Annotations: Dynamic Semantics in Stream Processing

Juan Amiguet*
CTIT Database Group
University of Twente
5, Drienerlolaan
Enschede, The Netherlands
J.Amiguet@utwente.nl

Andreas Wombacher†
CTIT Database Group
University of Twente
5, Drienerlolaan
Enschede, The Netherlands
A.Wombacher@utwente.nl

Tim E. Klifman
Database Group Student
University of Twente
5, Drienerlolaan
Enschede, The Netherlands
t.e.klifman-1@student.utwente.nl

ABSTRACT

In the field of e-science stream data processing is common place facilitating sensor networks, in particular for prediction and supporting decision making. However, sensor data may be erroneous, like e.g. due to measurement errors (outliers) or changes of the environment. While it can be foreseen that there will be outliers, there are a lot of environmental changes which are not foreseen by scientists and therefore are not considered in the data processing. However, these unforeseen semantic changes - represented as annotations - have to be propagated through the processing. Since the annotations represent an unforeseen, hence un-understandable, annotation, the propagation has to be independent of the annotation semantics. It nevertheless has to preserve the significance of the annotation on the data despite structural and temporal transformations. And should remain meaningful for a user at the end of the data processing. In this paper, we identify the relevant research questions.In particular, the propagation of annotations is based on structural, temporal, and significance contribution. While the consumption of the annotation by the user is focusing on clustering information to ease accessibility.

Categories and Subject Descriptors

J.2 [**Earth and atmospheric sciences**]: Miscellaneous; H.2.m [**Information Systems**]: Database management—*Stream data processing*

General Terms

Annotations

Keywords

Sensor networks, Streaming, Semantics, Annotations

*PhD Student
†Supervisor

PIKM'10, October 30, 2010, Toronto, Ontario, Canada.

1. INTRODUCTION

Sensor networks are used to gather information about environmental, or industrial processes usually resulting in large data volumes. Sensor data are processed as a stream supporting warning, decision support and forecasting applications [18]. The characteristics of streaming applications are often that (1) data elements are monotonically organized by order of arrival, (2) each element is seen at most once by the processing algorithm, (3) results have to be available with the least amount of delay [5]. Thus, stream processing consists of processing elements. A stream processing element continuously processes new sensor data of incoming streams resulting in an outgoing stream of processed data. The output stream can be an input for further processing elements.

The designer of a stream processing knows the semantics of the input streams and the processing algorithm, which results in the semantics of the output stream. These semantic relations are often maintained as provenance information [17]. However, the intended semantics of data tuples in a stream might be changed by unforeseen events e.g. in the environment. However, provenance information is not intended to handle these dynamic changes of semantics. As a consequence, semantic change of data tuples contained in a stream and consumed by an application might result in wrong decisions or predictions.

2. RELATED WORK

The various streaming systems, such as e.g. Borealis [1], System S processing core [3], Global Sensor Network [2], STREAM [4], or TelegraphCQ [8] address continuous queries and support scenarios as described in Section 3, however, they do not provide any notion of annotation or any propagation mechanism as addressed in this paper.

Alternatively initiatives such as the Sensor Web Enablement [7] define standards for sensor interoperability which include the interaction of systems as well as the description of metadata. However, the associated standards of the initiative focus on a description of a fixed set of metadata instead of propagation of metadata along the processing chain.

Another kind of metadata is data provenance. Provenance data as defined in [17] is data "[enabling e-science users] to reproduce their results by replaying previous computations, understand why two seemingly identical runs with the same inputs produce different results, and determine which data sets, algorithms, or services were involved in their derivation."

Data provenance on workflow level has been addressed in

many different flavors especially in the e-science community. An overview and a taxonomy of provenance is provided by [20].

Example infrastructures are Karma2 [21] and Tupelo [13] implementing the Open Provenance Model [10] for the capture and query of provenance data. However some differences exist amongst them. Karma2 enables provenance data to be communicated as updates in a de-coupled and asynchronous publish subscribe model. Tupelo libraries however enable the deferral of the sending of the provenance data until after the operation is finished in order to minimise the impact of proveance on the workflow operations. Tupelo also enables agregation from external data sources in order to infer semantic facts as part of querying. In order to support this query functionality Tupelo is configurable with ontologies, and understands queries specified in semantic web languages such as RDF and SPARQL.

A more infrastructure oriented example has been proposed in [23] describing a method for recording and reasoning over data provenance in web and grid services. In particular, it describes which services have been used, what kind of data has been passed between those services and what results have been generated by the services. The e-science community has developed the Open Provenance Model [10] as a result of three provenance challenges. It gives a good idea on the kind of information addressed by provenance systems. However, the type of unforeseen annotations as addressed in this paper are not contained. Provenance information is collected and not propagated. Further, provenance information does not change over time, but is recorded when a processing element is executed.

Special annotations which are propagated along the workflow are technical data quality measures. In literature there are several approaches described. Most of them are based around the notion of fitness for purpose [15, 16, 19] describing how well the actual data represent the intended purpose, i.e., semantics. Some of them rely on ground truth [19] others rely on method specific assumptions[15, 16].

The semantic element for the data fitness can only be found in frameworks for identifying quality data attributes [6, 22]. The 5WH (why, when, where, what, who, how) framework as introduced by Bisdikian in [6] aims to support decision in the compatibility of an element of data with an application, using two axis in the evaluation, semantic suitability and accuracy constraints. The 5WH framework is used for identifying which data quality measurements should be part of the application. Conversely the work introduced by Strong in [22] introduces in a business setting four dimensions of data quality. The four dimensions are: (1) intrinsic, (2) accesibility, (3) contextual and (4) representational. For each dimension several values are defined. The values are then used to identify shortcomings in the data quality, or data quality defects. Not all of the values can be easily quantified but intrinsic data quality values such as accuracy or contextual data quality values such as timeliness, completeness, and amount of data can be useful in workflow data processing.

However, the quality measurements are comparable to foreseen annotations in this paper. While annotations are boolean - there is either an annotation or not - data quality measures can be gradual - the accuracy is $\pm 0.23^{\circ}C$ for a temperature sensor. This difference has implications on the complexity of the required propagation mechanism. As a consequence, unforeseen non boolean annotations to our understanding can not be propagated.

There is quite some literature on storing annotations. Annotations are in particular often used in biological applications [12]. However, the propagation of these annotations is not considered. In the BDBMS system [11] the authors propose an annotation management based on a relational storage model supporting annotations at various granularity levels (table, tuple, column, cell). Mondrian [14] supports to annotate both single value and the associations between multiple values. The both cases the focus is on optimizing storage structures of annotations.

The DBNotes approach [9] proposes a 'post-it note' system for relational data allowing to annotate each tuple element in a relation. The annotations stick to the tuple elements, thus if a tuple is queried and the tuple element is in the result set, then also the annotation is in the result set. The support query language seem not to support aggregation. Thus, annotation propagation can not be represented along temporal aggregations, and structural changes are limited to one-to-one mappings of tuple elements ignoring the effects of the data transformation.

3. USE CASE

The use case is a simplified climate forecasting application which has been modified to capture all issues addressed in this paper. The use case is based on weather stations placed in different locations of a know region in the Alps. The weather stations contain amongst others temperature sensors. Sensor data is made available in a streaming system by a processing element represented in the upper part of Fig 1 as hexagons. The three temperature sensors used in this use case are *temp@S1*, *temp@S2* and *temp@S3* Since sensors do not have an input stream and the output cannot be determined on the input stream, the processing element is called time variant. The temperature sensors measure temperature every 10 minutes. Sample measurements of the three sensors are depicted in the lower part of Fig 1.

To get an idea of the temperature distribution over the mountains, the temperature sensor data are used as input for an *interpolation* processing step represented in the upper part of Fig 1 as a rectangle. The *interpolation* is a time invariant processing element, since its output solely depends on the data of the input streams. The interpolation is performed for every new sensor measurement using the current temperature measurements. The interpolation results in a matrix of temperature measurements corresponding to a grid spanned over the Alps. An example matrix is depicted in the lower part of Fig 1.

For climate research time spans of several years are relevant, thus, the interpolation results are further aggregated to monthly temperature interpolations represented as the *avg/mth* processing element in the upper part of Fig 1. The processing element provides again a matrix of measurements as depicted in the lower part of Fig 1.

Finally, to visualize the monthly temperature interpolations, a *visualization* processing step is added (see upper part Fig 1). This processing element consumes the streaming data, also known as data sink, since it does not provide any output stream. The visualization results in a contour plot, which is directly displayed in an application (see lower part of Fig 1).

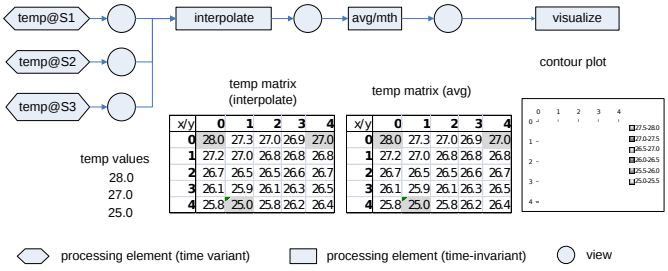

Figure 1: Temperature contour map workflow

3.1 Semantic Change

Stream semantics may change over time, i.e., the meaning of the data tuple in the stream is no longer the intended meaning. The new meaning of the data tuple can be represented as an annotation of the changed semantics on data tuples. Annotations are hence metadata tokens associated with a data tuple on the stream. The tokens consists of an arbitrarily small text, which represents a unique description of the semantic change. The text has to be both human and machine readable. Further several annotations can be associated with one data tuple.

Based on the use case (see Sect 3), the three weather stations may function correctly and send the data for processing in a timely fashion. Thus, the measurement has the intended semantics. Besides of the intended semantics three cases of changing semantics can be distinguished:

- One weather station measures a temperature of 40 degree Celsius, which is obviously a faulty measurement in an Alpine scenario. The semantics of the measurement changes from temperature measurement to measurement error. These kind of semantic changes can be detected algorithmically and therefore the detection of these changes can be added to the stream processing by automatically adding an *outlier* annotation to data tuples. Thus, represents a **foreseen** semantic change.

- Another foreseen semantic change is at the end of the month to go manually over the temperature data and annotate additional data tuples with *outlier* based on the knowledge of the deployment of the sensors and experience of the human doing the assessment. Although this is a manual annotation done after the data have been processed in the stream processing, the semantic change is **foreseen**, since all consumers of the stream are aware that an *outlier* annotation can be used on the particular stream.

- The weather stations produce reasonable measurements (see Fig 2 at time T1 and T2). During a routine inspection of the stations it is noticed that one of the three stations is covered in snow, as depicted in Figure 2 at time T3. Thus, the measurement is the temperature of air under an unknown or varying depth of snow. This semantic change is not indicated by the data. The scientists deploying the weather stations were not expecting that a station could be snowed in. Thus, the snowed in station is an **unforeseen** semantic change.

Foreseen semantic changes, like e.g. detected outliers, may be integrated in the stream processing. As a consequence, user consuming a certain stream are aware of possible annotations of data tuples before they actually use the stream. Due to this knowledge the user of the stream can decide whether and if so how she wants to handle annotations. Possible actions are: ignore the outlier annotation, since it is not relevant for the application, remove the annotated data tuples, or propagate the annotation and let subsequent processing elements deal with the annotation.

In case of unforeseen semantic changes, like e.g. an unexpected change in the environment of the sensor, annotations are not expected and therefore can only be propagated. It is characteristic for unforeseen semantic changes that the annotation is associated manually, with a significant delay and for a time interval of data tuples, where the lower bound of the time interval is unclear. In the use case introduced in section 3 a scientist performs a visual inspection (see Fig 2 T3) of the station and finds it covered in snow. The station is then dug out of the snow and this incident logged in the log book of the scientist. When returning to the lab the lower bound of the annotation is estimated based on additional information like snowfall, snow height, and wind speed and direction as well as knowledge of the terrain. The scientist then reviews the previously captured data and attaches a *snowed in* annotation to it. Hence the starting point of the annotation is estimated to be T2(see Fig 2), in the current example. The end of the annotation is clear since the scientists dig out the weather station when they observed that it is snowed in.

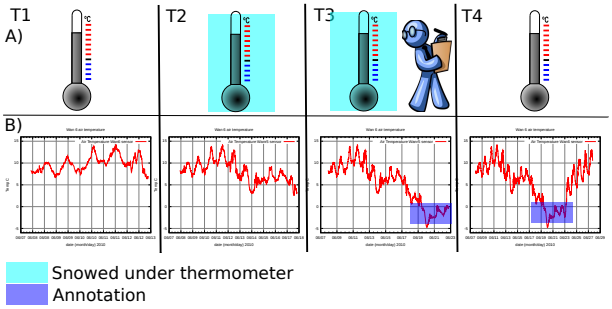

Figure 2: Annotation of the temperature data for S1 after a visual inspection

4. ANALYSIS

Semantic changes observed in a workflow and explicated by annotations have to be handled in the workflow. With regard to the input and output behavior processing elements can be classified as

- *source*: a time-invariant processing element without any input streams, e.g. a temperature sensor,

- *sink*: a processing element without any output streams, e.g. a visualization component,

- *transform*: a processing element with input and output streams, e.g. an interpolation processing element.

Algorithmic or manual annotations at a processing element are annotation sources. If an annotation is understood by a processing element one or several annotations may be consumed, thus the processing element acts as an annotation

sink. If an annotation is not understood the only option is to propagate the annotation, thus the processing element acts as an annotation transform.

Unforeseen annotations are not understood by the processing elements, because otherwise they are foreseen annotations. In particular, the change of semantics expressed in unforeseen annotations are only intelligible by the end user. Thus, unforeseen annotations have to be transformed by transform processing elements and have to be communicated to the user by sink processing elements.

With regard to the example (see Sect 3) the processing element *temp@S1* is an annotation source, the processing elements *interpolation* and *avg/mth* are annotation transforms, and *visualize* is an annotation sink.

The propagation and communication of unforeseen annotations introduces a series of issues discussed in the following subsections.

4.1 Annotation propagation

The issues related to propagation of unforeseen annotations are the effect of annotations on changing *data structures* or *temporal resolution* of streaming data, and the *significance of annotations* due to structural and/or temporal resolution changes.

4.1.1 Data structures

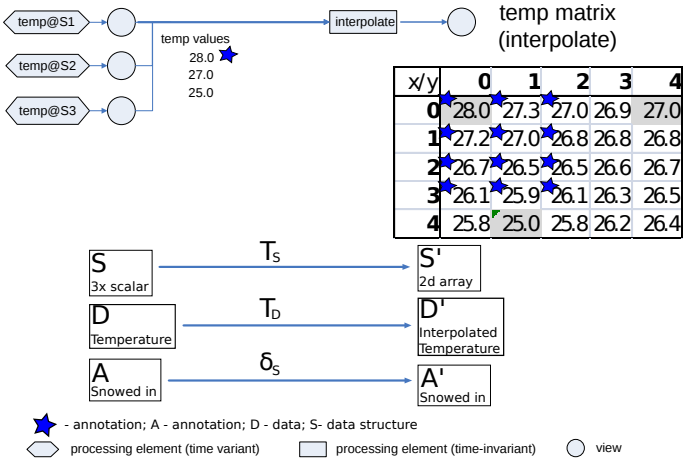

Figure 3: Annotation data structure transform

The use case introduced in Section 3 is based on three sensors each producing a temperature reading (see Fig 3 *temp values*). The processing element *interpolate* performs a spatial interpolation of the point temperature measurements resulting in a grid (two dimensional matrix) of interpolated temperature measurements (see Fig 1).

The processing element *interpolate* accepts scalar values as input data and produces a two dimensional matrix as output (see Fig 3 *temp matrix (interpolate)*). Thus, the processing element contains a structural transformation $T_S : S \rightarrow S'$ of the input structure S to the output structure S'. The structural transformation requires extra information such as the relative position of each sensor with regards to the others.

Based on the structural transformation of input and output a transformation $T_D : D \rightarrow D'$ of the data values can be defined. In particular, the input data out of D adhering to structure S are transformed into data out of D' adhering to structure S'. In the *interpolate* processing element the output array is determined using an inverse distance weighting algorithm based on the three temperature values 28.0 from *temp@S1*, 27.0 from *temp@S2* and 25.0 from *temp@S3* (see Fig 1).

The observed change in semantics after a visual inspection of the weather stations (see T3 in Fig 2) results in an annotation *snowed in* of the temperature value from *temp@S1*. I.e., the sensor measurements observed in the time span of the annotation are associated with the annotation. In Figure 3 the processing of the *interpolate* processing element is depicted for an example set of sensor data. The data of sensor S1 is annotated, where the annotation is represented as a star. Since this sensor is an input of processing element *interpolate*, the output of processing element *interpolate* may also contain the annotation. The annotation of the *temp matrix* in Figure 3 is again represented as a star. Thus, annotations also have to be transformed. However, since the annotation is unforeseen, the implementer of the processing element *interpolate* cannot specify how to transform annotation *snowed in*.

Since the processing element *interpolate* is executed for each new sensor measurement, the annotation transformation can be described solely as a structural transformation, further called δ_S. In the above example, the structural transformation δ_S specifies how the annotation of a single temperature value is transformed into annotations of cells in the output matrix. This transformation depends on the used interpolation algorithm and its parametrization T_D and the definition of the data structure transformation T_S.

In particular, the data structure transformation contains information about the size of the matrix and the data types of the cells. Further the data transformation T_D contains the interpolation algorithm and its parametrization. In particular, the distance of cells in the output structure from the cell representing the sensor observation indicates whether or not the cell should be annotated. The parametrization of the influence of distance on the cell data also influences the structural annotation transformation δ_S.

Since the structural annotation transform δ_S is independent of the annotation, it can be provided by the implementer of the transform. The basic assumption is that annotated data are either used and therefore the annotation is ignored or the annotation is acknowledged and thus the data are ignored.

RESEARCH QUESTION 1. *How to build the structural annotation transform δ_S based on data structure and data transforms T_s and T_D?*

An initial idea is to identify a finite set of data structures often used in e-science applications and to investigate their structural transformations. In particular, we plan to use weighing models to express whether an element of a data structure should be annotated or not. The ideas will be evaluated in several case studies.

4.1.2 Temporal resolution

Following the weather monitoring application introduced in section 3 the next step is to compute the monthly average of the previously interpolated values as discussed in the previous section (see Fig 3). This is performed in the average per month (*avg/mth*) processing element (see Fig

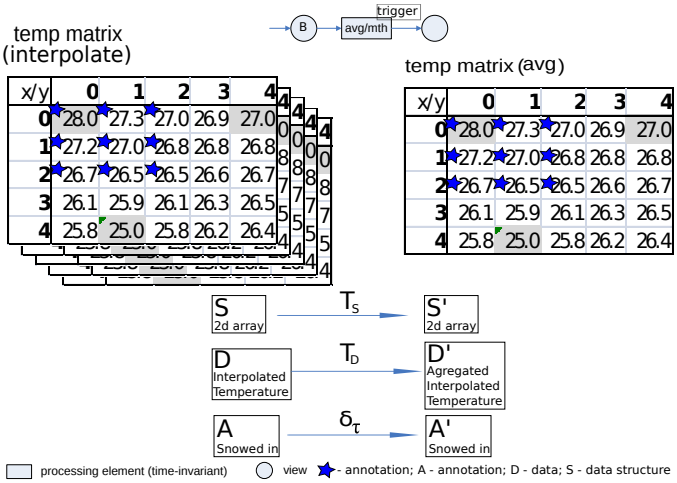

Figure 4: Annotation temporal resolution transform

4). The processing element performs an average over all the *temp matrix(interpolate)* arrays of one month at midnight of the first day of a month. The predicate of using one month of data *temp matrix(interpolate)* arrays is called the buffer predicate B selecting the relevant data from a view, and the predicate when to execute a processing element is called the trigger predicate *Trigger*. The specification of the buffer B and the *Trigger* predicates are part of the specification of the processing element *avg/mth* (See Fig 4).

The input and output data structures, temperature matrices, of the (*avg/mth*) processing element are identical.

Thus, the structural transformation T_s is the identity transform S being identical to S'. Therefore the annotation transform δ_S is also the identity transform, hence can be neglected in further discussion.

The data transformation T_D is based on all matrices selected by the buffer predicate B using a weighted average calculation per cell. Thus, a cell of the output array is given by the weighted average of the cell in all arrays contained in the buffer.

As a consequence, the transformation T_A of annotation A into annotation A' depends on the buffer predicate B and the data transform T_D. These effects are covered by a temporal annotation transform δ_τ. In particular, the temporal annotation transform δ_τ provides a weight of having an annotation per element of the output structure S', in case of the example this is per cell of the *temp matrix (avg)* array.

The weight is influenced by the amount of annotations per cell in the buffer. I.e., the more annotations on a cell per array the higher the weight of having an annotation at the corresponding cell of the output array.

Further, dependent on the data transformation T_D the contribution of each input array to the calculation of the weight varies. If the weighted average calculation is based on a constant weight, then all arrays contribute equally, while a progressive weighting favors the latest temperature arrays.

We have hence identified the two elements contributing to the temporal annotation transform δ_τ: (1) the data transform T_D and (2) the buffer predicate B. Also δ_τ like δ_S is an annotation independent transform and therefore can be provided by the implementer of the transform.

RESEARCH QUESTION 2. *How to build the temporal annotation transform δ_τ based on the data transform T_D and the buffer predicate B?*

The initial idea is to identify a set of aggregation functions often used in e-science applications and to investigate their relationship to the temporal annotation transform δ_τ. While average and sum aggregations appear rather straight forward, extrema aggregation functions like *min* and *max* appear to be more challenging. Since extrema operations only select a single value the temporal annotation transform intuitively is binary. However, we expect that the uniformity of the data influences the weight of the temporal annotation transform. More complex or general aggregation functions like convolution require further investigation.

4.1.3 Annotation significance

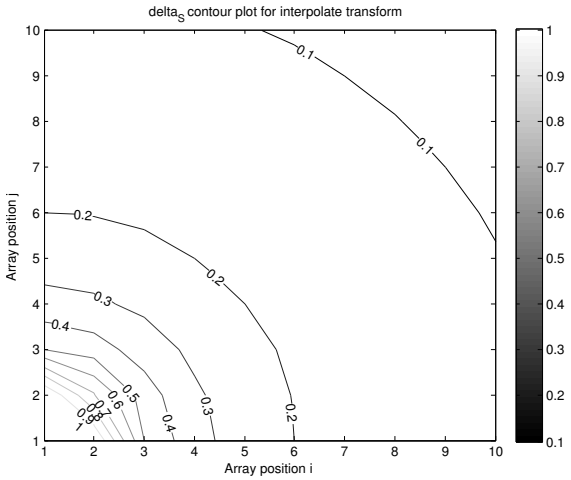

Figure 5: δ_τ contour plot

One question that has not been examined so far with regards to unforeseen annotations is their significance. So far we have looked to the contributions of the annotation transform T_A from the structural δ_S (Sect 4.1.1) and the temporal perspective δ_τ (Sect 4.1.2). Both contributions δ_S and δ_τ can be combined for each output cell of the transform in order to obtain the combined weight of the annotation transform. However the combined value forming T_A does not enable to decide if the annotation is to be propagated or not.

The interpolation transform present in our sample application (See Fig 3 *interpolate*) requires interpolation weights for each one of the three sensors to be built. The weights reflect the contribution of the data from a given sensor to the output values. Hence also the contribution to the propagation of the annotation. Figure 5 represents a contour plot of the propagation weights of sensor *temp@S1*, from our sample application introduced in Section 3 (See Fig 1). However annotations are boolean and there is the need to decide if an annotation is propagated or not. The decision can be made by transforming the weights array (See Fig 5) into a binary array specifying which are the cells that get annotated on output for a given sensor containing the input. To create the binary array all that is required is to threshold the weights array (See Fig 5). A thresholding value, which constitutes the point from which an annotation is significant enough to be propagated, has to be chosen basing the decision on the interpolation weights. Several techniques exist for selecting

such a thresholding value. However, the desired technique has to be applicable to all forms of T_A not only those in which the structural component δ_S is prevalent.

Further the same occurs in the monthly average step (see Fig 4 *avg/mth*) in our use case weather monitoring application introduced in Section 3. In the case of an annotation transform T_A in which the temporal component δ_τ is the only contribution to the transform the weights originate from the weighting function associated with the elements of the input buffer (See Fig 4 B), as introduced in Section 4.1.2. In such a scenario in order to warrant propagation of the weighted sum of all the annotations has to be greater than or equal than the chosen significance threshold. Here too the technique for selecting the threshold value has to be general purpose and avoid local effects such as preventing annotations from being propagated or propagating too many annotations which would then render the interpretation of the data with the annotation difficult.

Besides avoiding local effects in the propagation of the annotations, It is desirable for the technique to be transform, annotation, and data independent. Transform independence would ensure the usability of the same decision technique for all steps in a workflow. Annotation independence would enable the development by the transform developer. Data independence ensures stability of the propagation for a given transform regardless of the data being processed.

This brings us to the following research questions.

RESEARCH QUESTION 3. *Are δ_S and δ_τ sufficient for warranting annotation propagation?*

RESEARCH QUESTION 4. *Can a propagation threshold be found whilst minimizing undesirable local effects?*

The initial idea is to investigate different techniques such as adaptive thresholding, or clustering in order to determine which is the best technique for selecting the propagation threshold of an annotation. Several metrics have to be developed in order to asses the suitability and compare the different techniques.

4.2 Annotation processing

The final step of the weather monitoring application introduced in Section 3 is the visualization step (See Fig 1 *visualize*). During the step the output of the the monthly interpolation operation (See Fig 4 *avg/mth*) is ranked and each of the ranks is then assigned a colour. Each of the cells of the output array (See Fig 4 *temp matrix (avg)*) is mapped to a surface area ($\text{Surface}_{cell}(i,j)$) in the output image (See Fig 4 *contour plot*). The assigned area is then painted with the colour associated to the rank corresponding the data value. The process so far provides a plain contour map (See Fig 6 *i)*) depicting only the data.

The annotations are then displayed over the data following the same cell to image area mapping ($\text{Surface}_{cell}(i,j)$). In the current case the *snowed in* annotation attached to the S1 sensor is displayed over the monthly average temperature contour map (See Fig 6 *ii)*).

This method of representing the annotation is perfectly suited for cases in which the number of annotations is relatively small. For example it is not too difficult to visualize the data if a full system re-calibration annotation is present (See Fig 6 *iii)*). The annotation covers all the area but since there is only one uniform annotation the data can still be visualized correctly.

Further in the case of the same annotation originating from two different sources, i.e. two sensors (S1 and S3) get *snowed in*, the potential areas of overlap render more difficult the visualization of the data (See Fig 6 *iv)*).

With a larger number of annotations being displayed over the data it becomes more difficult to visualize the data. In the case of two different annotations, (See fig 6 *v)*) a different colour is used to represent each annotation. The annotation-to-colour mapping C_A is responsible of supplying a unique colour for each annotation. This mapping enables the user to decide which annotations to display, and which annotations to ignore. However the annotation-to-colour mapping C_A alone is not sufficient to ensure correct visualization of the data and annotations. Since due to the unforeseen nature of the annotations, they may be potentially infinite.

There is then the need to provide a mechanism to organize the annotations and assist the user in selecting which annotations are to be displayed. Such a classification or clustering mechanism is hard to define for unforeseen hence not understandable annotations.

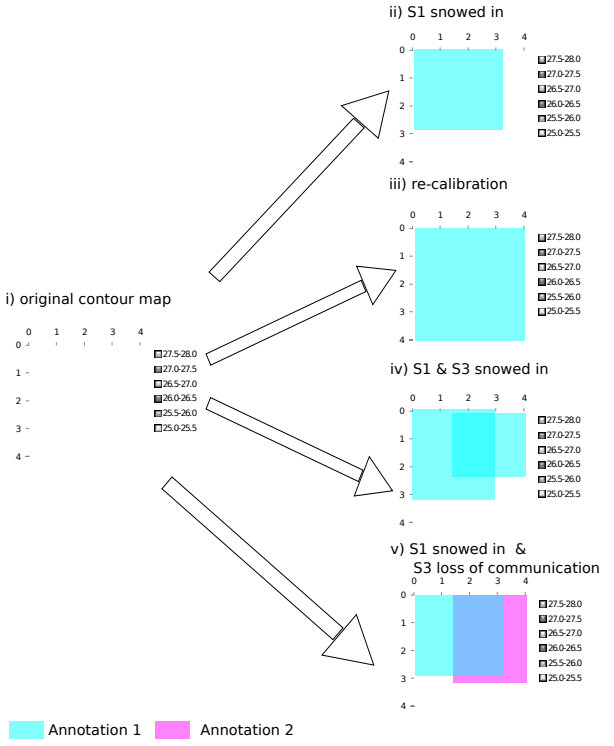

Figure 6: Incomplete understanding of annotations

RESEARCH QUESTION 5. *How can unforeseen annotations, hence not understood, be organized enabling the end use to choose which annotations to be visualized for their application?*

The initial idea for the classification of the annotations would be to use the combination of the annotation transforms T_A across the whole workflow and the significance values. These two values combined can then be used to cluster and separate annotations based on their origin, different weights in the annotation transform T_A or their significance,

closer or further away from the thresholding value. User supplied weights can also be attached to each of the annotations to assist the ranking algorithm.

4.3 Global annotation propagation

So far we have only considered the local propagation of annotations, that is the propagation of annotations across a single processing element. However there is the need to control the propagation of the annotations at a workflow level.

The local solution is based on the annotation transform T_A which facilitates making the decision based on the annotation significance as introduced in Section 4.1.3. However this mechanism depends on the structural and temporal contributions of a transform δ_S and δ_τ which have dependencies on the local transform and not on global propagation parameters. Such a propagation technique may need to be enhanced in order to accommodate for global parameters. It may be suitable for the number or annotations not to decrease or increase more than a certain percent. This would enable for the cumulative effect of all the annotation transforms to deliver a suitable amount of annotations to the visualization step.

The global propagation mechanism described here does however introduce a series of issues.

Research question 6. *Can a suitable local metric be found that warranties suitable global results?*

Research question 7. *Can the local metric be achieved without the need of back propagation?*

The initial idea is to build a metric which would assist in adjusting the significance level of the annotations being propagated trying to maintain constant the number of annotations before and after a transform. Since the transform T can modify the number of output data points through an aggregation or an interpolation the number of annotations is not a suitable metric. However the number of annotations per data point or annotation density ρ_A may be more suitable.

5. CONCLUSION

In stream data processing applications, notably for e-science, unforeseen semantic changes can occur leading to wrong decisions or predictions. An example of unforeseen semantic changes are temperature sensors in an alpine environment, which get snowed in (see Sect 3). The semantic change is explicated by an annotation after the change has been detected in order to provide awareness of the semantic change to the end user via the data processing.

Annotations are propagated alongside the data as part of the data processing. Hence propagating the semantic change through the workflow. The propagation of the annotation occurs across varying data structures, and varying time granularities in the workflow.

Whether an annotation gets propagated or not, depends on three factors: (1) structural contribution, (2) temporal contribution, and (3) annotation significance. The structural and temporal contribution factors are identified to depend on the data transformation algorithms used in the data processing, and are found to be annotation independent. This independence enables the implementation of the annotation propagation, but also implies limitations on the properties of the propagation. Annotation propagation is locally controlled but also requires stability on a workflow, application, level.

Based on the provided research questions, several case studies will be conducted to get further insides in the specific issues and to experiment with the indicated approaches. The case studies will be taken from different data processing of meteorological and hydrological data.

6. REFERENCES

[1] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryvkina, et al. The design of the borealis stream processing engine. In *Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA*. Citeseer, 2005.

[2] K. Aberer, M. Hauswirth, and A. Salehi. Infrastructure for data processing in large-scale interconnected sensor networks. *Mobile Data Management, 2007 International Conference on*, pages 198–205, May 2007.

[3] L. Amini, H. Andrade, R. Bhagwan, F. Eskesen, R. King, P. Selo, Y. Park, and C. Venkatramani. Spc: A distributed, scalable platform for data mining. In *Proceedings of the 4th international workshop on Data mining standards, services and platforms*, page 37. ACM, 2006.

[4] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, and J. Widom. *Data-Stream Management: Processing High-Speed Data Streams*, chapter STREAM: The Stanford Data Stream Management System. Springer, 2006.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16, New York, NY, USA, 2002. ACM.

[6] C. Bisdikian, J. Branch, K. K. Leung, and R. I. Young. A letter soup for the quality of information in sensor networks. *Pervasive Computing and Communications, IEEE International Conference on*, 0:1–6, 2009.

[7] M. Botts, G. Percivall, C. Reed, and J. Davidson. Ogc® sensor web enablement: Overview and high level architecture. *GeoSensor Networks*, pages 175–190, 2008.

[8] S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, V. Raman, F. Reiss, and M. Shah. Telegraphcq: Continuous dataflow processing for an uncertain world. In *CIDR*, 2003.

[9] L. Chiticariu, W. C. Tan, and G. Vijayvargiya. DBNotes: a post-it system for relational databases based on provenance. In F. Özcan, editor, *SIGMOD Conference*, pages 942–944. ACM, 2005.

[10] L. M. (Editor), B. Clifford, J. Freirei, Y. Gilk, P. Grothj, J. Futrelleg, N. Kwasnikowskah, S. Milesd, P. Missiere, J. Myersg, Y. Simmhanf, E. Stephang, and J. V. den Bussche. The open provenance model core specification (v1.1).

http://eprints.ecs.soton.ac.uk/18332/1/opm.pdf, 2009.

[11] M. Y. Eltabakh, W. G. Aref, A. K. Elmagarmid, M. Ouzzani, and Y. N. Silva. Supporting annotations on relations. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 379–390, New York, NY, USA, 2009. ACM.

[12] M. Y. Eltabakh, M. Ouzzani, and W. G. Aref. bdbms - A database management system for biological data. In *CIDR*, pages 196–206. www.crdrdb.org, 2007.

[13] J. Futrelle, J. Gaynor, J. Plutchak, J. D. Myers, R. E. McGrath, P. Bajcsy, J. Kastner, K. Kotwani, J. S. Lee, L. Marini, R. Kooper, T. McLaren, and Y. Liu. Semantic middleware for e-science knowledge spaces. In *MGC '09: Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, pages 1–6, New York, NY, USA, 2009. ACM.

[14] F. Geerts, A. Kementsietsidis, and D. Milano. MONDRIAN: Annotating and querying databases through colors and blocks. In L. Liu, A. Reuter, K.-Y. Whang, and J. Zhang, editors, *ICDE*, page 82. IEEE Computer Society, 2006.

[15] A. Klein, H.-H. Do, G. Hackenbroich, M. Karnstedt, and W. Lehner. Representing data quality for streaming and static data. In *ICDEW '07: Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, pages 3–10, Washington, DC, USA, 2007. IEEE Computer Society.

[16] A. Klein and W. Lehner. Representing data quality in sensor data streaming environments. *J. Data and Information Quality*, 1(2):1–28, 2009.

[17] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan, and L. Varga. The provenance of electronic data. *Commun. ACM*, 51(4):52–58, 2008.

[18] C. on the collaborative large-scale engineering analysis network for environmental research (CLEANER). *CLEANER and NSF's environmental observatories*. The National Academic Press, 2006. http://www.nap.edu/catalog/11657.html.

[19] M. A. Sharaf, J. Beaver, A. Labrinidis, and P. K. Chrysanthis. Balancing energy efficiency and quality of aggregate data in sensor networks. *VLDB J.*, 13(4):384–403, 2004.

[20] Y. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.

[21] Y. L. Simmhan, B. Plale, and D. Gannon. Karma2: Provenance management for data-driven workflows. *International Journal of Web Services Research*, 5(2):1–22, 2008.

[22] D. M. Strong, Y. W. Lee, and R. Y. Wang. Data quality in context. *Commun. ACM*, 40(5):103–110, 1997.

[23] M. Szomszor and L. Moreau. Recording and reasoning over data provenance in web and grid services. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *CoopIS/DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 603–620. Springer, 2003.