(α) *are data types with respective carrier sets*

$$A_{\textit{simpleTfield}(\alpha)} \equiv \{f \mid f : \bar{A}_{\textit{instant}} \rightarrow A_\alpha\} \cup \{\perp\},$$
$$A_{\textit{simpleSTfield}(\alpha)} \equiv \{f \mid f : \bar{A}_{\textit{point}} \times \bar{A}_{\textit{instant}} \rightarrow A_\alpha\} \cup \{\perp\}$$

where α is a data type applicable to the type constructor simpleTfield and simpleSTfield, respectively, with the carrier set A_α denoting any possible data type of BASE kind.

The final types – the *complexSfield*, *complexTfield*, and *complexSTfield* type constructors and their vector analogues – are high-level abstractions that hide individual sensors, their locations, and measurement values and provide an integrated field view.

Definition 8. *complexSfield*(α)*, complexTfield*(α)*, and complexSTfield*(α) *are data types with the respective carrier sets*

$$\begin{aligned} A_{\mathit{complexSfield}(\alpha)} &\equiv \{f \mid f : \bar{A}_{\mathit{point}} \to A_\alpha\} \cup \{\bot\} \\ A_{\mathit{complexTfield}(\alpha)} &\equiv \{f \mid f : A_{\mathit{instant}} \to A_\alpha\} \cup \{\bot\} \\ A_{\mathit{complexSTfield}(\alpha)} &\equiv \{f \mid f : \bar{A}_{\mathit{point}} \times \bar{A}_{\mathit{instant}} \to A_\alpha\} \cup \{\bot\} \end{aligned}$$

where α *is a data type applicable to the type constructor complexSfield, complexTfield and complexSTfield, respectively, with the carrier set* A_α *denoting any possible data type in* $A_{\mathit{BASE}} \cup A_{\mathit{point}} \cup A_{\mathit{instant}} \cup A_{\mathit{simpleSfield}} \cup A_{\mathit{simpleTfield}} \cup A_{\mathit{simpleSTfield}} \cup A_{\mathit{simpleSfieldvector}} \cup A_{\mathit{simpleTfieldvector}} \cup A_{\mathit{simpleSTfieldvector}}$.

Definition 9. *complexSfieldvector*(α)*, complexTfieldvector*(α) *and complexSTfieldvector*(α) *are data types with the respective carrier sets*

$$\begin{aligned} A_{\mathit{complexSfieldvector}(\alpha)} &\equiv \{f \mid f : \bar{A}_{\mathit{point}} \to A_\alpha\} \cup \{\bot\} \\ A_{\mathit{complexTfieldvector}(\alpha)} &\equiv \{f \mid f : A_{\mathit{instant}} \to A_\alpha\} \cup \{\bot\} \\ A_{\mathit{complexSTfieldvector}(\alpha)} &\equiv \{f \mid f : \bar{A}_{\mathit{point}} \times \bar{A}_{\mathit{instant}} \to A_\alpha\} \cup \{\bot\} \end{aligned}$$

where α *is a data type applicable to the type constructor complexSfieldvector, complexTfieldvector and complexSTfieldvector, respectively, with the carrier set* A_α *denoting any possible data type in*

$$\begin{aligned} \big\{ \{s_1, \ldots, s_n\} \mid \forall i\ [s_i \in\ & A_{\mathit{BASE}} \cup A_{\mathit{point}} \cup A_{\mathit{instant}} \cup \\ & A_{\mathit{simpleSfield}} \cup A_{\mathit{simpleTfield}} \cup A_{\mathit{simpleSTfield}} \cup \\ & A_{\mathit{simpleSfieldvector}} \cup A_{\mathit{simpleTfieldvector}} \cup A_{\mathit{simpleSTfieldvector}} \cup \{\bot\}]\big\} \end{aligned}$$

We now have specified the data model from an abstract perspective by defining the range of spatio-temporal fields we support in our stream data model. We have defined which field types are possible, and how they are constructed.

5 Extending Stream Data Models with Field Data Types

The previous section formalized the spatio-temporal fields data types that our data model supports on an abstract level that assumes continuity in space and time. This section presents the discrete versions of these field data types that are necessary to implement the data types on the basis of streams of *discrete observations* and *discretized views of space and time*.

5.1 From Spatio-Temporal Streams to Continuous Spatio-Temporal Fields

As mentioned in Section 2, this paper addresses fields that are approximated based on observations, so-called *sampled fields* (in contrast to equation fields that are purely defined via equations). In particular, we introduce fields types that are canonically constructed from spatio-temporal streams. Our approach starts with the definition of a *spatio-temporal relation*, which is a finite set of tuples, and each tuple is an observation. Using

Table 2. Discrete model of spatio-temporal field types extending stream data models.

Argument sorts	Kind	Type Constructor
	→ BASE	integer float string bool
	→ SPATIAL	point, geometry
	→ TIME	instant
BASE^+	→ SDATA	sensordata
SDATA	→SDSTREAM	sdstream
	→ WINDOW	slidingwindow
SDSTREAM×WINDOW	→ STREL	observationfield
	→ INTERPOL	interpolator
observationfield×INTERPOL	→ CSTFIELD	continuousSTfield continuousSfield continuousTfield
$(\text{BASE} \cup \text{SPATIAL} \cup \text{TIME} \cup \text{CSTFIELD})^+$	→ CCSTFIELD	complexcontinuousSTfield

a spatio-temporal relation that contains tuples with timestamps within a well-defined interval and spatial locations within a well-defined spatial region, we can construct an *observation field*. An observation field is simply a collection of raw sensor samples. Each sample represents a point in a spatio-temporal volume; other values are not available. Once we add an *interpolator* to an observation field, we can create a *continuous spatio-temporal field*. On behalf of the raw samples of the observation field and the interpolator, the continuous spatio-temporal field can produce *all* values within its continuous spatial and temporal domain. Some of these values correspond to actual samples; others are interpolated based on the samples. Below, we introduce streaming versions of the discussed types; this includes *sdstream*, *observationfield*, and *continuousSTfield*. The mapping between relational types and streaming types is bidirectional. Due to space constraints, we refer the reader to more details of this discussion in [23].

5.2 Discrete Data Model for Continuous Spatio-Temporal Field Data Types

The kinds BASE and TIME are maintained from the abstract data model. For the SPATIAL kind we add a *geometry* type that can represent various geometric objects such as points, lines, 2D regions, and aggregations thereof. This geometric type is introduced to better represent the results of operators over fields, whose discussion is beyond the scope of this paper.

Data type for sensor tuples: This data type is defined to represent an individual sensor sample using the corresponding basic data types and the type constructor *sensordata*. A *sensor data tuple* represents a single update from a sensor, while the time series of a sensor's updates is a stream. The *sensordata*'s constructor uses BASE^+ as input sorts. The rationale behind this choice is that a sensor node can combine all or a subset of its sensor samples from different attached sensors that are taken at one time instant and forward them compactly as a single message, creating an update tuple. Sensors without an update at that time report a NULL value as part of the update tuple. We assume that a sensor data tuple will always contain at least one time stamp of the kind TIME and a location value of type *point* since a data value without any time or any location information is meaningless. A second time stamp might be created at the DSE to represent the arrival time of the update tuple.

Definition 10. *A sensordata(α) is a data type with a carrier set*

$$A_{\textit{sensordata}}(\alpha) \equiv \bar{A}_{\textit{point}} \times \bar{A}_{\textit{instant}} \times A_{\textit{integer}} \times A_{\alpha}$$

where α is a data type in sort $BASE^+$ that is applicable to the type constructor sensordata and produces an output of the kind SDATA.

The carrier set of data type α is a *ValueVector*, defined in Def. 4. The three other parameters A_{point}, A_{instant}, and A_{integer} represent the carrier sets for location, explicit time stamp (time of observation), and implicit time stamp (arrival time at the DSE), respectively. The explicit time stamp is required whereas the implicit time stamp is an optional parameter, as indicated by the missing bar across A_{integer}.

Sensor Data Stream Type: Next, we define the data type for representing streams of sensor data tuples. Each value from the carrier set of this new data type *sdstream*(α) is a function that maps each implicit time stamp to a finite number (possibly zero) of sensor data tuples [4].

Definition 11. *A sdstream(α) is a data type with a carrier set*

$$A_{\textit{sdstream}(\alpha)} \equiv \big\{f \;\big|\; A_{\textit{instant}} \rightarrow \{S \subseteq A_{\alpha} \;\big|\; |S| < \infty\}\big\}$$

where α is a data type of sort SDATA that is applicable to the type constructor sdstream.

Sliding Window Type: For kind WINDOW, the data type constructor *slidingwindow* represents the concept of sliding windows in DSE. We adopt the two commonly used parameters to specify sliding windows: window size *ws* and update interval *ui*. We only consider tuple-based (count-based) or time-based sliding windows.

Definition 12. *A slidingwindow is a data type with a carrier set*

$$A_{\textit{slidingwindow}} \equiv \Big\{f \;\big|\; f : T_{\textit{eval}} \rightarrow T_S \times T_E \quad \text{such that } \forall(t_{\textit{start}_i}, t_{\textit{end}_i}) \in T_{\textit{eval}} \; \big[t_{\textit{start}_i} \leq t_{\textit{end}_i} \text{ and } t_{\textit{end}_i} - t_{\textit{start}_i} = \textit{ws} \text{ and } t_{\textit{start}_i} - t_{\textit{start}_{i-1}} = \textit{ui}\big]\Big\}$$

where ws is the sliding window size, ui is the sliding window update interval, T_{eval} is the carrier set of time stamps when a sliding window will be evaluated, and $T_S \times T_E$ is the carrier set that indicates the start and end time of a specific sliding window.

For a tuple-based (count-based) window $T_{eval}, T_S, T_E \subseteq \bar{A}_{integer}$ holds, while for a time-based window $T_{eval}, T_S, T_E \subseteq \bar{A}_{instant}$ holds. This is due to the fact, that we consider two different types of windows semantics: count-based window change their designated time interval with the arrival of a new streaming tuple, while time-based windows change their time interval with the advancement of time.

Observation Field Type: Using sliding windows, the most recent portion of a data stream can be regarded as a temporalized relation [4]. However, we prefer to define the more meaningful type *observationfield* that captures the finite set of sensor data points – each containing a spatial location, a timestamp, and a measurement value – from a window and defined them as a single field. An *observationfield* represents raw sensor data measurement streams as a ***discrete*** *spatio-temporal field* where values are only valid at the spatial/temporal locations where actual sensor measurement are available.

Definition 13. *An observationfield is a data type with carrier set*

$$A_{\mathit{observationfield}(\mathit{sdstream}(\alpha),\omega)} \equiv \Big\{ f \mid f : T_{\mathit{eval}} \to S_{\mathit{eval}} \quad \text{such that } \forall f(t_i) \in S_{\mathit{eval}_i} \\ [f(t_i).\mathit{timestamp} \in R(\omega, t_i)] \Big\}$$

where $S_{\mathit{eval}} = \{S \subseteq A_\alpha \bigm| |S| < \infty\}$, α *is a data type of sort SDATA,* $\mathit{sdstream}(\alpha)$ *is a data type of sort SDSTREAM,* ω *is a data type of sort WINDOW,* T_{eval} *is the carrier set of timestamps when a sliding window* ω *is evaluated, and* $R(\omega, t_{\mathit{eval}})$ *is the time range at each evaluation timestamp* $t_{eval} \in T_{eval}$ *of the sliding window* ω*.*

Interpolator Type: Now, we introduce the interpolator type required to convert streams of discrete sensor measurements into continuous spatio-temporal fields. More precisely, it is intended to estimate any values at arbitrary spatio-temporal locations within a specific instance of type *observationfield*, even if no sensor measurements are available at those precise spatio-temporal locations.

Definition 14. *An interpolator is a data type with a carrier set*

$$A_{\mathit{interpolator}} \equiv \{f \mid f : A_{\mathit{observationfield}(\mathit{sdstream}(\alpha),\omega)} \times \bar{A}_{\mathit{point}} \times \bar{A}_{\mathit{instant}} \to A_\sigma\}$$

where α *is a data type of sort SDATA,* $\mathit{sdstream}(\alpha)$ *is a data type of sort SDSTREAM,* ω *is a data type of sort WINDOW,* $\mathit{observationfield}(\mathit{sdstream}(\alpha), \omega)$ *is of data type observationfield, and* σ *is a data type in sort* BASE^+*.*

Continuous Spatial Temporal Field Type: Since we define the continuous spatio-temporal field type in the context of handling discrete streaming sensor data, the new type *continuousSTfield* is explicitly based on the observation field and the corresponding interpolator function. The resulting *continuousSTfield* is continuously updated with the windows parameters from the underlying observation field.

Definition 15. *Assume a given interpolator with output of type* σ *and an observationfield denoted as* $\mathit{observationfield}(\mathit{sdstream}((\alpha), \omega)$*. Then a continuousSTfield is a data type with a carrier set*

$$A_{\mathit{continuousSTfield}(\mathit{observationfield}(\mathit{sdstream}(\alpha),\omega))} \equiv \\ \Big\{ f \Bigm| f : T_{\mathit{eval}} \to \{f^c \mid f^c : \bar{A}_{\mathit{point}} \times \bar{A}_{\mathit{instant}} \to A_\beta \text{ such that } A_\beta \subseteq A_\sigma\} \Big\}$$

where α *is a data type of sort SDATA,* $\mathit{sdstream}(\alpha)$ *is a data type of sort SDSTREAM,* ω *is a data type of sort WINDOW,* T_{eval} *is the carrier set of timestamps when a sliding window* ω *is evaluated, and* β *is a data type in sort* BASE^+*.*

Other continuous spatial/temporal field types *continuousSfield* and *continuousTfield* and the complex type *complexcontinuousSTfield* closely follow the definitions from the abstract data model, except that they are now streaming versions correspondingly to the definition of a *continuousSTfield*.

6 Conclusions and Future Work

In this paper, we presented our formal extension of stream data models with field data types. Using the field data types, users can define high-level abstractions of continuous ST phenomena based on large numbers of concurrent, bursty and unpredictable sensor data streams that are now known to a DSE. Therefore, they can be supported through queries and processing optimization, thus, unburdening the user and application code. We introduced field types specifically for sampled fields, and their streaming counterparts. Our stream model extension formally integrates spatio-temporal streams, spatio-temporal relations and field types. We formalized the proposed types using second order signature to achieve independence from the details of a specific data model language implementation, and formalized the syntax as well as semantics of the proposed types. As for future work, several aspects closely related to this work require more research. For instance, the challenging question of generic operators over the proposed fields types that can be integrated into stream query languages requires further investigation. Similarly, a prototypical implementation of the type system as part of various actual data model languages is of significant interest.

Acknowledgement. The authors would like to thank Mark Plummer for many, fruitful discussions and the National Science Foundation for supporting this work via Award No. 1527504.

References

1. Apache Spark (2016), `http://spark.apache.org`
2. Ali, M.H., Aref, W.G., Bose, R., Elmagarmid, A.K., Helal, A., Kamel, I., Mokbel, M.F.: NILE-PDT: A Phenomenon Detection and Tracking Framework for Data Stream Management Systems. In: VLDB'05. pp. 1295–1298. VLDB Endowment (2005)
3. Ali, M., Chandramouli, B., Raman, B., Katibah, E.: Spatio-Temporal Stream Processing in Microsoft StreamInsight. IEEE Data Eng. Bull. 33(2), 69–74 (2010)
4. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: semantic foundations and query execution. The VLDB Journal 15(2), 121–142 (Jul 2005)
5. Babu, S., Widom, J.: Continuous queries over data streams. ACM Sigmod Record 30(3), 109 (Sep 2001)
6. Baumann, P.: The OGC Web Coverage Processing Service (WCPS) Standard. Geoinformatica 14(4), 447–479 (2010)
7. Camara, G., Egenhofer, M.J., Ferreira, K., Andrade, P., Queiroz, G., Sanchez, A., Jones, J., Vinhas, L.: Fields as a Generic Data Type for Big Spatial Data. In: Duckham, M., Pebesma, E., Stewart, K., Frank, A.U. (eds.) Proceedings of 8th International Conference "Geographic Information Science" (GIScience 2014), pp. 159–172. Springer LNCS 8728, Vienna, Austria (2014)
8. Couclelis, H.: People manipulate objects (but cultivate fields): Beyond the raster-vector debate in gis. In: Frank, A.U., Campari, I., Formentini, U. (eds.) Theories and Methods of Spatio-Temporal Reasoning in Geographic Space: International Conference GIS — From Space to

Territory: Theories and Methods of Spatio-Temporal Reasoning Pisa, Italy, September 21–23, 1992 Proceedings, pp. 65–77. Springer Berlin Heidelberg, Berlin, Heidelberg (1992)

9. Cova, T., Goodchild, M.: Extending geographical representation to include fields of spatial objects. International Journal of Geographical Information Science 16(6), 509–532 (2002)
10. Duckham, M., Zhong, X., Toohey, K.: Challenges to Using Decentralized Spatial Algorithms in the Field: The RISERnet Geosensor Network Case Study. SIGSPATIAL Newletter, Special Issue "Geosensor Networks" 7(2), 14–21 (2015)
11. Erwig, M., Güting, R.H., Schneider, M., Vazirgiannis, M.: Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases. GeoInformatica 3(3), 269–296 (1999)
12. Faulkner, M., Olson, M., Chandy, R., Krause, J., Chandy, K., Krause, A.: The Next Big One: Detecting Earthquakes and other Rare Events from Community-based Sensors. In: Int. Conf. on Information Processing in Sensor Networks (IPSN). pp. 13–24 (2011)
13. Ferreira, K.R., Camara, G., Monteiro, A.M.V.: An algebra for spatiotemporal data: From observations to events. Transactions in GIS 18(2), 253–269 (2014)
14. Galić, Z., Baranović, M., Križanović, K., Mešković, E.: Geospatial data streams: Formal framework and implementation. Data & Knowledge Engineering 91, 1–16 (2014)
15. Galton, A.: A Formal Theory of Objects and Fields. In: COSIT. pp. 458–473 (2001)
16. Galton, A.: Fields and objects in space, time, and space-time. Spatial Cognition and Computation 1, 39–68 (2004)
17. Güting, R.: Second-order signature: a tool for specifying data models, query processing, and optimization. In: SIGMOD '93. pp. 277–286. ACM, New York, NY, USA (1993)
18. Güting, R., Michael, H., Erwig, M., Jensen, C., Lorentzos, N., Schneider, M., Vazirgiannis, M.: A Foundation for Representing and Querying Moving Objects 1(212), 1–37 (2000)
19. Hart, Q., Gertz, M.: Querying Streaming Geospatial Image Data. In: SSDBM'05. pp. 147–150. Santa Barbara, CA, USA (2005)
20. Huang, Y., Zhang, C.: New Data Types and Operations to Support Geo-streams. In: Cova, T.J., Miller, H.J., Beard, K., Frank, A.U., Goodchild, M.F. (eds.) Proceedings of 5th International Conference "Geographic Information Science" (GIScience 2008), pp. 106–118. No. 003594, Lecture Notes in Computer Science, 2008, Volume 5266/2008, Park City, Utah (2008)
21. Kemp, K.: Fields as a framework for integrating GIS and environmental process models. Part 1: Representing spatial continuity. Transactions in GIS (3) (1996)
22. Laurini, R., Paolino, L., Sebillo, M., Tortora, G., Vitiello, G.: A spatial SQL extension for continuous field querying. Intern. Computer Software and Applications Conf. (COMPSAC 2004) 2, 78–81 (2004)
23. Liang, Q.: Towards the Continuous Spatio-Temporal Field Model for Sensor Data Streams. Dissertation, University of Maine (2015)
24. Nittel, S.: Real-time Sensor Data Streams. SIGSPATIAL Newsletter, Special Issue "Geosensor Networks" 7(2), 22–28 (2015)
25. Sanchez, L., Galache, J., Gutierrez, V., Hernandez, J.M., Bernat, J., Gluhak, A., Garcia, T.: SmartSantander: The meeting point between Future Internet research and experimentation and the smart cities. In: Future Network & Mobile Summit (FutureNetw). pp. 1–8 (2011)
26. Stonebraker, M., Çetintemel, U., Zdonik, S.: The 8 requirements of real-time stream processing. ACM SIGMOD Record 34(4), 42–47 (Dec 2005)
27. Whittier, J., Liang, Q., Nittel, S.: Evaluating Stream Predicates over Dynamic Fields. In: Proc of 5th International ACM SIGSPATIAL Workshop on GeoStreaming. pp. 2–11 (2014)
28. Whittier, J., Nittel, S., Liang, Q., Plummer, M.: Towards Window Stream Queries over Continuous Phenomena. In: Proc of 4th International ACM SIGSPATIAL Workshop on GeoStreaming. pp. 1–10. Orlando, FL (2013)