Abnormal Situations Interpretation in Industry 4.0 using Stream Reasoning
Franco Giustozzi, Julien Saunier, Cecilia Zanni-Merk

To cite this version:
Franco Giustozzi, Julien Saunier, Cecilia Zanni-Merk. Abnormal Situations Interpretation in Industry 4.0 using Stream Reasoning. Procedia Computer Science, 2019, 159, pp.620-629. 10.1016/j.procs.2019.09.217. hal-02317612

HAL Id: hal-02317612
https://normandie-univ.hal.science/hal-02317612
Submitted on 20 Jul 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial 4.0 International License
Abnormal Situations Interpretation in Industry 4.0 using Stream Reasoning
Franco Giustozzi\textsuperscript{a,}, Julien Saunier\textsuperscript{a}, Cecilia Zanni-Merk\textsuperscript{a}
\textsuperscript{a}Normandie Universit\textsuperscript{e}, INSA Rouen, LITIS, Rouen 76000, France

Abstract
With the coming era of Industry 4.0, more assets and machines in plants are equipped with sensors which collect big amount of data for effective on-line equipment condition monitoring. Monitoring equipment conditions can not only reduce unplanned downtime by early detection of relevant situations like anomalies but also avoid unnecessary routine maintenance. For the detection of these situations it is necessary to integrate distributed, heterogeneous data sources and data streams. In this context, semantic web technologies are increasingly considered as key technologies to improve data integration. However, they are mainly used for data that is assumed not to change very often in time. In order to tackle this issue, stream reasoning combines reasoning and stream processing methods. Such a combination enables the processing of dynamic and heterogeneous data continuously produced from a large number of sources and implementing real-time services. This paper presents an approach that uses stream reasoning to identify in real time certain situations that lead to potential failures. Early detection enables to choose the most appropriate decision to avoid the interruption of manufacturing processes. In order to achieve this, data collected from sensors are enriched with contextual information. The use of stream reasoning allows the integration of data from different data sources, with different underlying meanings, different temporal resolutions as well as the processing of these data in real time.

© 2019 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0/)
Peer-review under responsibility of KES International.

Keywords: Industry 4.0; Stream Reasoning; Condition Monitoring; Ontology.

1. Introduction
In recent years, a new trend called Industry 4.0 has emerged. Also known as the fourth industrial revolution, its main objective is to improve the production and associated services through the digitalization and automation of manufacturing processes. Several fields and technologies, such as Cyber Physical Systems (CPS), Internet of Things
(IoT), etc., are fundamental to this vision in order to build intelligent machines, storage systems and production facilities capable of exchanging information autonomously and intelligently.

According to lean manufacturing metrics [12], as measured by Overall Equipment Effectiveness (OEE), world-class manufacturing factories are working at 85% of their theoretical capacity, while medium-sized ones are at around 60%. Some of the reasons for these deficiencies are detailed in [10]. The main ones are unnecessary maintenance tasks and the breakdown of several critical types of equipment (such as compressors, fans, pumps, motors, and generators, etc.). This leads to increased maintenance costs and production stoppages. Furthermore, in some cases it may lead to severe safety and environmental incidents. In order to tackle these issues, factories rely on condition monitoring. It is the task of monitoring all the equipment involved in a manufacturing process for early detection of anomalies.

In this context, a large number of machines and plant resources are equipped with sensors that collect data continuously and make them available for analysis. In this article, our focus is on the real time use of this data to detect situations that may lead to failures disrupting production processes. A situation is a combination of one or several sensor measurements linked through spatial, temporal and/or spatio-temporal relationships. The detection of these situations requires the interpretation of observations considering their context (other spatially or temporally related observations). In other words, it involves distributed and heterogeneous data sources. Therefore, data integration is a key point to consider. Semantic web technologies are increasingly used to improve the interoperability in heterogeneous scenarios. The Semantic Web is an extension of the current World Wide Web, where the semantics of information is encoded as a set of statements such as RDF statements [20]. The choice of RDF as a data model, in combination with ontological languages (e.g., OWL [15]), enables reasoning on data to infer new knowledge. However, current solutions for reasoning on RDF or OWL data are not appropriate for the dynamic nature of the streams in the industrial scenario where the manufacturing processes are executed over time and under different contexts. To bridge this gap, a number of recent works propose to unify reasoning and stream processing, giving rise to the research field of stream reasoning. Stream reasoning supports decision systems based on the continuous processing of data streams together with rich background knowledge [26].

This paper presents an approach that uses stream reasoning to identify certain situations that lead to potential failures in order to choose the most appropriate decision to avoid the interruption of manufacturing processes. The proposed approach enriches the data collected from sensors with contextual information to allow real-time situations detection.

The remainder of the paper is structured as follows: section 2 presents the related work. In section 3, the general approach is presented, from data acquisition to situation representation and detection. An illustrative case study is detailed in section 4. Finally in section 5 some conclusions and perspective of future work are presented.

2. Related Work

In this section, we review related work on the most commonly used approaches for condition monitoring and the application of stream reasoning over other domains that have similar requirements to the Industry 4.0 scenario.

There are different methods to diagnose single machine defects occurring during the machine operation: vibration monitoring, ultrasonic analysis, and many others [8, 21]. They monitor the behaviour of different properties of a machine. Most of them use data mining that allows the extraction of knowledge from large amounts of data. They are suitable and acceptably efficient when consuming data streams to detect abnormal patterns in the values of a machine’s property (e.g. temperature, vibration, etc.). However, they have two drawbacks: (i) the need, in advance, for labeled data for model training; and (ii) the lack of explicit model to explain decisions. This make it difficult to interpret the data and it also complicates the interoperability and re-usability of the models.

To deal with this issue, the Semantic Sensor Web (SSW) approach provides tools that allow the integration of data from multiple data sources [24]. It introduces semantic annotations for describing: (i) the data produced by the sensors, introducing spatial, temporal, or situation/context semantics; and (ii) the sensors and the sensor networks that provide such data. Furthermore, there are also works on defining suitable ontologies for data and sensors to enable both the integration of data from multiple sensor networks and external sources, and reasoning on such data. As an example, the W3C Semantic Sensor Network Incubator Group [9] developed an ontology to describe sensors and sensor networks, the Semantic Sensor Network Ontology (SSN). However, as mentioned in the introduction current
solutions to perform reasoning on ontologies are limited to work on rather static scenarios. They are not appropriate for the dynamic nature of the streams on the Industry 4.0 scenario.

Stream reasoning appeared as an initiative to tackle this problem. It has been applied in many fields, such as smart cities, to process and understand the information relevant for the life of a city and use it to make the city services run better and faster [27, 13], remote health monitoring, to generate automated and personalized systems for remote patient monitoring [25, 3], maritime safety and security, to represent and to perform reasoning over ship trajectories [22], semantic analysis of social media, to extend traditional analysis based on graphs enriching the connections between people and concepts with semantic annotations [14, 6], among others.

However, although the Industry 4.0 domain could be considered as a particular application field for the Semantic Sensor Web to the best of our knowledge stream reasoning approaches are so far unexplored in it. Sensor data in the manufacturing domain represents an ideal scenario for stream reasoning mainly for two reasons. Firstly, the amount of data collected from sensors is considerable, and it is produced at high (and low) frequencies. Secondly, integrating data coming from different sensors (and from different sensor networks) that measure different properties of the manufacturing processes and machines involved in them is necessary in many settings for deriving useful information such as the detection of abnormal behaviour of machines or processes.

**Stream Reasoning engines.** There are various existing approaches aiming to perform reasoning over data streams. Two of those are C-SPARQL and CQELS.

C-SPARQL (Continuous SPARQL) [2] proposes a query language to process RDF streams. It provides continuous query capabilities in SPARQL query language [23]. It supports timestamped RDF triples as input and uses periodic execution strategy to continuously execute queries over these RDF streams. It has the capability of integrating both RDF streams and static background knowledge represented as RDF triples. Given that streams are intrinsically infinite, data are usually read through time windows using the CQL window concept [1]. Queries are executed on all the triples which happen during a given time interval.

CQELS [18] is another system that combines static and streaming data in RDF format. Similar to C-SPARQL, it provides windowing and relational operators together with ad-hoc operators for generating new streams from the obtained results. CQELS queries deal with triples in element-based window (a given number of triples). The main difference with C-SPARQL is that CQELS offers a processing model in which query evaluation is not periodic, but triggered by the arrival of new triples. These different execution methods lead to the possibility of having different query results produced for the same query and input data.

These approaches do not perform complex reasoning tasks because (1) they do not manage incomplete information and (2) they only consider a snapshot of the stream. Therefore, classical reasoning approaches can be used as complements to stream reasoning.

As mentioned in the introduction, early detection of situations that may lead to failures in the Industry 4.0 scenario requires the integration of data from different data sources, with different underlying meanings, different temporal resolutions as well as the need to process these data in real time. Thus, we propose to use stream reasoning to face these issues but also considering the application of classical reasoning approaches to overcome the limitations of the stream reasoning method. In other words, our proposal uses a combination of these approaches to meet the requirements of Industry 4.0 for the detection of relevant situations.

### 3. System Overview

The idea driving the proposed approach is that certain situations that may lead to machine failures can be detected by interpreting observations in their contexts, *i.e.* if an observation has an abnormal value it may be because another parameter is having abnormal values too. For example, consider the case where the temperature of a machine and the temperature of one of its components are being monitored. It is known that an increase in the temperature of the machine may be due to an increase in the temperature of the component, or vice versa. This allows the exploitation or use of expert knowledge about relationships between the values of certain parameters from the machines, processes and their context for interpreting observations. Therefore, through the early detection of the aforementioned situations types, operators have enough time to adapt the maintenance schedule or take further measures to prevent unexpected downtime.
3.1. General framework

In [28], the authors focus on the conceptual model, architecture, and key elements needed for the support and enhancement of Industry 4.0 with knowledge-based and intelligent systems. They emphasize that contextual knowledge, user experience, and semantics involved in manufacturing and production processes should be taken into consideration if a company aims to improve equipment and production systems’ safety, availability, to reduce the number of unnecessary maintenance tasks and to optimize production costs.

In order to interpret raw sensor readings combined with contextual knowledge for the detection of certain situations that may lead to failures, our proposal relies on an adaptation of this framework. This adaptation is depicted in Figure 1 and is described below.

- **Sensors and Actuators layer** contains the sensors and actuators that are deployed in the machines or the environment. A sensor is a device that detects and responds to some type of input from the physical environment. Some examples are pressure sensors, accelerometers for measuring vibration, acoustic sensors for detecting leaks, temperature detectors, etc. The output is generally a signal that can be converted to human-readable format or transmitted electronically over a network for reading or further processing. An actuator is a component of a machine that is responsible for moving and controlling a mechanism or system, for example by opening a valve. It can be activated by electric voltage or current, pneumatic or hydraulic pressure, or even human power. When it is activated the actuator converts the signal’s energy into mechanical movement.

- **Communication layer** is in charge of pre-processing the data coming from the Sensors and Actuators layer through processes such as filling missing values, smoothing the noisy data, etc. Furthermore, the data can be normalized, aggregated and/or generalized as required. This layer is also able to divide and distribute information and must ensure security and anonymity where necessary.

- **Semantic Enhanced CPS Representation layer** contains the reasoning process and modules that allow the semantic enrichment of the raw data coming from sensors. The Reasoner module makes use of First Order Logic (FOL) under the Open World Assumption (OWA). In this layer, we propose to add a module named Stream Reasoner to process data in real time. It executes continuous queries over the streams of data. The Reasoner and the Stream Reasoner are not only fed with data but also with rules provided by domain experts. The Expert Knowledge module is in charge of storing such rules. It also embraces data from other external sources like Business Information Systems (BIS) and Enterprise Resource Planning (ERP).

- **Application layer** comprises different applications that exploit the semantic enriched information, for example, a human to machine interaction module.

Our proposal is part of the Semantic Enhanced CPS Representation layer. It makes use of all its components to go from raw sensor readings combined with background knowledge to the detection of relevant situations to support decision-making tasks at the Application layer.
3.2. Relevant Situations Detection and Identification

The main components of our proposal are shown in Figure 2. The workflow starts from raw sensor readings to the detection of relevant situations to support decision making tasks. The modules to detect situations in real time are (1) Translation, (2) Temporal relations, (3) Cause determination, while the (4) Update/Create situations module manages classical reasoning. All these modules rely on a context ontology to semantically enrich the data representation and treatment.

3.2.1. Semantic representation: The Context Ontology

The Context ontology is an essential component in our approach. Firstly, it provides the formal representation of the manufacturing domain. It enables to represent a production line: the machines that compose it, the tasks they perform and the observations made by the sensors on certain properties of interest about the machines, processes, products and the environment. Secondly, it allows to represent the context in which an observation is measured. This is a key point to consider because that information is useful to interpret the observation not just as a single value allowing to determine the causes and what actions can be taken in a more informed way.

The ontology is an adaptation from a previous work [7] based on the definition of context given by [5]. This ontology-based context model for industry is built upon the following head concepts: resources, processes, sensors, time, location and situations. It reuses different ontologies in order to obtain a model that satisfies the Industry 4.0 requirements and facilitates context representation of the resources and processes.

The main concepts of the ontology and their relationships are presented in Figure 3. Among the reused ontologies it is worth mentioning the use of the Time Ontology [4] (used as a temporal model to guarantee a consistent representation of temporal content and properties), the GeoSPARQL Ontology [17] (for representing the abstraction of physical spatial places and relations among them), the SSN Ontology [9] (for annotating heterogeneous sensor data with formalized semantics), the ontologies to describe the resources of a factory and the manufacturing processes, and the ontology to represent the relevant situations. For condition monitoring, a crucial concept is the concept of situation. A situation defines an abstract state of affairs that represents a particular scenario of interest and involves observations, resources and processes. In other words, a situation is a combination of one or several sensor measurements linked through spatial, temporal and/or spatio-temporal relationships.

The data models and ontology languages used in this approach are RDF [20], RDFS [19] and OWL [15] [11]. These languages are used to infer implicit knowledge through inference process. This is also known as reasoning and can be
performed for several purposes. For example, it can be used to execute automatic consistency check when integrating multiple sources of knowledge, or to classify new information, or enrich query answers with new knowledge.

3.2.2. RDF Stream Reasoning for Situation Detection

The three modules of processing involved in our approach for situation and cause detection are Translation, Temporal Relations, and Cause Determination. The Decision Making module is not part of the task of detecting situations, but it exploits this information to support decision making tasks.

Translation. This module is mainly responsible for acquiring data from sensors and (i) converting it to RDF streams, and (ii) inserting it as instances into the ontology. Both tasks are performed by the Stream Generator and the Instance creator sub-modules respectively.

The Stream Generator component performs semantic annotation of the acquired data, using the concepts and relations among them as defined in the ontology. This allows the module to stream out semantic annotated data streams that are then consumed by the Stream Reasoner. The output streams are RDF streams. An RDF stream is defined as an ordered sequence of pairs, where each pair is constituted by an RDF triple and its timestamp $t$:

$\langle \text{Subject}, \text{Predicate}, \text{Object}, t \rangle$.

An RDF triple is defined as $\langle \text{Subject}, \text{Predicate}, \text{Object} \rangle \in (I \cup B) \times I \times (I \cup B \cup L)$ where $I$ is a set of IRIs (Internationalized Resource Identifiers), $B$ is a set of blank nodes and $L$ is a set of literals.

The Instance creator component creates instances from the received data and inserts them in the ontology, i.e. it is in charge of populating the ontology with observations and their corresponding metadata, such as the sensors which made the observation, the observed property, the time, etc.

Temporal Relations. Once the data from the distributed and heterogeneous data sources is available in a homogeneous, contextualized and ordered representation, the streams can be explored to generate new information. A set of queries, which combine background knowledge extracted from the ontology and some parts of the streams that are relevant, are registered and executed by the Stream Reasoner over the data streams. These queries represent particular situations to be identified and they include mainly temporal dependencies between observations and/or anomalies. The Stream Reasoner represents queries as query graphs, with query evaluation performed through graph pattern matching over graphs formed by the incoming data streams. It offers two types of processing models: either the query evaluation is periodic through windows of time, or the query evaluation is not periodic but triggered by the arrival of new triples.
For this component, C-SPARQL was chosen over CQELS because although both have similar characteristics C-SPARQL is open source and handles continuous queries. Therefore, C-SPARQL [2] is used to execute queries against the streaming data.

This module produces situations streams as output. This output feeds another stream reasoner in the cause determination module and the situations detected are stored in the ontology. In this way, the Stream Reasoner itself can be seen as an advanced sensor able to produce high level data.

**Cause/s Determination.** The purpose of this module is to identify the possible causes that generated a situation detected by the previous module. For this, two components are used separately. In the case where situations are generated at high frequency, the Stream Reasoner identifies the causes. Otherwise, the Reasoner is used over the ontology to infer the causes. This last option has some advantages over the previous one. For example, it is possible that some situations do not have identified causes in a real scenario, in which case the system notifies that the causes are unknown. Therefore, it is necessary to consider the Open World Assumption, which states that the absence of a statement alone cannot be used to infer that the statement is false. If the cause is identified later, it can be added to the knowledge base and linked to the situation for future use. In both cases, the module provides the Decision Making module with the identified situation together with the possible causes in case they are known. The association between the situation and the causes is also added to the Ontology.

**Decision Making.** Considering the situations and its causes, it is possible to support decision making tasks to determine what actions to launch to correct the behaviour of machines or avoid breakdowns. The actions to be triggered are diverse and depend on other external factors. For example, certain maintenance tasks can be launched remotely and performed by the machines themselves or the application can issue an alert to notify the operator closest to the machine to inspect it, if human action is needed. Each application can perform more advanced reasoning and processing on the received data.

3.2.3. **Static Reasoning for Situation Identification or Refinement**

The Translation module, in addition to generating the data streams, populates the ontology with semantically enriched data. The data models and the ontology languages used in this approach are RDF [20], RDFS [19] and OWL [15], because they facilitate the execution of automatic consistency checking when integrating multiple sources of knowledge, or the classification of new information, or the enrichment of query answers with new knowledge after an inference process.

Compared to the processing performed by stream reasoning, reasoning is more computationally expensive. Depending on the complexity of the ontology language adopted, reasoning can even become undecidable. To alleviate these problems, the W3C has defined several profiles [16] of OWL that reduce the complexity of reasoning making it more tractable and scalable. Despite these efforts, reasoning remains a complex task. Because of this, it is mainly applied to static data that is assumed not to change, or to not change very often in time.

Thus, the main purpose of the Update/Create situations module is to exploit the historical data stored in the knowledge base (ontology) to refine the already defined situations or even to find new ones. This can be accomplished by, (i) reasoning over the ontology to derive implicit information, and (ii) interacting with the ontology by issuing one-time queries, expressed in the SPARQL [23] language.

4. Case study

In this section, a case study is presented to illustrate the application and the advantages of detecting abnormal situations by interpreting anomalies or observations in their contexts. Firstly, a sample case of two properties which values have a causal relationship correlated in time is described formally. Secondly, an illustrative case in an industrial scenario with two sensors deployed in a machine is presented.

The formal definition, illustrated in Figure 4, is the following: Consider two properties $P_X$ and $P_Y$. At time $t_x$ the property $P_X$ exceeds its threshold $T_X$ indicating a positive deviation from their normal values. At time $t_j$ the property $P_Y$ exceeds its threshold $T_Y$ also indicating a positive deviation from their normal values. This effect is a consequence resulting from the increased of $P_X$. At time $t_a$ an action is triggered to correct the behaviour of $P_X$ and at time $t_n$ the property $P_X$ is under $T_X$. Then at $t_m$ the property $P_Y$ is also under $T_Y$. In this case, the value sensed of $P_Y$ at $t_j$
is interpreted in the context that since \( t_r \) the property \( P_X \) exceeds its threshold. The action launched at \( t_a \) is decided considering this interpretation.

In the industrial scenario, consider a machine \( M_1 \) that is part of a production line in a company. This machine has a component \( M_2 \), that is a main component on which the operation of the machine depends. Both the machine and its component have sensors, called \( \text{SensorTM}_1 \) and \( \text{SensorTM}_2 \), which measure respectively the temperatures of the machine and its component. The focus of the example is to monitor the temperature of the machine and of its component. We note \( P_Y \) and \( P_X \) the temperature properties of the machine and its component.

As shown in Figure 4, during the execution of the task performed by the machine, an increase in \( M_2 \) temperature is observed. Let us assume this is due to dirty filters in the cooling system. The \( M_2 \) temperature deviates from normal behavior at \( t_r \), while \( P_Y \) is normal. The observation made by \( \text{SensorTM}_2 \) at time \( t_r \) is represented in RDF format using the structure of our ontology in Listing 1. Afterwards, at time \( t_j \) the \( M_1 \) temperature also leaves normal behavior and indicates a positive deviation.

This effect is a consequence of the increased \( M_2 \) temperature, as \( M_2 \) is the main heat source in \( M_1 \). Consequentially a higher \( M_2 \) temperature leads to a higher \( M_1 \) temperature. Our approach allows to exploit this knowledge about the link between the values of the two temperatures. Through this interpretation of the anomaly in \( P_Y \) it is possible to detect that situation, named \( \text{AbnormalT-M1M2} \). The C-SPARQL query shown in Listing 2 detects this situation and selects the observations and the resources involved. Once the situation is detected an instance (\( \text{AbnormalT-M1M2} \)) is added to the ontology, as are the relations with the corresponding observations (\( \text{obsTempM1} \) and \( \text{obsTempM2} \)). The representation of the described scenario is visualized using our model in Figure 5.
The detection of this situation and its interpretation avoids a general inspection of $M_1$ due to its high temperature. Only the state of its component $M_2$ has to be verified. This allows to take action to correct the behavior of $P_Y$ considering its context. At time $t_a$, an action is taken about $M_2$ (cleaning or replacement of the cooling system filters) that solves the problem, which enables the temperature of $M_2$ to decrease. After some time, this leads to a decrease in the temperature of $M_1$.

**Listing 1.** Example of an observation in RDF using the structure of our ontology.

```
:obsTempM2 a sosa:Observation;
  sosa:observedProperty Px;
  sosa:madeBySensor :SensorTM2;
  sosa:hasSimpleResult Tx^^xsd:float;
  :hasTimeObs tr;
  :isInSituation AbnormalT-M1M2.
```

**Listing 2.** C-SPARQL query to detect the described situation.

```
REGISTER QUERY sit-AbnormalT-M1M2 AS
PREFIX :<http://contextOntology#>
PREFIX f: <http://larkc.eu/csparql/sparql/jena/ext#>
SELECT ?o1 ?o2 ?m1 ?m2
FROM STREAM <http://.../sensorTM2> [RANGE 5s 1s]
FROM STREAM <http://.../sensorTM1> [RANGE 5s 1s]
WHERE {
  :sensorTM1 :isHostedBy ?m1 .
  :sensorTM1 :madeObservation ?o1 .
  ?o1 :hasTime ?t1 .
  ?s2 :isHostedBy ?m2 .
  ?o2 :hasSimpleResult ?res2 .
  ?o2 :hasTime ?t2 .
  FILTER ( f:timestamp (:sensorTM1,:madeObservation,:o1) < f:timestamp (:sensorTM2,:madeObservation,:o2) 
    && ?res1 >= Ty
    && ?res2 >= Tx
    && ?m1 = M1 && ?m2 = M2 ) .
}
```

**Listing 3.** C-SPARQL query considering the average of $P_Y$.

```
REGISTER QUERY sit-AbnormalT-M1M2-avg AS
PREFIX :<http://contextOntology#>
SELECT ?o1 ?o2 ?m1 ?m2
FROM STREAM <http://.../sensorTM1> [RANGE 5s 1s]
FROM STREAM <http://.../sensorTM2> [RANGE 5s 1s]
WHERE {
  :sensorTM1 :isHostedBy ?m1 .
  :sensorTM1 :madeObservation ?o1 .
  ?o1 :hasTime ?t1 .
  ?s2 :isHostedBy ?m2 .
  { SELECT ?s2 ( avg(?p2) AS ?average )
    WHERE {
      ?o2 :hasSimpleResult ?res2 .
      ?o2 :hasTime ?t2 .
    } GROUP BY ?s2
  HAVING ( avg(?res2) >= Tx )
  }
  FILTER ( ?res1 >= Ty && ?s2 != :sensorTM1 ) .
}
```

Although the case study presented in this section involves only two properties, it shows the advantages of interpreting an observation in its context and the exploitation of background knowledge. In real cases, more complex situations can be detected using the proposed approach: for example, as an extension to the previous case, it is possible to interpret the anomaly of the temperature $P_Y$ regarding the average of the temperature $P_X$ measured during the time window $w$ instead of only considering if one of these values is greater than the threshold $T_X$. The corresponding C-SPARQL query is displayed in Listing 3.

5. Conclusions & Future Work

This paper presents an approach that uses stream reasoning to detect certain situations that could lead to failures in order to make the most appropriate decision to avoid the interruption of manufacturing processes. Data collected from sensors are enriched with contextual information to allow real-time situations detection. The use of stream reasoning allows the integration of data from different data sources, with different underlying meanings, different temporal resolutions as well as the processing of these data in real time. Furthermore, our proposal also uses classical reasoning approaches to compensate for possible non-detections of causes by the stream reasoning method.

In future work the following lines will be addressed. Firstly, a broader case study with more complex situations, involving more properties measurements that are temporal and spatially related, will be explored. This raises scalability and complexity issues to detect situations in real time. Therefore, tests will be performed with different variants of stream reasoning engines to evaluate their efficiency and scalability. Secondly, a complete study on how to update and refine the situations (queries) will be inspected as well as how to register and deregister queries as appropriate in each case. For example, in case a particular situation is detected, it would be necessary to deregister that query until the problem(s) are solved, to avoid detecting it continuously. Finally, the identification of causality patterns could enable to define generic queries for certain types of anomalies. These generic queries could be reused in different cases and

---

Franco Giustozzi et al. / Procedia Computer Science 00 (2019) 000–000
would perform more complex operations among the sensed values than those presented in the previous case study, such as the comparison between two measurements or the average of a set of measurements in a given time interval.

Acknowledgements

These PhD works are funded by the Normandy Region (France) in the framework of the STEaMINg (Semantic Time Evolving Models for Industry 4.0) project.

References