TR-QC-02-2016

On Spatio-temporal model-checking of vehicular movement in public transport systems

Preliminary Version

Revision: 0.0; Feb. 19, 2016

Author(s): Vincenzo Ciancia (CNR), Stephen Gilmore (UEDIN), Gianluca Grilletti (Scuola Normale Superiore), Diego Latella (CNR), Michele Loreti (UDF-IMT) and Mieke Massink (CNR)

Publication date: Feb. 19, 2016

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: `Jane.Hillston@ed.ac.uk`
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK
7	Institut National de Recherche en Informatique et en Automatique	INRIA	France

Abstract

In this paper we present the use of a novel spatio-temporal model-checker to detect problems in the data and operation of an adaptive system. We categorise received data as being plausible, implausible, possible or problematic. Data correctness is essential to ensure behavioural correctness in systems which adapt in response to data and our categorisation suggests the degree of caution which should be used in acting in response to this received data. We illustrate the theory with several concrete examples, addressing both the detection of errors in vehicle location data for buses in the city of Edinburgh and the behavioural phenomenon of "clumping", the undesired reduction of separation between subsequent buses serving the same route. Vehicle location data is visualised symbolically on a street map, and categories of problems identified by the spatial part of the model-checker are rendered by repainting the symbols for vehicles in different colours. Behavioural correctness makes use of both the spatial and temporal aspects of the model-checker to determine from a series of observations of vehicle locations whether the system is failing to meet the expected quality of service demanded by system regulators.

1 Introduction

Users, operators and regulators of managed services want to have systems which behave correctly across a wide range of conditions. Behavioural correctness is monitored by regulators and it is the responsibility of system operators to try to ensure that system operation lies within allowable bounds as often as possible in practice. A typical example of behavioural correctness is to guarantee a constant and sufficient amount of separation between subsequent buses in so-called "frequent" bus services that do not follow a fixed time table. A necessary precursor for achieving behavioural correctness in adaptive systems is *data correctness*. 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 might not be very problematic because it can be intelligently checked, or even discarded, before any action is taken as a consequence. 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. The task of achieving behavioural correctness comes after the task of achieving data correctness. In this paper we address examples of both, exploiting novel purely spatial and combined spatio-temporal model-checking techniques.

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. AVL data allows operators to produce quantitative measures of service quality.

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. Humans are both outside the system (in the role of passengers, they are users of the system), and also inside the system (in the role of drivers, they are crucial for carrying out the function of a transport system). 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.

In this paper, we consider both purely spatial aspects, namely, data correctness, and spatio-temporal issues, related to the behaviour of buses in the network. The spatial verification problem we consider is 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.

We show how to identify such problems in the data by the means of a recently developed *spatial* model-checker [9]. Using a formal language (namely, the logic SLCS, featuring boolean and *spatial* operators), all requirements on data can be mathematically formalised, and the model-checker can verify data correctness in a fully automated way. The use of logic formulas makes the approach declarative, in that subtle aspects of the analysis can be changed by editing the (very short) formulas, without the need to modify analysis algorithms. Depending on the intended result, 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).

Taking different sections from the data allows us to address different spatial model-checking problems. We could choose to project an instantaneous snapshot of the current location of all of the buses in the fleet (a "satellite view", allowing us to see all buses at one time). Alternatively we could choose to project 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 satellite 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 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 paper we will use single satellite views when we are investigating data correctness ("Is this bus really in a pond?") and sequences of successive satellite views when we are investigating behavioural correctness ("Is this bus catching up with the one in front?").

Besides the purely spatial aspects, we also experimented with spatio-temporal model-checking, in the light of detecting behavioural problems of the bus transport network. For the purpose, we developed a novel model-checker, extending the spatial model checker of [9] with temporal features. The temporal fragment is developed as a variant of the well-known *Computation Tree Logic* (CTL). To exemplify the descriptive capabilities of such spatio-temporal logics, we studied the problem of checking the requirement of avoiding *clumping* on a bus route. On bus routes with frequent service, the most relevant metric may not be adherence to a pre-defined bus time table, but rather maintaining a certain distance between different buses passing by each stop. This is needed to maintain a good distribution of passengers across buses. Such a specification of the headway requirement is rather ambiguous, and may be interpreted in different ways. We discuss how such different interpretations correspond to different spatio-temporal logic formulas, that can be machine-checked on spatial-behavioural models of the bus network.

Besides detecting the presence of clumping in the system log, consisting of the GPS traces of buses, one can also check for clumping in *branching* models of execution, that provide non-deterministic choice points where several different actions may be taken. Such branching models can be, e.g., derived from execution strategies aimed at reducing a specific problem (such as clumping). Combining the analysis of traces and branching models, it is possible to check whether a specific strategy would have worked well in the cases where the problem happened. We exemplify this idea by augmenting an existing GPS trace, featuring clumping, with choice points implementing a simple correction strategy that permits a given bus to wait at a bus stop for a short amount of time, in order to avoid clumping. Since the action of waiting may or may not be done, it generates choice points, giving rise to a branching model. In the branching model, one of the possible traces is the one that happened (that is, wait actions are never taken), but there are also other traces, where wait actions have been executed. The model checker is then used to verify that, indeed, there are correct traces, where clumping does not happen, in the branching models, proving that the correction strategy would have mitigated the problem in its known instances.

This paper is organised as follows. In Sect. 2 we address pure spatial model-checking, without the temporal dimension, providing an informal introduction and examples. Section 3 addresses data correctness in the context of AVL data for buses. Section 4 presents the various categories of data issues and how they are visualised on a street map. Section 5 illustrates how these data issues can be identified exploiting spatial model-checking on a portion of a city map. In Sect. 6 we extend spatial model checking with a temporal dimension introducing a spatio-temporal logic. Section 7 addresses some bus operational issues and behavioural correctness concerning the problem of "clumping", both in time and in space. Section 8 illustrates the use of spatio-temporal model-checking to detect situations of clumping, whereas in Sect. 9 we analyse the effect of some simple correction strategies to alleviate clumping. We conclude the paper by an overview of related work in Sect. 10 and conclusions and outlook on future work in Sect. 11.

Note for the reviewers The work we present enhances and extends the results of [11]. The spatio-temporal model-checker and the applications presented in Sects. 6-9 are new in this paper.

2 Spatial model checking

In its original conception, model checking [2] is a technique that was 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 [9, 10] by some of the co-authors of this paper, a *spatial* model-checking approach has been developed. This technique builds on *spatial logics*, that is, topological interpretations of modal logics [21], dating back to Tarski. Of particular interest to us is a spatial *until*-operator, inspired by the temporal until-operator, that first appeared in [1]. In [9], the operator has been studied in the setting of closure spaces (see below), together with a model-checking algorithm. The spatial variant of the connective 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, we consider specific graphs, namely digital maps, seen as regular grids, where the nodes are the points in the map and where edges connect each node to the adjacent nodes in the north, south, east and west direction.

Moreover, in [9], 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. In this paper, we use the spatial model-checker for some very pragmatic purposes that are concerned with the automatic identification of potential errors in 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*, *disjunction* and *conjunction*), the *closure* operator and the *spatial until* operator. The closure operator has its origin in topological spatial logics. More precisely, following [9], models of the logic are *closure spaces*. A closure space is a pair $(X, \mathcal{C})$, consisting of a set of points X and a function $\mathcal{C}$, from 2^X to 2^X such that, for all sets $A, B \subseteq X$ we have: $\mathcal{C}(\emptyset) = \emptyset$, $A \subseteq \mathcal{C}(A)$ and $\mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B)$. Note that it is not required that the closure is idempotent; more precisely, the class of closure spaces whose closure operator is idempotent coincides with topological spaces (the Kuratowski definition of topological spaces is based on closure). The latter are a subset of closure spaces. In the logic, the closure operator is denoted by $\Diamond$. 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. The spatial until operator $\mathcal{U}$ is a spatial version of the temporal until operator. A point x in space satisfies the formula $\phi\mathcal{U}\psi$ 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 Fig. 1. Finally, we assume a set of basic propositions, which in our specific case identify the colour of a pixel in the digital map.

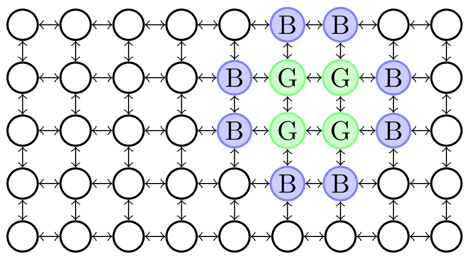


Figure 1: The green nodes satisfy green until blue ($G\mathcal{U}B$)

We will use a few derived operators in the properties shown in the next section. The first derived operator is the reachability operator $\phi\,\mathcal{R}\psi$. It is defined in terms of the spatial until operator as follows $\phi\,\mathcal{R}\psi \triangleq \neg((\neg\psi)\,\mathcal{U}(\neg\phi))$ and, abstracting 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 uses a double negation because the spatial until-formula is weak in the sense that $\phi\mathcal{U}\psi$ also holds when all points in the space satisfy ϕ and none of them satisfy ψ. Further examples of derived operators can be found in [10].

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) [12, 2]. 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 $\phi\mathcal{U}\psi$. 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 $\mathcal{O}(k.(|X| + |R|))$, where k is the number of operators

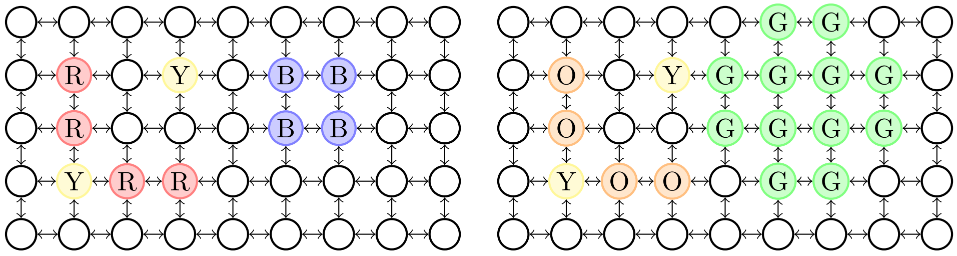


Figure 2: The green (G) nodes (on the right) satisfy the closure of the blue (B) ones (on the left), and the orange (O) nodes (on the right) satisfy the reachability formula $R \wedge ((R \vee Y)\, \mathcal{R}\, Y)$ applied on the left figure, reaching yellow (Y) nodes only from and via red (R) ones

in the formula, $|X|$ the number of nodes and $|R|$ the number of edges in the graph [9].

3 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.

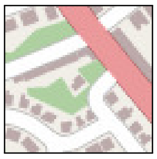
GPS-based AVL systems cannot function accurately without frequent location reports. When GPS signals are unavailable for an extended period of time Dead Reckoning Navigation (DRN) units come into play by computing changes in direction by measuring speed and direction, monitoring the speed and direction 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

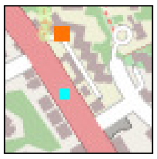
Edinburgh) so we seek a scalable approach to detecting GPS errors which can be automated. We are using a novel spatial model-checker to implement this scalable approach.

4 Categories of data issues

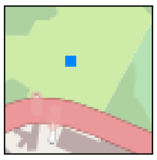
In this section we introduce the features of the map representation which we use and see the effect of the spatial properties which we evaluate. Figure 3 explains our conventions for representing buses, bus stops and the progress in time through observations on the map.



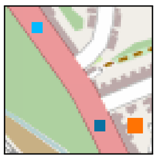
We use an OpenStreetMap representation for the map from which the names of streets have been removed for clarity. Main streets are pink, side streets are white.



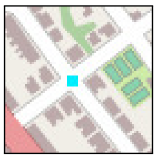
Buses are represented by blue squares on the map in a range of shades of blue. Bus stops are represented by orange squares.



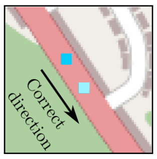
Errors in GPS data show up as bus observations which are not on any road.



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.



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



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

Figure 3: Representations of buses, roads and bus stops on maps

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.

5 Identifying data issues using model-checking

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 Sect. 4. 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 Fig. 4. The red circles, as well as the yellow balloon with text, have been superimposed on the images and are not part of input or output of the tool (this is done for readability, also in Sects. 8-9).

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 $\neg$, $\wedge$, $\vee$, $\mathcal{C}$, $\mathcal{U}$, are denoted by the characters `!`, `&`, `|`, `C`, and `U`, respectively.

Spatial features of data points in Fig. 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

[1]In this paper, 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.

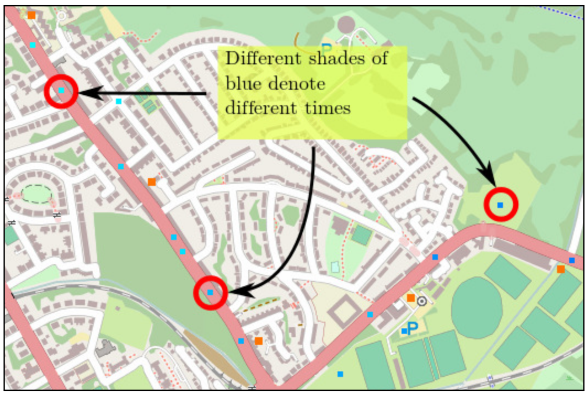


Figure 4: Input model. Blue squares are bus positions; their order in time is described by the increasing darkness. Orange squares are the bus stops.

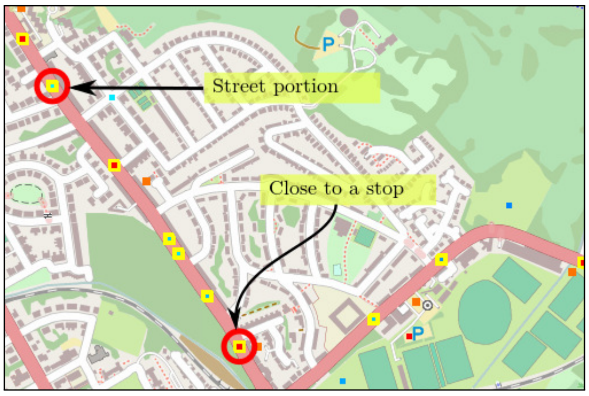


Figure 5: Positions close to a stop are repainted in red, the areas of the road surrounding all bus positions are repainted in yellow.

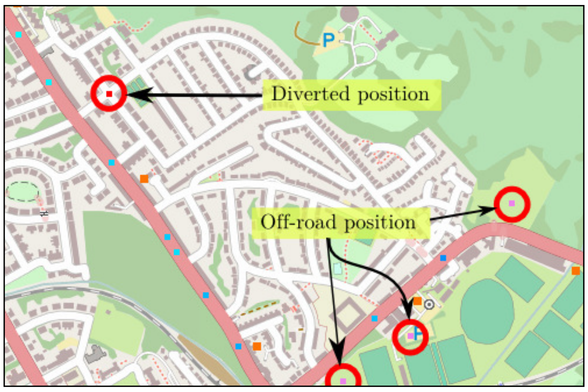


Figure 6: Diverted positions (neither off road, nor on a main street) are repainted in red, off-road positions are repainted in violet.

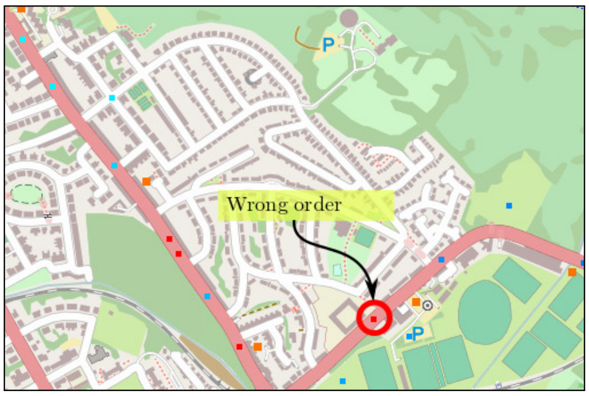


Figure 7: Positions that are found to be out of order (i.e., not between the previous and next position) are repainted in red.

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.

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 Fig. 6.

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

Figure 8: Definition of `consecutivePos`

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 Fig. 6.

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 Fig. 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 Fig. 8. The definition uses the *reachability* predicate $a\ \mathcal{R}\ b$, written as `reach(a,b)`. The definition of `consecutivePos` uses the previously defined predicate `streetPortion` to identify 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.

6 Spatio-temporal model checking

Spatio-temporal logics may be defined by permitting mutually recursive nesting of spatial and temporal operators. Several combinations are possible, depending on the chosen spatial and temporal fragments, and the permitted combinations of the two. A great deal of possibilities are explored in [17], for spatial logics based on topological spaces. We investigated one such structure in this work, in the setting of closure spaces. We implemented[2] a prototype spatio-temporal model-checker, that enhances the spatial variant with temporal operators, in the spirit of the branching temporal logic CTL.

[2]OCaml source code for the prototype is available at `https://github.com/cherosene/ctl_logic`.

In this section we provide a lightweight informal introduction to spatio-temporal model checking; we refer the interested reader to [8] for further technical details. In particular, the spatio-temporal logic that we use is a combination of SLCS (see Sect. 2) and the well-known temporal logic CTL [12]. Like SLCS, this spatio-temporal logic is developed in the setting of closure spaces. In addition to the already discussed spatial operators of SLCS, the augmented logic STLCS (Spatio-Temporal Logic for Closure Spaces) features the CTL path quantifiers `A` ("for all paths"), and `E` ("there exists a path"). As in CTL, such quantifiers must necessarily be followed by one of the path-specific temporal operators $\mathtt{X}\Phi$ ("next"), $\mathtt{F}\Phi$ ("eventually"), $\mathtt{G}\Phi$ ("globally"), $\Phi_1 \mathtt{U} \Phi_2$ ("until"), but unlike CTL, in this case Φ, Φ_1 and Φ_2 are STLCS formulas. For a further introduction to and more details on CTL and CTL model checking, the reader may consult [2]. In the sequel we will refer to the spatial until operator by `S` (from "surrounded by"), and use `U` for the temporal until operator.

A model $\mathcal{M}$ of the STLCS logic is composed of a Kripke structure $(S, \mathcal{T})$, where S is a non-empty set of *states*, and $\mathcal{T}$ is a non-empty *accessibility relation* on states, and a closure space $(X, \mathcal{C})$, where X is a set of points and $\mathcal{C}$ the closure operator (see Sect. 2). Every state s has an associated valuation $\mathcal{V}_s$, making $((X, \mathcal{C}), \mathcal{V}_s)$ a *closure model* according to Definition 6 of [9]. An equivalent interpretation is that the valuation function has type $S \times X \to 2^P$, where P is the set of atomic propositions, thus, the valuation of atomic propositions depends both on states and points of the space. The evaluation contexts are of the form $\mathcal{M}, s, x \models \Phi$, where Φ is a STLCS formula, s is a state of a Kripke structure, and x is a point in space X. In both notations, the intuition is that there is a set of possible worlds, i.e. the states in S, and a spatial structure represented by a closure space. At each possible world there is a different valuation of atomic propositions, inducing a different "snapshot" of the spatial situation which "evolves" over time. This is made clear along a temporal path. A path in the Kripke structure denotes a sequence of digital pictures indexed by instants of time; see Fig. 9 for a pictorial illustration.

Let us proceed with a few examples. Consider the STLCS formula `EG (green S blue)`. This formula is satisfied in a point x in the graph, associated to the initial state s_0, if there exists a (possible) evolution of the system, starting from s_0, in which point x is always, i.e. in every state in the path, green and surrounded by blue ($green \mathcal{U} blue$ in the terminology of Sect. 2). Note that the model-checker will return (or colour) *all* the points x that satisfy the formula. A further, nested, example is the STLCS formula `EF (green S (AX blue))`. This formula is satisfied in a point x in the graph associated to the initial state s_0, if there is a (possible) evolution of the system, starting from s_0, in which point x is eventually green and surrounded by points y that, for every possible evolution of the system from then on, will be blue in the next step.

Spatio-temporal model checking is performed using a variant of the classical CTL labelling algorithm [12, 2], augmented with the algorithm in [9] for the spatial fragment. In the implementation, models are represented by a Kripke structure, that is, a graph, which uses the plain text graph description language `dot` for graph representation[3], and a folder of images, one for each state in the Kripke structure. Images must have the same size, and the corresponding grid is taken as the reference closure space $(X, \mathcal{C})$ (grids are instances of closure spaces, see [9] for further details). Colours of the points of the picture for each state s determine the valuation function $\mathcal{V}_s$.

7 Bus operational issues

Public transport services are managed services, subject to published regulations governing the safety, punctuality and reliability of the service. Adherence to the published timetable is an important punctuality metric for timetabled bus services. So-called "frequent" services – those where a timetable is not published – are susceptible to a phenomenon known as *clumping*. Clumping occurs where one bus catches up with – or at least comes too close to – the bus which is in front of it. In the absence

[3]Further information on the dot notation can, for example, be found at http://www.graphviz.org/Documentation.php.

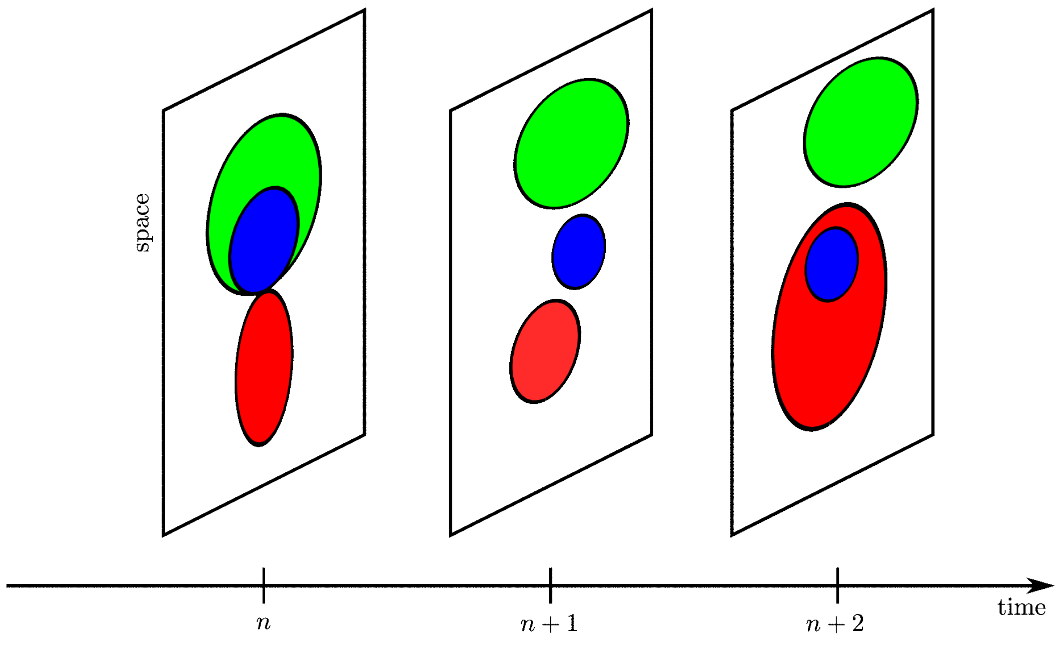


Figure 9: In spatio-temporal logics, a temporal path represents a sequence of *snapshots* that are induced by the time-dependent valuations of the atomic propositions.

of a published timetable for frequent services the important performance metric to consider is not timetable adherence but *headway*, a measure of the separation between subsequent buses.

- If the headway is too large then passengers are forced to wait at the bus stop for longer than they should, causing levels of passenger satisfaction with the service to fall.
- If the headway is too short then this has the consequence that some buses end up carrying too many passengers while others are carrying too few. A subsequent bus following closely behind an earlier one is unlikely to find passengers at a bus stop because a short time ago the passengers who were waiting boarded the earlier bus, and there has not been an opportunity for a queue to build up in the brief time since. Unfortunately this tends to further reduce the headway between buses because the subsequent bus is not delayed by boarding passengers, whereas the earlier bus was.

An instance of a short headway problem is illustrated in Fig. 10 using a series of successive "satellite view" images of the bus data.

We are describing the scenario of a richly-instrumented real-time-informed system where data cleaning has been applied as described above to result in a plausible set of observations of bus positions. In this scenario, the central authority of our adaptive system could intervene when short headway problems are detected between one bus and the subsequent one by sending a message to the second bus saying "WAIT". The bus waits extra time at the next bus stop (say 30 seconds). This has the effect of correcting the system behaviour away from the problematic behaviour by lengthening the headway. This intervention has not necessarily solved the problem, but it has lessened it, and it is always possible to ask a bus to wait again later in its lifetime if necessary.

Crucially, despite appearances to the contrary, such an arrangement is *not* a centrally-controlled system and does not operate as such primarily because the central server which manages the communication to and from buses has only partial information about the local traffic conditions in the immediate area around each bus. It is necessarily the case that the centre has only partial information

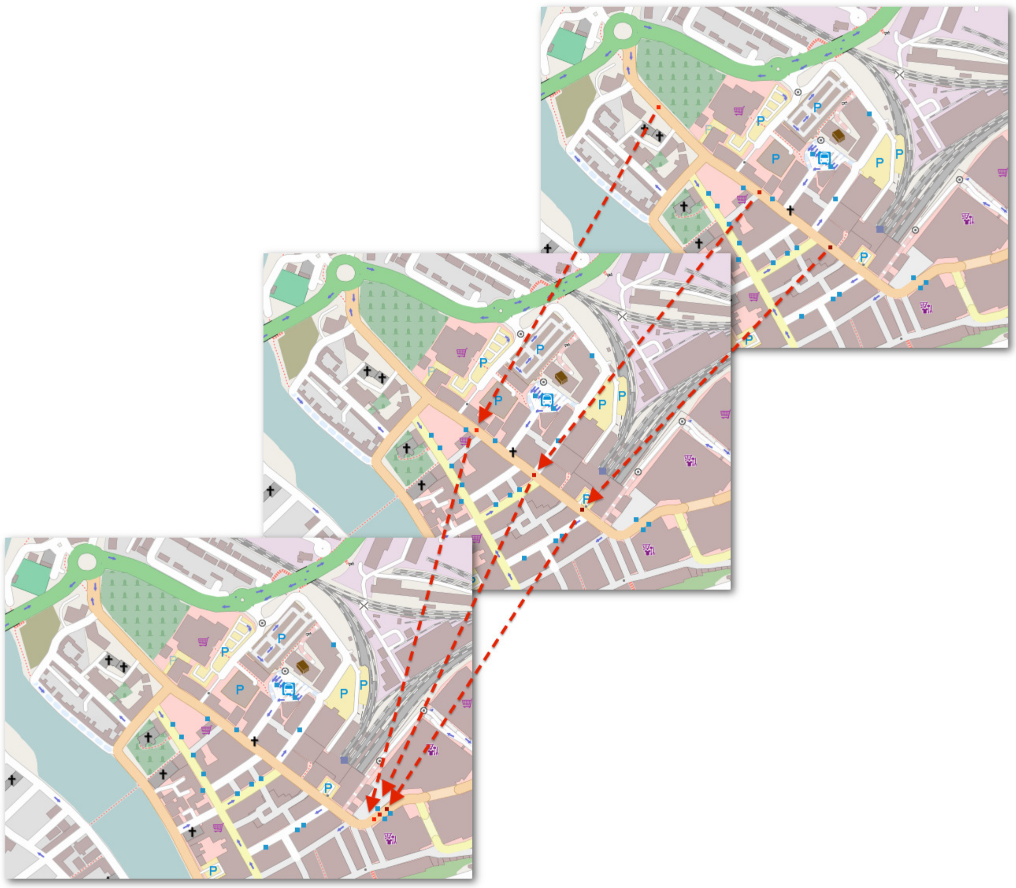


Figure 10: Because of delays caused by boarding passengers the headway between buses is successively eroded over time until the buses are essentially working as a conglomerate.

because the overhead of keeping the centre up-to-date with every aspect of the local traffic conditions would be completely unworkable. Because it is understood by all that the centre has only partial information, messages issued from the centre are *requests*, not *commands*. The system as a whole is a collective adaptive system with distributed control because the driver of the bus may choose not to honour the request because of local specific knowledge about the vehicular conditions of the road at this point. For example, drivers might not honour a request to wait if they can see that other bus operators have vehicles immediately behind this bus and are waiting to access this bus stop; or they might not honour a request because waiting here would delay emergency service vehicles; or many other reasons. The central communications server of the bus company does not have this information.

Interventions – such as requesting a bus to wait – must be guided by *policies*. The central authority has a goal in mind for the system, and will try to direct the system behaviour closer to the desired goal while respecting the policy which is being followed. Several factors must be considered in shaping a useful and practically viable policy.

- Policies should be designed to improve the service's score on a pre-defined metric or service-level agreement specified by service regulators. Such service-level agreements typically require that a high percentage of service instances are satisfactory according to some agreed definition of satisfactory, but a small percentage may be unsatisfactory. Policies decide when and where to make best use of this flexibility.

- Frequency-dependence is an important issue within a vehicular management policy. A carefully considered policy would limit the number of "WAIT" requests which can be sent at a particular time to buses serving a particular route. For example, if all buses serving a route were sent a "WAIT" instruction and all obeyed this, then this would not impact on the short headway problem at all. For this reason, "WAIT" requests should be rare: a reasonable policy could be to send a "WAIT" request to at most one bus on a route at any time.

- Location-dependence is also an important dimension for policies. It may be impractical or significantly disruptive to traffic flow for buses to wait in some areas of the city such as busy shopping streets or main arteries for traffic flow. Thus, even though a bus *should* be requested to wait because a short headway problem has been detected, the bus will not be requested to wait in these problematic areas of the city. Such requests could always be ignored by drivers but it would be better not to issue them in the first place.

- Time-of-day-dependence is also an important factor in transport policies. The policy could refrain from sending any "WAIT" requests during the rush hour periods.

We have discussed interventions to reduce and eventually eliminate short headway problems in the absence of a precise metric for measuring headway. Perhaps the most obvious measure of headway is *chainage*, the distance by road between two vehicles. However, because buses change speed on different parts of the route, this is not the best metric to use. We wish to address time-related problems with the service (passengers waiting for too long a time between buses; and there not being enough time between buses for passengers to arrive at a bus stop). For this reason, we should rather focus on the temporal separation between buses. This amounts to considering "How long ago was the previous bus at this location?" in addition to "How far behind the previous bus are we?" These two different properties illustrate that spatio-temporal properties may have rather subtle aspects to consider, and may require careful formalization. In Sect. 8 we shall see how to make such formalisation mathematically precise, so that it can be machine-checked by model checking. This methodology achieves a declarative approach to verification of spatio-temporal properties, having the advantage, over the ad-hoc implementation of analysis algorithms, that such slight changes in the interpretation of given requirements do not impact the implementation of the verification tools, as only the logical formalisation of requirements needs to be changed.

```
let bus1 = <RED [155,155]> & <GREEN [0,0]> & <BLUE [0,0]>;
let bus2 = <RED [188,188]> & <GREEN [0,0]> & <BLUE [0,0]>;
let bus3 = <RED [221,221]> & <GREEN [0,0]> & <BLUE [0,0]>;
let bus = bus1 | bus2 | bus3;
let busStop = <RED [55,55]> & <GREEN [55,55]> & <BLUE [255,255]>;

let close(x) = C^7 x;

let busAtStop(x) = busStop & close(x);

let busAfterBus1 = busAtStop(bus1) &
     EX busAtStop(bus2 | bus3);
let busAfterBus2 = busAtStop(bus2) &
     EX busAtStop(bus1 | bus3);
let busAfterBus3 = busAtStop(bus3) &
     EX busAtStop(bus1 | bus2);

let closeToOtherBus(x) = (x & close(bus & !x));
let conglomerate = busStop & close(closeToOtherBus(bus1)
                   | closeToOtherBus(bus2) | closeToOtherBus(bus3));

let timeConglomerate = (busAfterBus1 | busAfterBus2 | busAfterBus3);
```

Figure 11: Spatio-temporal formulas for conglomerates

8 Identifying operational issues using model checking

Using spatio-temporal logic, clumping of buses can be detected, both in a system trace (e.g., a GPS trace obtained at run-time), and in more complex *branching* models, that are used in Sect. 9. Consider a single bus route, served by k buses. At each instant of time, the state of the system is completely described by a tuple of k GPS positions; therefore, a system trace is a finite sequence of such tuples. As already discussed, there may be several different ways to formalise the notion of clumping. We describe two possible variants. Figure 11 contains the input code of a model-checking session using three buses. Formulas `bus1`, `bus2`, `bus3`, and `busStop`, are colour ranges, that serve the purpose to identify bus positions on the map. In this example, colours are used to separate the different buses serving the same route, so that each bus has a specific colour through time, whereas at the same time, two buses of the same route are coloured differently. Similarly, formula `busStop` identifies the position of a bus stop. The formulas that we explain below are true at points of a bus stop whenever clumping is happening (formation of a conglomerate) at that particular stop.

1. A *spatial* conglomerate happens when two buses serving the same route, at some point in time, are spatially close to each other, and also close to a bus stop. This event is described by the formula `conglomerate`. Points satisfying this formula are those that are close to a bus, which is in turn close to another bus. Formula `closeToOtherBus`, which is parametrised by the chosen bus, is responsible for checking that a bus is close to another one. The notion of "closeness" is defined by formula `close`, using a nested application of the basic closure operator of the logic.

2. A *spatio-temporal* conglomerate happens when two buses serving the same route pass by the same stop in a short amount of time. This case is subtler than the previous one, as it does not necessary imply that the headway between two buses becomes too small. This event is described by the formula `timeConglomerate`, which features a combination of spatial operators (used to

detect that a bus is close to a stop) and temporal operators (used to identify the spatio-temporal conglomerate). For instance, consider the formula `busAfterBus1`. This formula is true on points that are: i) part of a bus stop, and close to `bus1`, because `busAtStop` must be true for `bus1`; ii) such that, in one[4] time step, these will be part of a bus stop, and close to either `bus2` or `bus3`. Note that the use of spatial and temporal connectives in the same formula permits one to refer to the colour of points at a specific time, and at subsequent time instants.

Once established what is the kind of clumping one is interested in, one may use temporal operators to detect points where, e.g., clumping will happen at some point in the future. Figure 12 is obtained with our tool set, starting from the positions of three buses serving the same route. Figures 12a-12e are obtained by mapping bus coordinates over a base map. Buses are represented by squares of different shades of red. The dark blue square is a bus stop. Figure 12f shows the output of the model checker when checking the formula `EF timeConglomerate`. Indeed, it is the same as Fig. 12a, except for the colour of the bus stop, whose points are coloured green, as clumping happens at that stop, at some point in the future. As the model-checker is a simple prototype, with no optimisations in place, we do not provide accurate performance information. We just remark that execution time is in the order of ten seconds on a standard laptop for this example, in which over one million points (approximate size of the image) are examined several times (proportional to the number of sub-formulas of the formula to be verified).

9 Analysing the effect of correction strategies for operational issues

In this section, we study a method to analyse the effect of correction strategies for spatio-temporal issues on the bus network. In particular, we incorporate existing data (e.g., system logs) in estimating the impact of introducing new policies in a system. First, we note that our model checker can be used both to detect clumping in a system trace, as we have seen, or to analyse a *branching* model, that is, a system where at each state, non-deterministically, there may be several possible steps to different future states. Such non-deterministic models represent in a concise way a great number of possible system behaviours, depending on the choices that may be made at each execution step. We use this fact in conjunction with the idea, discussed in Sect. 7, to send to buses "WAIT" instructions in order to reduce clumping. In our case, it is relevant that the possibility of issuing "WAIT" instructions to specific buses, or not doing so, introduces non-deterministic choice points.

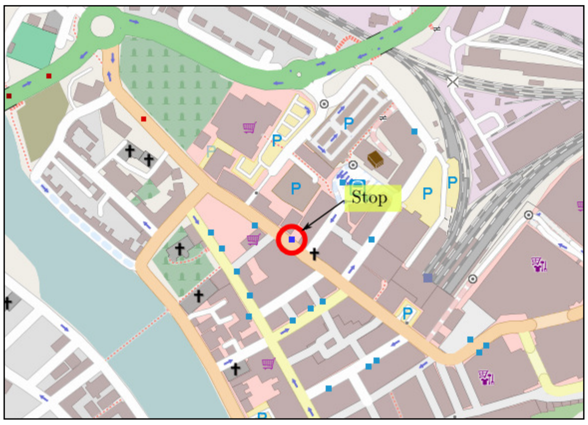
In our test, we consider as an input the (linear) traces of AVL data (e.g., provided by the bus company[5]), as discussed in Sect. 8. Whenever in some states of the given system trace there is clumping, the crucial point is then to turn such trace into a more complex, branching model, where choice points denote the usage of "WAIT" instructions. Then, the branching model is analysed using the same formulas again, in order to verify that the problem has been mitigated, by the fact that some traces exist in the model that avoid clumping.

Consider a system trace, that is, a sequence of tuples of k elements, where k is the number of buses serving a route. The length of the sequence is the number of samples. Element at position i in each tuple is the position of bus i. At each step, besides the already existing transition to the next step, more transitions are added, to new states, where one or more buses wait, (therefore, their position does not change), and the other ones move as they actually did in the system trace.

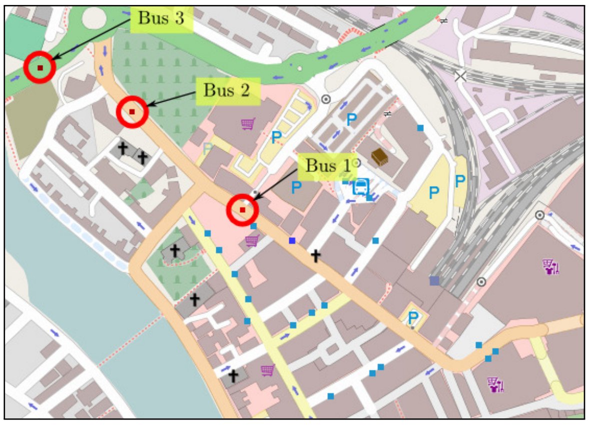
As a proof-of-concept, in order to demonstrate our approach, we implemented such a transformation algorithm, and tested it on the example described in Sect. 8. The implementation is parametric with respect to the maximum number of buses that are allowed to wait simultaneously, and the maximum number of wait instructions issued to the same bus. We remark that in normal situations, it

[4]More than one time step can be required. This can be achieved by repeated nesting of the `EX` operator. We did not do so for the sake of clarity in Fig. 12.

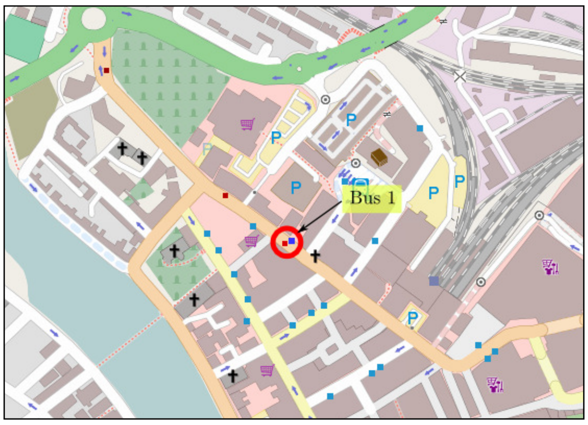
[5]As in Sect. 5, we use artificial data for the sake of simplicity, but usage of our model-checker does not differ on real data.



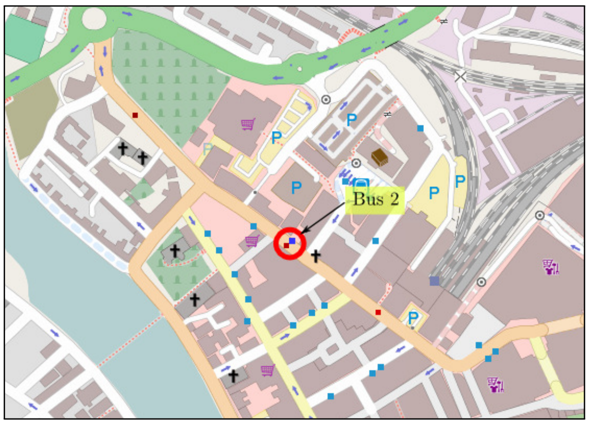
(a) Initial state



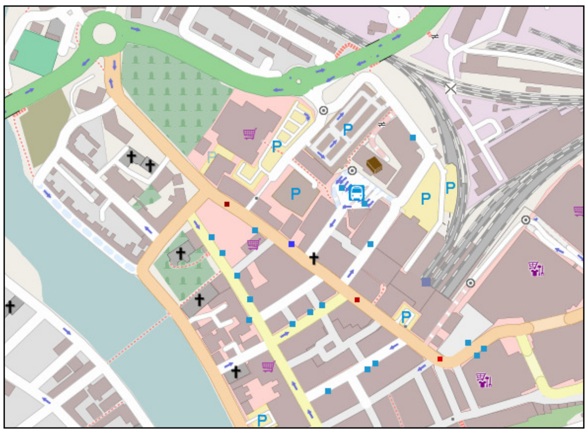
(b) Second state



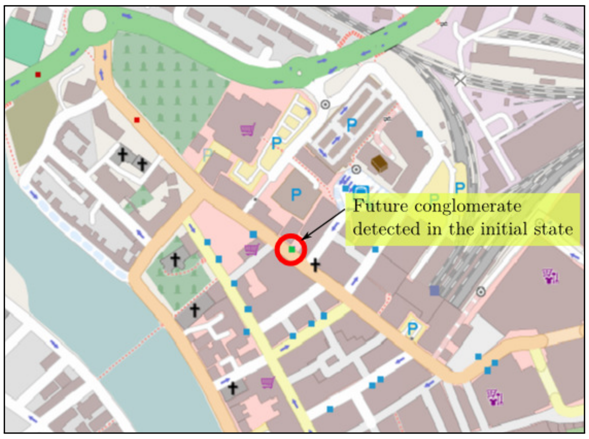
(c) Bus 1 passes by the stop



(d) Bus 2 passes by the stop



(e) Final state



(f) Result from the model-checker. Points of the initial state that will be involved in a conglomerate are coloured

Figure 12: Spatio-temporal conglomerate.

does not make much sense to let many buses wait simultaneously, as this would result in a general delay of the whole route. Similarly, it is not advisable to issue too many wait instructions to the same bus, as this might result in unacceptable delay for the passengers running on that particular bus.

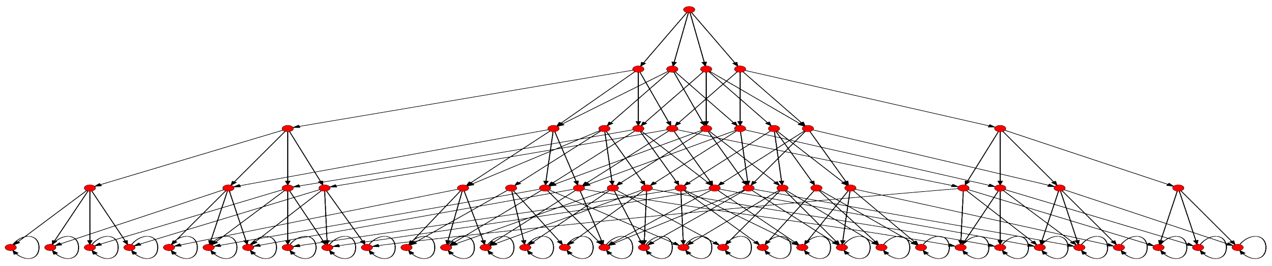


Figure 13: Branching model obtained by augmenting a linear trace.

The input of the algorithm consists of the system trace as described above, and of a map. The tool generates a branching model suitable for STLCS model-checking (see Sect. 6). The state space of the branching model generated from the system presented in Fig. 12 is shown in Figure 13; as typical in CTL model-checking, self-loops have been added to terminal states, since CTL-path formulas express properties of infinite paths, i.e. infinite sequences of states. In Fig. 14 we show one of the several traces of the system, in particular one where the conglomerate does not happen. To see that such a trace exists, the model checker is run for the formula `AF timeConglomerate`, which is true only if all system paths encounter a conglomerate. The result of the model-checker is that no point is coloured, signalling that there are “good” paths for each point in space. Model-checking over the branching model has an execution time of around five minutes on the same machine, where the analysis of the linear trace takes around ten seconds. Even though optimization of the algorithm and performance evaluation are left for future research, this is already a hint about the feasibility of spatio-temporal model-checking in realistic use cases.

10 Related work

In [22] 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 [18] 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 [14] 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 [19] 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.

The problems of headway computation are considered in [20], where the authors use Monte Carlo simulation and time series analysis to evaluate a family of interpretations of an ambiguous regulation governing headway for frequent bus services.

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. [5]) and mobile ambients with the related ambient logic (e.g. [7]). 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. Works such as [16, 6] introduce notions of physical space in the context of process calculi; it is an interesting future work to connect this research line to spatio-temporal model checking.. Furthermore, in a stochastic setting, the Mobile Stochastic Logic (MoSL) [13] 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. Also logics for reasoning on *signals* have been enhanced with spatial aspects [4].

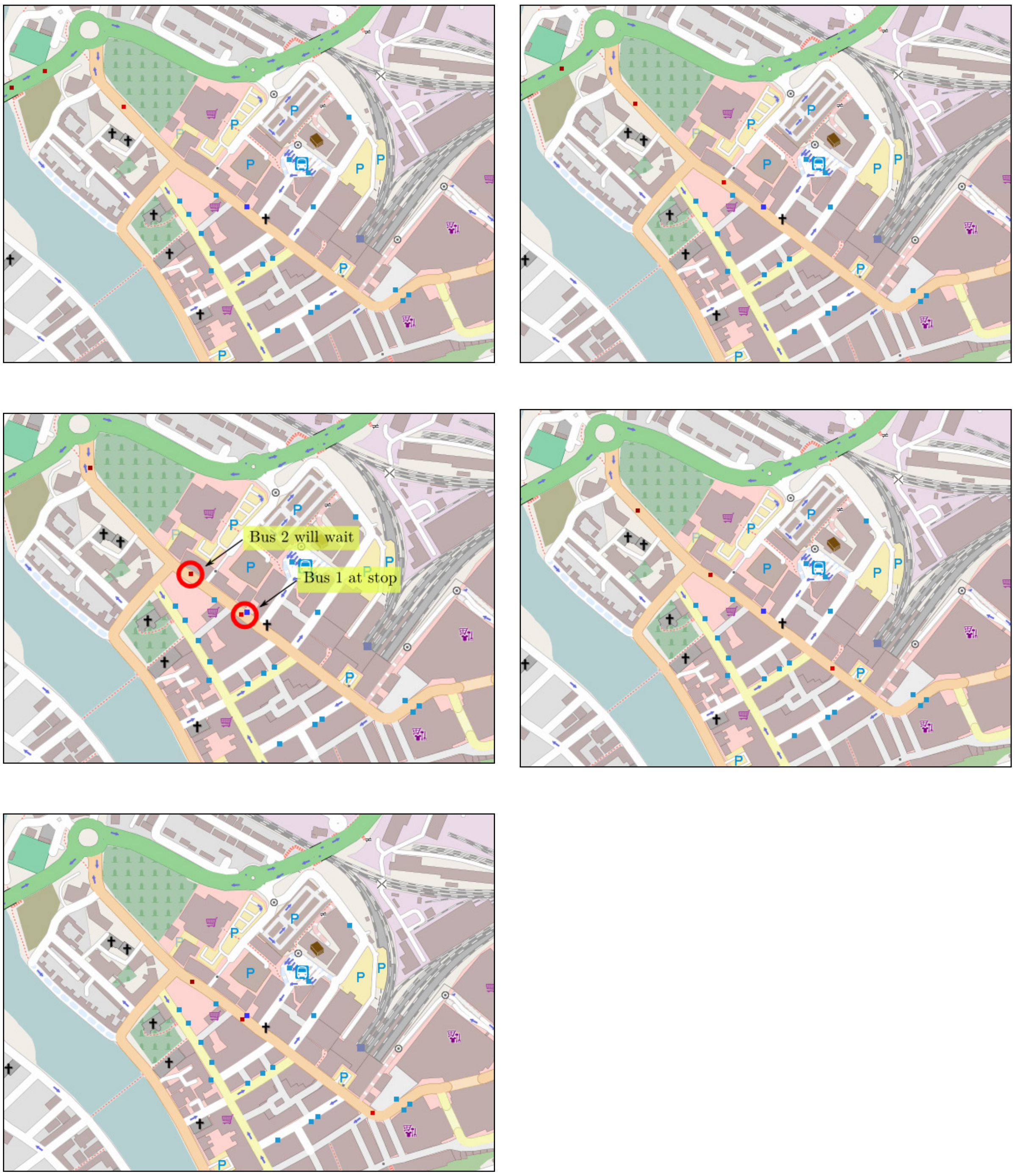


Figure 14: A trace that does not contain conglomerates, since the second bus waits, when the first bus arrives at the bus stop.

Other variants of spatial logics concern the symbolic representation of the contents of images, and, combined with temporal logics, for sequences of images [3]. 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 [15] 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.

11 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. By enhancing the logic and the model checker with a temporal perspective, the interplay of space and time has allowed us to define complex spatio-temporal formulas, predicating over the relation between points of a coloured image that evolves over traces or branching models.

Current work is focused on defining *collective* variants of spatial and spatio-temporal properties; that is, the satisfaction value of a formula is defined on a set of points, rather than on a single point, so that the satisfaction value of a formula with respect to a set of points (a collective property) is not necessarily determined by the satisfaction values over the points composing the set (an individual property). Such interpretation of spatio-temporal logics is particularly motivated by the setting of collective adaptive systems.

High priority in future work will be given to the investigation of various kinds of optimisations for spatio-temporal model-checking, including partition refinement of models, symbolic methods, and on-line algorithms taking advantage of differential descriptions of the changes between system states.

An orthogonal, but nevertheless interesting, aspect of computation is the introduction of probability and of stochastic features. Such features can also be added to our spatio-temporal logic, and investigating efficient model checking algorithms in this setting is important for practical applications, which are very often quantitative rather than boolean. The applications in the evaluation of correction strategies that we presented would be very much enhanced by forms of quantitative analysis. Such methodologies would be able to address questions like “how likely it is that a given strategy fixes the problem?” or “how frequently does the problem manifest itself, before and after applying a given strategy?”.

References

[1] M. Aiello. *Spatial Reasoning: Theory and Practice.* PhD thesis, ILLC, University of Amsterdam, 2002.

[2] Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.

[3] Alberto Del Bimbo, Enrico Vicario, and Daniele Zingoni. Symbolic description and visual querying of image sequences using spatio-temporal logic. *IEEE Trans. Knowl. Data Eng.*, 7(4):609–622, 1995.

[4] Luca Bortolussi and Laura Nenzi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In *VALUETOOLS*, 2014.

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

[6] Luca Cardelli and Philippa Gardner. Processes in space. *Theoretical Computer Science*, 431(0):40 – 55, 2012. Modelling and Analysis of Biological Systems Based on papers presented at the Workshop on Membrane Computing and Bio-logically Inspired Process Calculi (MeCBIC) held in 2008 (Iasi), 2009 (Bologna) and 2010 (Jena).

[7] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings of the 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, pages 365–377, 2000.

[8] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink. A spatio-temporal model-checker. Technical Report TR-QC-10-2014, QUANTICOL, 2014. `http://milner.inf.ed.ac.uk/wiki/pages/J8N4c8/QUANTICOL_Technical_Reports.html`

[9] V. Ciancia, D. Latella, M. Loreti, and M. Massink. Specifying and Verifying Properties of Space. In Springer, editor, *The 8th IFIP International Conference on Theoretical Computer Science, TCS 2014, Track B*, volume 8705 of *Lecture Notes in Computer Science*, pages 222–235, 2014.

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

[11] Vincenzo Ciancia, Stephen Gilmore, Diego Latella, Michele Loreti, and Mieke Massink. Data verification for collective adaptive systems: spatial model-checking of vehicle location data. In *2nd FoCAS Workshop on Fundamentals of Collective Systems*, IEEE Eight International Conference on Self-Adaptive and Self-Organizing Systems, page to appear. IEEE Computer Society, 2014.

[12] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 53–71. Springer, 1986.

[13] Rocco De Nicola, Joost-Pieter Katoen, Diego Latella, Michele Loreti, and Mieke Massink. Model checking mobile stochastic logic. *Theor. Comput. Sci.*, 382(1):42–70, 2007.

[14] Alireza Fathi and John Krumm. Detecting road intersections from GPS traces. In SaraIrina Fabrikant, Tumasch Reichenbacher, Marc van Kreveld, and Christoph Schlieder, editors, *Geographic Information Science*, volume 6292 of *Lecture Notes in Computer Science*, pages 56–69. Springer Berlin Heidelberg, 2010.

[15] Radu Grosu, Scott A. Smolka, Flavio Corradini, Anita Wasilewska, Emilia Entcheva, and Ezio Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Commun. ACM*, 52(3):97–105, 2009.

[16] Mathias John, Roland Ewald, and Adelinde M. Uhrmacher. A spatial extension to the pi calculus. *Electronic Notes in Theoretical Computer Science*, 194(3):133 – 148, 2008. Proceedings of the First Workshop From Biology To Concurrency and back (FBTC 2007).

[17] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Spatial logic + temporal logic = ? In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 497–564. Springer, 2007.

[18] Julia Letchner, John Krumm, and Eric 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*, IAAI'06, pages 1795–1800. AAAI Press, 2006.

[19] 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*, 137(4):150–155, 2011.

[20] Daniël Reijsbergen and Stephen Gilmore. Formal punctuality analysis of frequent bus services using headway data. In András Horváth and Katinka Wolter, editors, *Computer Performance Engineering - 11th European Workshop, EPEW 2014, Florence, Italy, September 11-12, 2014. Proceedings*, volume 8721 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2014.

[21] Johan van Benthem and Guram Bezhanishvili. Modal logics of space. In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 217–298. Springer, 2007.

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