Semantic Stream Processing for IoT Devices in the Food Safety Domain *

Milan Markovic
Computing Science
University of Aberdeen
Aberdeen, AB24 5UA
milan.markovic@abdn.ac.uk

Peter Edwards
Computing Science
University of Aberdeen
Aberdeen, AB24 5UA
p.edwards@abdn.ac.uk

ABSTRACT

The Hazard Analysis and Critical Control Point (HACCP) approach is widely used to develop food safety procedures in restaurants and other food related businesses. IoT devices could be deployed in this context to automate monitoring of critical control points (such as the maximum storage temperature of raw meat). However, such sensor infrastructures will result in the generation of significant amounts of data as well as associated meta-data describing the context for these readings. In this paper, we demonstrate how streams of semantically annotated sensor data can be automatically transformed into concise records describing significant events required to check compliance of business operations against HACCP-based food safety rules.

Keywords

Internet of Things, Provenance, Semantic Web, Food Safety

1. INTRODUCTION

To implement a HACCP food safety management system a business first has to identify the steps involved in its food handling processes (e.g. cold storage, cooking) and any health hazards associated with these steps. Furthermore, they have to identify critical control points associated with these hazards, and appropriate constraints for the food handling processes (e.g. the maximum permitted temperature for a food item in cold storage); these then have to be regularly monitored. IoT devices can be utilised to provide real-time monitoring of a range of critical control points, in order to support HACCP-based compliance checks [2]. However, continuous monitoring of this kind inevitably results in scalability issues - due to the significant volume of real-time sensor data. One approach to address such issues is on-the-fly detection of significant events, which are then stored in a suitably abstracted form. For example, events marking the start and the end of a food item in the cold storage phase would be sufficient to determine compliance with respect to the relevant HACCP constraint (in this case, the requirement that chilled food should be stored below 5°C). Descriptions of such events can then be stored instead of the raw sensor data, potentially resulting in significant overhead savings.

In our previous work [4], we introduced an approach for semantic modelling of the provenance of food items in the context of HACCP-based workflows. For example, for a food preparation workflow that includes a series of steps (storage, preparation, cooking) and any associated HACCP constraints (such as the minimum core temperature of cooked meat), a provenance record would include descriptions of entities (food items) and activities which used and produced such entities (e.g. a cooking activity used a raw burger entity and produced a cooked burger entity). Furthermore, activities and entities can be linked to the abstract descriptions defined in the corresponding workflow plan (i.e. a cooking *activity* is defined by a cooking *step*). In our approach, we also provided mechanisms to record if a particular constraint was satisfied in the context of a single execution of the planned step. For example, to record that an entity representing a cooked burger was produced by an activity defined by a cooking step and that this entity satisfied the minimum core temperature constraint associated with products of this step.

In this paper, we describe a semantic stream-based data processing framework for automated compliance checking. The framework can be adapted to various stages of a food supply chain and integrate diverse sources of sensor data to infer HACCP-based food safety compliance records via continuous stream querying and rule-based reasoning. To demonstrate its utility, we present results for an evaluation conducted with sensor data collected from a commercial kitchen environment. We conclude this paper with discussions of benefits and limitations in using provenance abstractions for food safety sensor data.

2. CASE STUDY

To test our approach we deployed sensors in a commercial kitchen to continuously monitor the temperature of individual meat items during *storage*, *preparation* and *cooking*. Two kinds of sensor were used: plastic wireless tags[1] mea-

*The research described here was funded by an award made by the RCUK Digital Economy IT as a Utility Network+ (EP/K003569/1) and the UK Food Standards Agency. We would like to thank Niels Christensen for his contributions to development of the ISI framework.

This is a preprint copy. This paper was published in Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems - SEMANTiCS 2016, Leipzig, September 2016.

[1]http://mytaglist.com/specs.html

sured the surface temperature and were attached to outer packaging of raw meat, continuously updating the temperature readings every 30s; a wireless meat probe[2] measured the core temperature of cooked meat with a sampling rate of 10s. Temperature readings produced by these sensors were annotated using the Semantic Sensor Ontology (SSN)[3]. The SSN ontology provides a rich vocabulary for describing sensor platforms, sensor deployments, capabilities of sensors and observations they produce. This ontology is the *de facto* standard for semantic modelling of sensor data and information about sensors and their deployment configurations. In our system, the sensor data included references to the identity of individual *sensing device*(s), individual *observation values*, *features of interest* (such as a particular meat item observed by a sensing device), and *properties* that were observed (surface temperature, core temperature).

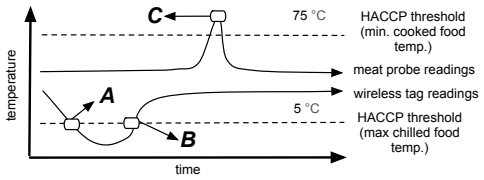

Figure 1: Temperature reading thresholds used to determine provenance events.

Figure 1 illustrates three relevant events that can be identified from sensor data in the context of our HACCP workflow, namely a meat item entering the cold storage stage (event A), a meat item entering the preparation stage (event B), and a meat item being cooked (event C).

To describe these events we used the FS-PROV[4] ontology [4], which extends a suite of ontologies including PROV-O[5], P-PLAN[6] and SC-PROV[7]. The ontology models HACCP-based workflow plans as a series of *fs-prov:Step*(s) associated with *fs-prov:Resource*(s) that represent inputs and outputs of these steps (e.g. food items). An *fs-prov: HACCPConstraint* can be associated with a resource in order to determine a threshold value for some property of the resource (e.g. the maximum surface temperature allowed for a meat item in cold storage). Compliance records can then be created by linking events and items that actually occurred during a business operation to a workflow plan. For example, we can record that an entity representing chilled meat which satisfied the HACCP surface temperature constraint (below 5°C) existed between 13:00 and 14:00 on 01 Jan 2016.

Using the HACCP thresholds and sensor temperature readings, we can infer provenance compliance records by creating corresponding descriptions of meat item states and linking them to the workflow plan. For example, if sensor observations reporting the surface temperature drop below the HACCP threshold for cold storage, we can use an inference rule in the form of a SPARQL INSERT query (Appendix, Query 2) to create a description of a meat item entering the chilled state. The timestamp of a sensor reading that determines the state change (e.g. the first observed temperature below the HACCP threshold) will be associated with the description of this new state. Provenance abstractions thus preserve the exact time of the state change, as reported by the sensor readings. This is in contrast, for example, with an approach for abstracting IoT data that utilises the Symbolic Aggregate Approximation (SAX) method [3]. In the SAX algorithm, following a split of time series sensor data into equal segments, the sensor readings in each segment are averaged. While this approach could potentially abstract sensor data into segments representing different food item states, the exact time when a HACCP threshold was exceeded would not be accurately recorded.

The rule listed in Appendix, Query 2 also captures events when items in a preparation state are returned back into cold storage. The rule infers appropriate triples to denote that the entity describing the meat item in the preparation state is no longer valid, and thus requires access to previously inferred provenance triples describing that state.

In our sensor deployment, we recorded temperature readings associated with 12 meat items over periods ranging from 2 to 2.5 hours. In a real-life deployment, recording and processing all sensor data would require significant resources. Therefore, we decided to test a stream-based approach where semantically annotated data are represented as time-ordered streams of sensor updates. Inference rules are then used to generate on-the-fly compliance records using the FS-PROV vocabulary.

3. IOT STREAM INSPECTOR

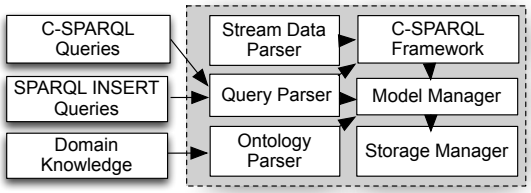

Figure 2: The IoT Stream Inspector architecture.

To simplify processing of semantic sensor data streams generated by IoT devices we have developed a framework called IoT Stream Inspector[8] (ISI). The framework is written in Java and builds on Jena[9] and the C-SPARQL framework[10]. ISI enables automatic creation of new data structures (e.g. provenance abstractions) based on streams of linked data. The framework could be deployed directly as part of a sensing device or as part of a smart stream data gateway within a larger IoT ecosystem.

Figure 2 illustrates the main components of the ISI framework. It integrates the *C-SPARQL framework*, which utilises the C-SPARQL extension of SPARQL to construct continuous queries on streams of linked data [1]. The *C-SPARQL framework* integrates relational stream processing framework

[2]http://www.filesthrutheair.com/product/EL-WiFi-TC-Thermocouple-Sensor
[3]http://purl.oclc.org/NET/ssnx/ssn
[4]http://w3id.org/abdn/foodsafety/fs-prov
[5]http://www.w3.org/TR/prov-o/
[6]http://vocab.linkeddata.es/p-plan/
[7]https://w3id.org/abdn/socialcomp/sc-prov
[8]https://github.com/m-markovic/IoT-Stream-Inspector/
[9]https://jena.apache.org/
[10]https://github.com/streamreasoning/CSPARQL-engine

ESPER[11] and SPARQL engine to associate timestamps with individual triples, which are transformed into so-called quadruples. By extending SPARQL with the concepts *stream* and *window*, continuous SPARQL queries can be used to retrieve triples within some time range (i.e. a window) from a specific quadruple sequence. The *C-SPARQL framework* provides methods to query only the most recent windows (e.g. a window spanning the last 20s). However, to enable comparisons of current and previously detected food item states (e.g. to compare the current state to the one detected 40s ago) ISI extends the framework with caching capabilities to enable previous results of individual C-SPARQL queries to be stored. ISI generates a single linked data stream, which is populated using an extensible generic method (*Stream Data Loader*) and managed by the *C-SPARQL framework*.

Every time any sensor reading is received, *Stream Data Parser* pushes the content of JENA's Ont model containing semantically annotated sensor data onto the linked data stream. This stream is then queried by a series of continuous user-defined C-SPARQL queries loaded from text files by the *Query Parser*. To detect a single food item, ISI requires a C-SPARQL query that will retrieve the latest sensor observations (i.e. within some window) that correspond to this item from the linked data stream. Such queries need to be defined by a user for each observed food item and are stored in an individual txt file in a predefined system directory. An example C-SPARQL query to retrieve sensor data for a particular item is shown in Appendix, Query 1.

When the framework is initialised (i.e. a cold start), no previous inferences exist. This makes design of inference rules which rely on previously inferred states difficult. Therefore, for each C-SPARQL query, the framework allows the user to define two types of inference rules; the first type is executed during the cold start, while the second is executed after some initial inferences have been created (e.g. a record of a meat item in its chilled state). C-SPARQL queries produce continuous results (e.g. observations for the last 20s) which *Model Manager* loads into a fresh instance of a JENA Ont model and merges with the user-defined static domain knowledge loaded by the *Ontology Parser* (e.g. a description of a specific HACCP workflow plan). If some inferences already exist, the latest interfered triples are added into the model. The user-defined SPARQL INSERT queries (inference rules) are then executed on this model. This process repeats with every new set of triples returned by the continuous stream query. Storage of the inferred triples is handled by the extensible *Storage Manager* methods.

4. EVALUATION

We based our evaluation efforts upon analysis of the data collected during our experiments, and the overall performance of our prototype food safety monitoring system based on the ISI framework. To enable evaluations based on average values observed over multiple system executions, we extended the ISI framework with a simulator module[12]. The simulator was used to reconstruct semantic (SSN) data streams from sensor observations[13] captured during the aforementioned sensor deployment in a commercial kitchen.

The extended ISI framework was run on a virtual server instance with Intel(R) Xeon(R) CPU 2.40GHz and 2GB memory. To evaluate how the framework operates under different workloads, ISI was tested in three settings with inputs from 2, 4, and 7 sensors where one of the sensors was always the meat probe (i.e. to enable inference of a cooked state). The domain knowledge loaded by the framework represented a HACCP workflow plan including three steps (*storage*, *preparation* and *cooking*) and corresponding constraints. During the sensor deployment, one sensor generated between 600 - 1000 raw observations depending on the reliability of the sensor and length of the collection period. Replaying data for all sensors in a single system setting (i.e. 2, 4, or 7 sensors) constituted a single run. For each system setting we performed 5 warm up runs, followed by 10 real (measured) runs. Times required to execute inference queries and the number of triples/quadruples were averaged and recorded for each system setting.

Provenance queries such as the one shown in Appendix, Query 3 were used to validate the provenance abstractions, by comparing them with a manual record of the actual events observed by a researcher during the sensor deployment. This was to confirm that the inference rules were able to identify all changes in the meat item state as they actually occurred during the sensor deployment. Timestamps associated with events in the provenance record deviated by up to one minute from the actual (observed) timings; this was a consequence of the reaction time of the sensors in registering temperature increases (e.g. when an item was removed from a fridge). Figure 3 illustrates the impact of abstracting the

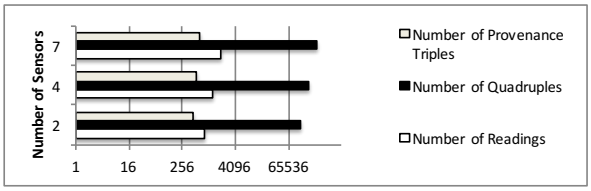

Figure 3: A log scale comparison of the number of raw sensor readings, quadruples (i.e. SSN annotations on the stream), and provenance triples.

sensor observations from our trial dataset using FS-PROV provenance descriptions. The results include descriptions of changing item states and a three-step HACCP workflow plan (422 triples). The very large number of quadruples was caused by the richness of SSN descriptions required to describe a single sensor reading and the default triples present in each instance of a JENA Ont model that were pushed onto the stream with each new sensor reading. In three tested ISI configurations containing 2,4, and 7 sensors, 95% of provenance inferences for one meat item were performed within 32ms, with mode values depending on the number of deployed sensors and ranging between 12.8ms – 13.8ms; median values were between 13.4ms – 15.8ms.

5. DISCUSSION

Our results demonstrate that provenance abstractions can be used to represent concise food safety compliance records and that these can be generated automatically from semantic stream data using simple SPARQL inference rules. However, it is important to acknowledge the trade-offs between

[11]http://www.espertech.com/esper/
[12]https://github.com/m-markovic/FoodSafety
[13]https://github.com/m-markovic/FoodSafety-Data

storage of abstracted and original sensor observations. For example, our approach would not support audit of the exact temperatures recorded at specific times, as this would be part of the original data. Original observations may also contain additional context such as information about sensor calibration and accuracy which influenced the sensor data quality. Capturing such information within provenance records might be possible; however, this was out of scope for the work presented here.

In future work, we plan to evaluate the ISI platform in various settings including public transport and domestic (smart home) environments. For these deployments we will explore use of single board resource-constrained devices as IoT gateways to support abstraction and obfuscation of the sensor data.

6. REFERENCES

[1] Barbieri, D. F., Braga, D., Ceri, S., Valle, D. E., and Grossniklaus, M. Querying rdf streams with c-sparql. *SIGMOD Rec. 39*, 1 (2010).

[2] Hiroshi, H., Shimura, T., and Fukui, T. Sensor network for haccp food safety management. In *Proceedings of IET International Conference on Communication Technology and Application (ICCTA 2011)* (2011), pp. 662–666.

[3] Kolozali, S., Bermudez-Edo, M., Puschmann, D., Ganz, F., and Barnaghi, P. A knowledge-based approach for real-time iot data stream annotation and processing. In *Proceedings of 2014 IEEE International Conference on Internet of Things (iThings), and Green Computing and Communications (GreenCom)* (2014), pp. 215–222.

[4] Markovic, M., Edwards, P., Kollingbaum, M., and Rowe, A. Modelling provenance of sensor data for food safety compliance checking. In *Proceedings of the 6th International Provenance & Annotation Workshop - IPAW* (Virginia, June 2016), vol. 9672, Springer, pp. 134–145.

APPENDIX

```
REGISTER QUERY meatItem11 AS

SELECT distinct ?s ?p ?o
FROM STREAM
<http://foodsafety/ssn> [RANGE 20s TUMBLING]
WHERE {
{?s  ssn:featureOfInterest <example.org/meatItem11>.
?s ?p ?o.}
Union
{?o  ssn:featureOfInterest <example.org/meatItem11>.
?s ?p ?o.}
Union
{?obs  ssn:featureOfInterest <example.org/meatItem11>.
?obs ssn:observationResult ?s.
?s ?p ?o.}
Union
{?obs  ssn:featureOfInterest <example.org/meatItem11>.
?obs ssn:observationResult ?res.
?res ssn:hasValue ?s.
?s ?p ?o.}
Union
{?obs  ssn:featureOfInterest <example.org/meatItem11>.
?obs  ssn:observationResult ?out.
?out ssn:isProducedBy ?s.
?s ?p ?o.}}
```

Query 1: A C-SPARQL query to retrieve sensor readings corresponding to a single meat item.

```
INSERT {
?activityId a fs:WorkflowActivity.
?evalContextId  a sc-prov:EvaluationContext.
?new a fs:WorkflowEntity.
?new prov:wasGeneratedBy ?activityId.
?new prov:generatedAtTime ?firstHighObsTime.
?new prov:specializationOf  ?foi.
?old prov:invalidatedAtTime ?firstHighObsTime.
?evalContextId fs:entity ?new.
?evalContextId sc-prov:hadCondition fs-ex:HACCPChilled.
?evalContextId sc-prov:hadResult ?resultId.
?resultId prov:hasValue "true".
fs-ex:cooling  fs:instantiatedByActivity ?activityId.
fs-ex:chilledMeat  fs:instantiatedByEntity ?new.
}
WHERE {
?old a fs:WorkflowEntity.
?old prov:generatedAtTime ?lastoldTime.
?old prov:specializationOf ?foi.
fs-ex:meatOutOfStorage fs:instantiatedByEntity ?old.
{
Select (MAX(?timeStored) as ?lastoldTimeComp ) ?foi
WHERE {
?entity prov:generatedAtTime ?timeStored.
?entity prov:specializationOf ?foi.
?thisSensor ssn:observes fs-ex:meatSurfaceTemp.
?obsRes ssn:isProducedBy ?thisSensor.
?obs ssn:observationResult ?obsRes.
?obs ssn:featureOfInterest ?foi}
Group by  ?foi }
{
SELECT  *
WHERE {
?obs ssn:observationSamplingTime ?firstHighObsTime.
{
SELECT (MIN(?obsTime) as ?TimeCheck)
WHERE {
?obs ssn:observationResult ?sensorOutput.
?obs ssn:observationSamplingTime ?obsTime.
?sensorOutput  ssn:isProducedBy ?sensor.
?sensorOutput ssn:hasValue ?obsValue.
?obsValue sk:hasQuantityValue ?followedReading.
?sensor ssn:observes fs-ex:meatSurfaceTemp.
FILTER (?followedReading < 5)
}}}}

FILTER (BOUND(?obs)&&?firstHighObsTime=?TimeCheck
&&?lastoldTime=?lastoldTimeComp)

BIND (UUID() as ?activityId)
BIND (UUID() as ?evalContextId)
BIND (UUID() as ?new)
BIND (UUID() as ?resultId)}
```

Query 2: A rule to infer the chilled state of an item.

```
SELECT ?item ?cookedAt

WHERE {
?storageStep a fs-ex:Cooking.
?result a fs:WorkflowEntity.
?result prov:specializationOf  ?item.
?result prov:generatedAtTime ?cookedAt.

?resultResource fs:isResultOf ?storageStep.
?resultResource fs:instantiatedByEntity ?result.

?evalContext fs:entity ?result.
?evalContext sc-prov:hadCondition fs-ex:HACCPCooked.
?evalContext sc-prov:hadResult ?evalResult.
?evalResult  a fs:WorkflowEntity.
?evalResult prov:hasValue "true".

VALUES (?item) {(<http://example.org/meatItem1>) }  }
```

Query 3: A provenance query to check if a food item has been cooked in compliance with the corresponding HACCP constraint.