2014 IEEE Eighth International Conference on Self-Adaptive and Self-Organizing Systems Workshops

Data verification for collective adaptive systems: spatial model-checking of vehicle location data

Vincenzo Ciancia*, Stephen Gilmore†, Diego Latella*, Michele Loreti‡§, and Mieke Massink*

*Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo", Italy
†Laboratory for Foundations of Computer Science, University of Edinburgh, Edinburgh, Scotland
‡Dipartimento di Statistica, Informatica, Applicazioni "G. Parenti", Università di Firenze, Italy
§IMT Advanced Studies Lucca, Italy

***Abstract*—In this paper we present the use of a novel spatial model-checker to detect problems in the data which an adaptive system gathers in order to inform future action. We categorise received data as being plausible, implausible, possible or problematic. Data correctness is essential to ensure correct functionality in systems which adapt in response to data and our categorisation influences the degree of caution which should be used in acting in response to this received data. We illustrate the theory with a concrete example of detecting errors in vehicle location data for buses in the city of Edinburgh. Vehicle location data is visualised symbolically on a street map, and categories of problems identified by the spatial model-checker are rendered by repainting the symbols for vehicles in different colours.**

I. Introduction

Collective adaptive systems depend crucially on real-time data collection subsystems which allow them to reflect on their operation, detect problems in their service, and report these problems back to system operators or to system users. These data collection systems are often built from physical components such as sensors and receivers which have limits to their engineering, meaning that they can, and do, sometimes deliver inaccurate measurement data.

In a small-scale supervised system where the measurement data is interpreted by a human before any action is taken, erroneous data such as this may not be very problematic because it can be intelligently checked, or even discarded, before any action is taken as a consequence of the erroneous data. In a collective adaptive system however, the demands of scale and responsive adaptivity may mean that there is no human oversight of the data before action is taken as a consequence of the data received.

Physical components can only deliver accurate information up to their physical limits and size and weight considerations often severely curtail the amount of processing which can be done on an embedded device. These practicalities mean that the responsibility for dealing with erroneous data then lies with the system itself, and that it must make efforts to automate the process of data checking and cleaning before acting in response to data received.

Spatial considerations and location play an increasing role in systems. Knowing where the system's assets are located is often of critical importance for correct functionality. In a modern public transport system many aspects of well-regulated operation depend on accurate fleet management, supported by an automatic vehicle location (AVL) system. AVL data drives other systems providing information to passengers and system operators such as bus arrival prediction systems.

A smart transport system is a collective adaptive system because it is a large-scale system which adapts to unexpected problems in service delivery such as road closures and mechanical failure of vehicles by re-routing vehicles and automatically adjusting schedules to compensate for the delays introduced while simultaneously keeping passengers informed in real-time about changes to the service delivered. In order that the decisions made and the changes effected to the service delivery are the appropriate ones the system must have a high-quality supply of accurate and timely data, and the ability to detect problems in the data received.

We consider a spatial verification problem here, that of determining whether or not the AVL data received about vehicles in the fleet indicates an error condition in one of several categories:

- the AVL data received should be classified as being suspect or corrupt, by virtue of being too far from the expected data—this may indicate a problem with the hardware of the measurement device;
- the data indicates a problem in that, if it is correct, then one of the vehicles in the fleet is far from its expected position on its planned route—this may indicate a problem with the delivery of the service or it may be due to driver error; or
- the data seems plausible, and suggests a valid position for a vehicle but this vehicle seems to have deviated from its planned route for reasons of operational problems such as road closures or engineering works—this may indicate a problem with the road network.

The methods presented here could either be used for on-line or off-line data processing. In on-line data processing the smart infrastructure within the collective adaptive system would attempt to identify and classify problems in real-time, alerting operators to problems as they occur. In off-line data processing the infrastructure would attempt to identify and classify problems with the benefit of hindsight, providing plausible retrospective explanations for events so that the service operators can review their service and provide reports for service regulators or other authorities, and use the insights gained to improve the service at the next offering (for example, the following day).

978-1-4799-6378-2/14 $31.00 © 2014 IEEE
DOI 10.1109/SASOW.2014.16

II. Data correctness

In order to have a sense of when data is incorrect, we should have a sense of when it is correct. At a broad granularity of view, buses belong to a bus operator who has a specific responsibility for serving a geographical region of a country and any data which indicates that a bus is geographically very far from the operator's area of service (say, in another country) can be relatively easily eliminated. Surprisingly, examples of such gross incorrectness are not as uncommon as you might at first presume.

In the case of vehicle location data for a bus fleet, we have a natural sense of data correctness. Buses are not off-road vehicles so they should at all times be positioned on a road or in a small number of other specific locations, such as a garage.

Our overall goal is to classify bus fleet vehicle location data points as being plausible or suspect, and this closely aligns with their being on a road (or, allowing for inevitable measurement error, at least near to a road) and subsequently, on (or at least near to) the correct road. If data is classified as being plausible, then we have greater reassurance that it is appropriate for the system to take action based on the data received. If instead the data is classified as being suspect then we would like to proceed more cautiously and perhaps ask for human intervention at this stage to approve the course of action.

Some errors in GPS readings are always to be expected. Satellite-based triangulation requires very precise measurement of extremely fast signals. Environmental factors impact on the accuracy of this triangulation. Signal bounce off tall buildings can interrupt the GPS signal. Even heavy cloud cover, humidity and atmospheric pressure can have an impact on measurements.

Cold starts are particularly problematic because the GPS receiver does not have a current almanac, ephemeris, initial position or time and because of this it will give misleading measurements until its time-to-first-fix. These cold starts do not always occur at the start of the working day, but happen when a bus is brought into service at any point in the day, so it is not trivial to exclude them, and separate out the signal from the noise.

GPS-based AVL systems cannot function accurately without frequent location reports. When GPS signals are unavailable for an extended period of time Dead Reckoning Navigation (DRN) units come into play by computing changes in direction by measuring speed and direction, monitoring the speed and direction of the vehicle through on-board communication sensors. Dead reckoning navigation will become less accurate over time without a position update from the GPS receiver. This is another source of data errors.

A human observer can apply human intelligence to detect an erroneous GPS reading by plotting it on a map and comparing against the position of roads and buildings. However, this type of intelligent watchfulness cannot scale to tracking an entire fleet of buses (around 700 in the case of the city of Edinburgh) so we seek a scalable approach to detecting GPS errors which can be automated. We are using a novel spatial model-checker to implement this scalable approach.

III. Model checking

In its original conception, model checking [1] is a technique that is developed for the formal verification of properties of the behaviour of distributed and concurrent systems. It takes a formal model of the system and a property of interest, usually expressed as a temporal logic formula, and then checks, in an automatic way, whether the model satisfies the property. This way, properties of the temporal evolution of a system are considered, but properties of (physical) space typically are not. In recent work [2], [3] by some of the co-authors of this paper, a *spatial* model-checking approach has been developed. This technique builds on a topological interpretation of modal logics [4], dating back to Tarski, and has been extended with a spatial *until*-operator, inspired by the temporal until-operator. The latter can be used to define conditional reachability properties in a spatial setting. The logical operators have been lifted to a more general setting such that they can also be used on discrete, graph-based structures, which include geographic maps. In this paper the specific graph is a digital map, seen as a regular grid, where the nodes are the points in the map and where transitions connect each node to the adjacent nodes in the north, south, east and west direction. Moreover, an efficient proof-of-concept model-checker for this Spatial Logic for Closure Spaces (SLCS) has been implemented. The tool is able to interpret spatial logic formulas on digital images, providing graphical understanding of the meaning of formulas, and thus an immediate form of counterexample visualisation. Points that satisfy a particular formula can be visualised by a colour of choice on the digital map.

Although on the longer term we are interested to address both the temporal and spatial evolution of systems, in this paper we use the purely spatial model-checker for some very pragmatic purposes that are concerning the automatic identification of potential errors in automatically obtained large scale AVL data. The approach we follow is to project the AVL data, including the exact position of the vehicles, on an existing digital map of the relevant geographical area. We then use the spatial model-checker to identify and highlight regions of interest on this map. Examples of such regions may be buses that are positioned in an unlikely environment like a meadow or a lake, or not on the route of the bus.

The spatial logic consists of a very small number of basic operators that, in turn, are used to define a number of useful derived operators. Among the basic operators are the standard boolean connectives, negation and conjunction, the closure operator and the spatial until-operator. The closure operator $\Diamond$ has its origin in topological spatial logics. The formula $\Diamond\phi$ is satisfied by a point x in space if x satisfies ϕ or it is a direct neighbour of a point satisfying ϕ, see Fig. 2. The spatial until-operator $\mathscr{U}$ is a spatial version of the temporal until operator. A point x in space satisfies the formula $\phi\,\mathscr{U}\,\psi$ whenever it is included in an area A consisting of points satisfying formula ϕ and there is "no way out" from A unless passing through points in an area B that satisfies ψ (see Figure 1). Finally, we assume a set of basic propositions, which in our specific case identify the colour of a pixel in the digital map.

We will use a few derived operators in the properties shown in the next section. The first derived operator is the reachability operator $\phi\,\mathscr{R}\,\psi$. It is defined in terms of the spatial until operator as follows $\phi\,\mathscr{R}\,\psi \triangleq \neg((\neg\psi)\,\mathscr{U}\,(\neg\phi))$ and, abstracting

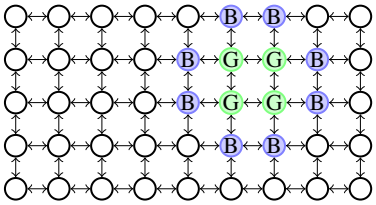


Fig. 1. The green nodes satisfy green until blue ($G \mathscr{U} B$)

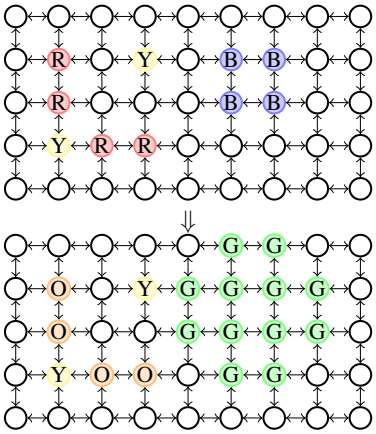


Fig. 2. The green (G) nodes (bottom) satisfy the closure of the blue (B) ones (top), and the orange (O) nodes (bottom) satisfy the reachability formula $R \wedge ((R \vee Y) \mathscr{R} Y)$ applied on the top figure, reaching yellow (Y) nodes only from and via red (R) ones

from some details, it identifies those points from which Ψ can be reached passing only by points satisfying Φ. Essentially it expresses that it is not possible to reach a point that does not satisfy ϕ anymore, while passing only by points that do, without having passed by a point that satisfies ψ, see Fig. 2. The formulation of reachability is using a double negation because the spatial until-formula is weak in the sense that $\phi \mathscr{U} \psi$ also holds when all points in the space satisfy ϕ and none of them ψ. Further examples of derived operators can be found in [2].

Roughly speaking, the SLCS model-checker takes as input a finite discrete model, and a formula ϕ and returns all the nodes in the graph that satisfy ϕ. The algorithm is a spatial variant of the global model-checking algorithm for CTL (Computation Tree Logic) [1]. The function that computes the satisfaction set is defined inductively on the structure of ϕ following a bottom-up approach. The original part of the algorithm concerns the spatial until operator $\phi \mathscr{U} \psi$. The algorithm performs a backwards search from the set of nodes that do not satisfy ϕ or ψ but that can be reached in one step from such nodes. The algorithm terminates in $\mathscr{O}(k.(|X|+|R|))$, where k is the number of operators in the formula, $|X|$ the number of nodes and $|R|$ the number of transitions in the graph [3].

IV. Categories of data issues

In this section we introduce the features of the map representation which we use and see the effect of the spatial properties which we evaluate. Figure 3 explains our conventions for representing buses, bus stops and the progress in time through observations on the map.

We have several categories of data issues to distinguish and condition action on:

- Plausible: The bus is positioned on a road and it is a road where we would expect to see a bus. Nothing about this data point seems problematic: adaptive behaviour based on this data observation would seem to be acting on good data.
- Implausible: This data point seems suspicious: the bus is positioned in an area of the city where we would not normally expect to see a bus (such as in a field, or a wood, or a pond). Unsupervised adaptive behaviour based on this data would be inadvisable.
- Possible: This data point has a bus positioned on a road but it is a side street when we were expecting to see the bus on a main road. The data is not implausible but it indicates that an unexpected event has perhaps occurred (a road closure, a traffic accident, or a diversion caused for another reason). Adaptive behaviour based on this data observation might be appropriate here.
- Problematic: This data point shows a bus on the expected route but it is less far along the route than previously reported. It is likely that either this data point is putting the bus behind its current position, or the previous data point put it ahead of its current position (or, possibly, the bus is reversing). Adaptive behaviour based on this data should be delayed until the uncertainty about which point is incorrect is resolved.

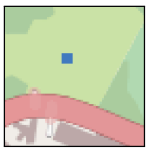
V. Data collection and selection

The vehicle location data which we have available to us comes from Lothian Buses, a well-respected bus operator serving the Edinburgh and Lothians region in Scotland. The data provides us with the GPS position of each bus in the fleet at any point in the day, at a granularity of thirty seconds. Network capacity limitations with the current positioning technology used mean that it is not possible to obtain polled data from all of the buses in the fleet more frequently than this. The installation of wi-fi access on buses and the adoption of a 4G service for data transmission will eliminate this restriction in future.

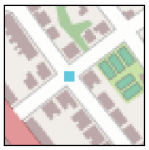
Taking different sections from the data allows us to address different spatial model-checking problems. We could choose to project out a single row of the data, in which case we would have an instantaneous snapshot of the current location of all of the buses in the fleet (a "helicopter view", allowing us to see all buses at one time). Alternatively we could choose to project out a single column of the data, in which case we would see the journey of a particular bus in the fleet as a function of time (a "passenger view", allowing us to see one bus at all times).

The helicopter view allows us to ask questions about congestion and adjacency allowing questions such as "Are there too many buses on Princes Street?". The passenger view

We use an OpenStreetMap representation for the map from which the names of streets have been removed for clarity. Main streets are pink, side streets are white.



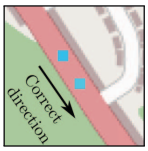
Errors in GPS data show up as bus observations which are not on any road.



Diverted buses are seen as being on side streets which are off the main bus route.

Buses are represented by blue squares on the map in a range of shades of blue. Bus stops are represented by orange squares. Main streets used for bus routes are pink.

The colour of a bus darkens slightly from one observation to the next. In this case the observation nearer to the bus stop has a later timestamp than the observation which is further away from the bus stop.



Unexpected changes in position are significant. If a later timestamped observation shows the bus less further along its route then this usually indicates a data error because buses rarely reverse along a road.

Fig. 3. Representations of buses, roads and bus stops on maps

allows us to ask questions about journeys and routes allowing questions such as "Did this bus deviate from its route and travel on any side roads today?" and "Which roads did this bus travel on?". In this example we will use the passenger view (seeing one bus at all times).

VI. Model-checking results

The SLCS model-checker has been used to analyse a coloured picture, representing a portion of the map of the city of Edinburgh, augmented with squares filled in special colours, denoting particular kinds of entities, as described in Section IV. Logic formulas are used in this example to detect problems in the data. We also show how to detect other features of interest of bus positions, and of roads. We shall now describe the formulas we use more in detail; these are interpreted on the model depicted in Figure 4.

Atomic predicates, when working on images, are actually colour ranges. Because of this, it is possible for us to analyse an image directly, without the need for additional meta-data. For this example, we selected 14 data points to represent buses[1], represented by different blue squares. The shade of blue depends upon the time at which the bus was in the given position. The shade ranges from light to dark, where lighter shades precede darker ones in time. Then, using atomic predicates on colours and colour ranges, we defined various basic formulas, among which: formula `bus` representing all the bus positions (using a colour range); formulas `pos1`, ..., `pos14`, representing the separate bus positions; formulas identifying streets (`street`) and main streets (`mainStreet`).

The SLCS spatial model-checker acts as a transformer, accepting an image as input and producing an updated image as its result. The model presented to the model-checker is the map image with reported bus positions marked (using blue squares) and the positions of bus stops marked (using orange squares). The model-checker evaluates spatial logic formulae and represents its results by *repainting* the bus positions which satisfy a predicate using a colour chosen for that predicate. (For example, positions of diverted buses can be repainted in red and positions of off-road buses can be repainted in violet.)

In the examples of formulae which follow we use the concrete input syntax of the SLCS model-checker where logical operator symbols such as $\neg$, $\wedge$ and $\vee$ are replaced by the keyboard characters `!`, `&` and `|`, respectively.

a) Spatial features of data points: in Figure 5 we show the result of identifying a portion of the main street for each position (depicted in yellow); this is done using the formula:

```
Let streetPortion(b) =
    mainStreet & (C^3 b)
    & ( ! (C^5 (bus & (!b))))
```

where `b` is instantiated to `pos1`, ..., `pos14`, and `C^n`, for n a natural number, is the nested application of the *closure*, or *dilation* operator. The formula dilates `b` by three pixels, and avoids points too close to *other* buses, in order to minimise possible overlaps. In the same figure, in red, we colour positions that are close to a bus stop. This uses the formula `close(bus,stop)`, where `close` is defined as follows:

```
Let close(a,b) = a & (C^30(b))
```

Thus, the formula intersects the points where the bus has passed with the points reachable from a stop by 30 pixels.

b) Implausible points: data points that are not positioned on a street are implausible. This is described by the formula:

```
Let busOutOfStreet =
    bus & (! (bus U street))
```

The formula characterises points that are part of any of the regions denoting bus positions, and are not surrounded by a street. Points satisfying this formula (thus, not plausible) are repainted in violet to produce the spatial model-checking result shown in Figure 6.

[1]The points have been selected artificially, in order to present a clean working example, but use of the spatial model-checker does not differ when dealing directly with the vehicle location data supplied by Lothian Buses.

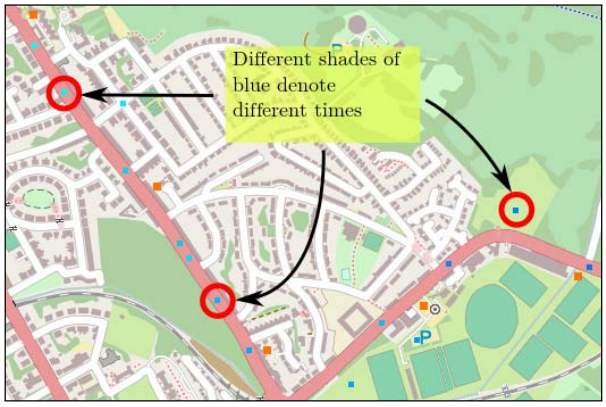


Fig. 4. Input model. Blue squares are bus positions; their order in time is described by the increasing darkness. Orange squares are the bus stops.

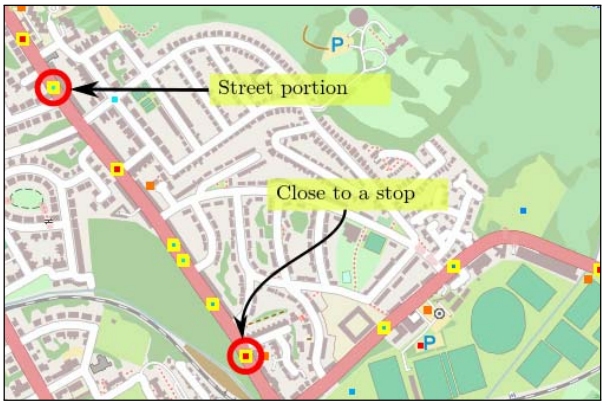


Fig. 5. Positions close to a stop are repainted in red, areas of the road surrounding all positions are repainted in yellow

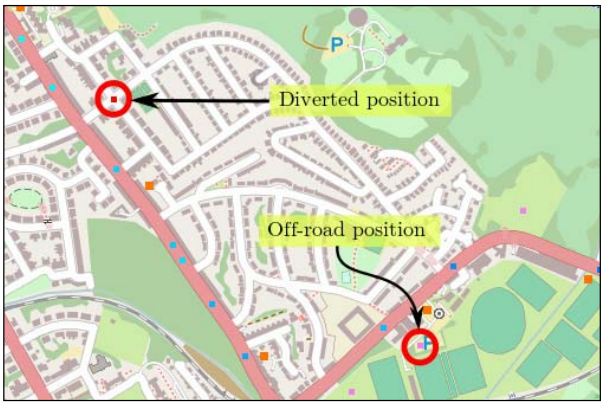


Fig. 6. Diverted positions (neither off road, nor on a main street) are repainted in red, off-road positions are repainted in violet

Fig. 7. Positions that are out of order (i.e., not between the previous and next position) are repainted in red

c) Possible points: diverted bus positions are represented by the formula:

```
Let divertedBus = bus U smallStreet
```

and are then repainted in red to produce the result shown in Figure 6.

d) Problematic points: our spatial logic is expressive enough to define properties related to the order of positions of the same bus at different times on a given road. We analyse the bus positions. For each position, we detect its immediately neighbouring positions, in order to check that these correspond to a preceding and following position, respectively. If this is not the case, the position is misplaced, even though it may still be on the main street. In Figure 7, such misplaced bus positions are painted in red. The formal specification of this property is complicated by the fact that the underlying graph of an image is not directed, thus it is not completely straightforward to specify precedence relations between points. The most important step is the definition of predicate `consecutivePos(p1,p2)`, given in Figure 8. The definition uses the *reachability* predicate $a \mathscr{R} b$, written as `reach(a,b)`. The definition of `consecutivePos` uses the previously defined predicate `streetPortion` to identify

```
Let consecutivePos(p1,p2) =  p1 U
   reach((streetPortion(p1) |
          mainStreet |
          streetPortion(p2))
         &
         (!(streetPortion(bus &
                    (!(p1 | p2))))),
       streetPortion(p2))
```

Fig. 8. Definition of `consecutivePos`

the region of points in `p1`, surrounded by the area from which `streetPortion(p2)` can be reached, passing by the main street, including the areas surrounding `p1` and `p2`, avoiding the areas of the street surrounding other buses. Note that `streetPortion(p)` is an area at least as wide as the main street. So no next bus positions can be reached following only points belonging to the main street. Using `consecutivePos`, predicate `wrongOrderPos` is defined as follows:

```
Let wrongOrderPos(p1,p2,p3) =
   (p2 U mainStreet) &
   (!(consecutivePos(p2,p1) &
```

```
consecutivePos(p2,p3)))
```

Given three positions `p1`, `p2`, `p3`, position `p2` is selected only if it is not strictly between `p1` and `p3`. The three positions are instantiated to all the strictly consecutive triplets between `pos1` and `pos14` in order to identify out-of-order positions.

VII. Related work

In [5] the authors present a spatial time-series model for tracking planned journeys although their main area of concern is vehicle speed forecasting rather than detection of outliers. The detection of outliers have been addressed by applying stochastic approaches. In [6] the authors present a method for snapping GPS data onto a road network using a Hidden Markov Model. They identify noisy GPS data as being the largest problem with snapping GPS readings onto road maps. They report that GPS signals can be reasonably modelled as a zero-mean Gaussian with a standard deviation of 10 metres. In [7] the authors present an approach to inferring the lane structure of roads from GPS data by fitting a mixture of Gaussians to GPS traces. This probabilistic approach naturally models the inherent noise in GPS data. In [8] the authors apply the concept of functional depth to the identification of outliers in GPS observations. Outliers are identified by detecting curves rather than central values as in traditional statistical tests for comparing distributions.

Different forms of spatial logic have also been proposed in computer science to refer to logics expressing properties of structured objects such as processes or data structures, in particular in the context of π-calculus (e.g. [9]) and mobile ambients with the related ambient logic (e.g. [10]). For example a binary logic operator has been introduced, $\Phi|\Psi$, that holds for a process P when this process is a parallel composition of two processes Q, satisfying Φ, and R, satisfying Ψ. Furthermore, in a stochastic setting, the Mobile Stochastic Logic (MoSL) [11] has been proposed to predicate on mobile processes in models specified in StoKLAIM, a stochastic extension of KLAIM based on the tuple-space model of computations.

Other variants of spatial logics concern the symbolic representation of the contents of images, and, combined with temporal logics, for sequences of images [12]. The latter is based on a discretisation of the space of the images in rectangular regions and the orthogonal projection of objects and regions onto Cartesian coordinate axes such that their possible intersections can be analysed from different perspectives, whereas in [13] a linear spatial superposition logic is defined for the specification of emergent behaviour. The logic is applied in the context of medical image analysis for the recognition of patterns.

The spatial logic SLCS used in the current paper, instead, addresses properties of discrete, graph-based models that include geographical maps.

VIII. Conclusions

The use of a spatial model checker provides us with a sophisticated tool for checking complex properties over systems where location plays an important role, as it does in many collective adaptive systems. Using this tool we have been able to detect and correct a wide range of location-related errors in vehicle location data.

Future work will focus on enhancing the logic and the model checker with a temporal perspective. Even though we were able to model simple time-related properties by turning time into a spatial feature (a particular shade of a color), the interplay of space and properly specified time allows one to define complex formulas. Interesting examples could be "the stops close to where the bus will pass in the near future", or "the area surrounding the positions where the bus passed before breaking". Stochastic and probabilistic aspects are also of high interest in collective adaptive systems, and are inherently related to time. The study of such aspects is also planned in the near future.

Acknowledgements: This work is supported by the EU project *QUANTICOL: A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours*, 600708. The authors thank Bill Johnston of Lothian Buses and Stuart Lowrie of the City of Edinburgh council for providing access to the data related to bus positions.

References

[1] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[2] V. Ciancia, D. Latella, M. Loreti, and M. Massink, "Specifying and Verifying Properties of Space - Extendend version," QUANTICOL, Tech. Rep. TR-QC-06-2014 also available as arXiv:1406.6393, 2014.

[3] ——, "Specifying and Verifying Properties of Space," in *IFIP Theoretical Computer Science 2014 Track B*, Springer, Ed., 2014, to Appear.

[4] J. van Benthem and G. Bezhanishvili, "Modal logics of space," in *Handbook of Spatial Logics*, 2007, pp. 217–298.

[5] Z. Yan, "Traj-ARIMA: A spatial-time series model for network-constrained trajectory," in *Proceedings of the Second International Workshop on Computational Transportation Science*, ser. IWCTS '10. New York, NY, USA: ACM, 2010, pp. 11–16.

[6] J. Letchner, J. Krumm, and E. Horvitz, "Trip Router with Individualized Preferences (TRIP): Incorporating personalization into route planning," in *Proceedings of the 18th Conference on Innovative Applications of Artificial Intelligence - Volume 2*, ser. IAAI'06. AAAI Press, 2006, pp. 1795–1800.

[7] A. Fathi and J. Krumm, "Detecting road intersections from GPS traces," in *Geographic Information Science*, ser. Lecture Notes in Computer Science, S. Fabrikant, T. Reichenbacher, M. van Kreveld, and C. Schlieder, Eds. Springer Berlin Heidelberg, 2010, vol. 6292, pp. 56–69.

[8] C. Ordoñez, J. Martínez, J. Rodríguez-Pérez, and A. Reyes, "Detection of outliers in GPS measurements by using functional-data analysis," *Journal of Surveying Engineering*, vol. 137, no. 4, pp. 150–155, 2011.

[9] L. Caires, "Behavioral and spatial observations in a logic for the π-calculus," in *Proceedings of the 7th International Conference on Foundations of Software Science and Computation Structures (FOSSACS'04)*, ser. LNCS, I. Walukiewicz, Ed., vol. 2987. Springer, 2004, pp. 72–87.

[10] L. Cardelli and A. D. Gordon, "Anytime, anywhere: Modal logics for mobile ambients," in *Proceedings of the 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, 2000, pp. 365–377.

[11] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink, "Model checking mobile stochastic logic," *Theor. Comput. Sci.*, vol. 382, no. 1, pp. 42–70, 2007.

[12] A. D. Bimbo, E. Vicario, and D. Zingoni, "Symbolic description and visual querying of image sequences using spatio-temporal logic," *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 4, pp. 609–622, 1995.

[13] R. Grosu, S. A. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci, "Learning and detecting emergent behavior in networks of cardiac myocytes," *Commun. ACM*, vol. 52, no. 3, pp. 97–105, 2009.