Anomaly Detection in a Boxed Beef Supply Chain

PETER BAUMGARTNER, Data61|CSIRO, Australia
ALEXANDER KRUMPHOLZ, Data61|CSIRO, Australia

An approach to simulating and analysing sensor events in a boxed beef supply chain is presented. The simulation component reflects our industrial partner's transport routes and parameters under normal and abnormal conditions. The simulated transport events are fed into our situational awareness system for detecting temperature anomalies or potential box tampering. The situational awareness system features a logic-based modeling language and an inference engine that tolerates incomplete or erroneous observations. The paper describes the approach and experimental results in more detail.

CCS Concepts: • **Computing methodologies → Model verification and validation**; **Modeling and simulation**; **Model development and analysis**; **Discrete-event simulation**.

Additional Key Words and Phrases: situational awareness, logic programming, knowledge-based approach

1 INTRODUCTION

To boost the value of exports and secure new agricultural markets, Australia needs to grow its high-value markets where premiums are paid for quality, safety and provenance. This requires building robust supply chains that deliver products global consumers can trust and enhancing product integrity across the Australian meat industry [8]. CSIRO's Data61 recent multidisciplinary project *Supply Chain Integrity Initiative*[1] has developed technologies that validate claims about the origin of a product and its authenticity, among others.

In this paper, we report on the methods and results of our sub-project on *event inference*. Current industry challenges include timely management of recall for food safety issues, fraud detection and monitoring compliance with cold chain requirements. The event inference sub-project addresses this challenge by providing tool support for situational awareness in the red meat supply chain with a focus on transport. The technology developed aims to improve visibility of goods as they move through the supply chain and detect anomalies using rule-based analytics.

Our approach depends on the availability of sensor data sampling transport parameters. A neighbouring sub-project developed a novel technology that uses vibration energy harvesting for both sensing and generating power. This technology was trialed in collaboration with the Australian company BeefLedger [1] to collect data from frozen meat transport routes in south-east Queensland. The sensors monitor motion, temperature, and location using GPS.

Our approach combines such sensor data with domain knowledge on how goods move through the supply chain. This makes it possible to more accurately identify anomalies, missing data and fraud than is possible in isolation. We developed a situational awareness model and applied it to a synthetic data set based on the BeefLedger frozen meat use case. The model was able to identify anomalies with very high accuracy and capability to distinguishing between different types of anomalies. Such a model can provide plausible explanations in the context of real world events to improve tracking and managing anomalies in the supply chain.

Our modelling language is a logical specification language tailored for situational awareness applications [3]. It succeeds an earlier approach based on state automata [5] and is related to semantic approaches for the IoT [12] such as contextual querying (see e.g. [9]), ontological models for (EPCIS) supply chains [13], logic-based event recognition [2] and stream reasoning [6], among

[1]See https://algorithm.data61.csiro.au/advancing-supply-chain-integrity-in-australian-agriculture-workshop/.

This research is supported by the Science and Industry Endowment Fund.

others. It contributes novel language features and reasoning capabilities, which we exploit in this paper for a situational awareness case study.

2 SIMULATOR AND DATA

BeefLedger [1] specializes in providing a complete meat supply chain from producer to (international) consumers. In our project we focus on a small segment of the chain, where boxed beef products are transported within Brisbane, Queensland. Our situational awareness model (in Section 4 below) requires as input corresponding transport parameters in the form of way points, trip durations, cooling requirements and concrete transport sensor data. As concrete sensor data collection was not available at the time of our developments, we wrote a simulator that synthesizes reasonably realistic sensor data. In our experiments we used exclusively that data instead.

We wrote the simulator with flexibility in mind. It makes it easy to generate scenarios for exercising several types of anomalies and under random perturbations for simulating real-world sensor behaviour. It is controlled by parameters such as way points and number of boxes to be transported, and commands for describing how the transport evolves in terms of box and truck sensor data. In the following we describe the simulator and the generated data in some more detail. See Figure 1 for reference.

The simulated data is based on a BeefLedger route from Morningside via Woolloongabba to Annerley. The route is given in terms of warpaint latitudes and longitudes in the form of a Google Earth kml-file. (It is easy to make with the help of the Google Earther route planner.) In the first step, the kml-file is processed by a utility directions_2_csv, which is controlled by parameters to set the start time and speed of the truck, a list of stopping times and the frequency of GPS location events to be created. Based on this configuration directions_2_csv outputs a csv-file[2] of randomized events that match the specification.

In the second step, event files for a specified number of boxes are generated based on the trucks event file and further configuration parameters. These event files are then manipulated to derive specific scenarios. For that, we wrote (in Scala) a set of *library programs*, each providing a specific functionality delivered through pipe-like input/output. (The library programs are listed at the bottom of Figure 1.) In broad terms, the functionalities involve the manipulation of location and temperature sensor data over time for given subsets of sensors, and the simulation of sensor dropouts. The library programs are orchestrated by the `Event Simulator Script` (a shell script) which creates box and truck event csv files for concrete transport scenarios.

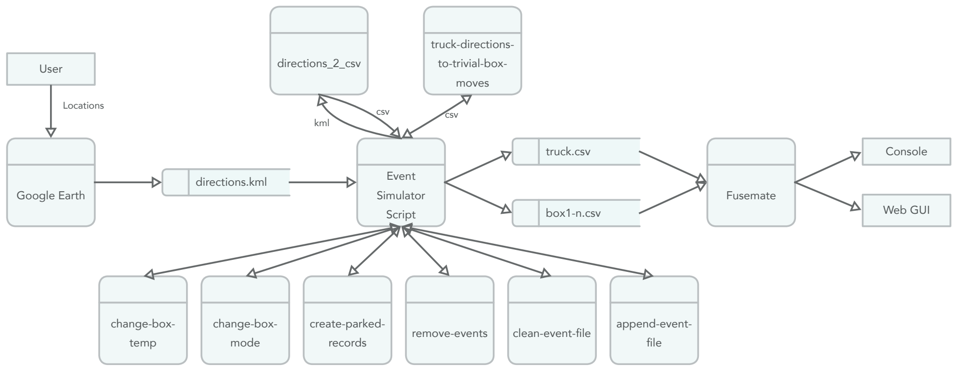


Fig. 1. Data flow through the event simulator.

[2]csv: "comma separated values"

With the help of the simulator, we created four scenarios reflecting various anomalies:

Normal. This is the baseline scenario. Ten boxes are loaded onto the truck at the storage site at Morningside. The truck moves to Woolloongabba where five boxes are delivered, put into the local cool room and sensors switched off. The rest of the boxes continue to the final destination at Annerley, where they are unloaded and put into the cool room. The temperature events of the box sensors are considered "normal": all boxes are properly cooled in the truck; at Woolloongabba, the cargo doors are opened and all boxes slightly rise in temperature; as the cargo doors are closed, the temperature of the boxes inside fall again, but the temperature of the boxes at the loading dock rise further and faster until they are placed in the cool room. Similar events can be observed at Annerley.

Latecool. In this scenario, the boxes unloaded at Woolloongabba are not immediately put into the cool room and their temperature consequently raise too high.

Missingbox. In this scenario, the sensor of one box stops to send data and it never reaches its destination.

Cabinbox. In this scenario one box' temperature raises after the first stop at Woolloongabba as it has been taken into the driver cabin. The box reaches Annerley, where it is cooled down along with the other boxes after unloading. Figures 2 shows the change in temperature of the different groups of boxes over time.

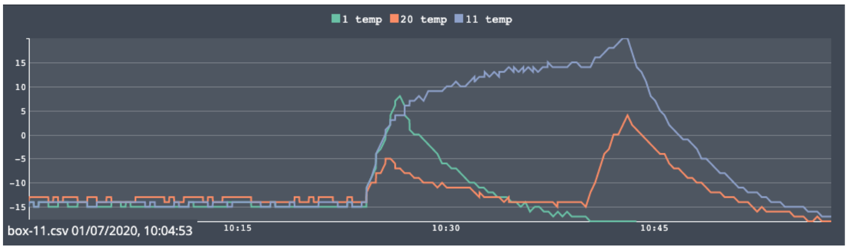


Fig. 2. Cabinbox scenario: In contrast to the normal scenario, one box is not cooled properly on the way to Annerley (blue curve). Its temperature raises to 15°C, which may indicate that it travelled in the driver cabin. Once the truck arrives at the destination, the box temperature climbs as it is now outside together with the other boxes. Once in the freezer room all temperatures fall again towards -20°C.

3 FUSEMATE SITUATIONAL AWARENESS SYSTEM

We used our *Fusemate* situational awareness system [3] for domain modelling. At its core, Fusemate is a logic programming system that provides the user with a knowledge representation formalism for describing state transitions with if-then rules. It follows the answer-set paradigm [7] where rules are processed in a bottom-up way for computing logical models of a given program and (in our case time stamped) input data. Fusemate's inference engine supports reasoning with erroneous or incomplete data by maintaining multiple possible logical models at a time. The logical models can be refined or revoked in light of new data arriving, by means of a belief revision operator [3]. Fusemate has (also) been designed with re-usability and strong external interfaceability in mind (e.g., databases, XML/JSON, web servers) [4].

Specifically for situational awareness applications, Fusemate processes a stream of events and proposes plausible explanations for the current state of the world (Fig 3). The explanations are meant to be semantically rich and meaningful to a human decision maker. For that, Fusemate needs to be equipped with a model that describes structural, causal and temporal relationships of the domain under consideration.

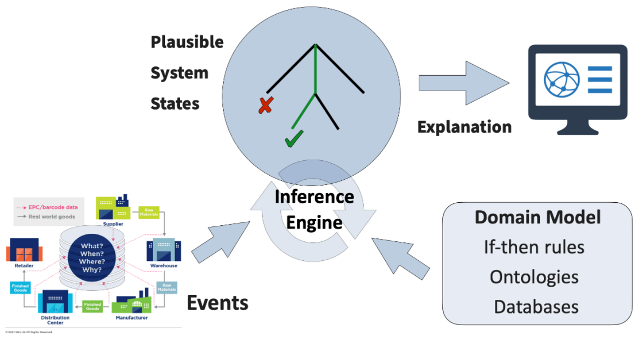


Fig. 3. Fusemate situational awareness system architecture.

In this project we instantiated Fusemate with a rule model for traceability and anomaly detection. Each rule contributes a tiny fragment of domain knowledge. For example (if-then rules are of the form *conclusions* `:-` *conditions*):

```
EnRouteToLoc(time, since, id, from, to) :-
  Leg(from, to), TruckEvent(time, at id), LeftLoc(since < time, id, from),
  NOT ( EntersLoc(t, id, to), since < t ∧ t < time )
```

This rule roughly says: "At any current `time`, if a given truck `id` has left most recently at time `since` a location `from` (e.g. "Morningside warehouse"), and there is a defined leg to location `to` (e.g. "Annerley client") and the truck has not yet arrived at `to` then the truck is en route to location `to`.

4 FUSEMATE BEEFLEDGER MODEL

Developing the full rule model for the BeefLedger transport chain was a major activity and outcome of this project. An in-depth description is beyond the scope of this paper, but we can provide an overview in terms of *five main categories*:[3]

Rules category	Summary description or example	Nr of rules
Fixing the event stream	Deal with sensor dropouts	9
Event Calculus		
General	Framework for actions and their consequences	6
Domain specific	E.g. "loading" a box causes "on truck"	13
Adding actions in retrospect	Provides explanations for observations E.g., if box is no longer close to truck and truck is at warehouse then action was "unload"	5
Domain specific notifications		
Info	E.g., "truck left Morningside"	7
Warn	E.g., "box temperature high for one minute"	5
Anomaly	E.g., "box disappeared during transport from Morningside to Annerley"	5
Domain agnostic anomaly	E.g., "box 10 disappeared for 10 minutes"	2
Other		6

[3]The Fusemate system including the model and data sets described in this paper can be downloaded at https://bitbucket.csiro.au/users/bau050/repos/fusemate/.

Rules for fixing the event stream. These rules retrospectively add events for recovering sensor dropouts *if context supports it.* As an example, there is a rule for recovering the GPS location of a box at time t that applies if the box is known to be on the truck at time t_1 shortly before t and time t_2 shortly after t. The rule then infers the GPS location of the box at time t to be the same as the truck location at that time t (if available). The rationale is that nothing out of the ordinary can have happened to the box between t_1 and t_2.

In our experiments we found that even if at a small rate, event recovery is important for avoiding false positives, such as mistaking a brief sensor dropout for a box tampering event.

Event Calculus. The Event Calculus is a general logical language for representing and reasoning about actions and their consequences [11]. We encoded it with Fusemate rules and applied it to our domain objects of interest, such as trucks, boxes and named locations (warehouses). For example:

```
Initiates(time, Unload(boxId, truckId), BoxAtLoc(boxId, loc)) :-
  Box(boxId), HoldsAt(time, AtLoc(truckId, loc))
```

This rule expresses the "obvious" fact that, if a box is unloaded (the action) from a truck that is at a given location at a give time, then the box is at that location at that time (the consequence). The `Initiates` predicate has the effect that the `BoxAtLoc` consequence becomes true if a concrete event `Happens(`*`time`*`, Unload(`*`boxId`*`, `*`truckId`*`))` happens (it remains true until a corresponding "Terminates" invalidates it). The following table summarizes our specific Event Calculus rules:

Action	**Consequences**	**Real-world event**
`Load(boxId, truckId)`	`On(boxId, truckId)`	Box movements on/off a truck
`Unload(boxId, truckId)`	`BoxAtLoc(boxId, loc)`	
`LeftLoc(truckId)`	`EnRouteToLoc(truckId, from, to)`	Truck movements in terms of named
`EntersLoc(truckId)`	`TruckAtLoc(truckId, loc)`	locations (warehouses)
`SwitchOff(boxId)`	`SwitchedOff(boxId)`	Box sensors: switch on/off; high temperature
`HighTempOn(boxId)`	`HighTemp(boxId)`	periods
`HighTempOff(boxId)`		

The actions and their consequences in the table above are the main device for explaining Fusemate's plausible current state computation to the user. Such explanations become even more relevant when actions are inferred by Fusemate rules, as per the following.

Rules for adding actions in retrospect. While Event Calculus actions can be part of an event stream, none were part of the BeefLedger data sets. In order to keep the experiment self-contained we defined rules for inferring such actions from the given events. These rules work similarly to the "rules for fixing the event stream" above. A good example is a rule that concludes a `Happens(`*`time`*`, Unload(`*`boxId`*`, `*`truckId`*`))` action of a box from a truck if the GPS coordinates of a box and a truck have become sufficiently distant while the box was Initiated to be on the truck.

Rules for domain specific notifications. We wrote domain specific rules that summarize informative findings, warnings and anomalies in an informal way. They build on the Event Calculus predicates above. For example:

```
Anomaly(time, s"[MissingBox] Box $boxId is not known to be on Truck $truckId on
                arrival at $toLoc, however truck left $fromLoc with
                box $boxId on board at $prev") :-
  Happens(time, EntersLoc(truckId, toLoc)),
  Happens(prev < time, LeftLoc(truckId, fromLoc)),
  Leg(fromLoc, toLoc), HoldsAt(prev, On(boxId, truckId)),
  NOT( HoldsAt(time, On(boxId, truckId)) )
```

The anomaly-rule above applies if a truck completed travelling a certain leg that started with a certain box loaded on it and ended at the destination without evidence of the box still on it. (That could be explained by, e.g., an Unload action along the trip.)

Other rules under this heading deal with box temperature anomalies and box movement anomalies (see below for more details).

Rules for domain agnostic anomalies. These rules are generic and can be applied in many domains. The idea is to identify clusters of "similar" objects and propose anomalies via cluster outliers. This scheme is parameterized only in a similarity measure and a feature distance function. There are two kinds of outliers:

Static outliers. Given a cluster of similar objects, a static outlier is one of these objects that deviates from the majority of the objects with respect to certain feature values (a nearest-neighborhood technique).
In our concrete case, we defined similarity of boxes by being close to each other at the same time (all the boxes loaded on a truck at a given time, for instance). As a feature value we used the box temperature. This is enough to identify the "cabinbox" scenario above as anomalous.

Dynamic outliers. This notion of outlier detection monitors the development of clusters over time. A dynamic outlier is, in the BeefLedger domain, a box disappearing from a cluster that reappears in the same cluster sometime later. In practice, this could indicate box tampering.

5 USER EXPERIENCE

Fusemate is a research prototype. The logic programs can be made with any text editor, and data can be provided in CSV, XML or JSON. Input data can be provided in a bunch or streamed in real-time (or faster) to a running Fusemate server. Providing and maintaining infrastructure around Fusemate is straightforward, thanks to Fusemate being implemented as an embedding into a full fledged programming language, Scala [10]. Fusemate's output goes to the terminal, see Figure 4.

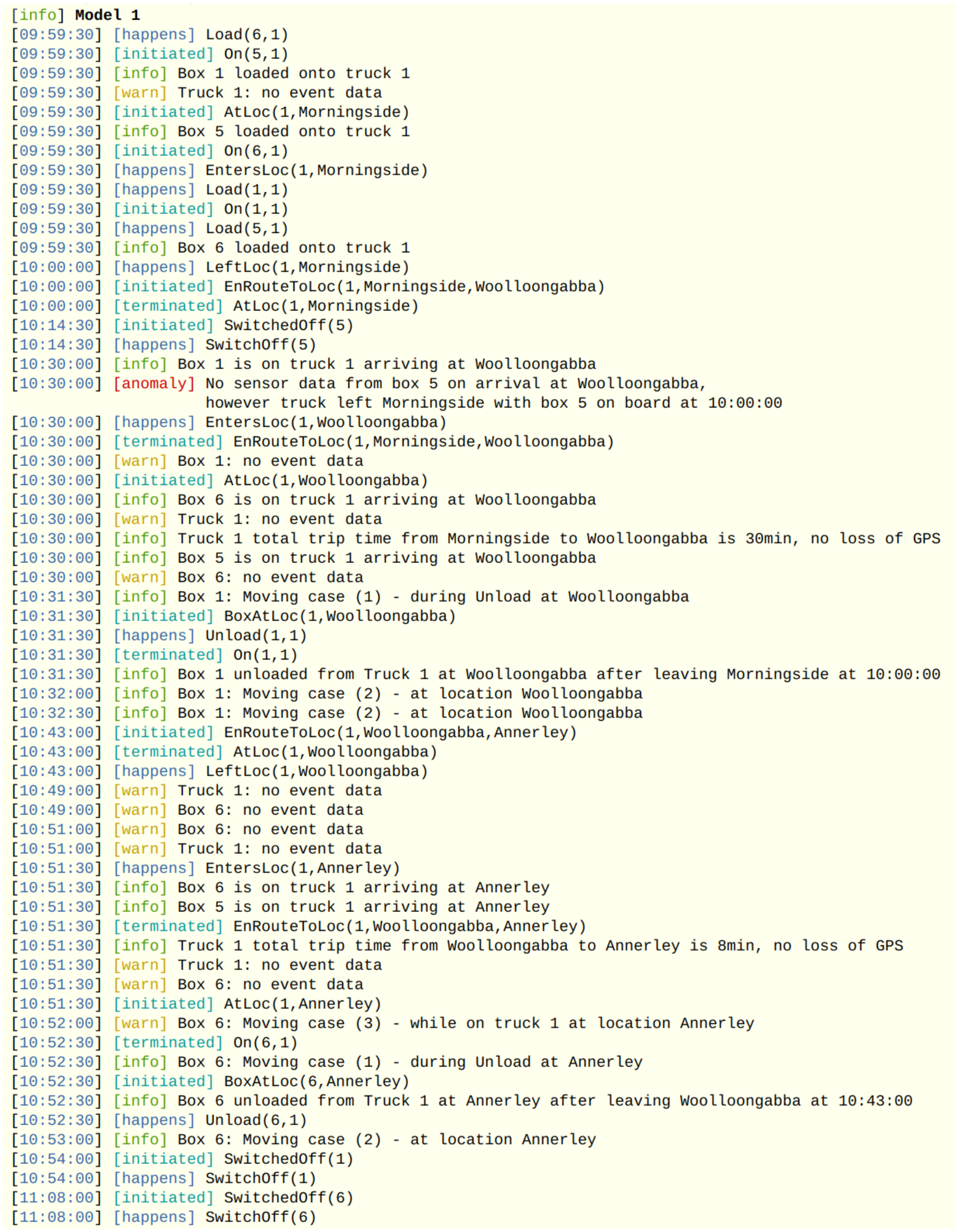
For this project, we implemented a web-based tool for the visualization of Fusemate runs and state analysis, see Figure 5. To allow for events to be displayed immediately when they occur, we designed the system as a Node.js web service that listens for event data, log messages and other calls. Incoming data is then being passed on to a client running in the user's web browser and the visualization is thereby updated continuously as time progresses. The client uses a leaflet map component and is implemented using the MERN stack. The interface is useful for visually correlating the Fusemate output with the locations at which they occur.

6 EXPERIMENTAL RESULTS

We applied Fusemate with the rule set in Section 4 to our five data sets in Section 2 . The runtime in each case is about 30sec. The purpose of our experiments was to evaluate if Fusemate is able to compute the expected warnings and anomalies and without too many false positives/negatives.

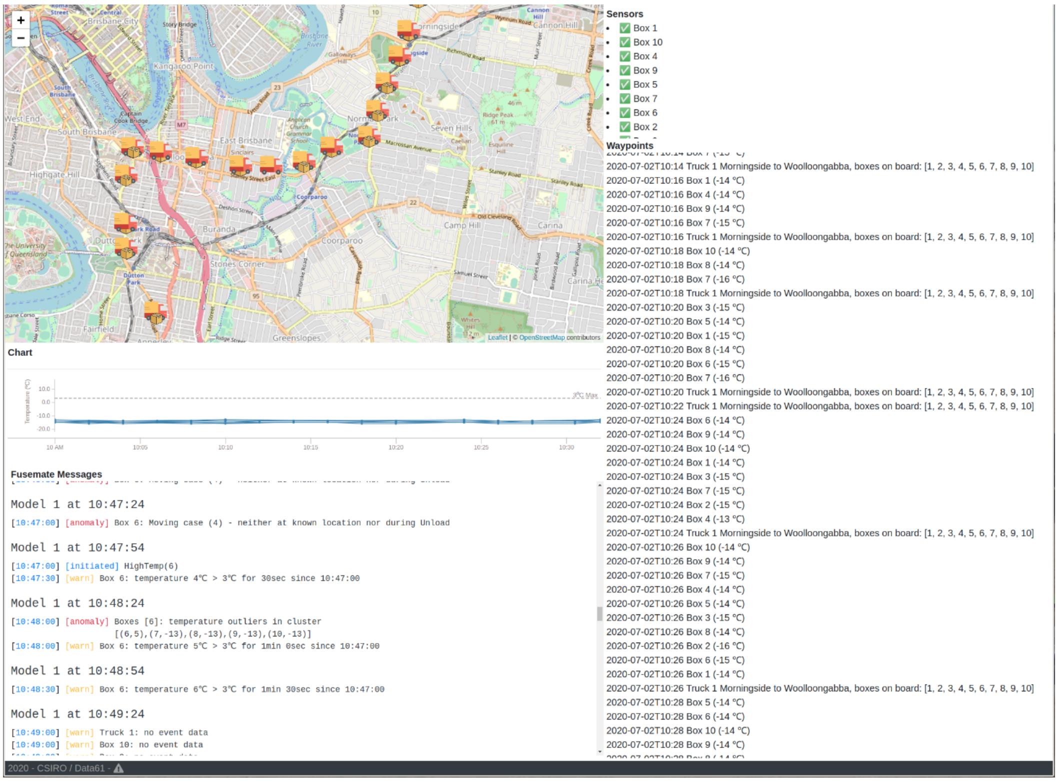
Six types of anomalies. The Fusemate model distinguishes between six different types of anomalies, which are the following::

MissingBox: a box was loaded onto the truck on departure but did not arrive at a waypoint or destination.
BoxHighTemp: the temperature of a box exceeds a threshold for a prolonged time.
BoxMove: a box is moved in an unexpected way, see "WP2.1 box movements" below.
GPSDead: a box ceases emitting GPS and temperature events.



```
[info] Model 1
[09:59:30] [happens] Load(6,1)
[09:59:30] [initiated] On(5,1)
[09:59:30] [info] Box 1 loaded onto truck 1
[09:59:30] [warn] Truck 1: no event data
[09:59:30] [initiated] AtLoc(1,Morningside)
[09:59:30] [info] Box 5 loaded onto truck 1
[09:59:30] [initiated] On(6,1)
[09:59:30] [happens] EntersLoc(1,Morningside)
[09:59:30] [happens] Load(1,1)
[09:59:30] [initiated] On(1,1)
[09:59:30] [happens] Load(5,1)
[09:59:30] [info] Box 6 loaded onto truck 1
[10:00:00] [happens] LeftLoc(1,Morningside)
[10:00:00] [initiated] EnRouteToLoc(1,Morningside,Woolloongabba)
[10:00:00] [terminated] AtLoc(1,Morningside)
[10:14:30] [initiated] SwitchedOff(5)
[10:14:30] [happens] SwitchOff(5)
[10:30:00] [info] Box 1 is on truck 1 arriving at Woolloongabba
[10:30:00] [anomaly] No sensor data from box 5 on arrival at Woolloongabba,
                     however truck left Morningside with box 5 on board at 10:00:00
[10:30:00] [happens] EntersLoc(1,Woolloongabba)
[10:30:00] [terminated] EnRouteToLoc(1,Morningside,Woolloongabba)
[10:30:00] [warn] Box 1: no event data
[10:30:00] [initiated] AtLoc(1,Woolloongabba)
[10:30:00] [info] Box 6 is on truck 1 arriving at Woolloongabba
[10:30:00] [warn] Truck 1: no event data
[10:30:00] [info] Truck 1 total trip time from Morningside to Woolloongabba is 30min, no loss of GPS
[10:30:00] [info] Box 5 is on truck 1 arriving at Woolloongabba
[10:30:00] [warn] Box 6: no event data
[10:31:30] [info] Box 1: Moving case (1) - during Unload at Woolloongabba
[10:31:30] [initiated] BoxAtLoc(1,Woolloongabba)
[10:31:30] [happens] Unload(1,1)
[10:31:30] [terminated] On(1,1)
[10:31:30] [info] Box 1 unloaded from Truck 1 at Woolloongabba after leaving Morningside at 10:00:00
[10:32:00] [info] Box 1: Moving case (2) - at location Woolloongabba
[10:32:30] [info] Box 1: Moving case (2) - at location Woolloongabba
[10:43:00] [initiated] EnRouteToLoc(1,Woolloongabba,Annerley)
[10:43:00] [terminated] AtLoc(1,Woolloongabba)
[10:43:00] [happens] LeftLoc(1,Woolloongabba)
[10:49:00] [warn] Truck 1: no event data
[10:49:00] [warn] Box 6: no event data
[10:51:00] [warn] Box 6: no event data
[10:51:00] [warn] Truck 1: no event data
[10:51:30] [happens] EntersLoc(1,Annerley)
[10:51:30] [info] Box 6 is on truck 1 arriving at Annerley
[10:51:30] [info] Box 5 is on truck 1 arriving at Annerley
[10:51:30] [terminated] EnRouteToLoc(1,Woolloongabba,Annerley)
[10:51:30] [info] Truck 1 total trip time from Woolloongabba to Annerley is 8min, no loss of GPS
[10:51:30] [warn] Truck 1: no event data
[10:51:30] [warn] Box 6: no event data
[10:51:30] [initiated] AtLoc(1,Annerley)
[10:52:00] [warn] Box 6: Moving case (3) - while on truck 1 at location Annerley
[10:52:30] [terminated] On(6,1)
[10:52:30] [info] Box 6: Moving case (1) - during Unload at Annerley
[10:52:30] [initiated] BoxAtLoc(6,Annerley)
[10:52:30] [info] Box 6 unloaded from Truck 1 at Annerley after leaving Woolloongabba at 10:43:00
[10:52:30] [happens] Unload(6,1)
[10:53:00] [info] Box 6: Moving case (2) - at location Annerley
[10:54:00] [initiated] SwitchedOff(1)
[10:54:00] [happens] SwitchOff(1)
[11:08:00] [initiated] SwitchedOff(6)
[11:08:00] [happens] SwitchOff(6)
```

Fig. 4. Fusemate log output for the synthetic ”missingbox” example. It shows most aspects of the Fusemate modelling explained above, including the inferred actions as plausible explanations for the current system state. The example is slightly abridged for readability, by omitting 7 of the 10 boxes. With the full set of 10 boxes, there are two false positive anomalies which are due to the “fixing the event stream rules” not being effective.



Sensors

- ✅ Box 1
- ✅ Box 10
- ✅ Box 4
- ✅ Box 9
- ✅ Box 5
- ✅ Box 7
- ✅ Box 6
- ✅ Box 2

Waypoints

```
2020-07-02T10:14 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:16 Box 1 (-14 °C)
2020-07-02T10:16 Box 4 (-14 °C)
2020-07-02T10:16 Box 9 (-14 °C)
2020-07-02T10:16 Box 7 (-15 °C)
2020-07-02T10:16 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:18 Box 10 (-14 °C)
2020-07-02T10:18 Box 8 (-14 °C)
2020-07-02T10:18 Box 7 (-16 °C)
2020-07-02T10:18 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:20 Box 3 (-15 °C)
2020-07-02T10:20 Box 5 (-14 °C)
2020-07-02T10:20 Box 1 (-15 °C)
2020-07-02T10:20 Box 8 (-14 °C)
2020-07-02T10:20 Box 6 (-15 °C)
2020-07-02T10:20 Box 7 (-16 °C)
2020-07-02T10:20 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:22 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:24 Box 6 (-14 °C)
2020-07-02T10:24 Box 9 (-14 °C)
2020-07-02T10:24 Box 10 (-14 °C)
2020-07-02T10:24 Box 1 (-14 °C)
2020-07-02T10:24 Box 3 (-15 °C)
2020-07-02T10:24 Box 7 (-15 °C)
2020-07-02T10:24 Box 2 (-15 °C)
2020-07-02T10:24 Box 4 (-13 °C)
2020-07-02T10:24 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:26 Box 10 (-14 °C)
2020-07-02T10:26 Box 9 (-14 °C)
2020-07-02T10:26 Box 7 (-15 °C)
2020-07-02T10:26 Box 4 (-14 °C)
2020-07-02T10:26 Box 5 (-14 °C)
2020-07-02T10:26 Box 3 (-15 °C)
2020-07-02T10:26 Box 8 (-14 °C)
2020-07-02T10:26 Box 2 (-16 °C)
2020-07-02T10:26 Box 6 (-15 °C)
2020-07-02T10:26 Box 1 (-14 °C)
2020-07-02T10:26 Truck 1 Morningside to Woolloongabba, boxes on board: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2020-07-02T10:28 Box 5 (-14 °C)
2020-07-02T10:28 Box 6 (-14 °C)
2020-07-02T10:28 Box 10 (-14 °C)
2020-07-02T10:28 Box 9 (-14 °C)
```

Fusemate Messages

```
Model 1 at 10:47:24
[10:47:00] [anomaly] Box 6: Moving case (4) - neither at known location nor during Unload

Model 1 at 10:47:54
[10:47:00] [initiated] HighTemp(6)
[10:47:30] [warn] Box 6: temperature 4℃ > 3℃ for 30sec since 10:47:00

Model 1 at 10:48:24
[10:48:00] [anomaly] Boxes [6]: temperature outliers in cluster
                     [(6,5),(7,-13),(8,-13),(9,-13),(10,-13)]
[10:48:00] [warn] Box 6: temperature 5℃ > 3℃ for 1min 0sec since 10:47:00

Model 1 at 10:48:54
[10:48:30] [warn] Box 6: temperature 6℃ > 3℃ for 1min 30sec since 10:47:00

Model 1 at 10:49:24
[10:49:00] [warn] Truck 1: no event data
[10:49:00] [warn] Box 10: no event data
```

Fig. 5. Web-based visualisation of Fusemate for boxed meat transport experiments. Depicted is the truck route (top left), box temperature graph (below map), Fusemate conclusions regarding the current position of the truck and the boxes loaded (right), and a console log with updates on the current model over time in terms of actions, warnings and alerts (bottom left).

BoxClusterDisappear: see "rules for domain agnostic anomaly detection" above.
BoxClusterTemp: see "rules for domain agnostic anomaly detection" above.

The following table lists how often these anomalies were issued. Each number given is the total over all 10 boxes for a given anomaly type described in the first column Numbers in parenthesis are the correct numbers.

Scenario	**Missing Box**	**BoxHigh Temp**	**BoxMove**	**GPSDead**	**BoxCluster Disappear**	**BoxCluster Temp**
Normal	0 (0)	0 (0)	0 (0)	0 (0)	2 (0)	0 (0)
Latecool	0 (0)	14 (14)	0 (0)	0 (0)	2 (0)	0 (0)
Cabinbox	0 (0)	13 (13)	4 (4)	0 (0)	3 (1)	6 (6)
Missingbox	1 (1)	0 (0)	0 (0)	0 (0)	2 (0)	0 (0)

The results confirm a strong correlation between the type of anomalies detected and the test case designed to exercise them. The BoxClusterTemp anomaly in the cabinbox scenario is the expected result, as the box heats up but is still in proximity to the other boxes on the truck.

The BoxClusterDisappear anomalies in the cabinbox scenario are false positives, with one exception for the cabinbox scenario. These could be avoided by tuning parameters. No other false positives were issued. In summary, of the 47 reported anomalies, 45 were correct (95%).

Box movement anomalies. Of particular interest to us was to integrate box movement events into the Fusemate modelling. Here, a "box movement" is a spontaneous change of a box's orientation, which may happen due to, e.g., the box falling from a stack of boxes, or the box being moved around by the driver.

Our interest is motivated by novel techniques developed in a neighbouring Data61 project for detecting such movements. This capability is actually a side effect of utilizing piezo sensors for harvesting electric energy from vibration movement. With machine learning techniques, sensor movements are classified as anomalous (or not) by comparing transport trips with historical ones.

With Fusemate, we took a complementary approach which allows us to classify box movements based on domain knowledge and the context conditions they occur in. This way we achieve differentiated explanations of the movements that are difficult to achieve by history-based assessment. For example, a box movement in the context of the box being unloaded from the truck is likely to be innocuous; it may be even expected, instead of constituting an anomaly, proper.

We wrote Fusemate rules for analysing box movements in context of the BeefLedger scenarios. These rules classify each box movement into a severity level "Info", "Warn" or "Anomaly". They conclude one of the following four categories:

Case (1) - Info: DuringUnload: the box was moved during unloading at a known location (e.g. warehouse)

Case (2) - Info: AtLoc: the box was moved at a known location

Case (3) - Warn: OnTruckAtLock: the box was moved while on the truck, and the truck was at a known location.

Case (4) - Anomaly: the box was moved in any other case, e.g. during transport The rationale behind this last rule is that movements during transport should not happen (box fell from the stack? Tampering occurred after opening?)

The log output in Figure 4 shows some concrete examples for cases (1) – (3).

We applied the box movement analysis to the test scenarios. The following table shows the number of notifications in each category:

	Info	Info	Warn	Anomaly
Scenario	DuringUnload	AtLoc	OnTruckAtLoc	None of the left
Normal	10	13	5	0
Latecool	10	13	5	0
Cabinbox	10	13	5	4
Missingbox	9	12	5	0

In sum, the data sets contain 114 box movements, which were classified by Fusemate into 90 Info, 20 Warn and 4 Anomaly notifications. This analysis fully corresponds to our expectations. In particular, the only expected anomalies are the ones in the cabinbox scenario. In numerical terms, the Fusemate analysis correctly eliminates 96% of all box movements as non-anomalous.

7 CONCLUSIONS

The main research question in this paper is of applied nature: how can we model complex dependencies in a food supply chain in way that enables logical inference for situational awareness? This is a non-trivial question, as realistically, modelling of a complex domain often needs to be a compromise because extracting all relevant aspects and their dependencies is not achievable in full breadth and depth.

In this paper we proposed an approach based on logic programming. From a knowledge engineer's perspective, its purpose was to evaluate if our modelling language is expressive enough to adequately model (certain) situational awareness tasks in the food supply chain. Correspondingly, the model that we obtained was a major activity and outcome of this project. Thanks to the declarative nature of logic programming, the model could be developed and tested in an incremental and modular way. Moreover, with experiments on simulated data we were able to show that our model can detect the defined anomalies with high accurracy.

The model could be improved for coverage, say, by integrating more sensor outputs or "paperwork" events stored in a database. Also its functionality could be improved, e.g., for quality assessment as a function of actual transport temperature conditions, and pricing of the product as it arrives at the final destination. With the experiences gained, these extensions would be routine exercises by now.

Next steps include advancing the modelling language capabilities of Fusemate. Of major interest will be the development of a comprehensive "logical theory of situational awareness". The provision of the Event Calculus is already a good starting point, but it would be useful to include other aspects as well, like reasoning on space (e.g., notions of neighbourhood or overlapping areas) and argumentation frameworks (where rules can take priority over others).

A more general research question is how to enhance the developed software tool into a fully-fledged method for situational awareness in supply chain and other applications. Ultimately, that method should also make logical modelling accessible to supply chain experts and end users.

REFERENCES

[1] 2021. BeefLedger. https://beefledger.io/.

[2] A. Artikis, Anastasios Skarlatidis, François Portet, and G. Paliouras. 2012. Logic-based event recognition. *Knowl. Eng. Rev.* 27 (2012), 469–506.

[3] Peter Baumgartner. 2020. Possible Models Computation and Revision – A Practical Approach. In *International Joint Conference on Automated Reasoning (LNAI, Vol. 12166)*, N. Peltier and V. Sofronie-Stokkermans (Eds.). Springer International Publishing, Cham, 337–355. https://doi.org/10.1007/978-3-030-51074-9_19

[4] Peter Baumgartner. 2021. The Fusemate Logic Programming System (System Description). https://arxiv.org/abs/2103.01395

[5] Peter Baumgartner and Patrik Haslum. 2021. Situational Awareness for Industrial Operations. In *Data and Decision Sciences in Action 2*, Andreas T. Ernst, Simon Dunstall, Rodolfo García-Flores, Marthie Grobler, and David Marlow (Eds.). Springer International Publishing, Cham, 125–137. ASOR-2018.pdf

[6] Harald Beck, Minh Dao-Tran, and Thomas Eiter. 2018. LARS: A Logic-based framework for Analytic Reasoning over Streams. *Artificial Intelligence* 261 (08 2018), 16–70. https://doi.org/10.1016/j.artint.2018.04.003

[7] Wolfgang Faber. 2020. An Introduction to Answer Set Programming and Some of Its Extensions. In *Reasoning Web. Declarative Artificial Intelligence - 16th International Summer School 2020, Oslo, Norway, June 24-26, 2020, Tutorial Lectures (Lecture Notes in Computer Science, Vol. 12258)*, Marco Manna and Andreas Pieris (Eds.). Springer, 149–185. https://doi.org/10.1007/978-3-030-60067-9_6

[8] David McKinna and Catherine Wall. 2020. *Commercial application of supply chain integrity and shelf life systems.* Technical Report. Meat and Livestock Australia Limited, NORTH SYDNEY NSW 2059. https://www.mla.com.au/research-and-development/reports/2020/commercial-application-of-supply-chain-integrity-and-shelf-life-systems/.

[9] A. Medvedev, A. Hassani, P. D. Haghighi, S. Ling, M. Indrawan-Santiago, A. Zaslavsky, U. Fastenrath, F. Mayer, P. P. Jayaraman, and N. Kolbe. 2018. Situation Modelling, Representation, and Querying in Context-as-a-Service IoT Platform. In *2018 Global Internet of Things Summit (GIoTS).* 1–6. https://doi.org/10.1109/GIOTS.2018.8534571

[10] Scala [n.d.]. The Scala Programming Language. https://www.scala-lang.org.

[11] Murray Shanahan. 1999. *The Event Calculus Explained.* Springer Berlin Heidelberg, Berlin, Heidelberg, 409–430. https://doi.org/10.1007/3-540-48317-9_17

[12] Dhananjay Singh, Gaurav Tripathi, and Antonio J. Jara. 2014. A Survey of Internet-of-Things: Future Vision, Architecture, Challenges and Services. In *2014 IEEE World Forum on Internet of Things, WF-IoT 2014.* IEEE.

[13] Monika Solanki and Christopher Brewster. 2014. Detecting EPCIS exceptions in linked traceability streams across supply chain business processes. In *SEMANTICS.* ACM, 24–31.