
Properties Relevant for Inferring Provenance

Author:
Abdul Ghani
RAJPUT

Supervisors:
Dr. Andreas WOMBACHER
Rezwan Huq, M.Sc

MASTER THESIS

UNIVERSITY OF TWENTE
THE NETHERLANDS

August 16, 2011

Properties Relevant for Inferring Provenance

A thesis submitted to the faculty of Electrical Engineering, Mathematics and
Computer Science, University of Twente, the Netherlands in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCES IN COMPUTER SCIENCE

with specialization in

INFORMATION SYSTEM ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE,
UNIVERSITY OF TWENTE
THE NETHERLANDS

August 16, 2011

Contents

Abstract	v
Acknowledgment	vii
List of Figures	ix
1 Introduction	1
1.1 Motivating Scenarios	2
1.1.1 Supervisory Control and Data Acquisition	2
1.1.2 SwissEX RECORD	3
1.2 Workflow Description	3
1.3 Objectives of Thesis	4
1.4 Research Questions	5
1.5 Thesis Outline	6
2 Related work	7
2.1 Existing Stream Processing Systems	7
2.2 Data Provenance	8
2.3 Existing Data Provenance Techniques	8
2.4 Provenance in Stream Data Processing	9
3 Formal Stream Processing Model	13
3.1 Syntactic Entities of Formal Model	14
3.2 Discrete Time Signal	14
3.3 General Definitions	18
3.4 Simple Stream Processing	22
3.5 Representation of Multiple Output Streams	22
3.6 Representation of Multiple Input Streams	24
3.7 Formalization	24
3.8 Continuity	25
4 Transformation Properties	29
4.1 Classification of Operations	29
4.2 Mapping of Operations	31

4.3	Input Sources	31
4.4	Contributing Sources	32
4.5	Input Tuple Mapping	32
4.6	Output Tuple Mapping	33
5	Case Studies	35
5.1	Case 1: Project Operation	35
5.1.1	Transformation	35
5.1.2	Properties	38
5.2	Case 2: Average Operation	40
5.2.1	Transformation	40
5.2.2	Properties	42
5.3	Case 3: Interpolation	43
5.3.1	Transformation	43
5.3.2	Properties	46
5.4	Case 4: Cartesian Product	47
5.4.1	Transformation	47
5.4.2	Properties	50
5.5	Provenance Example	51
6	Conclusion	55
6.1	Answers to Research Questions	55
6.2	Contributions	57
6.3	Future Work	57
	References	59

Abstract

Provenance is an important requirement for real-time applications, especially when sensors act as a source of streams for large-scale, automated process control and decision control applications. Provenance provides important information that is essential to identify the origin of data, to reproduce the results in real-time applications as well as to interpret and validate the associated scientific results. The term provenance documents the origin of data by explicating the relationship among the input samples, the transformation and the output samples. In this thesis, we present a formal stream processing model based on discrete time signal processing. We use the formal stream processing model to investigate different data transformations and the provenance relevant characteristics of these transformations. The validity of the formal stream processing model and transformation properties is demonstrated by providing the four case studies.

Acknowledgment

Over the last two years, I have received a lot of help and support by many people whom I would like to thank here.

I would not have been able to successfully complete this thesis without the support of supervisors during past seven months. My sincere thanks to Dr. Andreas Wombacher, Dr. Brahmananda Sapkota and Rezwan Huq. They have been a source of inspiration for me throughout the process of the research and writing. Their feedback and insights were always valuable, and never went unused.

I owe my deep gratitude to all of my teachers, who have taught me at Twente. Their wonderful teaching methods enhanced my knowledge of the respective subject and enabled me to complete my studies in time. I also like to extend my sincere thanks to the staff of international office. Special thanks go to Jan Schut because without his support it is not possible for me to come here and complete my studies.

My roommates at the third floor at Zilverling provided a great working environment. I thank them for the laughs and talks we had. I would like to thank the following colleagues and friends whose help in the study period has contributed to achieve this dream. Thanks to Fiazan Ahmed, Fiaza Ahemd, Irfan Ali, Irfan Zafar, M.Aamir, Martin, Klifman, T.Tamoor and Mudassir.

Of course, this acknowledgment would not complete without thanking my mother, brother and sister. Having supported me throughout my university study, I cannot express my gratitude enough. I hope this achievement will cheer them up during these stressful times.

My family (Nida, Fatin and Abdullah) more than deserves to be named here too. Throughout the process of my studies and my graduation research, they have been loving and supportive.

ABDUL GHANI RAJPUT

August 16, 2011.

List of Figures

1.1	Workflow model based on RECORD project scenario	4
2.1	Taxonomy of Provenance	10
3.1	Logical components of the formal model and the idea of figure is taken from [18]	15
3.2	The generic Transformation function	16
3.3	Unit impulse sequence	17
3.4	Unit step Sequence	17
3.5	Example of a Sequence	18
3.6	Sensor Signal Produces an Input Sequence	19
3.7	Input Sequence	20
3.8	Window Sequence	21
3.9	Simple stream processing	22
3.10	Multiple outputs based on the same window sequence	23
3.11	Example of increasing chain of sequences	26
4.1	Types of Transfer Function	34
5.1	Transformation Process of Project Operation	36
5.2	Input Sequence and Window Sequence	37
5.3	Several Transfer Functions is Executed in Parallel	38
5.4	Average Transformation	41
5.5	Interpolation Transformation	44
5.6	Distance based interpolation	46
5.7	Cartesian Product Transformation	48
5.8	Example for overlapping windows	52
5.9	Example for non-overlapping windows	53

Chapter 1

Introduction

Stream data processing has been a hot topic in the database community in this decade. The research on stream data processing has resulted in several publications, formal systems and commercial products.

In this digital era, there are many real-time applications of stream data processing such as location based services (LBSs identify a location of a person) based on user's continuously changing location, e-health care monitoring systems for monitoring patient medical conditions and many more. Most of the real-time applications collect data from source. The source (sensor) produces data continuously. The real-time applications also connect with multiple sources that are spread over wide geographic locations (also called data collection points). The examples of sources are scientific data, sensor data, wireless and sensor networks. These sources are called data streams [11].

A data stream is an infinite sequence of tuples with the timestamps. A tuple is an ordered list of elements in the sequence and the timestamp is used to define the total order over the tuples. Real-time applications are specialized forms of stream data processing. In real-time applications, a large amount of sensor data is processed and transformed in various steps.

In real-time applications, reproducibility is a key requirement and reproducibility means the ability to reproduce the data items. In order to reproduce the data items, data provenance is important. Data provenance [23] documents the origin of data by explicating the relationship among the input data, the algorithm and the processed data. It can be used to identify data because it provides the key facts about the origin of the data.

The research on data provenance has focused on static databases and also in stream data processing, which are discussed in Chapter 2. But there is still a lot to be investigated such as reproducibility in real-time applications. Suppose in a stream processing setup, we have a transformation process T . It is executed on

an input stream X at time n and produces output stream Y . We can re-execute the same transformation process T at any later point in time n_0 (with $n_0 > n$) on the same input stream X and generate exactly the same output stream Y [1]. The ability to reproduce the transformation process for a particular data item in a stream requires transformation properties. The transformation has a number of properties for instance constant mapping. For example, if a user wants to trace back the problem to the corresponding data stream then he needs to have a constant rate of output tuple otherwise user can not handle that. Therefore one important property for inferring provenance is constant mapping or fixed mapping and we have more properties which are discussed in Chapter 4. These transformation properties are used to infer data provenance.

To this end, this thesis will present a formal stream processing model based on discrete time signal processing theory. The formal stream processing model is used to investigate different data transformations and transformation properties relevant for inferring data provenance to ensure reproducibility of data items in real-time applications.

This chapter is organized as follows. In Section 1.1, two motivating scenarios are presented. In Section 1.2, we give a detailed description of a workflow model which is based on a motivating scenario Section 1.1.2. In Section 1.3, we present the objectives of the thesis. In Section 1.4, we state our research questions and sub research questions followed by Section 1.5 that states the complete thesis outline.

1.1 Motivating Scenarios

Due to the growth in technology, the use of real-time application is increasing day-by-day in many domains such as environmental research and medical research. In most of these domains, the real-time applications are designed to collect and process the real-time data which is produced by sensor. In these applications, provenance information is required. In order to show the importance of data provenance in stream data processing we will present two motivating scenarios in the following subsections.

1.1.1 Supervisory Control and Data Acquisition

The Supervisory Control And Data Acquisition (SCADA) application is a real-time application. The SCADA application collects data from multiple sensors and these sensors produce data continuously. The SCADA is a data-acquisition-oriented and an event-driven application [4]. The SCADA is a centralized system which performs process control activities. It also controls entire sites (electrical power transmission and distribution station) from a remote location. For instance, the SCADA electrical system contains up to 50,000 data collection

points and over 3,000 public/ private electric utilities. In that system, failure of any single data collection point can disrupt the entire process flow and cause financial losses to all the customers that receive electricity from the source, due to a blackout [4].

When a blackout event occurs, the actual measured sensor data can be compared with the observed source data. In case of a discrepancy, the SCADA system analysts need to understand what caused the discrepancy and have to understand the data processed on the basis of the streamed sensor data. Thus, analysts must have a mechanism to reproduce the same processing result from past sensor data so that they can find the cause of the discrepancy.

1.1.2 SwissEX RECORD

Another data stream based application is the RECORD project [28]. It is a project of the Swiss Experiment (SwissEx) platform [6]. The SwissEX platform provides a large scale sensor network for environmental research in Switzerland. One of the objectives of the RECORD project is to identify how river restoration affects water quality, both in the river itself and in the groundwater.

In order to collect the environmental changes data due to river restoration, SwissEX deployed several sensors at the weather station. One of them is the sensorscope Meteo station [6]. At the weather station, the deployed sensors measure water temperature, air temperature, wind speed and some other factors related to the experiment like electric conductivity of water [28]. These sensors are deployed in a distributed environment and send the data as streaming data to the data transformation element through a wireless sensor network.

At the research centre, the researchers can collect and use the sensor data to produce graphs and tables for various purposes. For instance, a data transformation element may produce several graphs and tables of an experiment. If researchers want to publish these graphs and tables in scientific journals than the reproducibility of these graphs and tables from original data is required to be able to validate the result afterwards. Therefore, one of the main requirements of the RECORD project is the reproducibility of results.

1.2 Workflow Description

In the previous section, a motivating scenario SwissEX RECORD has been introduced. In which the researchers want to identify how river restoration affects the quality of water. To achieve this objective, a streaming workflow model is required. This section illustrates how the streaming workflow model works. Figure 1.1 shows a workflow model which is based on the RECORD project scenario.

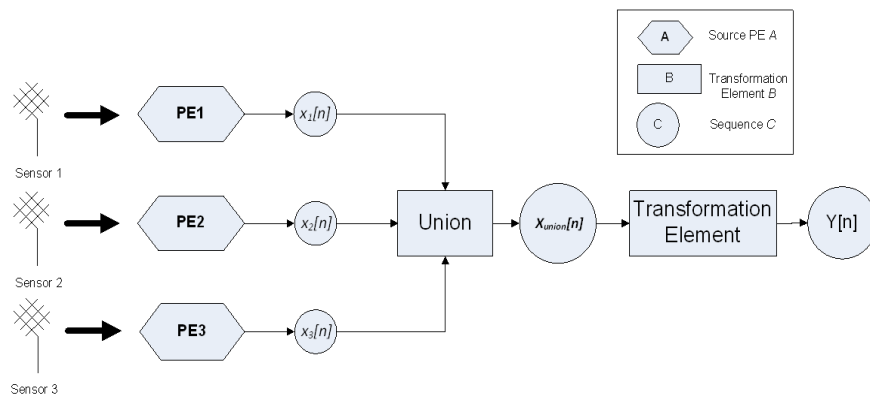


Figure 1.1: Workflow model based on RECORD project scenario

In Figure 1.1, three sensors are collecting the real-time data. These sensors are deployed in three different geographic location of a known region of the river and the region is divided into 3×3 cells of a grid. These sensors send readings of electric conductivity of water to a data transformation element. In order to convert the sensor data in a streaming processing system, we propose a wrapper called *source processing element*. Each sensor is associated with a processing element named PE_1 , PE_2 and PE_3 which provides the data tuples in a sequence $x_1[n]$, $x_2[n]$ and $x_3[n]$ respectively. A sequence is an infinite set of tuples/data with timestamps. These sequences are combined together (by a union operation) which generates a sequence $x_{union}[n]$ as output. It contains all data tuples sent from all the three sensors. The sequence $x_{union}[n]$ will work as input to the transformation element. The transformation element processes the tuples of the input sequence and produces an output sequence or multiple output sequences $y[n]$, depending on the transformation operations used.

Let us look at a concrete example, at the transformation element (as shown in Figure 1.1), an average operation is configured. The average operation acquires tuple from $x_{union}[n]$ and computing last 10 tuples/time space of the input sequence and it executed every 5 seconds. The tuples/time space which is configured for the average operation is called a window and how often average operation is executed, we call a trigger. The details of the trigger and the window are discussed in Chapter 3.

For the rest of the thesis, the example workflow model is used to define the transformation of any operation and answer the potential research questions.

1.3 Objectives of Thesis

The following are the objectives of the thesis.

- Define a formal stream processing model to do calculations over stream processing which is based on an existing stream processing model [9].
- Investigate the data transformations of SQL operations such as Project, Average, Interpolation and Cartesian product using the formal stream processing model.
- Define the formal definitions of data transformation properties.
- Prove the continuity property of the formal stream processing model.

$$F(\cup\chi) = \cup F(\chi)$$

1.4 Research Questions

In order to achieve the objectives of the thesis, the following main research questions are addressed.

- What are the formal definitions of the basic elements of a stream processing model that can be applied to any stream processing systems?
- What are the suitable definitions of transformation properties for inferring provenance?

In order to answers the main research questions, the following sub questions have been defined.

- What is the mathematical formulation of a simple stream processing model?
- What are the mathematical definitions of Project, Average, Interpolation and Cartesian product transformations?
- What are the suitable properties of the data transformations?
- What are the formulas of the data transformation properties?

The formal stream processing model is a mathematical model and an important property of this mathematical model is the continuity property. It is used to provide a constructive procedure for finding the one unique behavior of the transformation. Therefore, we have another research question which is:

- How to prove the continuity property for formal stream processing model?

The answers of these sub-questions provide the answer to the main research questions.

1.5 Thesis Outline

The thesis is organized as follows

- Chapter 2 gives a short review of existing stream data processing systems. It will describe what provenance metadata is, why it is essential in stream data processing and how this can be recorded and retrieved. Chapter 2 also provides the review of provenance in streaming processing.
- To derive the transfer functions of the operations, we need an existing simple stream processing model. In Chapter 3, we presented a short introduction to discrete time signal processing for the formalization of the formal stream processing model. Based on discrete time signal, we provide the definitions of basic elements of the formal stream processing model and discrete time representation of the stream processing.
- Chapter 4 provides the details of transformation properties and formal definitions of properties relevant for tracing provenance.
- In Chapter 5, four case studies are described where the formal stream processing model has been used and tested. At the end of the chapter, two examples are given for the case of overlapping and non-overlapping windows.
- Finally in Chapter 6, conclusions are drawn and future work is discussed.

Chapter 2

Related work

This chapter introduces preliminary concepts which is used throughout this thesis. Section 2.1 starts with a brief discussion on existing stream processing systems. This includes discussions on how stream processing systems handle and process continuous data streams. Section 2.2 introduces the concept of data provenance and the importance of data provenance in stream processing systems. Section 2.3 introduces existing data provenance techniques. This chapter is concluded in Section 2.4, which discusses the data provenance in stream processing system.

2.1 Existing Stream Processing Systems

Stream data processing systems are more and more supporting the execution of continuous tasks. These tasks can be defined as database queries [12]. In [12] data stream processing system is defined as follows:

Data stream processing systems take continuous streams of input data, process that data in certain ways, and produce ongoing results.

Stream data processing systems are used in decision making, process control and real-time applications. Several stream data processing systems have been developed in the research as well as in the commercial sector. Some of which are described below.

STREAM [16] is a stream data processing system. The main objective of the STREAM project was memory management and computing approximate query results. It is an all purpose stream processing system but this system can not support reproducibility of query results.

TelegraphCQ at UC Berkeley [17] is a dataflow system for processing continues queries over data streams. The primary objective of the Telegraph project is

to design for adaptive query processing and shared query evaluation of sensor data. CACQ is an improved form of the Telegraph project and it has the ability to execute multiple queries concurrently [14].

Another popular system in the field of stream data processing is the Aurora system. Aurora system allows users to create the query plans by visually arranging query operators using boxes (corresponding to query operators) and links (corresponding to data flow) paradigm [18]. The extended version of Aurora system is the Borealis [21] system. It supports distributed functionality as well.

IBM delivers a System S [19] solution for the commercial sector. The System S is a stream data processing system (it is also called stream computing system). The System S is designed specifically to handle and process massive amounts of incoming data streams. It supports structured as well as unstructured data stream processing. It can be scaled from one to thousands of computer nodes. For instance, System S can analyze hundreds or thousands of simultaneous data streams (such as stock prices, retail sales, weather reports) and deliver nearly instantaneous analysis to users who need to make split-second decisions [20].

The System S does not support the data provenance functionality and in this system data provenance is important, because later on users may want to track how data are derived as they flow through the system.

All of the above approaches do not provide the functionality of data provenance and cannot regenerate the results. Therefore, a provenance subsystem is needed to collect and store metadata, in order to support reproducibility of results.

2.2 Data Provenance

Provenance means, where is the particular tuple/data item coming from or the origin of data item or the source of a data item. In [7] provenance also defined as the history of ownership of a valued object or work of art or literature. It was originated in the field of Art and it is also called metadata. Provenance can also help to determine the quality of a data item or the authenticity of a data item [13]. In stream data processing, data provenance is important because it not only ensures the integrity of a data item but also identifies the source or origin of a data tuple. In decision support applications, data provenance can be used to validate the decision made by application.

2.3 Existing Data Provenance Techniques

In the domain of information/data processing, [27] is one the first to use the notion of provenance. In [27], authors introduce two ideas of data provenance i.e. *where* and *why* provenance. When executing a query, a set of input data

items is used to produce a set of output data items. To reproduce the output data set, one needs the query as well as the input data items. The set of input data items are referred to as Why-provenance. Where-provenance refers to the location(s) in the source database from which the data was extracted [27]. In [27], authors did not address how to deal with streaming data and associated overlapping windows. It only shows case studies for traditional data.

In [29], authors proposed a method for recording and reasoning over data provenance in web and grid services. The proposed method captures all information on workflow, activities and all datasets to provide provenance data. They created a service oriented architecture (SOA), where they use a specific web service for the recording and querying of provenance data. The method is only works for coarse grained data provenance (the coarse grained data provenance can be defined on relation-level) ; therefore this method cannot achieve reproducibility of results.

In [30], authors recognized a specific class of workflow called data driven workflows. In data driven workflows, data items are first class input parameters to processes that consume and transform the input to generate derived output data. They proposed a framework called Karma2 that records the provenance information on processes as well as on data items. While their proposed framework is closer to the stream processing system than the majority of the research papers on workflows, it does not address the problem, specifically related to stream data processing.

To design a standard provenance model, a series of workshops and conferences have been arranged. During these workshops and conferences participants have discussed a standard provenance model, which is called the Open Provenance Model (OPM)[31]. The OPM is a model for provenance which allows provenance information to be exchanged between systems, by means of a compatibility layer based on a shared provenance model [1]. The OPM define a notion of graphs. The provenance graph is used to identify the casual relationship between artifacts, processes and agents. A limitation of the OPM is that it primarily focuses on the workflow aspect. It is not possible to define what exactly a process does. It also has an advantage that it might to be working with interoperability of different systems [31].

In [32] authors did a survey on data provenance techniques that were used in different projects. On the bases of their survey, they provide a taxonomy of provenance as shown in Figure 2.1¹.

2.4 Provenance in Stream Data Processing

In this era, lots of real-time applications have been developed. Most of the applications are based on mobile networks or sensors networks. Sensor networks,

¹Figure 2.1 is taken from [32].

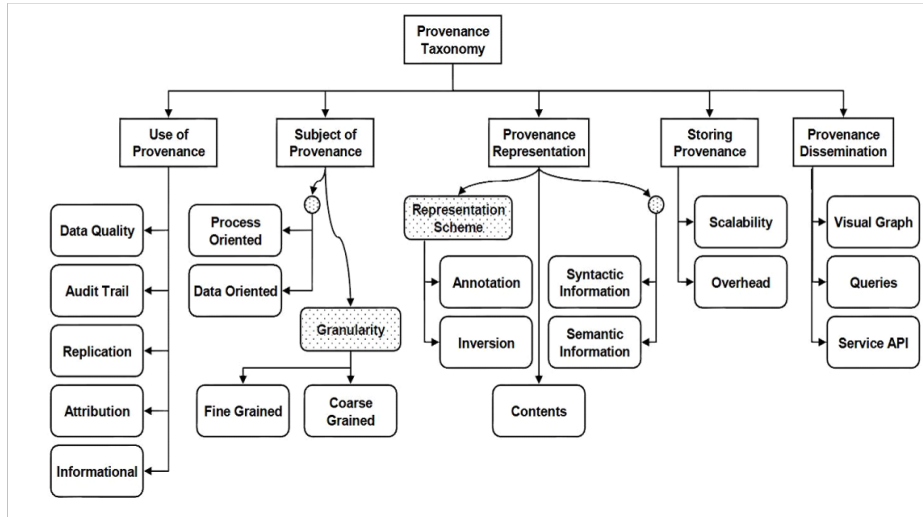


Figure 2.1: Taxonomy of Provenance

which are a typical example of stream data processing and commonly used in diverse applications, such as applications which monitor the water like RECORD project, temperature and earthquake [13].

In these real-time applications, data provenance is crucial because it helps to ensure reproducibility of results and also determining the authenticity as well as quality of data items [13]. Provenance information can be used to recover the input data from the output data item. As described earlier, reproducibility is the key requirement of streaming applications and it is only possible if we can document the provenance information such as where particular data item came from, how it was generated.

First research on data provenance in stream data processing was done by IBM T.J. Watson’s Century [13]. In [15], a framework is provided (referred to as Century) with the purpose of real time analysis of sensor based medical data with data provenance support is provided. In the architecture of Century, a subsystem called data provenance is attached. This subsystem allows users to authenticate and track the origin of events processed or generated by the system. To achieve this, authors designed a Time Value Centric (TVC) provenance model, which uses both process provenance (defined at workflow level) and data provenance (derivation history of the data) in order to define the data item and input source which contributed to a particular data item. However, the approach has only been applied in the medical domain. This paper did not mention formal description of properties (discussed in Chapter 4) relevant for inferring provenance.

Low Overhead Provenance Collection Model [34] is proposed for near-real time

provenance collection in sensor based environmental data stream. In this paper, authors focus on identifying properties that represent provenance of data item from real time environmental data streams. The three main challenges described in [34] are given below:

- Identifying the small unit (data item), for which provenance information is collected.
- Capturing the provenance history of streams and transformation states.
- Tracing the input source of a data stream after the transformation is completed.

A low overhead provenance collection model has been proposed for a meteorology forecasting application.

In [5], authors report their initial idea of achieving fine-grained data provenance using a temporal data model. They theoretically explain the application of the temporal data model to achieve the database state at a given point in time.

Recently [1], proposed an algorithm for inferring fine grained provenance information by applying a temporal data model and using coarse grained data provenance. The algorithm is based on four steps; first step is to identify the coarse grained data provenance (it contains information about the transformation process performed by that particular processing element). Second step is to retrieve the database state. Third step is to reconstruct the processing window based on information provided by the first two steps. The final step is to infer the fine grained data provenance information. In order to infer fine grained data provenance, authors have provided the classification of transformation properties of processing elements, i.e., operations only for constant mapping operations. Such properties are the input sources, contributing sources, input tuple mapping and output tuple mapping. Authors have implemented the algorithm into a real time stream processing system and validated their algorithm.

This thesis is based on the transformation properties of processing elements described in [1]. The details and formal definitions of these properties are discussed in Chapter 4.

Chapter 3

Formal Stream Processing Model

The goal of this chapter is to provide a mathematical framework for stream data processing which is based on discrete time signal processing theory. It can be named as formal stream processing model.

The discrete time signal processing is the theory of representation, transformation and manipulation of signals and the information they contain [22]. The discrete time signal can be represented as a sequence of numbers. The discrete time transformation is a process that maps an input sequence into an output sequence. There are a number of reasons to choose discrete time signal processing to formalize the stream data processing. One important reason is that, the discrete time signal processing allows for the system to be event-triggered, which is often the case in stream data processing. Another reason is that, one of the objectives of the stream data processing is to perform real-time processing on real-time data. Therefore, the discrete time signal processing is common to process real-time data in communication systems, radar, video encoding and stream data processing [22].

This chapter is organized as follows. Section 3.1 provides an overview of the syntactic entities, their graphical representation and symbols used in the formal stream processing model. Section 3.2 introduces the basic concepts of discrete time signal processing theory. This theory can be used to solve the research questions stated in the previous chapter. Section 3.3 provides the general definitions of the input sequence, transformation, window function and the trigger rate. Based on these general definitions, the simplest data stream processing is defined in Section 3.4. The representation of multiple outputs and multiple inputs is illustrated in Section 3.5 and Section 3.6 respectively. Section 3.7 provides the formalization of the model with and without considering the complex data structure. Finally in Section 3.8, a proof of continuity property of formal

stream processing model is given.

3.1 Syntactic Entities of Formal Model

The symbols, formulas and interpretation used in formal stream processing model are syntactic entities [24]. The syntactic entities are the basic requirements to design a formal model [24]. Figure 3.1 shows that the formal stream processing model is based on symbols, string of symbols, well-formed formulas, interpretation of the formulas and theorems. In order to define a transformation element, syntactic entities of the formal stream processing model are required because syntactic entities are used to define the transformation element.

The list of symbols, used in our formal stream processing model, and their description [25] are given in Table 3.1.

S.No	Symbols	Description
1	$x[n]$	Represents an input sequence, generated by an input source.
2	$y[n]$	Represents the output of the transformation.
3	n	Particular point in time in the input sequence.
4	$w(n, x[n])$	Represents a window function.
5	n_w	Is used to represent the window size of the window sequence.
6	τ	Is used to represent the trigger in the formal model.
7	o	Used to represent the offset.
8	I	Represents the number of input sources.
9	$T\{.\}$	Shows a transformation function T, that maps an input to an output.
10	m	Shows the total number of transformation or output
11	j'	Represents the particular output and its value goes to 1,2,3,...,m.
12	l	Represents the particular transformation and its value goes to 1,2,3,...,m.

Table 3.1: List of Symbols used in FSPM

3.2 Discrete Time Signal

The formal stream processing model is based on discrete time signal theory, which is a theory of representing discrete time signals by a sequence of number

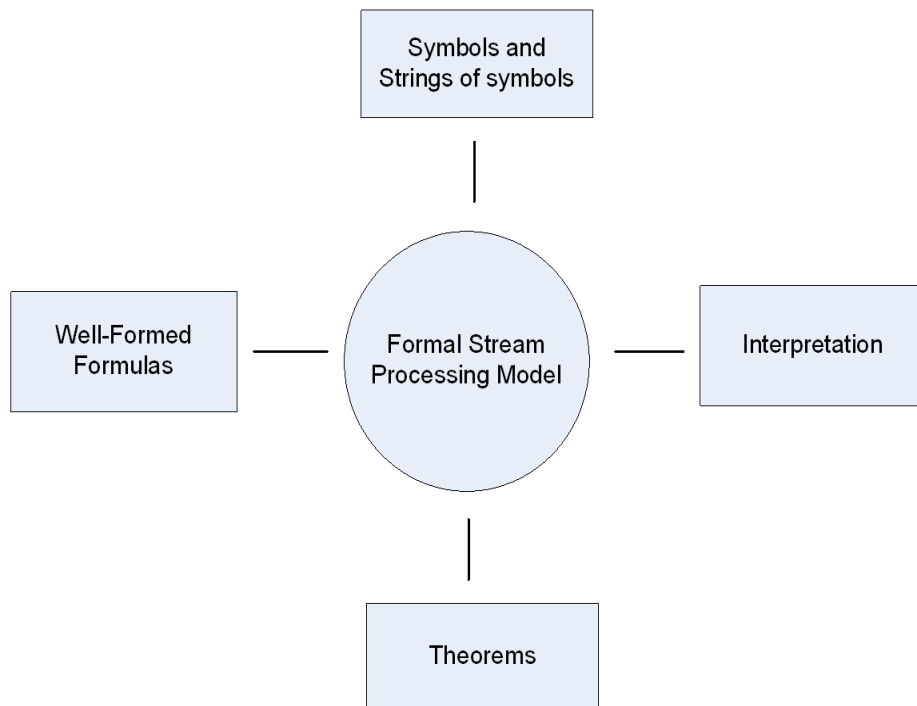


Figure 3.1: Logical components of the formal model and the idea of figure is taken from [18]

and the transformation of these signals [22]. The mathematical representation of the discrete time signal is defined below:

$$\text{Discrete time signal : } n \in \mathbb{Z} \rightarrow x[n]$$

Where

index n represents the sequential values of time,

$x[n]$, the n th number in the sequence, is called a sample.

the complete sequence is represented as $\{x[n]\}$.

In the used stream processing model, a stream is called a sequence. The formal model is process stream or set of streams. Therefore we can say that, a stream is simply a discrete time sequence or discrete time signal.

A transformation is a discrete time system. A discrete time system maps an input sequence $\{x[n]\}$ to an output sequence $\{y[n]\}$; An equivalent block diagram is shown in Figure 3.2.

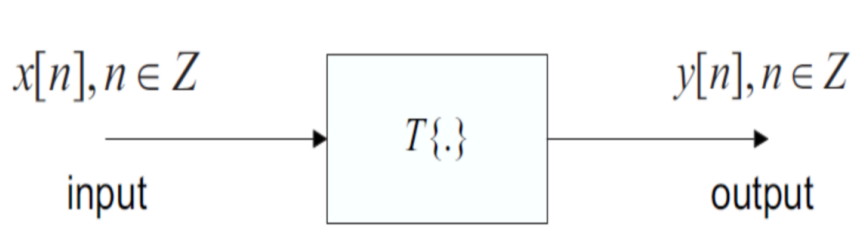


Figure 3.2: The generic Transformation function

In order to define a transformation of an operation, some basic sequences and sequence of operations are required.

Unit Impulse

The unit impulse or unit sample sequence (Figure 3.3) is a generalized function depending on n such that it is zero for all values of n except when the value of n is zero.

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$$

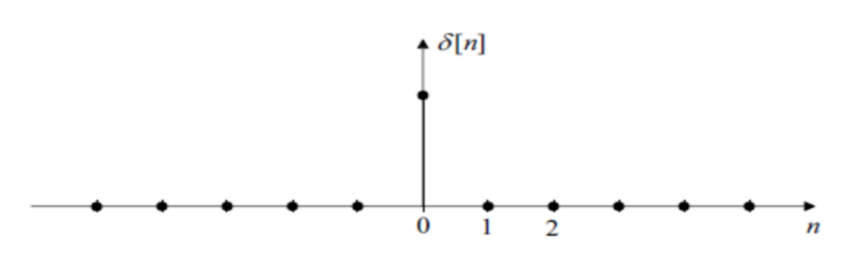


Figure 3.3: Unit impulse sequence

Unit Step

The unit step response or unit step sequence (Figure 3.5) is given by

$$u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & n < 0 \end{cases}$$

The unit step response is simply an on-off switch which is very useful in discrete time signal processing.

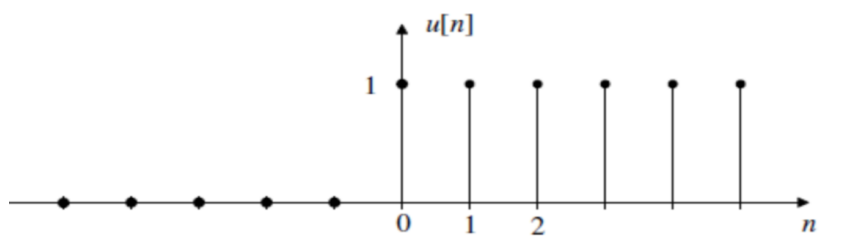


Figure 3.4: Unit step Sequence

Delay or shift by integer k

$$y[n] = x[n - k] \quad -\infty < n < \infty$$

when, $k \geq 0$, sequence of $x[n]$ shifted by k units to the right.

$k < 0$, sequence of $x[n]$ shifted by k units to the left.

Any sequence can be represented as a sum of scaled, delayed impulses. For example the sequence $x[n]$ in Figure 3.5 can be expressed as:

$$x[n] = a_{-3}\delta[n + 3] + a_1\delta[n - 1] + a_2\delta[n - 2] + a_5\delta[n - 5]$$

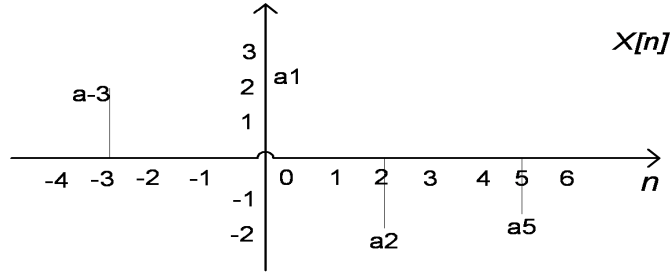


Figure 3.5: Example of a Sequence

More generally, any sequence can be represented as:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n - k]$$

Finally, these concepts of discrete time signal theory help us to represent the formal definitions of stream processing model. The formal model will allow us to define the properties relevant for inferring provenance.

3.3 General Definitions

The fundamental elements of any stream processing system are data streams, transformation element, window, trigger and offset. There are number of definitions available in the literature for these fundamental elements. In this thesis we try to provide the formal definition of these elements. The element definitions are as follows.

Input Sequence

In our model, the input data arrives from one or more continuous data streams. Normally, these data streams are produced by sensors. These data streams are represented as input sequences in our model. The input sequence represents the measurement/record of a sensor. The input sequence contains more than one element. Each of these elements is called a sample which represents one measurement [22].

Definition 1 *Input Sequence: An input sequence is a data stream used by a transformation function. It is a sequence of number x , where the n th number in the input sequence is denoted as $x[n]$:*

$$x = \{x[n]\} \quad -\infty < n < \infty \quad (3.1)$$

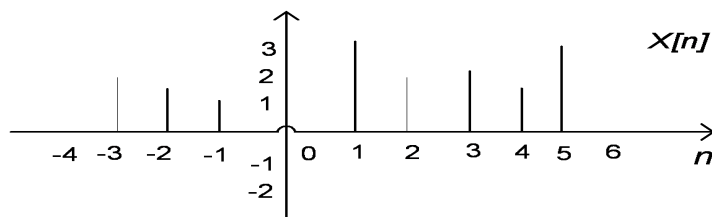


Figure 3.6: Sensor Signal Produces an Input Sequence

Where n is an integer number which represents the measurement/record of a sensor.

Note that from this definition, an input sequence is defined only for integer values of n and refers to the complete sequence simply by $\{x[n]\}$. For example, the infinite length sequence as shown in Figure 3.6 is represented by the following sequence of numbers.

$$x = \{\dots, x[1], x[2], x[3], \dots\}$$

Transformation

The transformation is a transfer function which takes finitely many input sequences as input and gives finitely many sequences as output. The number of input and output depends on an operation. The formal definition of the transformation is given as:

Definition 2 Transformation: Let $\{x_i[n]\}$ be the input sequences and $\{y_{j'}[n]\}$ be the output sequences for $1 \leq i \leq I$ and $1 \leq j' \leq m$. A transformation is a transfer function T defined as:

$$\prod_{j'=1}^m y_{j'}[n] = T\left\{\prod_{i=1}^I x_i[n]\right\}.$$

Where m is the total number of output and I is the total number of input sequence.

Window

For the processing of sensor data, most of the real-time applications are interested in the most recent samples of the input sequences. Therefore, a time

window is defined to select the most recent samples of the input sequence. A window always consists of two end-points and a window size. The end-points are moving or fixed. Windows are either time based or tuple based [1]. In this thesis, we used time based window. Why we used time based window? Because we have a model that supports only time instead of tuple which is based on IDs. The formal stream processing model supports time based sliding window because a time stamp is associated with each sample of the input sequence. The sliding window is a window type in which both end-points move. In the sliding window, the window size is always constant. To represent the window in our formal model, we have defined a window function. The formal definition is given as follows:

Definition 3 *Window Function:* A window function is applied on the input sequence (Definition 1) in order to select a subset of the input sequence $\{x[n]\}$. This function selects subset of n_w elements in the sequence where n_w is the window size. Window function is defined as:

$$w(n, \{x[n]\}) = \sum_{k=0}^{n_w-1} x[n']\delta[n - n' - k] \quad -\infty < n', n < \infty$$

which is also equivalent to:

$$w(n, \{x[n]\}) = \sum_{k=n-n_w+1}^n x[n']\delta[n' - k] \quad -\infty < n', n < \infty \quad (3.2)$$

The output of the window function $w(n, \{x[n]\})$ is called the window sequence. The window sequence is nothing more than a sum of delayed impulses (defined in Section 3.2) multiplied by the corresponding samples of the sequence $\{x[n]\}$ at the particular point in time n . The resulting sequence can be represented in terms of time. Example 3.1, describing the working of window function.

Example 3.1 Suppose we have an input sequence $\{x[n]\}$ as shown in the Figure 3.7. To select subset of the sequence, window function is applied on the input sequence.

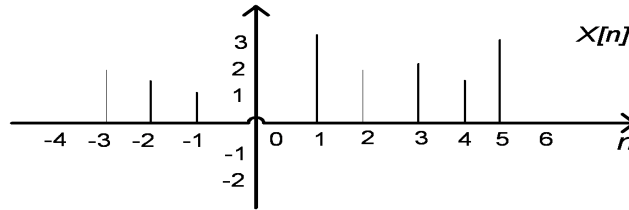


Figure 3.7: Input Sequence

The samples involved in computation of the window sequence are $k = 3$ to 5 with window size $n_w = 3$ and $n = 5$. The result of the window function is shown in Figure 3.8. By putting these parameters to the window function formula, we get:

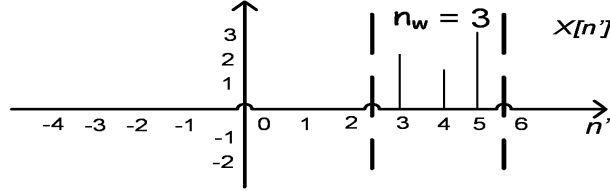


Figure 3.8: Window Sequence

$$w(5, \{x[n]\}) = \sum_{k=5-3+1}^5 x[n']\delta[n' - k]$$

$$w(5, \{x[n]\}) = \sum_{k=3}^5 x[n']\delta[n' - 3]$$

$$w(5, \{x[n]\}) = x[n']\delta[n' - 3] + x[n']\delta[n' - 4] + x[n']\delta[n' - 5]$$

when n' is 3 from replicated sequence therefore, we get:

$$w(5, \{x[n]\}) = x[3]$$

Similarly the output of the $w(5, \{x[n]\}) = x[4]$, when $n' = 4$ and $w(5, \{x[n]\}) = x[5]$, when $n' = 5$.

Trigger Rate

A trigger rate represents the data driven control flow of the data workflow. Data-driven workflows are executed in an order determined by conditional expressions [8]. Triggers are important in a stream processing. It is used to specify when a transformation element should execute. In general, there are two types of triggers, namely time based triggers and tuple based triggers. A time based trigger executes at fixed intervals, while a tuple based trigger executes when a new tuple arrives [1]. The formal model is based on time based triggers since the formal model is based only for time (we do not have a model for IDs).

Definition 4 *Trigger Rate*: τ is a trigger rate over a sequence which specifies when a transformation element is executed. It is defined for all values of n and applied again with a unit impulse function. The Trigger Offset, (o) determines

how many samples are skipped at the beginning of the total record before samples are transferred to the window. Which is defined as:

$$\delta[n\% \tau - o]$$

The transformation element is defined for all values of n , based on the trigger the transformation element is only supposed to be defined at the moments where the trigger is enabled. Thus, for a transformation $T\{\cdot\}$, a trigger is applied with a unit sample (i.e. $\delta[n\% \tau - o] = 1$).

3.4 Simple Stream Processing

The simple stream processing is based on a transformation function that maps input data contained in a window sequence producing an output sequence, where the transformation function is executed after arrival of every τ elements of the input sequence. It shows how to process and integrate the input sequence to produce the output sample as shown in Figure 3.9.



Figure 3.9: Simple stream processing

Based on the above definition, the simple possible stream processing can be defined mathematically as,

$$y[n] = \delta[n\% \tau - o] T \{w(n, \{x[n]\})\} \quad -\infty < n < \infty \quad (3.3)$$

with window size n_w , trigger offset o and trigger rate τ .

3.5 Representation of Multiple Output Streams

Equation 3.3 shows that when we execute a transformation function based on the same window sequence (where window size is one) that contained a single sample, it produces a single output value. The window sequence contains more than one sample, the transformation element produces different outputs. All these outputs must be associated with the same time index n . Since it is not possible, it is modeled as several transformation functions performed in parallel thereby producing several output sequences [9]. Thus,

$$y_1[n] = \delta[n\% \tau - o]T_1\{w(n, \{x[n]\})\}$$

⋮

$$y_l[n] = \delta[n\% \tau - o]T_l\{w(n, \{x[n]\})\}$$

To represent the multiple outputs of the transformation element, we used the concept of the direct product. The direct product is defined on two algebras X and Y, giving a new one. It can be represented as infix notation \times , or prefix notation \prod . The direct product of $X \times Y$ is given by the Cartesian product of X,Y together with a properly defined formation on the product set.

Definition 5 *Multiple outputs:* Let $y_1[n], y_2[n], y_3[n], \dots, y_m[n]$ be the outputs of $T_1\{.\}, T_2\{.\}, T_3\{.\}, \dots, T_m\{.\}$ based on the same window sequence $w(n, \{x[n]\})$ of input sequence for all values of n , then multiple output can be represented by

$$\prod_{j'=1}^m y_i[n] = \prod_{l=1}^m T_l\{.\}$$

$$\prod_{j'=1}^m y_i[n] = \prod_{l=1}^m \delta[n\% \tau - o]T_l\{w(n, \{x[n]\})\}$$

where m is the total number of output.

Figure 3.10 shows the graphical representation of multiple outputs based on the same window sequence. The direct product of the output sequence can be interpreted as a sequence of output tuples. In definition 5, we assumed that the number of output is fixed to m .

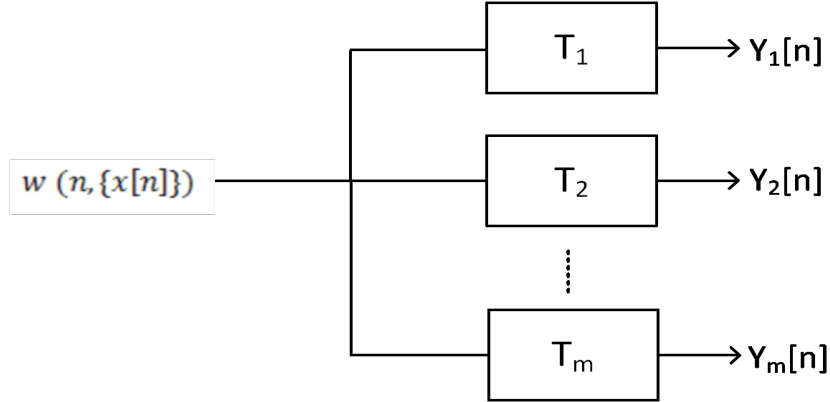


Figure 3.10: Multiple outputs based on the same window sequence

3.6 Representation of Multiple Input Streams

The concept of multiple input streams is common in stream data processing and in mathematics. For instance, union and Cartesian Product can take more than one sequence as input. In order to carry out the transformation of these processing elements, we have to extend the simple stream processing model to support multiple input streams.

Definition 6 *Multiple input streams: Let us we have multiple window sequences $w(n_1, \{x_1[n]\}), \dots, w(n_i, \{x_i[n]\})$ and each window has a different window size $n_{w_1} \dots n_{w_i}$. Let these windows are input to a transformation function such as:*

$$y[n] = \delta[n\% \tau - o]T\{w(n_1, \{x_1[n]\}), \dots, w(n_i, \{x_i[n]\})\} \quad -\infty < n', n < \infty$$

Multiple input streams can also be defined in terms of a direct product again, that is:

$$y[n] = \delta[n\% \tau - o]T\left\{\prod_{i=1}^I w(n_i, \{x_i[n]\})\right\} \quad -\infty < n < \infty$$

where I is the total number of input stream/source.

3.7 Formalization

This section combines the definitions introduced before in order to define the formal stream processing model. Equation 3.4 shows the mathematical description of the formal stream processing model. This formal model will be used to do calculations over stream processing. In Equation 3.4, the structure of the input sequence and the output sequence is not considered. It is, therefore, possible to include the more complex data structure of $y_{j'}[n]$ and $x_i[n]$ in Equation 3.4. The resulting formal stream processing which includes more complex data structure is given in Equation 3.5.

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty \quad (3.4)$$

$$\prod_{j'=1}^m \prod_{j''=1}^{d_{j', y_{j'}}} y_{j', j''}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \prod_{j_i=1}^{d_{x_i}} \{x_{i, j_i}[n]\}) \right\} \quad -\infty < n < \infty \quad (3.5)$$

Where,

j' and $l = 1, 2, 3 \dots m$, where m being the maximum number of outputs of the processing element

$d_{j', y_{j'}}$ is the dimensionality of the data structure of the j' th output sequence $y_{j'}$

I is the number of input sequences

d_{x_i} is the dimensionality of the data structure of the i th input sequence x_i

In this thesis, dimensionality of the input data is not considered because the data structure information of the input data is not available in advance. The formal model (without considering the complex data structure) Equation 3.4 is used to identify the data transformation and transformation properties for inferring provenance, which are discussed in next chapter.

3.8 Continuity

In this section, we provide a simple proof of a continuity property of the formal stream processing model. The proof of the method is essentially the same as in [26] but the contribution here the proof of continuity property using the notations of formal stream processing model.

As per the Kahn Process Network [26], let $\{x[n]\}$ denotes the sequence of values in the stream, which is itself totally ordered set. In our formal stream processing model, the order relationship is not present because every sequence is defined from $-\infty$ to ∞ , as shown in Figure 3.11.

To define a partial order relationship in our formal model, let us consider a prefix ordering of sequences, where $x_1[n] \sqsubseteq x_2[n]$, if $x_1[n]$ is a prefix of $x_2[n]$ (i.e., if the first values of $x_2[n]$ are exactly those in $x_1[n]$) in X . Where X denotes the set of finite and infinite sequences as shown in Equation 3.6.

$$X = \{x_1[n], x_2[n], x_3[n], \dots\} = \bigcup_{i=1}^{\infty} \{x_i[n]\} \quad 1 \leq i \leq \infty \quad (3.6)$$

In Equation 3.6, X is a complete partial order set, if it holds the following relationship between sequences.

$$x_i[n] \sqsubseteq x_j[n] \Leftrightarrow x_i[n] = x_j[n] \cdot u[-i]$$

The above relationship is defined as the complete partial order (CPO) in our formal stream processing model. Therefore, the set X is a complete partial order with the prefix order defining the ordering. A complete partial order is a partial order with a bottom element where every chain has a least upper bound (LUB)

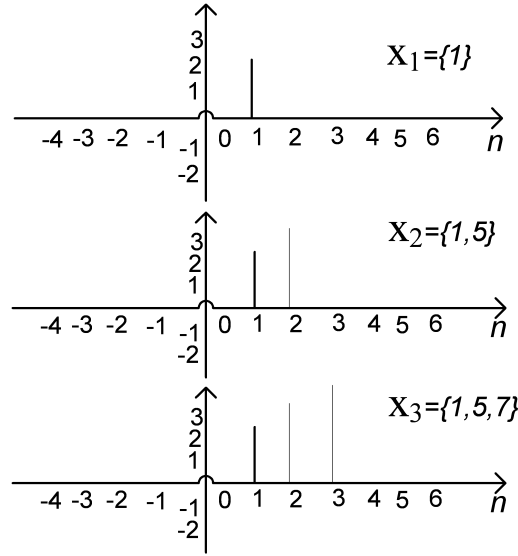


Figure 3.11: Example of increasing chain of sequences

[26]. A least upper bound (LUB), written $\sqcup X$, is an upper bound that is a prefix of every other upper bound. The term $(x_j[n] \cdot u[-i])$ indicates that when $x_j[n]$ is multiplied with the unit step sequence then we get the $x_i[n]$ sequence.

In our formal stream processing model, usually T is executed on a sequence. Now we can extend the definition of T in order to execute and support chain of sequences such as:

$$T(X) = \bigcup_{x[n] \in X} T\{x[n]\}$$

Definition: Let X and Y be the CPO's. A transformation $T : X \rightarrow Y$ is continuous if for each directed subset $x[n]$ of X , we have $T(\sqcup X) = \sqcup T(X)$. We denote the set of all continuous transformation from X to Y by $[X \rightarrow Y]$.

In our formal stream processing model, a transformation takes m input and n outputs such as $T : X^m \rightarrow Y^n$. Let a transformation is defined as below:

$$T(x[n]) = \begin{cases} y[n], & \text{if } x[n] \sqsubseteq X \\ 0 & \text{otherwise} \end{cases}$$

Theorem: The above transformation is Continuous.

Proof. Consider a chain of sequences $X = \{x_1[n], x_2[n], x_3[n], \dots\}$, we need to show that $T(\sqcup X) = \sqcup T(X)$. Write $T(X) = \bigcup_{x[n] \in X} T\{x[n]\}$. ■

Taking R.H.S:

$$\sqcup T(X) = \sqcup\{T(x_1[n]), T(x_2[n]), \dots\}$$

Since X is an increasing chain, it has a least upper bound as per the partial order relationship defined above. Suppose the LUB is $x[n]$, then output is:

$$\begin{aligned}\sqcup T(X) &= \sqcup\{T(x_1[n]), T(x_2[n]), \dots, T(x[n])\} \\ \sqcup T(X) &= x[n] = y[n]\end{aligned}$$

Similarly for L.H.S:

$$\begin{aligned}T(\sqcup X) &= T(\sqcup\{x_1[n], x_2[n], \dots, x[n]\}) \\ T(\sqcup X) &= x[n] = y[n]\end{aligned}$$

Thus, in both cases, $T(\sqcup X) = \sqcup T(X)$, so T is continuous.

Chapter 4

Transformation Properties

The goal of this chapter is to provide the formal definitions of transformation properties for inferring provenance. In Section 1.2, a workflow model was described, in which transformation is an important element. The transformation has a number of properties that makes it useful for inferring provenance. These transformation properties are: input sources, contributing sources, input tuple mapping, output tuple mapping and mapping of operations. These are classified and discussed in [1] as required for reproducibility of results in e-science applications. Based on this classification, the formal definitions of the transformation properties are provided in this chapter.

The remainder of the chapter is organized as follow. Section 4.1 provides the classification of operation. Section 4.2 explains the mapping of operations and provides the formal definition of mapping. Section 4.3 describes the input sources property and its formal definition. Section 4.4 discusses about the contributing sources property and provides the formal definition. Section 4.5 explains and defines the formal definition of input tuple mapping. Finally, Section 4.6 defines the output tuple mapping and its formal definition.

4.1 Classification of Operations

To formalize the definitions of transformation properties, the data transformation of four SQL operations are considered. These are: Project, Average, Interpolation and Cartesian product. Each of these data transformations have a set of properties, such as the ratio of mapping from input to output tuples is a transformation property. The explanation of all these properties is described in Table 4.1.

In Figure 4.1, the graphical representation of considered transformation is provided. The transformation of Project, Average, Interpolation and Cartesian

Properties	Description	Supporting operation
Mapping operation	SQL operations which maintain fixed ratio of mapping input to output are Constant mapping operations.	Project, Average, Cartesian Product and Interpolation operations.
	SQL operations which do not maintain fixed ratio of mapping input to output are called Variable mapping operation.	Select operation (did not considered in this thesis)
Input Sources	Transformations that have only a single sequence as input	Project, Average and Interpolation
	Transformations that have multiple sequences as input	Cartesian product
Contributing Sources	It checks whether the creation of an output sample depends on sample from single or multiple contributing sources.	Cartesian product
Input Tuple Mapping	Specifies whether only a single or multiple input samples contribute to producing exactly one output samples.	Project and Cartesian project operations are single input tuple mapping, while average and interpolation are multiple input tuple mappings.
Output Tuple Mapping	Distinguishes whether the execution of an operation produces a single or multiple output tuples per input tuple.	Project, average and Cartesian product operation are single output tuple mappings, while interpolation is a multiple output tuple mapping.

Table 4.1: Transformation Properties and their descriptions

product are constant mapping operations, which are separated by black solid line. The Select operation is a variable mapping operation which is not considered in this thesis.

Figure 4.1 shows that the Project, Average and Interpolation operation are single input source operations. The Cartesian product operation is a multiple input source operation.

It also shows that Project transformation takes a single element of the input sequence and produces a single element at the output sequence. Thus, the ratio is 1 : 1. The Average transformation takes three input elements of the input sequence and produced single output, therefore the ratio is 3 : 1.

Since the Cartesian product is a multiple input source operation it takes one input element from each source and produces one output element as shown in Figure 4.1. Therefore, the ratio of Cartesian product is (1,1) : 1. These ratios are again reflected in the input and output tuple mapping criteria in Table 4.2.

Operation	Mapping Operation	Input Sources	Contributing Sources	Input Tuple Mapping	Output Tuple Mapping
Project	Constant	Single	Not applicable	Single	Single
Average	Constant	Single	Not applicable	Multiple	Single
Interpolation	Constant	Single	Not applicable	Multiple	Multiple
Cartesian Product	Constant	Multiple	Multiple	Single	Multiple

Table 4.2 Classes of operation

4.2 Mapping of Operations

Based on the classification of operations described in Section 4.1, the formal definition of a mapping is defined in this section. The two types of transformations are possible: constant mapping and variable mapping transformations. The constant mapping transformations have a fixed ratio. The variable mapping transformations do not maintain a fixed ratio of input to output mapping as described in Table 4.1. Let us give the formal definition.

Definition 7 *Constant Mapping Transfer Function:* $T : \{w(n, \{x[n]\})\} \rightarrow \{y[n]\}$ is called constant mapping transfer function if the mapping ratio of $\{w(n, \{x[n]\})\}$ to $\{y[n]\}$ is fixed for all values of n . If it is not fixed, then it is a variable mapping.

4.3 Input Sources

In our formal stream processing model, one of the important transformation properties is input sources. This property is used to find the number of input sources that contribute to produce an output tuple.

The input sources are input sequences (see Definition 1). The transfer functions takes one or more input sources, processes them and produces one or more derived output sequences. The single input source transfer functions do have a single input sequence, while multiple input source transfer functions have multiple input sequences as inputs. Let us give a formal definition.

Definition 8 *Input Sources:* Let $y[n]$ be an output sequence of a transfer function T , where T is applied on one or more input sequences as per Definition 6, then:

$$y[n] = \delta[n\% \tau - o] T \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

where I is used to denote the number of input sources contributing to T to produce the output, $I \in \mathbb{N}$, where \mathbb{N} is the natural number. Therefore:

$$\text{Input sources} = \begin{cases} \text{Multiple} & \text{if } I > 1 \\ \text{Single} & \text{else} \end{cases}$$

4.4 Contributing Sources

The formal definition of this property will be used to find the creation of an output sample is based on samples from a single or multiple input sequences. This property is only applicable for those transformations which takes $I > 1$ input sequences as an input. The formal definition of the property is given below.

Definition 9 *Contributing Sources:* Let T be a transfer function which have multiple input sources as input, such as $w(n_1, \{x_1[n]\}) \times \dots \times w(n'_i, \{x_i[n']\})$, then contributing sources property defined as:

$$T \{w(n_1, \{x_1[n]\}) \times \dots \times w(n'_i, \{x_i[n']\})\} = T \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\}$$

$$\text{Contributing sources} = \begin{cases} \text{Multiple} & \text{For } I > 1 \text{ and} \\ & \text{each } I \text{ is contributed in } T \\ \text{Single} & \text{For } I > 1 \text{ and} \\ & \text{only a single source is contributed in } T \\ \text{Not Applicable} & \text{For } I = 1 \end{cases}$$

4.5 Input Tuple Mapping

The input tuple mapping property is used to find a given sample, related to the input source that is used by the transfer function. The formal definition is as follows:

Definition 10 *Input Tuple Mapping (ITM):* Let T be a transfer function and applied on a window (see definition 3) $\{w(n, \{x[n]\})\}$, which is equivalent to:

$$T \{w(n, \{x[n]\})\} = T \left\{ \sum_{k=n-n_w+1}^n x[n'] \delta[n' - k] \right\} \quad -\infty < n', n < \infty$$

If the output of the transfer function is an accumulated sum of the value at index n and all previous values of the input sequence $\{x[n]\}$ then input tuple mapping is multiple else input tuple mapping is single.

4.6 Output Tuple Mapping

The most important and difficult property is the output tuple mapping for inferring provenance data. It depends on input tuple mapping as well as on an input source. In this property, dimensionality of input data is important because the output data dimensionality is different from the input data dimensionality. But In this thesis, we did not consider the dimensionality of the input data. Output tuple mapping distinguishes whether the execution of a transformation produces a single or multiple output tuple per input tuple mapping [1]. The output tuple mapping is a decimal or a fractional number when it is calculated. The formal definition is given as:

Definition 11 *Output Tuple Mapping (OTM): Let T be a transformation that maps the n_w (window size) samples per source to produce the m number of output samples, then the output tuple mapping is defined as:*

$$OTM = r \times \sum_{i=1}^I ITM_i \quad \begin{cases} \text{Multiple} & OTM > 1 \\ \text{Single} & \text{otherwise} \end{cases}$$

where OTM = output tuple mapping

ITM_i = input tuple mapping per source

$$r = \frac{m}{\sum_{i=1}^I n_{wi}} \text{ where } I \text{ is the total number of input sources}$$

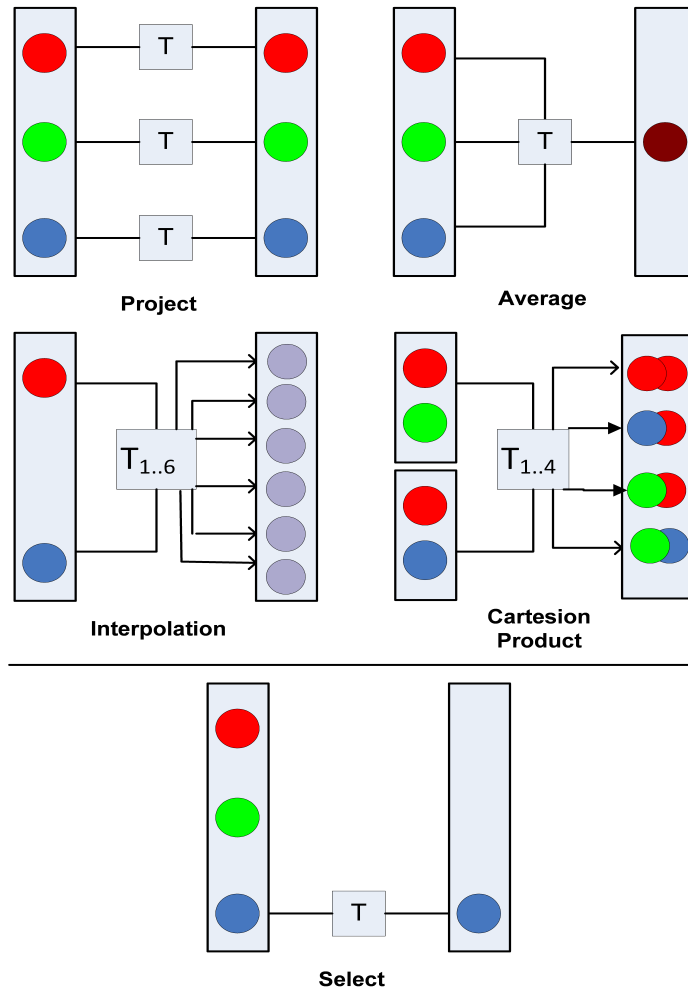


Figure 4.1: Types of Transfer Function

Chapter 5

Case Studies

The primary goal of this chapter is to derive the transformation of the Project, Average, Interpolation and Cartesian product to exemplify the formal stream processing model and formal definitions of transformation properties described in the previous chapters.

5.1 Case 1: Project Operation

5.1.1 Transformation

This section derives the transformation definition of Project operation using the formal stream processing model. We begin by explaining the concept of Project operation.

The Project operation is a SQL operation which is also called projection. A project is an unary transformation that can be applied on a single input sequence. The transformation process takes the input sequence (see Definition 1) and computes the sub-samples of the input sequence. In other words, it reduces the n th sample from the input sequence. Similarly in the databases, projection of a relational database table is a new table containing a subset of the original columns.

Figure 5.1 shows the graphical representation of the project transformation process and also shows that the sensor produces an input sequence which is $x[n]$. The input sequence is passed to the project transformation (in Figure 5.1, big square box represents the project transformation process). The window function (see Definition 2) is applied on the input sequence to cover the most recent samples of the input sequence since the sensor is producing the data continuously. The output of the window function is the window sequence. Based on the window size of the sequence, the multiple outputs are produced by project

transformation i.e. $Y_1[n]$, $Y_2[n]$ and $Y_m[n]$ as shown in Figure 5.1. All outputs are associated with the same time n .

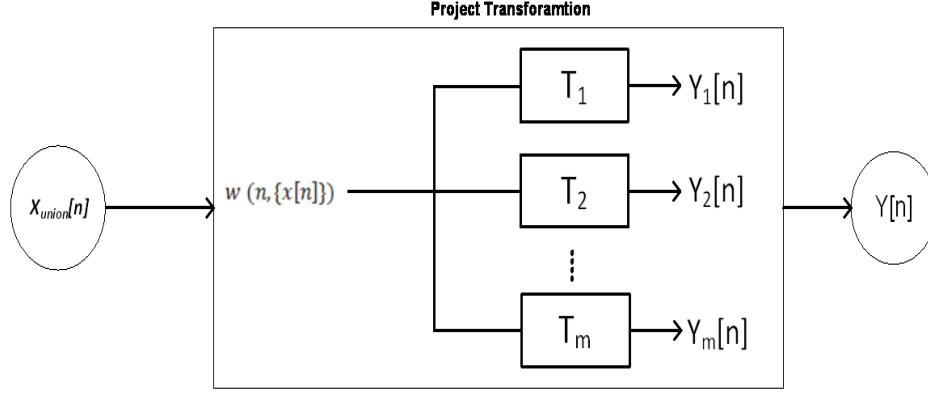


Figure 5.1: Transformation Process of Project Operation

Now using the concept of project operation which is defined above, the transfer function of project can be derived using the formalization Equation 3.4 which is:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

Put the value of $I = 1$ in the above equation, because the project is an unary operation. we get:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^1 w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

As we have described earlier that the total number of outputs for the project operation is equal to the window size which is $m = n_w$, therefore the above equation becomes:

$$\prod_{j'=1}^{n_w} y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^{n_w} T_l \left\{ \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n x_{i_i}[n'] \delta[n' - k] \right) \right\} \quad -\infty \leq n', n \leq \infty$$

The project transformation simply takes the input sequence $\{x[n]\}$ to the right by $l - n_w$ samples to form the output where T_l denotes the total number of transformation. Therefore, the final transformation of the project is defined by:

$$\prod_{j'=1}^{n_w} y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^{n_w} \prod_{i=1}^1 x_{i_l}[n - n_w + l] \quad -\infty < n < \infty \quad (5.1)$$

where

x_{i_l} is the input sequence, where the value of $i = 1$ which means that single input source is participating and l represents the particular point sample in time.

n_w is the window size and being the maximum number of outputs by the project operation.

o is the offset value initially we consider offset to be zero and τ is a trigger rate.

Example 5.1 Suppose an input sequence (as shown in Figure 5.2) is applied on a project transformation. The window function is applied on input sequence with $n_w = 3$ at the point in time $n = 5$. The transfer function is executed after arrival of every 3 elements in the sequence and the trigger offset is 2.

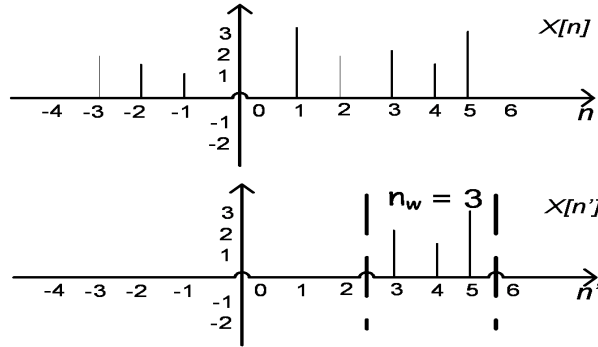


Figure 5.2: Input Sequence and Window Sequence

By putting the values $n_w = 3, I = 1, \tau = 3$ and $o = 2$ in Equation 5.1, we get:

$$\prod_{j'=1}^3 y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^3 x_{1_l}[n - n_w + l]$$

The output of the above equation is multiple as per the Definition 5. It can be modeled as transformations in parallel producing several outputs as shown in Figure 5.3.

In Figure 5.3, T_l takes the window sequence as an input sequence and it produces multiple outputs which are T_1, T_2 and T_3 . Therefore, the general output is described by:

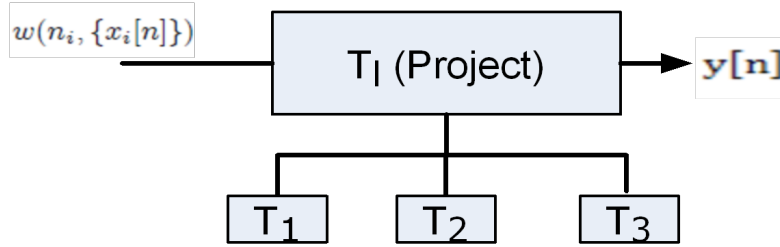


Figure 5.3: Several Transfer Functions is Executed in Parallel

$$\prod_{j'=1}^3 y_{j'}[n] = x_{1_1}[n - n_w + 1] \times x_{1_2}[n - n_w + 2] \times x_{1_3}[n - n_w + 3]$$

Let us start with the definition of T_1 . It takes window sequence with the following parameters $n_w = 3$, $l = 1$ and $n = 5$ and translate it in the following steps.

$$\begin{aligned} y_1[n] &= x_{1_1}[n - n_w + 1] \\ y_1[n] &= x_{1_1}[5 - 3 + 1] \\ y_1[n] &= x_{1_1}[3] = T_1 \end{aligned}$$

The transformation function T_2 executes the next sample with the following parameters $n_w = 3$, $l = 2$ and $n = 5$, the output becomes:

$$\begin{aligned} y_2[n] &= x_{1_2}[n - n_w + 2] \\ y_2[n] &= x_{1_2}[5 - 3 + 2] \\ y_2[n] &= x_{1_2}[4] = T_2 \end{aligned}$$

This process is performed continuously until all the transformations are executed.

5.1.2 Properties

In the previous section, the project transformation is defined to test the formal definitions of transformation properties (as we have defined in Chapter 4). We have to test the following properties:

- Input Sources
- Contributing Sources
- Input Tuple Mapping
- Output Tuple Mapping

Input Sources

This property checks that transfer functions take single or multiple input sources as input that contributes to produce an output sample. According to Definition 8, the value of I in our formal model is used to identify the number of input sources that contribute to produce the output. The derived project transformation is:

$$\prod_{j'=1}^{n_w} y_{j'}[n] = \delta[n \% \tau - o] \prod_{l=1}^{n_w-1} \prod_{i=1}^I x_{i_l}[n - n_w + l] \quad -\infty < n < \infty$$

Now compare the project transformation definition with the formal stream processing model, which is

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n \% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

As we can see, the value of I in the project transformation is equal to 1, therefore the project transformation is a single input source operation.

Contributing Sources

According to Definition 9, the contributing sources property is only applicable for those transformations which have multiple input sources. In case of the project transformation this property is not applicable because the project transformation has a single input source as shown in Equation 5.1.

Input Tuple Mapping

According to Definition 10, the input tuple mapping of the project transformation is single. The project transformation Equation 5.1 indicates that the output of the project transformation is a single sample i.e. $x_{i_l}[n - n_w + l]$ instead of accumulated sum of the value at index n and all previous values of the input sequence $\{x[n]\}$.

Output Tuple Mapping

It distinguishes whether the execution of a transformation produces a single or multiple output tuples per input tuple mapping. To check this, a formula is defined (see Definition 11) to calculate the output tuple mapping. The general formula is:

$$OTM = r \times \sum_{i=1}^I ITM_i \quad \begin{cases} \text{Multiple} & OTM > 1 \\ \text{Single} & \text{otherwise} \end{cases}$$

where $OTM =$ output tuple mapping

$ITM_i =$ Input tuple mapping per source

$$r = \frac{m}{\prod_{i=1}^I n_{wi}} \quad \text{where } I \text{ is the total number of input sources}$$

Therefore to calculate the output tuple mapping of the project transformation, the value of m and the value of the input tuple mapping is required. The input tuple mapping has been already calculated in previous section. The input tuple mapping is one. The total number of output can easily derived from the project transformation equation:

$$\prod_{j'=1}^{n_w} y_{j'}[n] = \delta[n\% \tau - o] \prod_{j'=1}^{n_w} \prod_{i=1}^1 x_{i_i}[n - n_w + l] \quad -\infty < n < \infty$$

The above equation indicates that the total number of output produced by the project transformation is $m = n_w$, therefore by putting the values in the output tuple mapping formula. we get:

$$\begin{aligned} OTM &= r \times \sum_{i=1}^I ITM_i \\ OTM &= \frac{m}{\prod_{i=1}^1 n_{wi}} \times \sum_{i=1}^1 ITM_i \\ OTM &= \frac{n_w}{n_w} \times 1 = 1 \end{aligned}$$

The result of Definition 11 is interpreted that the output tuple mapping of project transformation is 1.

5.2 Case 2: Average Operation

5.2.1 Transformation

The goal of this section is to derive the transformation of average operation using the formal stream processing model given in Equation 3.4. The average is

a SQL aggregate operation and returns a single value, using the values in a table column [35]. Figure 5.4 shows the generic process of the average transformation. It also shows that the average is calculated by combining the values from a set of input and computing a single number as being the average of the set.

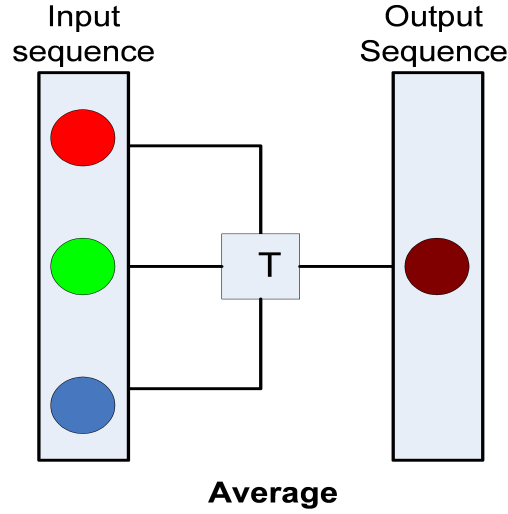


Figure 5.4: Average Transformation

From the concept of the average operation, the average transformation can be derived using the following equation:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

In the above equation, we can put the value of $I = 1$ and the value of $m = 1$ since the average returns a single sample, using samples in an input sequence. So, the resulting equation is:

$$\prod_{j'=1}^1 y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^1 T_l \left\{ \prod_{i=1}^1 w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

$$\prod_{j'=1}^1 y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^1 T_l \left\{ \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n x_{i_i}[n'] \delta[n' - k] \right) \right\} \quad -\infty \leq n', n \leq \infty$$

In mathematics, the average of n numbers is given as $1/n \sum_{i=1}^n a_i$ where a_i are numbers with $i = 1, 2, 3 \dots n$. Similarly, the average transformation is defined as:

$$\prod_{j'=1}^1 y_{j'}[n] = \delta[n\%o\tau - o] \prod_{l=1}^1 \prod_{i=1}^1 \frac{1}{n_w} \left(\sum_{k=n-n_w+1}^n x_1[k] \right) \quad -\infty < n < \infty \quad (5.2)$$

where

x_1 is the input sequence and $y_{j'}[n]$ is the output sequence where j' is the number of output which is equal to 1.

n_w is the window size and n is the point in time at which we are interested to start calculating the average.

o is the offset value, initially it is considered to be zero and τ is a trigger rate.

5.2.2 Properties

Input Sources

The input sources property checks whether the average transformation takes single or multiple input sources as input (as per Definition 8). The property can be checked by looking at the average transformation Equation 5.2. In Equation 5.2, the value of $I = 1$. Therefore, average is a single input source transformation.

Contributing Sources

Same as the project transformation, the contributing sources property is not applicable on the average transformation because Equation 5.2 indicates that the average transformation has a single input source as input.

Input Tuple Mapping

The input tuple mapping property of the average transformation is multiple since output of the average transformation is the accumulated sum of the value at index n and all previous values of the input sequence $x[n]$ that is:

$$\sum_{k=n-n_w+1}^n x_1[k] = x_1[n - n_w + 1] + x_1[n - n_w + 2] + \dots + x_1[k]$$

The value of k goes to n , therefore

$$k = (n - n + n_w + 1) - 1$$

$$k = n_w$$

So, the average transformation has multiple input tuple mapping i.e. n_w .

Output Tuple Mapping

To calculate the output tuple mapping of the average transformation, Definition 11 is used. The formula for output tuple mapping is:

$$OTM = r \times \sum_{i=1}^I ITM_i$$

$$OTM = \frac{m}{\prod_{i=1}^1 n_{wi}} \times \sum_{i=1}^1 ITM_i$$

$$OTM = \frac{1}{n_w} \times n_w = 1$$

The result of the OTM formula is 1 which means that the output tuple mapping of the average transformation is 1.

5.3 Case 3: Interpolation

5.3.1 Transformation

The Interpolation is an important function in many real-time applications such as the RECORD project (described in Section 1.2) and has been used for years to estimate the value at an unsampled location. It is important for visualization such as generation of contours.

There exist many different methods of interpolation. The most common approaches are weighted average distance and natural neighbors. The details of these approaches are available in [36]. In this thesis only weighted distance based interpolation transformation is described.

In Section 1.2, we described how the streaming workflow model use sensor data and combine them into a grid and how transformation element, interpolation are used to construct new samples. The RECORD case (defined in Section 1.2)

is used to derive the transformation of interpolation operation using the formal stream processing model.

Figure 5.5 shows the generic process of the interpolation transformation. It shows that the interpolation transformation takes a number of input samples from an input sequence and produces a set of output samples. In Figure 5.5, the interpolation takes 2 input samples and produces 6 output samples. Similarly, if it takes 3 input samples then it produces 9 output samples therefore interpolation is a constant mapping operation (as we have described in Section 4.1).

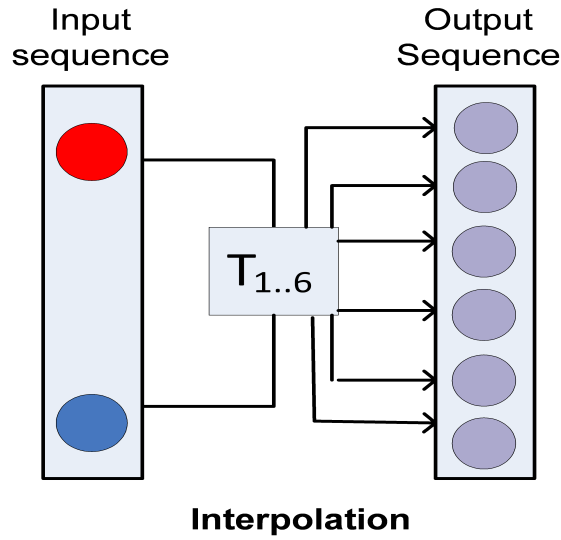


Figure 5.5: Interpolation Transformation

To derive the interpolation transformation, we can use the formal model Equation 3.4:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

As we know that the interpolation transformation takes a single input sequence as input, therefore we can put the value of $I = 1$ in the above equation, the resulting equation is:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^1 w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n x_i[n'] \delta[n' - k] \right) \right\} \quad -\infty < n, n' < \infty$$

Suppose that we have an input sequence $\{x[n]\}$ and we can apply the window function on the input sequence to select a subset of the samples (of window size n_w) at the given point in time n , therefore the above equation become,

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n x_1[k] \right)$$

Given a set of samples to which a point $P(x,y)$ is attached as shown in Figure 5.6. The point P is user-defined. In Figure 5.6, black circles are samples involved in the interpolation and gray circle is a new sample which is being estimated. The weight assigned to each sample is typically based on the square of the distance from the black to gray circle. Therefore the above equation becomes:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n \lambda_{i,l} \cdot x_1[k] \right)$$

where

$$\lambda_{i,l} = \frac{1/C + d_{n,l}^2}{\sum_{k'=n-n_w+1}^n 1/C + d_{n,l}^2}$$

$\lambda_{i,l}$ the weight of each sample (with respect to the interpolation sample i.e. gray circle) used in the interpolation process,

$d_{i,l}^2$ is the distance between sample n and the location being estimated

C is the small constant for avoiding ∞ condition

So, the interpolation transformation is defined as,

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m \prod_{i=1}^1 \left(\sum_{k=n-n_w+1}^n \frac{1/C + d_{n,l}^2}{\sum_{k'=n-n_w+1}^n 1/C + d_{n,l}^2} \cdot x_i[k] \right) \quad (5.3)$$

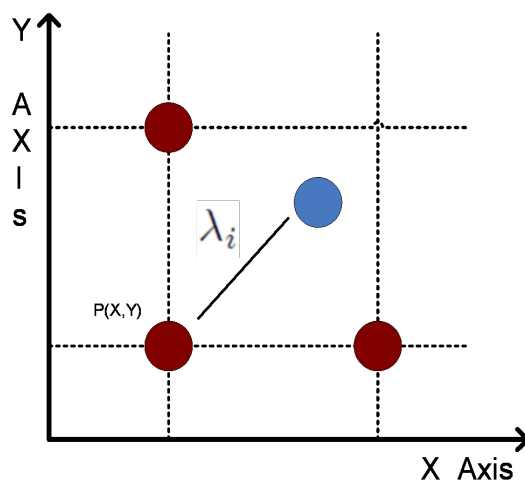


Figure 5.6: Distance based interpolation

5.3.2 Properties

Input Sources

The input sources property is a transformation property which is used to find the number of input sources participating in the transformation process. The interpolation transformation Equation 5.3 shows that the value of $I = 1$ therefore, the interpolation is a single source transformation. The interpolation transformation can take multiple input sources as input but with some assumption such as window size is one for each input sources. The alternative of the interpolation transformation is not chosen because it depends on the window size. On the other hand, all the case studies are independent of the window size.

Contributing Sources

The contributing sources property is not applicable to the interpolation transformation because the value of $I = 1$ in Equation 5.3.

Input Tuple Mapping

Same as the average transformation, the input tuple mapping property of the interpolation transformation is multiple since the output of the transformation is the accumulated sum of the value at index n and all previous values of the input sequence $x[n]$ that is:

$$\sum_{k=n-n_w+1}^n x_1[k] = \lambda_{1,l} \cdot x_1[n - n_w + 1] + \lambda_{1,l} \cdot x_1[n - n_w + 2] + \dots + \lambda_{1,l} \cdot x_1[k]$$

The value of k goes to n , therefore

$$\begin{aligned} k &= (n - n + n_w + 1) - 1 \\ k &= n_w \end{aligned}$$

The input tuple mapping of interpolation transformation is n_w which is multiple.

Output Tuple Mapping

The output tuple mapping of the interpolation transformation is multiple as per Definition 11. Equation 5.3 shows that the number of outputs is m . The input tuple mapping property defines that the interpolation transformation has multiple input tuple. Therefore, the output tuple mapping is:

$$\begin{aligned} OTM &= r \times \sum_{i=1}^I ITM_i \\ OTM &= \frac{m}{\prod_{i=1}^1 n_{wi}} \times \sum_{i=1}^1 ITM_i \\ OTM &= \frac{m}{n_w} \times n_w = m \end{aligned}$$

As a result of the OTM formula, the output tuple mapping of the interpolation transformation is multiple.

5.4 Case 4: Cartesian Product

5.4.1 Transformation

The Cartesian product is the direct product of two or more sources. It is also called the product set. Suppose we have two input sources $\{x_1[n]\}$ and $\{x_2[n]\}$, the Cartesian product of these sources is defined as the set of all ordered pairs whose first sample is an element of source $x_1[n]$, and whose second sample is

an element of source $x_2[n]$. The Cartesian product is written as $(x_1[n] \times x_2[n])$. The order of the input sources can not be changed because the ordered pairs is reversed. Although its elements remain the same but their pairing gets reversed.

In the workflow model described in Chapter 1, the Cartesian product operation is considered a transformation element. Figure 5.7 shows the Cartesian product transformation process. It takes two input sequences as input and produces four output samples i.e. $T_{1..4}$. Figure 5.7 also shows that the Cartesian product has a ratio of $(1, 1) : 1$ which means it takes one input sample from each source and then produces one output tuple. Therefore, it belongs to the constant mapping operations.

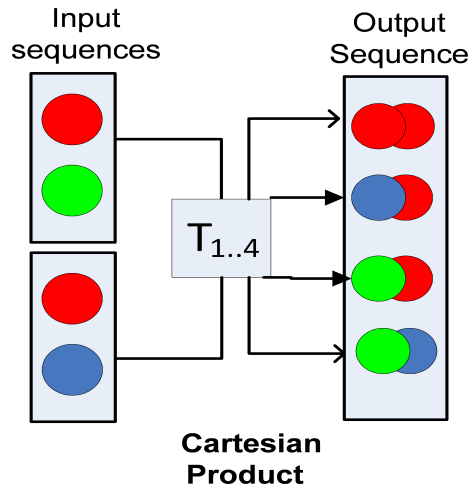


Figure 5.7: Cartesian Product Transformation

We can define the Cartesian product transformation using the formal model equation which is:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m T_l \left\{ \prod_{i=1}^I w(n_i, \{x_i[n]\}) \right\} \quad -\infty < n < \infty$$

The above equation also equal to:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\%_o\tau - o] \prod_{l=1}^m T_l \left\{ \begin{array}{l} \left(\sum_{k=n-n_{w_1}+1}^n x_1[n']\delta[n'-k] \right) \times \\ \left(\sum_{k=n-n_{w_2}+1}^n x_2[n']\delta[n'-k] \right) \times \dots \\ \cdot \times \left(\sum_{k=n-n_{w_i}+1}^n x_i[n']\delta[n'-k] \right) \end{array} \right\}$$

Now suppose we have 2 input sources, therefore the value of $I = 2$ and each source has constant window size i.e. $n_{w_1}, n_{w_2} = 2$. The number of output ($m = n_{w_1} \times n_{w_2} = 4$) is fixed which is a multiple of each source window size. Therefore, the above equation becomes:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\%_o\tau - o] \prod_{l=1}^m T_l \left\{ \begin{array}{l} \left(\sum_{k=n-n_{w_1}+1}^n x_1[n']\delta[n'-k] \right) \times \\ \left(\sum_{k=n-n_{w_2}+1}^n x_2[n']\delta[n'-k] \right) \end{array} \right\}$$

The Cartesian product of two input sequences $\{x_1[n]\}$ and $\{x_2[n]\}$ with window size n_{w_1}, n_{w_2} is the set of all possible combinations of $(x_1[n - n_{w_1} + 1], x_2[n - n_{w_2} + 1])$ where $x_1[n - n_{w_1} + 1]$ is a sample of input sequence $\{x_1[n]\}$ at the particular point in time and $x_2[n - n_{w_2} + 1]$ is a sample of $\{x_2[n]\}$ at the particular point in time. We can define the Cartesian product of two input sequences as follows:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\%_o\tau - o] \prod_{l=1}^m \{(x_1[n - n_{w_1} + l]) \times (x_2[n - n_{w_2} + l])\}$$

The generalized form of Cartesian product for I number of input sources is given as:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\%_o\tau - o] \prod_{l=1}^m \left(\prod_{i=1}^I x_i[n - n_{w_i} + l_i] \right) \quad (5.4)$$

Where

m is the total number of output i.e. $m = n_{w_1} \times n_{w_2} \times \dots n_{w_i}$.

I is the total number of input sources.

l_i shows the position of a sample in the i th source window which is

$l_i = 1, 2, \dots, n_{w_i}$

5.4.2 Properties

Input Sources

According to Definition 8, the transformation of the Cartesian product takes multiple input sources i.e. I as input to produce an output as shown in Equation 5.4. The value of $I \in \mathbb{N}$ where \mathbb{N} is the natural number. Therefore, the Cartesian product is a multiple input sources operation.

Contributing Sources

The contributing sources property is applicable on the Cartesian product transformation since the value of $I > 1$ in Equation 5.4. According to Definition 9, If the value of $I > 1$ and each I is contributed to produce an output sample then the contributing sources property is multiple. In the Cartesian product transformation each source is participating to produce an output sample, such as two input sequences $\{x_1[n]\}$ and $\{x_2[n]\}$ are contributed to produce an output. Therefore the contributing sources property of Cartesian product transformation is multiple.

Input Tuple Mapping

From the definition of the Cartesian product transformation, the input tuple mapping is single per input source. The derived transformation of Cartesian product is:

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m \left(\prod_{i=1}^I x_i[n - n_{wi} + l_i] \right)$$

$$\prod_{j'=1}^m y_{j'}[n] = \delta[n\% \tau - o] \prod_{l=1}^m (x_1[n - n_{w1} + l_1] \times x_2[n - n_{w2} + l_2] \dots \times x_i[n - n_{wi} + l_i])$$

The above equation shows that each source is contributing a sample to exactly produce the multiple output samples. Which means that each source is contributing a single sample and those samples are combined together to produce multiple output samples. So, the input tuple mapping is one per input source as per Definition 10.

Output Tuple Mapping

When the transformation of the Cartesian product is executed, it produces multiple output tuples as define by [1]. Now, we can prove it easily by using

Definition 11, which is:

$$OTM = r \times \sum_{i=1}^I ITM_i$$

$$OTM = \frac{m}{\prod_{i=1}^I n_{wi}} \times \sum_{i=1}^I ITM_i$$

The value of the input tuple mapping is 1 per input source and the total number of output $m = n_{w1} \times n_{w2} \times \dots n_{wi}$. Therefore, the above formula becomes:

$$OTM = \frac{n_{w1} \times n_{w2} \times \dots n_{wi}}{n_{w1} + n_{w2} + \dots n_{wi}} \times (1 + 1 + 1 + \dots 1_i)$$

$$OTM = \left(\frac{n_{w1} \times n_{w2} \times \dots n_{wi}}{n_{w1} + n_{w2} + \dots n_{wi}} \times (1 + 1 + 1 + \dots 1_i) \right) > 1$$

$$OTM = Multiple$$

As a result, the output tuple mapping of Cartesian product is multiple.

5.5 Provenance Example

In this section, we provide two examples for inferring provenance of a given sample for the case of overlapping windows and non-overlapping windows. The idea of the examples is taken from [1]. We can use transformation properties to infer provenance information for any particular output sample at a specific point in time n .

Example 1: Case of Overlapping Windows

For this case, we have considered a simple workflow where a project transformation takes one input sequence as input and produces an output sequence. In Figure 1, we considered that the window size is 3 and the transformation will be executed after arrival of every single sample (i.e. $\tau = 1$). In Figure 5.8, the starting time is 1 and 2,3,... are different points in time. For overlapping windows, we get the same type of output sequence.

Now, we have to choose the output sample for which the provenance information is inferred. Assume $y_3[4]$ (point in time 4) sample of the output sequence is chosen for inferring provenance information as shown in Figure 5.8. In Figure

5.8, the project transformation processes 3 samples (which are $x_1[2], x_1[3]$ and $x_1[4]$) of the window sequence as input and produces the multiple outputs. After that the transformation processes the next window sequence (from $x_1[3], x_1[4]$ and $x_1[5]$) and produces the next outputs.

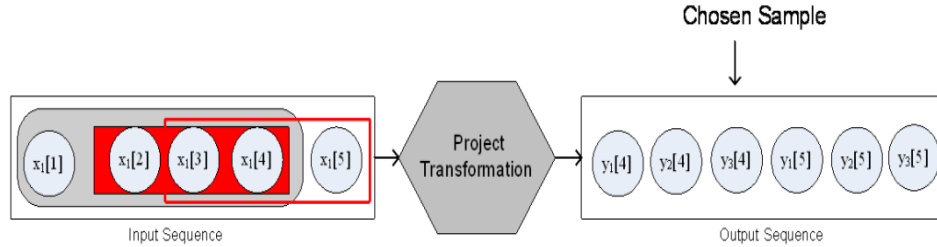


Figure 5.8: Example for overlapping windows

According to the transformatin properties, first we have to get the total number of input sources using input sources property (see Definition 8). Here, single input sources is contributing, which is x_1 . Now, we check the value of input tuple mapping. In this example, the input tuple mapping is 1 as per the ITM property. At last, we have to check which input sample is contributing from input source at a point in time 4. In order to check the input sample, we have to reconstruct the processing window. As we know that the window size is 3 and trigger rate is 1, so $y_3[4]$ (point in time 4) sample should be produced from input samples $x_1[2]$ to $x_1[4]$. We are interested in provenance information of y_3 at point in time 4. Now, we can count 3 samples of the input sequence which is started from $x_1[2]$ to $x_1[4]$ as shown in Figure 5.8. Therefore, the input sample $x_1[4]$ contributed to produce $y_3[4]$ at the output sequence.

Example 2: Case of Non-Overlapping Windows

For this case, we consider the project transformation to process the non-overlapping windows. Figure 5.9 shows an input sequence with 2 windows (in dark small square box), each window contains three samples and the project transformation is executed after arrival of every three samples.

The output sample, i.e. $y_3[7]$ is chosed for which provenance data is inferred. The project transformation processes first window and produces three output as the output sequence. Similarly, it processess second window and producess three more output, as shown in Figure 5.9.

Same as the first example, the single input sources x_1 is contributing and the input tuple mapping is also one as per the definition of ITM porperty. Now infer the provenance information of $y_3[7]$, since we know the window size and trigger rate. The samples $y_3[7]$ is produced from input sample $x_1[5]$ to $x_1[7]$. From point in time 7 of the input sequence, subtract three samples since window size

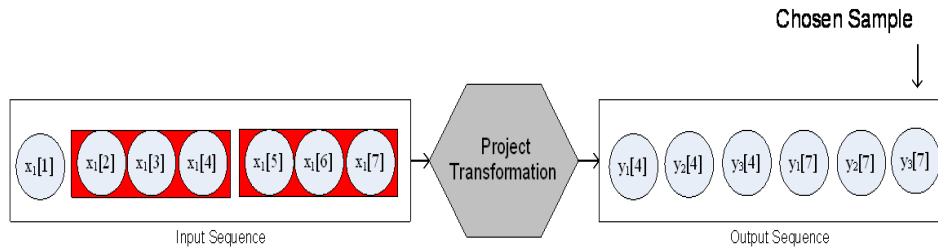


Figure 5.9: Example for non-overlapping windows

is 3, we get the desired window. The desired window sequence ($x_1[5]$ to $x_1[7]$) is processed by the project transformation to produce $y_3[7]$. So, the input sample $x_1[7]$ contributed to produce $y_3[7]$ at the output sequence as shown in Figure 5.9.

Chapter 6

Conclusion

This chapter summarizes the thesis by briefly discussing the conclusions of the previous chapters followed by discussing the contributions and the most important directions for future work.

This chapter is structured as follows: Section 6.1 gives answers to research questions, Section 6.2 explains the scientific contribution of this research and Section 6.3 identifies some potential research issues for future work.

6.1 Answers to Research Questions

This thesis discusses the properties relevant for inferring provenance in stream data processing. It introduced the formal definitions of the input sequence, transformation, window function, trigger rate and representation of multiple input and output streams using discrete time signal processing. Based on these definitions, a formal stream processing model and data transformation properties are given. These data transformation properties are one of our main contributions with regard to inference of provenance.

Now, we reflect on the results of our research by explicitly answering each research question presented in Chapter 1.

What are the formal definitions of the basic elements of a stream processing model that can be applied to any stream processing systems?

In Chapter 2, several stream processing systems have been summarized with their advantages and drawbacks. We identified that most of the data stream models consist of the input streams, stream transformer, trigger and windows.

There are many definitions of these elements available in the literature. We tried to provide the most general definitions of these elements.

In Chapter 3, the formal definitions of the basic elements for any stream processing model were defined. In the following Chapters 4,5 it was shown that these definitions are suitable to derive the definition of any transformation.

What are the formal definitions of transformation properties for inferring provenance?

In Chapter 1, a streaming workflow model was described. One of the important elements of the model is the transformation element. The transformation element has a number of properties that are useful for inferring provenance, such as a transformation consists of one or more input sequence as input and one or more output sequence as outputs. An input sequence can be an input source which originates and provides data to transformation element. A transformation element processes the input source and produces the output sequence. Based on the number of input sources, a classification of operations are provided in Chapter 4.

In Chapter 4, data transformation properties are introduced for tracking provenance. In this chapter, the formal stream processing model was used to provide the formal definitions of input sources, contributing sources, input tuple mapping, output tuple mapping and mapping type. These definitions were presented only for constant mapping operation.

What is the mathematical formulation of a simple stream processing model?

In Chapter 3, a simple stream processing model formula is introduced. In this model, we did not consider the dimension of the input data because we do not have any information about the input data in advance. For instance, when we apply $m \times n$ matrix as an input to a transformation. The output of the transformation has different dimensionality as compare to the input data dimensionality. Therefore the input and output data structure has an impact output tuple mapping property. In the later chapters, it was shown that this mathematical formula of stream processing model is suitable to derive any transformation definition.

What are the mathematical definitions of Project, Average, Interpolation and Cartesian product transformations?

Four case studies were presented in Chapter 5. The case studies were a very important part of this research. First, case studies proved that the formal stream

processing model can be used to derive any transformation such as Project, Average, Interpolation and Cartesian product. Second, the derived transformation is used to test the formal definitions of transformation properties.

In Chapter 5, transformation definitions of Project, Average, Interpolation and Cartesian product are provided.

Can we prove the continuity property for formal stream processing model?

In Chapter 3, we have proved that our formal stream processing model is continuous by given the proof of continuity theorem.

6.2 Contributions

This section summarizes the contributions of the thesis in the field of stream data processing and data provenance.

The main contribution of this master project is to formalize the transformation properties for inferring provenance information in stream processing. The formalization of transformation properties was done using the formal definitions of stream processing model. These properties allow scientists to reproduce the results in real-time applications. The generic properties can then be used in many domains such as monitoring systems, control systems and in academic settings.

The second contribution and difficult task of this thesis is to provide the definition of Project, Average, Interpolation and Cartesian product transformation to test the formal stream processing model. It has been shown that the formalization Equation 3.4 could be used to analyze and derive the definition of any transformation element for any streaming processing.

The third contribution is to prove the continuity property of the formal stream processing model using the notation of the discrete time signal processing.

We believe that the proposed formalism of transformation properties is a first step towards a unique theory for inferring provenance in stream processing.

6.3 Future Work

In this section, we provide a couple of interesting oppourtinities for data transformation properties that are left out from this thesis due to time constraints. The directions for future resarch are given below.

More research can be done by considering the input and output data structure in the formal stream processing model. In the output tuple mapping property, the dimensionality of input and output data structure is important, for instance when the average transformation is executed it combines multiple elements into one element in the output by reducing the dimension of the input data structure. Therefore, it would be interesting to add a dimensionality factor in the formal definitions of transformation properties.

The formalization of data transformation is not completed yet. More transformation elements could be distinguished, like variable mapping operations. Those operations which do not maintain a fixed ratio of output to input mapping are called variable mapping operations such as Select and Join operations. The Select operation may map an input sample to an output sample depending on the Select criteria, these operations have no fixed ratio. Therefore, future work could entail to find out how to derive the variable mapping transformation using the formal stream processing model.

References

- [1] Mohammad Rezwanul Huq, Andreas Wombacher, Peter M. G. Apers: "Inferring Fine-grained Data Provenance in Stream Data Processing: Reduced Storage Cost, High Accuracy". In DEXA 2011, Toulouse, France. Lecture notes in Computer Science (LNCS), Vol: 6861, part II, pp. 118-127.
- [2] P. Buneman and W. C. Tan, "Provenance in databases," in SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data. New York, NY, USA: ACM, 2007, pp. 1171– 1173.
- [3] http://en.wikipedia.org/wiki/Syntax_%28logic%29, Retrieved on 16/05/2011.
- [4] J. D. Fernandez and A. E. Fernandez, SCADA Systems: Vulnerabilities and Remediation," Journal of Computing Sciences in Colleges, Vol. 20, No. 4, pp. 160-168, Apr. 2005.
- [5] Mohammad Rezwanul Huq, Andreas Wombacher, Peter M. G. Apers: "Facilitating fine-grained data provenance using temporal data model". In: Proceedings of the Seventh International Workshop on Data Management for Sensor Networks, DMSN 2010, 13 Sep 2010, Singapore. pp. 8-13. ACM. ISBN 978-1-4503-0416-0.
- [6] <http://www.swiss-experiment.ch/index.php/Main:Home>, Retrieved on 12/07/2011.
- [7] M.Webster Online - The Language Center. <http://www.mw.com/home.htm>, Retrieved on 18/05/2011.
- [8] A. Wombacher, "Data workflow - a workflow model for continuous data processing," <http://eprints.eemcs.utwente.nl/17743/>, Centre for Telematics and Information Technology University of Twente, Enschede, Technical Report TR-CTIT-10-12, 2010.
- [9] A.Wombacher, M.R.Huq and J.Amiguuet, "Formal stream processing model", Database group, University of Twente, Enschede The Netherlands.
- [10] http://en.wikipedia.org/wiki/Direct_product, Retrieved on 08/04/2011.
- [11] <http://moa-datastream.posterous.com/>, Retrieved on 10/07/2011.

-
- [12] M. Branson, F. Douglass, B. Fawcett, Z. Liu, A. Riabov, and F. Ye, "CLASP: Collaborating, autonomous stream processing systems," in Proc. ACM Middleware, 2007.
- [13] H.Lim, Y.Moon and E.Bertino, "Research issues in data provenance for streaming environments" Proceedings of the 2009 ACM SPRINGL, November 3, 2009, Seattle, WA, USA, pp. 58 - 62.
- [14] S. Madden, M. Shah, J. Hellerstein, and V. Raman: "Continuously adaptive continuous queries over streams", in Proceedings of the 2002 ACM SIGMOD international conference on Management of data. ACM, 2002, pp. 49,60.
- [15] Blount, M., Davis II, J.S., Ebling, M., Kim, J.H., Kim, K.H., Lee, K., Misra, A., Park, S., Sow, D.M., Tak, Y.J., Wang, M., Witting, K. "Century: Automated Aspects of Patient Care" In Proc. of the 13th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA 2007), Daegu, Korea, pp.504-509, August 21-24, 2007.
- [16] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002, pp. 1–16.
- [17] S. Chandrasekar an, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Krishnamurthy, S. Madden, F. Reiss, and M. Shah, "TelegraphCQ: continuous dataflow processing," in Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 2003, p. 668.
- [18] L. Golab and T. Oszu: "Processing sliding window multi-joins in continuous queries over data streams", In Proc. of the 2003 Intl. Conf. on Very Large Data Bases, Sept. 2003.
- [19] http://public.dhe.ibm.com/software/data/sw-library/ii/whitepaper/SystemS_2008-1001.pdf, Retrieved on 18/07/11
- [20] <http://www.eweek.com/c/a/IT-Infrastructure/IBM-Debuts-System-S-Stream-Computing-Platform-614980/>, Retrieved on 18/07/11
- [21] D. Abadi, Y. Ahmad, M. Balazinska, U. Cetintemel, M. Cherniack, J. Hwang, W. Lindner, A. Maskey, A. Rasin, E. Ryzkina et al., "The design of the borealis stream processing engine," in Second Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, 2005, pp. 277–289.
- [22] A. V. Oppenheim. (1999). Introduction. In: T. Robbins Discrete-Time Signal Processing. 2nd ed. USA: Prentice-Hall, Inc. p1-70.
- [23] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," SIGMOD Rec., vol. 34, no. 3, pp. 31-36, 2005.

-
- [24] D. Miller. (1992). Abstract Syntax and Logic Programming. Logic Programming. Volume 592/1992, (2), p322-337.
- [25] http://en.wikipedia.org/wiki/Syntax_%28logic%29#cite_note-1, Retrieved on 18/05/2011.
- [26] E. A. Lee, "A Denotational semantics for dataflowwith firing," Electron. Res. Lab., Univ. of Cal., Berkeley, Tech. Rep. No. UCB/ERL M97/3, 1997.
- [27] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and where: A characterization of data provenance," in Database Theory – ICDT 2001, 2001, pp. 316–330.
- [28] Website: Record project <http://www.swissexperiment.ch/index.php/Record:Home>, Retrieved on 10/03/2011.
- [29] Szomszor, M., Moreau, L.: Recording and reasoning over data provenance in web and grid services. In: On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. (2003), pages 603 - 620.
- [30] Y. L. Simmhan, B. Plale, and D. Gannon, "Karma2: Provenance management for data driven workflows," International Journal of Web Services Research, Idea Group Publishing, vol. 5, pp. 1–23, 2008.
- [31] L. Moreau, J. Freire, J. Futrelle, R. McGrath, J. Myers, and P. Paulson, "The open provenance model: An overview," Provenance and Annotation of Data and Processes, pp. 323–326, 2008.
- [32] Y. Simmhan, B. Plale, D.G. (2005). A Survey of Data Provenance Techniques. Technical Report IUB-CS-TR618, Indiana University.
- [33] M. Stonebraker, U. Cetintemel, and S. Zdonik, "The 8 Requirements of Realtime Stream Processing." SIGMOD Record, 34(4):42–47, 2005.
- [34] N. Vijayakumar and B. Plale, "Towards low overhead provenance tracking in near real-time stream filtering". Lecture Notes in Computer Science, vol. 4145, I. Moreau and I. T. Foster Eds, Springer, pp.46-54.
- [35] <http://en.wikipedia.org/wiki/Average>, Retrieved on 22/06/2011.
- [36] I. Amidror. Scattered data interpolation methods for electronic imaging systems: A survey. Journal of Electronic Imaging, 2(11):157–176, 2002.
- [37] <http://www.tutornext.com/cartesian-product-two-sets/729>, Retrieved on 9/07/11.