Formal Analysis of Resilience in Transport Systems with Bigraphs

Susmoy Das, Ricardo Almeida, Blair Archibald, and Michele Sevegnani

School of Computing Science
University of Glasgow, Glasgow, United Kingdom
{susmoy.das,ricardo.almeida,blair.archibald,michele.sevegnani}@glasgow.ac.uk

Abstract. Transport system analysis often relies on simulation, but this makes it tricky to detect rare events and simulation languages are far from the transport domain. We propose modelling transport using a diagrammatic formal method called Bigraphs that allows exact analysis via model checking, and user-defined visual representation of systems. We apply this approach to modelling vehicles moving through a Port and show how it can be used to maximise efficiency and achieve more resilient shipping and logistics.

Keywords: Transport systems, Bigraphs, Maritime Ports, Model Checking, Resilience, Modelling

1 Introduction

Transport infrastructure plays a critical role in modern societies and it is essential we can give robust safety and reliability guarantees. This is particularly important as we approach a transition point: transport infrastructure must undergo radical changes if we want to meet decarbonisation targets (for instance, the transport sector is responsible for 21% of the global CO_2 emissions [27]). Given the costs involved (both financial and carbon), physically changing the infrastructure to experiment with new approaches is not feasible. Instead, transport experts rely on specific transport models [26] to determine what effects changes may have before implementing them in practice. This is no easy task: transport is a complex system involving human preference, economic supply and demand, and physical constraints.

Transport models roughly fit into two categories: macro-scale—that consider whole-system equilibrium—or individual models—that determine system behaviour through the behaviour of a set of individuals using the infrastructure. Given the abundance of computational power, individual models are increasingly popular and are at the core of *simulation-based approaches* found in tools such as MatSIM [3,30], SUMO [4,23], AnyLogic [1], and iTwin [2]. While powerful, simulation approaches struggle with low-probability events, despite the fact these are the events that are likely to have catastrophic impact, and simulation-model

specification is often far from the transport domain, requiring programming expertise (e.g. Java for MatSIM) and textual specification formats (e.g. XML).

We believe formal methods, in particular those based on diagrammatic/visual modelling notations, can alleviate these issues. They support robust quantitative analysis (e.g. probabilistic), via full-system state exploration, that ensures low-probability events are not missed, and the diagrammatic notation can be customised for domain, not programming, experts.

While formal methods are not new within the transport space, they have largely been applied to the verification of safety-critical components. For example, the design of railway signalling [19] and aviation-certification scenarios [13, 18]. We instead apply formal methods to the modelling of transport as a complex system: an area under-explored, but essential as we transition our transport systems.

We show how diagrammatic models, specifically Milner's Bigraphs [5, 25, 29] (Section 2), can be applied to transport by modelling a port-operations scenario. We outline the model (Section 3), and use it to show some relevant properties e.g. What is the probability that a ship leaves the port within t time-units? What is the impact of introducing new road segments on port performance? (Section 4). This is the first step in determining the suitability of these approaches in modelling complex transport systems.

2 Bigraphs

Bigraphs are a universal formal model that describes systems based on both spatial relationships between entities, e.g. a Car in a Road, as well as non-local linking, e.g. A Car connected to a Cell tower at another location sending telemetry data. A key feature is a diagrammatic notation that is useful for non-experts without compromising its formal rigour[1]. A rewriting theory, with *user-specified rules* that replace sub-bigraphs with other sub-bigraphs, models system evolution over time.

Entities are drawn using boxes or any other shape of choice, and spatial relationships are represented through nesting. These relationships are often physical containment, but can also be used for ownership, e.g. a X has a (virtual) Y. We allow *parameterised entities* to represent families of concrete entities (e.g. in Section 3 we name road segments using SName(s), an entity parameterised with the name of the Segment it is nested in).

The dashed unfilled rectangles are *regions* and represent adjacent portions of the system, like entities in a direct-sibling relationship (i.e. they share the same parent) or in completely separate locations. The dashed filled rectangles are *sites* and represent abstraction, that is, an unspecified bigraph (including the empty bigraph) might exist there. Sites are essential for the rewriting theory as they allow matches to be partial.

[1] This notation is **equivalent** to an algebraic representation that allows full compositional analysis of systems. We focus on the diagrammatic notation here.

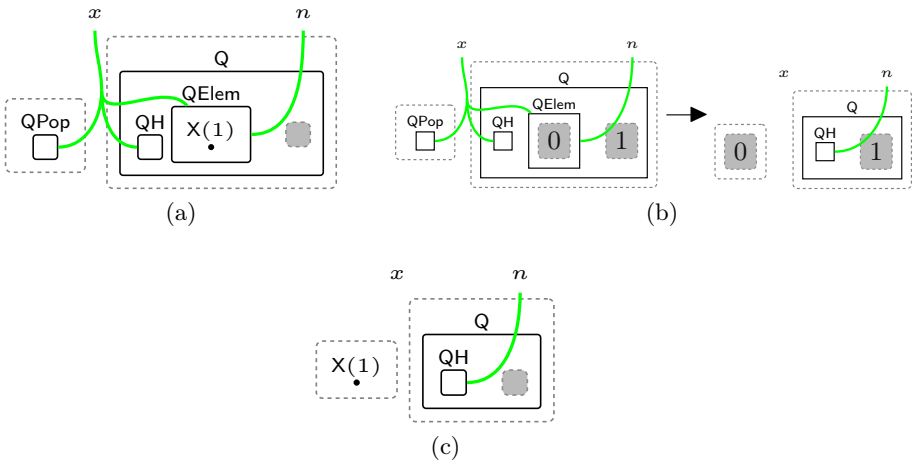

Fig. 1. (a) Bigraph modelling a queue (linked list) and requested pop operation; (b) Rewrite rule: pop an element from the queue; (c) Result of applying (b) to (a).

The green edges represent non-local relationships. In general, these are hyperedges (1–n links) rather than binary edges seen in standard graph models. Edges may be closed (1–0 links) which we show with an edge with a bar at the end. Names, e.g. $x, n, \ldots$, operate similarly to sites in that they represent abstraction: an unspecified set of entities (including no entities) might also be connected to this link.

We model vehicle queues as linked lists. Each queue (represented by entity Q) contains a head and a tail pointers (resp., QH and QT), and zero or more elements QElem where vehicles are nested. Fig. 1a presents a generic bigraph for modelling a queue with a requested pop operation. The actual operation is performed by the rewrite rule in Fig. 1b, which consists of a left-hand side and a right-hand side. To apply a rewrite to a larger bigraph, we find matches of the left-hand side and replace them with the right one. The numbers within the sites represent *where* they should appear in the right hand side. For example, here the contents of site 0 move into the leftmost region. This allows movement, copying, and deletion of site contents as required. The result of applying the rewrite of Fig. 1b to Fig. 1a is in Fig. 1c.

Extensions to bigraphs allow rewrite rules to be given specific labels. For example, in stochastic bigraphs [5, 6] (that we use here) we label rewrite rules with *rates*. We draw these above the arrow between the left and right sides.

3 A Graphical Model of Ports

For island countries, such as the UK, Singapore, and Japan, (sea)ports play an essential role in maintaining stable logistic chains, given their much larger

capacity compared to aviation transport, and their reliability is essential. This makes them an ideal candidate for this work.

A common port setup, particularly for a mix of tourism and low-range goods transport, is roll-on/roll-off (RoRo) shipping. With RoRo, vehicles (rather than goods only) are loaded directly onto vessels, allowing them to arrive and leave the ports without additional infrastructure (cranes, warehouses, etc.). Key components of a port are the entrance queues—often split into several lanes supporting heavy goods vehicles (HGVs) and cars—, border control (for the source country and possibly the destination country), random security checks, and pre-shipping queues.

In general, ship unloading is significantly faster than loading as there are no checks to be done (since they were done at the loading port). Because of this, we only model the outwards direction.

3.1 Modelling the Port

Ports often have special handling based on the type of vehicles. For example, goods vehicles undergo both custom checks and security checks for explosive materials and preventing large-scale smuggling. When loading ships, it is easier to park larger vehicles first. Here we define bigraph entities for two vehicle types: Cars, representing any lightweight vehicle (small vans, motorbikes, etc), and Trucks, representing HGVs and trade vehicles more generally.

Transport scenarios are spatial in nature: vehicles must exist *somewhere*. We capture this by splitting the port into sequences[2] of Segments (including parallel Segments for multi-lane roads). Each segment has an identifier SName(s), and a queue of vehicles Q (modelled with the queues from Section 2). To allow flexible interconnections, segments also include many SLinkE and SLinkS entities (represented by the unique shapes • and ▾ respectively) that determine which segments incoming traffic may arrive from, and where outgoing traffic can go to. The use of two entities for this is a common technique to encode directionality in bigraphs (where links are unordered) [7]. Using multiple SLinkE entities, instead of a single hyperedge for the segment, allows different movement rates based on the source/destination segments (see Section 3.2).

Our model assumes a fixed number of discrete vehicles entering the port, and their movement through the port, rather than considering the port as an infinite operating system of in/out flows[3], and so our queues are always finite.

We use segments to model the layout of a port as a bigraph, a simplified version of which is presented in Fig. 2, informed by review of ports across the UK. We have two entry points for vehicles, roads A1 and A2. Upon entering, vehicles are directed into one of three possible lanes (lane A, lane B or lane C), and each lane will only accept either cars or HGVs. Vehicles queue in their lanes and then proceed to the foreign passport office for the destination country, followed by the passport office for the home country. A review of port processes in

[2] We allow for flexible setups including loops, i.e. we have a graph of segments.
[3] This is a form of macro-modelling.

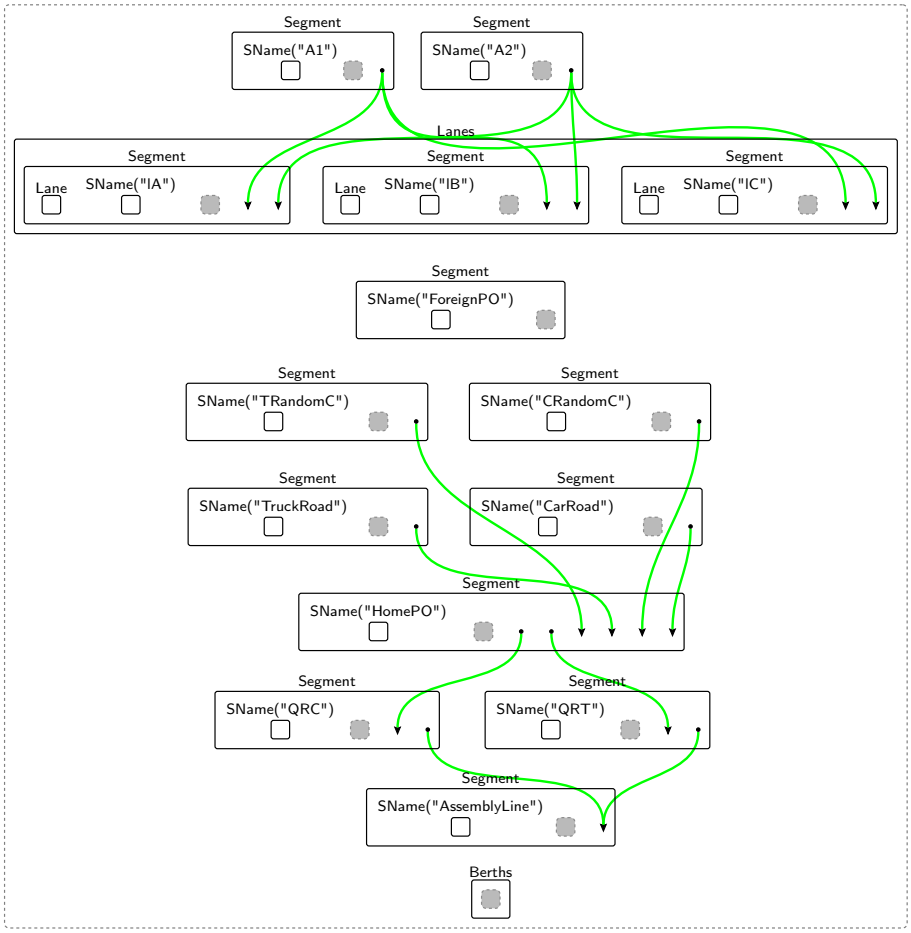

Fig. 2. A simplified bigraph model of a maritime port. We use Segments to illustrate the critical structure and substructure of the port, where sites abstract everything else away. Vehicles arrive via roads A1 and A2 and progress through the port following the movement links: from an SLinkS to one of the linked SLinkEs, as defined in the movement rule (see Fig. 3). The links to and from the foreign passport office are established dynamically (see Fig. 4).

the UK revealed that vehicles may be randomly picked to undergo an additional intermediary security check, which is often split into cars and trucks. After all checks have been performed, vehicles move to assembly lanes that are used to board a specific Ship once it arrives in the Berth. In real ports there is often some additional ferry operator checks (handling tickets, payments, etc.) but these are often significantly faster—and a target for removal through digitalisation in future—so we do not model these here. In the next section we describe how the movement of vehicles through the port is enabled by defining the appropriate reaction rules, and how these definitions can be static or dynamic depending on the nature of the inner logistics of the port.

3.2 Enabling Movement through the Port

The main operation of any transport scenario is the movement of vehicles between segments. In particular we are interested in the *rates* of movement under different scenarios, e.g. how congestion, or limited border control offices affect overall transit time. To enable rates with bigraphs, we use the stochastic bigraphs extension [5, 6] that associates rates with rewrite rules.

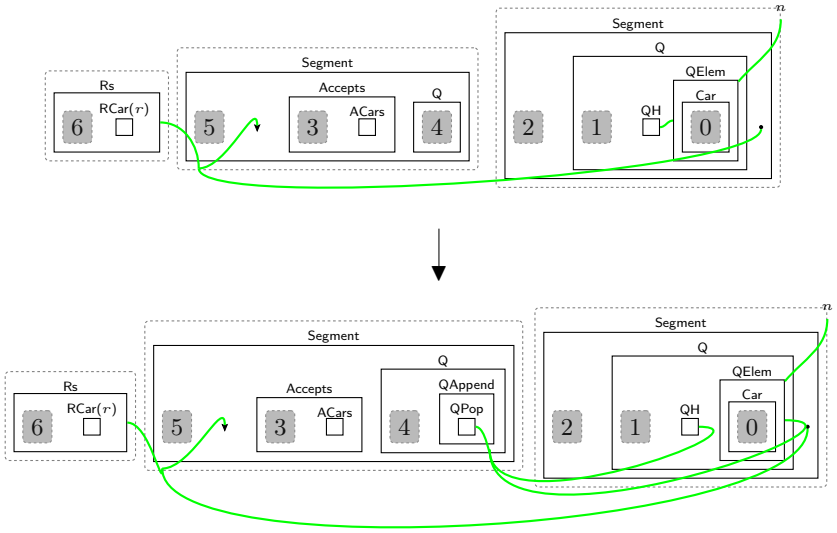

Fig. 3. Generic movement rule for cars. Movement happens at rate r into the next Segment so long as it Accepts cars (ACars).

The movement rate might depend on the specific segments involved, e.g. to model the fact that one road might have two lanes versus four lanes, and the type of vehicle moving (HGVs in general moving slower than cars). To allow

flexible rates, we encode the rate information within the model itself, utilising the hyperedges of bigraphs to attach rate information to a SLinkS/SLinkE pair. Specific rates based on the type of vehicle are encoded via nesting, i.e. RCar(r) and RTruck(r), where r is a user-specified nonnegative real number that can be inferred from historical data.

Ports have some control over which vehicles are allowed in which areas (for example, we might have a car-only lane). We encode this information by extending segments with an Accepts set (an entity with nested tokens ACar/ATruck) that determines when a particular vehicle is allowed in a segment.

The movement rule has the form shown in Fig. 3. This rule is specialised to cars, but a similar rule is available for trucks. Two separate rules are required since we must specialise on the accepts list and specific rate entity. Rules for queue management are similar to those in Section 2, with an additional (unshown) append rule that adds a vehicle to the end of the queue.

We use additional rules to enable flexible restructuring of the segments, i.e. adding and removing SLinkS/SLinkE dynamically. In particular, this models lane selection for the border control offices. We assume they only accept one vehicle at a time, which we enable by only linking the segment when the office is empty, and unlinking once a vehicle moves into it. An example is in Fig. 4 where lane A is dynamically linked to the foreign passport office. Another example of dynamic linking happens at the random checkpoints: we dynamically link a Car to TRandomC to encode that it has been picked for the additional check, and we weight the rule to simulate that there is a 5% probability of this happening. For trucks, the rule is identical but assumes a 10% probability.

The remaining rules in our model manage operations on ships (ships entering berths, being loaded, and leaving), whose load is tracked using a parametrised Load entity (e.g. Load(5) indicates the current load of the ship is 5). Whenever the berth is empty, a ship enters the berth immediately. Once a ship is docked at the berth, vehicles from the assembly lane can board it and update the current load of the ship. Once a ship is full it leaves the port.

4 Model Analysis: Safety and Reliability

We use BigraphER [28], an open-source framework for writing, manipulating, and executing bigraph models, to analyse the model. BigraphER enables model checking by generating a transition system, with states as bigraphs, and transitions as rewrites (up-to-bigraph-isomorphism). For the stochastic bigraphs used here, the state transitions are labelled with rates, that is, we export a continuous time Markov chain (CTMC). This transition system can be imported into external (probabilistic) model checkers such as PRISM [21] for analysis. Comprehensive details of the analysis—including the BigraphER models, the corresponding generated PRISM files, the queries executed on them, and the resulting data and plots—are available online[4].

[4] https://doi.org/10.5281/zenodo.15527329

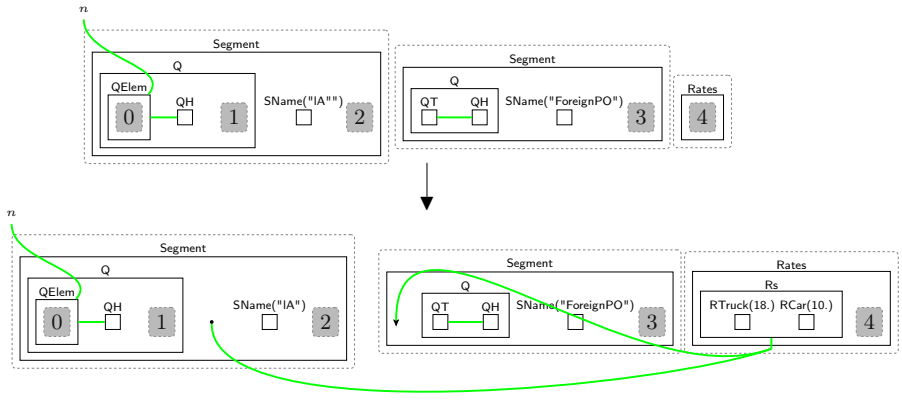

Fig. 4. A rule that dynamically links lane A to the foreign passport office ahead, by adding an SLinkS to the former, an SLinkE to the latter, movement rates as entities nested under Rates, and a hyperedge to link them all up.

It is useful to mark states that contain a domain-specific bigraph of interest, e.g. states where a particular entity exists. In BigraphER, this is done using bigraph predicates that are essentially left-hand sides of rules. States matching this left-hand side are labelled in the resulting transition system. These predicates can be used when generating logical formulae for the model checker. Here we use the stochastic fragment of Computational Tree Logic CSL [9, 10] supported by PRISM.

4.1 Safety

We begin with a set of queries that validate our model. These queries ensure that the model's implementation and its evolution are accurately captured and consistent with the realistic flow of traffic and operational practices at the port. To achieve this, we start by identifying erroneous scenarios and expressing them as Bigraph predicates. Examples include:

- Improper linking of elements within a segment (e.g. a vehicle currently located in one segment is incorrectly linked to a different segment).
- Transitions between segments without associated rates (in such cases, the transition will never occur).
- Misaligned segment arrangements, where queue segments are not connected end-to-end (i.e. the tail of one queue is not linked to the head of the next), or links follow an invalid pattern.

Once these predicates representing potential errors or safety violations are defined, we perform reachability analysis on the generated transition system. This allows us to check whether the system can reach any of these erroneous states.

If such a state is reachable, most model checkers can provide a counterexample trace, i.e. a sequence of transitions from the initial state to the error state. These traces help identify whether the problem lies in the model's construction or reflects an actual operational issue at the port.

In this paper, we primarily use these predicates as debugging tools to refine our model. The final model, used for the results presented in the subsequent sections, does not exhibit any of the errors previously identified.

In addition to error-checking predicates, the model includes two dedicated predicates to track the presence of vehicles at random checkpoints. Other predicates monitor system properties such as the load on the current ship, the total number of cars and trucks loaded, the berth's occupancy status, and whether the ship is full. These predicates enable further analysis of the model and its behaviour.

4.2 Reliability

Reliable logistics depends on efficient port operations and delays can cause, for example, refrigerated goods to spoil. We consider the following efficiency properties:

- What is the probability a ship leaves the port within t time units?
- What is the probability that m cars/trucks are on the ship within t time units?

A ship leaves the port when it is full (either by cars/trucks). We have a predicate 'n cars/trucks' which keeps track of how many cars/trucks have been on the ship (totalled across the entire model including multiple ships). Note that, it could be the scenario that, n cars need not fill a ship. Also, after a ship has departed and a new one has docked, m cars could be more than the load of a particular ship. We run our analysis with both two and three incoming lanes. For the three-lane configuration, we focus on specific lane assignments, i.e. designating specific lanes for cars or trucks. The lanes themselves are indistinguishable so the number of allocated lanes is the core difference. That is, it doesn't matter if lane one is assigned to cars or lane two, the overall effect is still a single car lane. This comparative analysis is valuable as it helps identify the optimal lane allocation for a given set of incoming vehicles, and insights gained can assist ports optimising for reliability, e.g. if a lane is blocked completely due to a break-down, what is a sensible reallocation.

Figure 5 shows the probability of specific events based on lane allocation as we vary the time. Importantly, these are *exact* probabilities (not simulation runs). That is, all possible paths are considered such that a probability of 1 implies this happens on all paths within some time. The road segments initially contain a total of 2 cars and 2 trucks across all models with one lane organised as a car followed by a truck and the other as a truck followed by a car, respectively. The maximum load a ship can take is 2 and a car weighs 1 unit whereas a truck weighs 2 units. The Any 2 and Any 3 cases are where there are no predetermined lane

assignments, for the 2-lane and 3-lane scenarios, respectively. Lane allocation cases are written as, for example, CCC indicating all three lanes accept **only** cars, while CCT means two lanes for cars and one for trucks. The all-truck (TTT) lane configuration serves as a validation case, as it prevents cars from progressing to the next segment.

Figure 5a, considers the probability that two cars[5] are loaded on the ship within t time. As expected, the probability reaches 1 the fastest when all lanes are designated for cars, and decreases as the number of car lanes is reduced. A similar trend is observed for trucks in Fig. 5b where we consider the probability that a one truck is loaded on the ship within t time.

Figure 5c considers the probability a ship has a load of 1, where the load of a car is 1 unit and truck is 2 units. This does not directly follow Fig. 5a as if a truck boards before a car the load of 1 is never seen (so there are many more negative cases). As expected having more car lanes allows better performance.

We also considered a bigger model with 9 vehicles comprising of 6 cars and 3 trucks (as realistically we have more cars than trucks). Their corresponding loads were the same, i.e. 1 and 2 for cars and trucks. The maximum load a ship can take is 6 is in this case. We used BigraphER to generate the state space of the CTMC up to $100,000$ states and ran a set of similar queries as discussed above for the smaller model. The corresponding results for the bigger model can be found in Fig. 5d, Fig. 5e, and Fig. 5f. In reality, multiple ships can arrive and depart from the port. However, the above query only evaluates performance up to the first ship's departure. Due to the semantics of CSL, subsequent ships are not accounted for in these initial queries.

Port operations are consistently more efficient with 3 lanes compared to 2 (comparing Any 3 to Any 2), even though there are sequential bottlenecks in the passport office (as shown by the Any 3 and Any 2 legends in the plots). This result is intuitive, but the analysis quantifies the exact performance gain obtained by adding an additional lane. These metrics can guide future infrastructure planning by locating the point where further expansion yields diminishing returns. The analysis can be repurposed to evaluate operational resilience. For example, when a lane must be temporarily closed for maintenance or a breakdown. This analysis provides valuable insights into the robustness of current infrastructure and helps authorities plan for contingencies, optimise maintenance schedules, and develop strategies to mitigate disruptions without significantly compromising reliability and throughput.

5 Conclusions

Utilising a port case study, we have shown bigraphs to be an effective tool for modelling complex transport scenarios. The diagrammatic nature of the formalism makes it useful for non-formal-methods experts without sacrificing on rigour, e.g. we can do full probabilistic model checking and have a strong logical

[5] We do not assign identifiers to vehicles, so can only say *a* vehicle, not a specific vehicle.

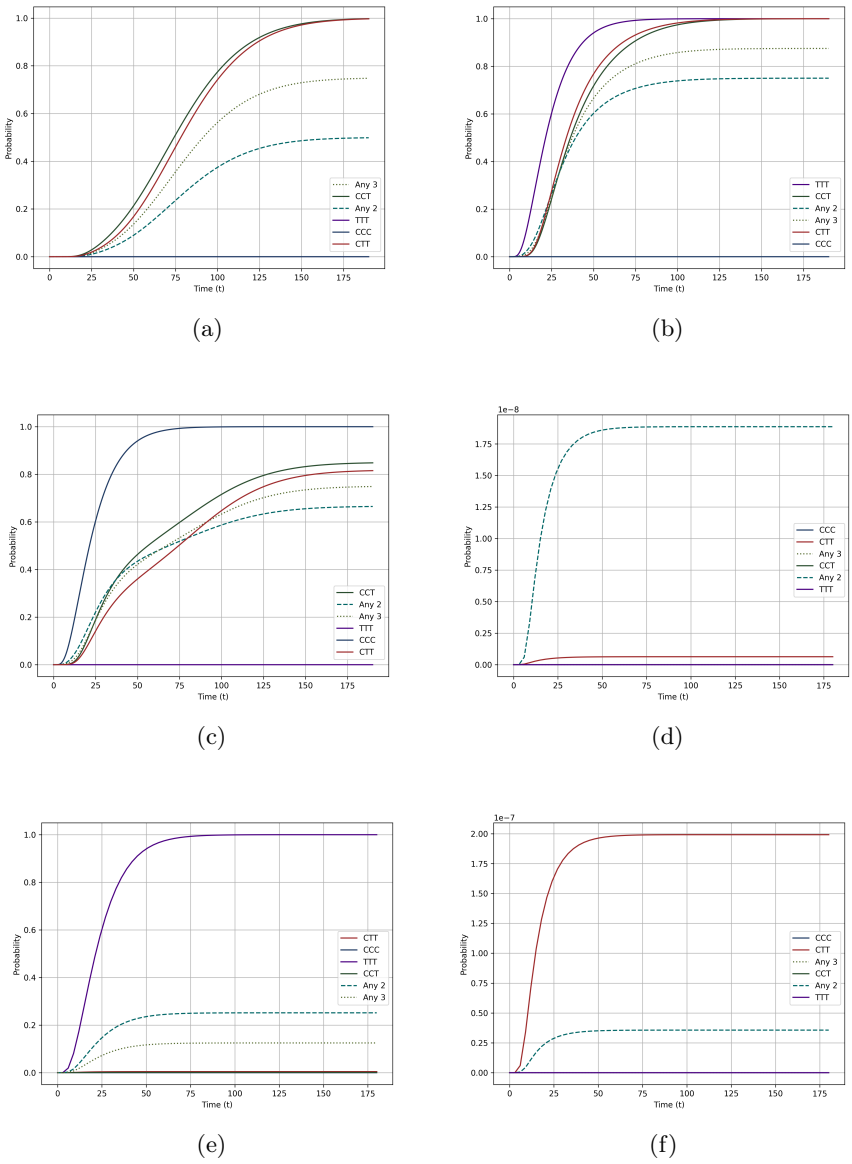

Fig. 5. Model checking results for two port operations models (initialised with 2 cars and 2 trucks). Any 2, has two lanes that accept any vehicle, while Any 3 has three lanes that accept any vehicle one with three lanes. CCT etc. are fixed lane assignments (e.g. car lanes and one truck lane). They give the probability (within time t): (a) two cars are on the ship; (b) a truck is on the ship; (c) the ship has exactly one car and no trucks. The next set of plots present the results for the same model initialised with 6 cars and 3 trucks: (d) two cars are on the ship; (e) a truck is on the ship; (f) the ship has either 2 trucks, 4 cars, or 2 cars and a truck.

structure to work with meaning we could extend the approach with other verification methods such as proofs, and simulation-based statistical model checking [14, 17, 20].

Bigraphs make it easy to capture the spatial structure of systems which is essential for transport, and *user-defined* rules allow it to pivot to different scenarios with ease. This makes it more useful for transport than other formal modelling techniques like manually building Discrete/Continuous time Markov chains [11], Petri-nets, π-calculus etc. For example, while we have used PRISM for verification, it is difficult to build this scenario directly in the PRISM reactive modules language as it has no support for complex-types, e.g. queues/linked-lists, spatial elements need to be encoded manually (by linking reactive modules on action names), and there is no easy to visualise diagrammatic output.

As future work, we want to relax some of our modelling assumptions (single border control, limited lanes, same rates etc.). There are currently only two types of vehicles, cars and trucks, but this set could be extended and more attributes added. For example, we are particularly interested in using these models for carbon reduction and this could warrant further classifying vehicles based on petrol/diesel/electric engine types. We also intend to compare our results with those produced by simulation-based tools such as MatSIM, SUMO, and others—particularly in terms of scalability, usability, and overall performance.

More generally we want to explore the scalability of the models, and believe there is scope to automatically generate, from the set of rules, population models [15, 22] (where we count vehicles rather than explicitly queue them), simulation models, differential equations, or by pre-computing envelopes of behaviour (as done by Calder et al. to estimate the probability of component failure in a critical communications service and avoid the cost of online model checking [16]). We also want to export the models as action-labelled models with rewrite rules as the transition labels between states. This will allow us to reason over probabilistic and stochastic variants of the μ-calculus [24], and later allow us to reason on complex specifications in logics over CSL with models having both actions and state labels, like asCSL [12]. For example, given an empty ship is loaded with a truck, what is the probability that no more than k trucks are loaded onto the ship before it is full. This can be considered as a fairness constraint that maintains the proportion of cars to trucks or number of trucks on a ship close to reality. Such queries can then be used to change the rates for consequent reaction rules or their priority classes to mimic reality better.

The transport sector is filled with sensors and an abundance of data, and we want to explore how this can interact with the model. For example, can live data feed both initial model states and dynamic rate updates to enable on-the-fly what-if scenarios to be explored, e.g. what-if we close this lane for maintenance now. This can be seen as a form of digital twinning over formal models [8].

Acknowledgments.

This work is supported by the Engineering and Physical Sciences Research Council, under grant EP/Z533221/1 (TransiT: Digital Twinning Research Hub for

Decarbonising Transport) and an Amazon Research Award on Automated Reasoning.

References

1. AnyLogic Simulation Software., https://www.anylogic.com/
2. iTwin Platform., https://www.bentley.com/software/itwin-platform/
3. MATSim: Multi-Agent Transport Simulation., https://matsim.org/
4. SUMO (simulation of Urban MObility), https://eclipse.dev/sumo/
5. Albalwe, M., Archibald, B., Sevegnani, M.: Modelling real-time systems with bigraphs. Electronic Proceedings in Theoretical Computer Science **417**, 96–116 (03 2025). https://doi.org/10.4204/EPTCS.417.6
6. Archibald, B., Calder, M., Sevegnani, M.: Probabilistic bigraphs. Formal Aspects Comput. **34**(2), 1–27 (2022). https://doi.org/10.1145/3545180
7. Archibald, B., Calder, M., Sevegnani, M.: Practical modelling with bigraphs (2024), https://arxiv.org/abs/2405.20745
8. Archibald, B., Harvey, P., Sevegnani, M.: A digital twinning approach to decarbonisation: Research challenges. In: 1st International Workshop on Low Carbon Computing (Dec 2024)
9. Aziz, A., Sanwal, K., Singhal, V., Brayton, R.K.: Verifying continuous time markov chains. In: Computer Aided Verification, 8th International Conference, CAV '96, New Brunswick, NJ, USA, July 31 - August 3, 1996, Proceedings. pp. 269–276 (1996)
10. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time markov chains. IEEE Transactions on Software Engineering **29**(6), 524–541 (2003). https://doi.org/10.1109/TSE.2003.1205180
11. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press (2008)
12. Baier, C., Cloth, L., Haverkort, B.R., Kuntz, M., Siegle, M.: Model checking markov chains with actions and state labels. IEEE Trans. Software Eng. **33**(4), 209–224 (2007). https://doi.org/10.1109/TSE.2007.36, https://doi.org/10.1109/TSE.2007.36
13. Blooshi, M.A., Jafer, S., Patel, K.: Review of formal agile methods as cost-effective airworthiness certification processes. Journal of Aerospace Information Systems **15**(8), 471 – 484 (2018). https://doi.org/10.2514/1.I010601, cited by: 5
14. Bogdoll, J., Ferrer Fioriti, L.M., Hartmanns, A., Hermanns, H.: Partial order methods for statistical model checking and simulation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **6722 LNCS**, 59 – 74 (2011). https://doi.org/10.1007/978-3-642-21461-5_4, cited by: 45
15. Bortolussi, L., Lanciani, R., Nenzi, L.: Model checking markov population models by stochastic approximations. Information and Computation **262**, 189 – 220 (2018). https://doi.org/10.1016/j.ic.2018.09.004, cited by: 7; All Open Access, Bronze Open Access, Green Open Access
16. Calder, M., Sevegnani, M.: Stochastic model checking for predicting component failures and service availability. IEEE Transactions on Dependable and Secure Computing **16**(1), 174–187 (2019). https://doi.org/10.1109/TDSC.2017.2650901
17. El Rabih, D., Pekergin, N.: Statistical model checking using perfect simulation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial

Intelligence and Lecture Notes in Bioinformatics) **5799 LNCS**, 120 – 134 (2009). https://doi.org/10.1007/978-3-642-04761-9_11, cited by: 38
18. Gigante, G., Pascarella, D.: Formal methods in avionic software certification: The do-178c perspective. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7610 LNCS**(PART 2), 205 – 215 (2012). https://doi.org/10.1007/978-3-642-34032-1_21, cited by: 18
19. Kacprzak, M., Lomuscio, A., Łasica, T., Penczek, W., Szreter, M.: Verifying multi-agent systems via unbounded model checking. In: Hinchey, M.G., Rash, J.L., Truszkowski, W.F., Rouff, C.A. (eds.) Formal Approaches to Agent-Based Systems. pp. 189–212. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)
20. Kroiß, C.: Simulation and statistical model checking of logic-based multi-agent system models. Advances in Intelligent Systems and Computing **296**, 151 – 160 (2014). https://doi.org/10.1007/978-3-319-07650-8_16, cited by: 3
21. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Proc. 23rd International Conference on Computer Aided Verification (CAV'11). LNCS, vol. 6806, pp. 585–591. Springer (2011)
22. Latella, D., Loreti, M., Massink, M.: On-the-fly fluid model checking via discrete time population models. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **9272**, 193 – 207 (2015). https://doi.org/10.1007/978-3-319-23267-6_13, cited by: 5
23. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: The 21st IEEE International Conference on Intelligent Transportation Systems. IEEE (2018), https://elib.dlr.de/124092/
24. Mateescu, R., Requeno, J.I.: On-the-fly model checking for extended action-based probabilistic operators. Int. J. Softw. Tools Technol. Transf. **20**(5), 563–587 (2018)
25. Milner, R.: The Space and Motion of Communicating Agents. Cambridge University Press (2009)
26. Ortúzar S., J.d.D., Willumsen, L.G.: Modelling transport. Wiley-Blackwell, Oxford, 4th edn. (2011)
27. Ritchie, H.: Cars, planes, trains: where do co_2 emissions from transport come from? Our World in Data (2020), https://ourworldindata.org/co2-emissions-from-transport
28. Sevegnani, M., Calder, M.: Bigrapher: Rewriting and analysis engine for bigraphs. In: Chaudhuri, S., Farzan, A. (eds.) Computer Aided Verification - 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II. Lecture Notes in Computer Science, vol. 9780, pp. 494–501. Springer (2016). https://doi.org/10.1007/978-3-319-41540-6_27
29. Sevegnani, M., Kabac, M., Calder, M., McCann, J.: Modelling and verification of large-scale sensor network infrastructures. In: 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS). pp. 71–81 (2018). https://doi.org/10.1109/ICECCS2018.2018.00016
30. W Axhausen, K., Horni, A., Nagel, K.: The multi-agent transport simulation MATSim. Ubiquity Press (2016)