The Pyramis Library: Efficient Numerical Evaluation of Hierarchical UML Statecharts applied to Stochastic Workflows

Laura Carnevali[1], Reinhard German[2], Leonardo Montecchi[3], Leonardo Scommegna[1], and Enrico Vicario[1]

[1] Information Engineering Dept., University of Florence, Italy
{laura.carnevali,leonardo.scommegna,enrico.vicario}@unifi.it
[2] Computer Science Dept., University of Erlangen-Nuremberg, Germany
reinhard.german@fau.de
[3] Computer Science Dept., Norwegian University of Science and Technology, Norway
leonardo.montecchi@ntnu.no

Abstract. Pyramis is a library for quantitative evaluation of hierarchical UML statecharts with non-Markovian stochastic timing and probabilistic choices. It implements an efficient numerical approach for transient analysis until absorption and steady-state analysis, separately evaluating the Semi-Markov Process (SMP) of each model component. As Pyramis facilitates code reusability, maintainability, and extensibility, it has been easily integrated with the FaultFlow library for dependability analysis of component-based systems, supporting efficient quantitative evaluation of stochastic static fault trees without repeated events.
In this paper, we use Pyramis for quantitative evaluation of workflows where activities have non-Markovian stochastic duration and where precedence constraints define a Directed Acyclic Graph (DAG). Workflows have a Service Level Objective (SLO) on their end-to-end (E2E) response time distribution at low workloads of requests. Pyramis efficiently derives the workflow E2E response time distribution, yielding a stochastic upper bound for topologies with non-well-nested precedence DAGs. We report experiments for a workflow with topology derived from a real benchmark and execution times obtained from a dataset of the literature. Results are promising in terms of tradeoff between accuracy and complexity.

Keywords: Statecharts, workflows with stochastic durations, non-Markovian distributions, semi-Markov processes, software tools and libraries.

1 Introduction

Motivation. Quantitative evaluation of non-Markovian models has the potential to effectively support performance and dependability engineering of complex time-critical systems. To this end, the adopted formalism should guarantee not only ease of model construction and harmonization with consolidated industrial practice [**?**,**?**], but also analyzability of the underlying stochastic process. In addition, a software tool should support stochastic modeling and analysis. Addressing these intertwined aspects is challenging and fundamental to success.

Related works. Model Driven Engineering (MDE) [?,?] pursues this goal by automatically deriving stochastic models from semi-formal artifacts [?,?,?]. These artifacts are typically specified by diagrams of the Unified Modeling Language (UML), annotated by the profiles for Modeling and Analysis of Real-Time Embedded systems (MARTE) [?,?] and for Dependability Analysis and Modeling (DAM) [?], or by diagrams of the Systems Modeling Language (SysML) [?].

Most of the literature on MDE translates semi-formal artifacts into Markovian models [?,?,?,?,?,?], leveraging consolidated and efficient solution techniques [?,?,?,?] while limiting the model expressivity by using only exponential timers, preventing representation of phenomena like synchronous events (e.g, periodic tasks) and behaviors depending on history (e.g., aging). Few MDE approaches exploit non-Markovian models [?,?,?,?,?,?], improving model expressivity with non-Markovian timers, while facing significantly harder complexities in the evaluation of the underlying stochastic process [?]. To mitigate the issue, these approaches either limit the number of concurrent non-Markovian timers, or consider classes of domain-specific models whose concurrency structure and stochastic timing guarantee feasibility of stochastic analysis [?,?,?,?,?,?]. The method of [?,?] performs availability analysis of dynamic fault trees, exploiting the model structure to compute stochastic bounds on the time to failure distribution and evaluate maintenance policies. The approach of [?] derives the execution time distribution of a workflow with precedence constraints defined by a Directed Acyclic Graph (DAG), by fitting general distributions with phase-type distributions, and by combining numerical evaluation of subgraphs having series or parallel topology with probabilistic model checking of the other subgraphs. [?] has suggested to address stochastic modeling and analysis of a hierarchical extension of UML statecharts with non-Markovian stochastic timing and probabilistic choices, termed Hierarchical Semi-Markov Process with parallel regions (HSMP). It has been extended in [?] by the time advance mechanism known from stochastic state classes to take exits in parallel regions with different time origins correctly into account. In [?] it was put on a formal semantical basis, extended further, and implemented in the Pyramis library. The HSMP formalism provides ease of representation, while reducing the model expressiveness with fairly lax restrictions that enable separate analysis of the Semi-Markov Process (SMP) underlying each model component, guaranteeing efficient evaluation of steady-state and transient behavior (until absorption).

Contribution. Pyramis [?] is a Java library implementing the approach of [?,?], using consolidated design patterns [10] and MDE principles to support code reusability, maintainability, and extensibility, as well as integration with other libraries and tools. It is available open source under the AGPLv3 licence [?].

In this paper, we illustrate how to use Pyramis for quantitative evaluation of workflows, a relevant class of stochastic models in performance engineering [?]. Specifically, we consider workflows composing activities with DAG precedence constraints and non-Markovian durations. Workflows have a Service Level Objective (SLO) on their end-to-end (E2E) response time distribution at low workloads of requests. We translate the UML activity diagram of a workflow into

an HSMP, disregarding the representation of dependencies in non-well-nested precedence DAGs, while efficiently deriving a stochastic upper bound on the workflow E2E response time distribution. We perform experiments for a workflow with topology derived from a real benchmark and execution times obtained from a dataset of the literature, obtaining promising results in the achievement of a tradeoff between accuracy of evaluation and complexity of computation.

Structure. In the following, first we illustrate the Pyramis metamodel (**??**). Then, we use Pyramis to model workflows with stochastic durations (Section 3) and we report experimental results for a workflow with topology derived from a real benchmark and execution times obtained from a dataset of the literature (Section 4). Finally, we draw our conclusions and discuss future work (Section 5).

2 An overview of the Pyramis library

Pyramis is a Java library for quantitative modeling and evaluation of HSMPs [**?**,**?**,**?**], an extension of UML statecharts capturing the concepts of concurrency, hierarchy, non-Markovian stochastic timing, and probabilistic choices.

HSMP syntax. **??** shows the Pyramis metamodel using UML activity diagram notation, and **??** shows the HSMP of a workflow (we discuss in Section 4 how to derive the HSMP of **??** from the UML activity diagram of **??**). Specifically, an HSMP (modeled by class `HSMP` in **??**) contains locations (modeled by class `LogicalLocation` in **??**) which can be either *steps* (modeled by class `Step` in **??** and depicted as rounded boxes in **??**), i.e., actions with stochastic duration, or *final locations* (modeled by class `FinalLocation` in **??**), i.e., terminated actions. In turn, steps can be either *simple* (modeled by class `SimpleStep` in **??**), i.e., atomic actions having a duration distribution, or *composite* (modeled by class `CompositeStep` in **??**), i.e., composed actions made of *concurrent regions* (modeled by class `Region` in **??** and separated by dashed lines in **??**). Regions can be of type either `ENDING` (i.e., terminating in a final location) or `NEVERENDING` (i.e., not terminating). We collect regions in composite step of type either `LAST` (i.e., terminating as soon as all its regions have terminated, with final locations

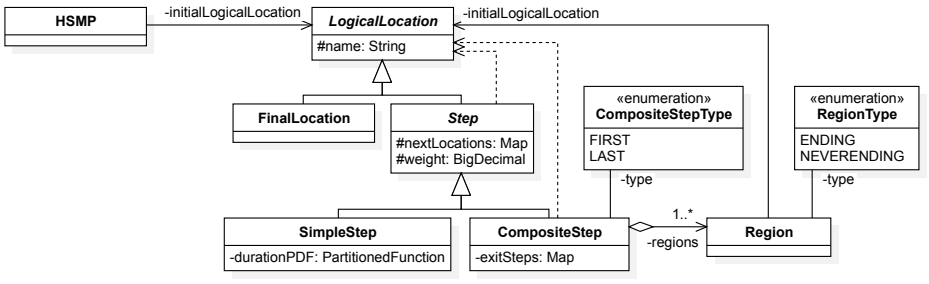


Fig. 1: Metamodel of the Pyramis library, using UML class diagram notation.

depicted in ?? as unfilled circles on the border of the region, with a smaller black-filled circle inside) or FIRST (i.e., terminating as soon as any of its regions terminates, with final locations depicted as unfilled circles on the border of the region, with an X–shaped cross inside)[4].

Composition of regions and steps yields a hierarchy of HSMPs where each composite step is defined by an HSMP at the next lower level (e.g., composite step B in ?? is itself defined by an HSMP), the top-level HSMP contains a single region (i.e., the top-level region of ?? contains a single region composing composite step AtoM and simple steps N and O in sequence), and non-top-level HSMPs contain at least one region of type ENDING (e.g., the regions of all the composite steps in ?? are of type ENDING).

HSMP semantics. The state of an HSMP consists of: an active location for each region of the composite step modeling the HSPM; a time to the next event (TNE) for each active step (modeling the time to the completion of the step); and, a state for each active composite step. As an example, we consider the initial state of the HSMP of ??, where all the simple steps contained in composite steps A′, A″, and B are active and sample a TNE from their duration distribution. After each event (i.e., step completion), the TNE of each active step is reduced by the TNE of the completed step. If all the steps of A′ terminate before all the steps of A″ and B have terminated, then C′ becomes the newly active step of region RACE′ and samples a TNE. Otherwise, if all the steps of A″ terminate before all the steps of A′ and B have terminated, then C″ becomes the newly active step of region RAC″ and samples a TNE; conversely (i.e., if all the steps of B terminate before all the steps of A′ and A″ have terminated), D becomes the newly active step of region RBD and samples a TNE. Then, the model execution proceeds event after event. Leveraging this semantics, the approach of [?] implemented by Pyramis separately derives the duration distribution of each composite step by separately evaluating the SMP of each of its regions. Then, the approach exploits these distributions to derive the steady-state probability of each step.

Pyramis features. Pyramis is designed following consolidated design patterns [10] and MDE principles [?,?] to support code reusability, maintainability, and extensibility, and integration with other libraries. As evidence of this fact, Pyramis has been easily integrated with FaultFlow [?,?], a Java library for dependability evaluation of component-based systems, with focus on fault propagations within individual components and between different components. The main outcome of such integration was the automated translation of stochastic static fault trees without repeated events from the FaultFlow metamodel to the Pyramis metamodel, enabling the exploitation of the solution method implemented by Pyramis to derive the distribution of the time to the occurrence of any component failure (including the system failure) and the Birnbaum and Fussell-Vesely importance measures of faults. Notably, Pyramis efficiently affords the analysis of very complex SSFTs with large depth and hundreds of different faults.

[4] Composite steps of type FIRST are not shown in the HSMP model of ?? due to the fact that they are not used to model the considered class of workflows.

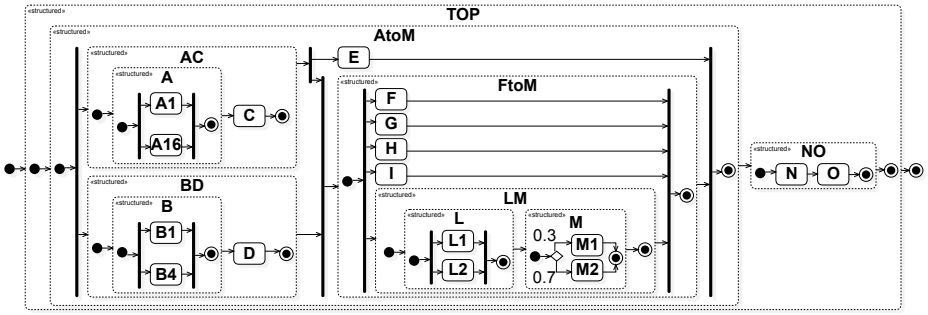


Fig. 2: UML activity diagram of a variant of the Sipht workflow of [**?**,**?**] (the name of a structured action indicates the names of the actions that it composes).

3 Quantitative modeling of workflows using Pyramis

We consider workflows composing actions with durations having non-Markovian distributions and with precedence constraints defining a DAG. **??** shows the UML activity diagram of a workflow, extending the topology of the Sipth workflow of the Pegasus repository [**?**,**?**] to include all action composition patterns for illustration purposes. Specifically, a workflow consists of *elementary actions* modeling atomic operations (e.g., `A1` to `A16`) and *structured actions* (labeled with stereotype `<<structured>>`) composing other (elementary or structured) actions (e.g., `AC` and `BD`). A workflow composes actions with different constructs: *sequence* (e.g., `AC` models the sequential execution of `A` and `C`); *fork-join* (e.g., `A` models the concurrent execution of `A1`, `A2`, ..., `A16`); *split-merge* (e.g., `M` models the exclusive execution of either `M1` or `M2` with probability 0.3 and 0.7, respectively); *irreducible DAG*, i.e., a DAG that is made of *individual* fork and join operators and cannot be modeled by sequence and fork-join constructs previously introduced[5] (e.g., `TOP` composing `AC`, `BD`, `E`, `FtoM`, and `NO`). A *non-well-nested* DAG includes irreducible DAG constructs, while a *well-nested* DAG does not.

By construction, HSMPs cannot represent a workflow that composes actions with irreducible DAG constructs. Rather, they can model a well-nested variant of the workflow where we replicate the actions belonging to different paths of irreducible DAG constructs (in a DAG backward visit, we replicate each action that is predecessor of more than one action or of a replicated action). **??** shows the UML activity diagram of such variant of the workflow of **??**, obtained by replicating `AC` (predecessor of `E` and `FtoM`) into `AC'` and `AC''`. Replication of predecessor actions guarantees that the E2E response time distribution $\tilde{F}$ of the obtained workflow is a *stochastic upper bound* of the E2E response time distribution F of the actual workflow (i.e., $\tilde{F}(t) \leq F(t)\ \forall\, t$), as proved in [**?**].

Then, we easily derive the HSMP of a workflow from the UML activity diagram of the mentioned variant of the workflow, by mapping: elementary actions into steps; sequence constructs into a sequence of steps, one predecessor of the

[5] Individual fork and join operators are different constructs than the fork-join one.

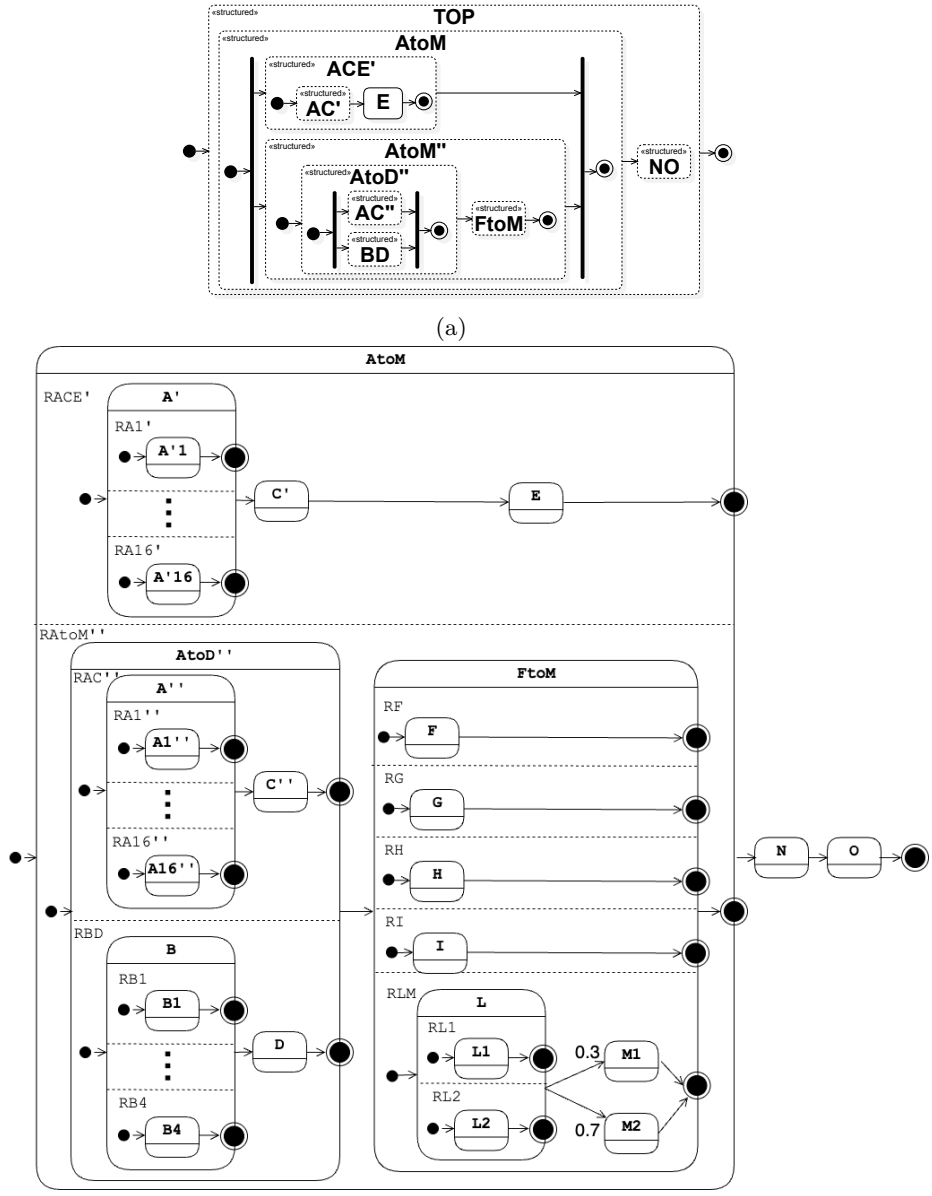


Fig. 3: (a) UML activity diagram of a variant of the workflow of **??**, obtained by replicating `AC` into `AC'` and `AC''` (the name of a structured action contains the symbol `'` or the symbol `''` depending on whether the structured action contains `AC'` or `AC''`, respectively). (b) HSMP of the workflow of **??**.

other; fork-join constructs into composite steps of type `LAST`; and, split-merge constructs into steps with multiple successor steps. **??** shows the HSMP of the workflow of **??** obtained from the UML activity diagram of **??**.

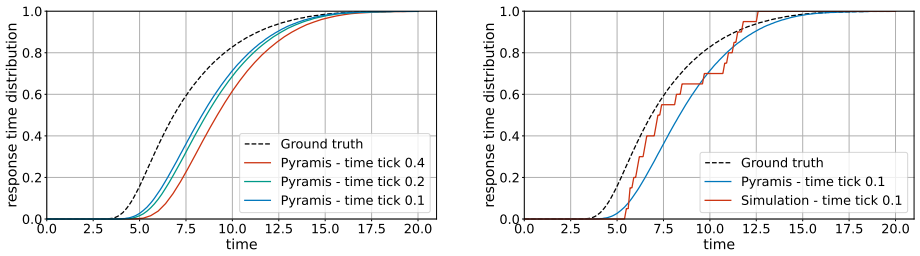
4 Quantitative evaluation of workflows using Pyramis

For each elementary action of the workflow of **??**, we define a duration distribution that approximates a histogram of samples derived from the WS-Dream dataset [**?**], which collects response times of many web services. Specifically, we consider 100 web services and, for each of them, we derive a histogram by collecting and regularizing the related samples according to the inter-quartile range rule [**?**]. Then, for each elementary action of our workflow, we randomly select a histogram and we approximate it with the distribution of a shifted truncated exponential random variable fitting its mean value and its coefficient of variation [**?**], i.e., a random variable having support $[a, b]$, probability density function $h(t) := \lambda\, e^{-\lambda t}\, e^{a\lambda}/(1 - e^{(a-b)\lambda})$ for some rate $\lambda > 0$, and probability distribution function $H(t) := (1 - e^{a\lambda} e^{-\lambda t})/(1 - e^{(a-b)\lambda})$.

Transient analysis until absorption of an HSMP yields the duration distribution of the top-level composite step. For HSMPs modeling the class of workflows considered in Section 3, this distribution represents the workflow E2E response time distribution. If the workflow is well-nested, the distribution is calculated in exact manner. Otherwise, the HSMP disregards dependencies in irreducible DAG constructs, and its transient analysis yields a stochastic upper bound on the workflow E2E response time distribution, as discussed in Section 3.

?? plots the ground-truth E2E response time distribution of the workflow of **??**, computed by performing a 1-million-run simulation of the Stochastic Time Petri Net (STPN) modeling the workflow through the Sirio library [19] of the ORIS tool [17], with time limit 20 and time tick 0.1, requiring around 30 min on a single core of an Intel i7-1260P CPU 4.70 GHz with 32.0 GB RAM. **??** also plots the E2E response time distribution of the workflow of **??** computed by Pyramis with the same time limit and with time tick 0.1, 0.2, and 0.4, achieving Jensen-Shannon (JS) distance from the ground truth nearly equal to 0.0024, 0.0034, and 0.0062, respectively, and requiring a computation time of nearly 63 ms, 64 ms, and 114 ms, respectively. As expected, the computation time and the accuracy (in terms of JS distance from the ground truth) increase as the time tick decreases. Results show that the approach is feasible even for a complex workflow with topology derived from a real benchmark application and that the computed E2E response time distribution is quite accurate with respect to the simulative ground truth, also confirming that this approximated distribution is actually a stochastic upper bound of the ground truth distribution.

?? compares the distribution provided by the ground truth and Pyramis with time limit 20 and time tick 0.1 with that computed by a 20-run simulation with the same computation time as Pyramis. Notably, simulation achieves nearly the same JS distance from the ground truth as Pyramis, while not guaranteeing that it is a stochastic upper bound of the ground truth distribution.



(a) (b)

Fig. 4: E2E response time distribution of the workflow of **??**.

5 Conclusions

We plan to integrate Pyramis with Eulero [**?**], a Java library for quantitative evaluation of the considered class of workflows. Specifically, Eulero computes a stochastic upper bound on the E2E response time distribution through a compositional approach that limits the number of simplifications performed to manage dependencies in non-well-nested precedence DAGs, achieving better evaluation accuracy than Pyramis while incurring greater computational complexity. We could extend Pyramis to accept workflows specified in the Eulero format to compare the two solution on a wide set of workflows. Moreover, the approach of Pyramis could be used also to analyze specific subworkflows to support the decisions taken by the compositional solution of Eulero (e.g., identify the most complex subworkflow, to be analyzed separately). We will leverage consolidated MDE principles to develop the software modules required to integrate Pyramis with Eulero, making them available open source under the AGPLv3 licence. We will also provide an artifact to replicate the experimental results of this paper.

References

1. ACEA: The 2030 urban mobility challenge. Tech. rep. (May 2016)
2. Balaji, P., Srinivasan, D.: Multi-agent system in urban traffic signal control. IEEE Computational Intelligence Magazine **5**(4), 43–51 (2010)
3. Balcombe, R., Mackett, R., Paulley, N., Preston, J., Shires, J., Titheridge, H., Wardman, M., White, P.: The demand for public transport: a practical guide (2004)
4. Bertocci, N., Carnevali, L., Scommegna, L., Vicario, E.: Efficient derivation of optimal signal schedules for multimodal intersections. Simulation Modelling Practice and Theory p. 102912 (2024)
5. Cheng, C., Du, Y., Sun, L., Ji, Y.: Review on theoretical delay estimation model for signalized intersections. Transport Reviews **36**(4), 479–499 (2016)
6. Dion, F., Hellinga, B.: A rule-based real-time traffic responsive signal control system with transit priority: application to an isolated intersection. Transportation Research Part B: Methodological **36**(4), 325–343 (2002)

7. Eom, M., Kim, B.I.: The traffic signal control problem for intersections: a review. European transport research review **12**, 1–20 (2020)
8. ERTRAC: Integrated urban mobility roadmap. Tech. rep., European Road Transport Research Advisory Council (February 2017)
9. Faria, R., Brito, L., Baras, K., Silva, J.: Smart mobility: a survey. In: Int. Conf. on IoT for the Global Community. pp. 1–8. IEEE (2017)
10. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Patterns, D.: Elements of reusable object-oriented software. Design Patterns (1995)
11. Guo, Q., Li, L., Ban, X.J.: Urban traffic signal control with connected and automated vehicles: a survey. Transportation Research Part C: emerging technologies **101**, 313–334 (2019)
12. Li, Y., Sun, D.: Microscopic car-following model for the traffic flow: the state of the art. J. Contr. Theory and Appl. **10**(2), 133–143 (2012)
13. Li, Z., Elefteriadou, L., Ranka, S.: Signal control optimization for automated vehicles at isolated signalized intersections. Transportation Research Part C: Emerging Technologies **49**, 1–18 (2014)
14. Lighthill, M.J., Whitham, G.B.: On kinematic waves II. A theory of traffic flow on long crowded roads. Proc. of the Royal Society of London. Series A. Mathematical and Physical Sciences **229**(1178), 317–345 (1955)
15. Maerivoet, S., De Moor, B.: Cellular automata models of road traffic. Physics reports **419**(1), 1–64 (2005)
16. Ng, K.M., Reaz, M.B.I., Ali, M.A.M.: A review on the applications of Petri nets in modeling, analysis, and control of urban traffic. IEEE Trans. on Int. Transp. Sys. **14**(2), 858–870 (June 2013). https://doi.org/10.1109/TITS.2013.2246153
17. Paolieri, M., Biagi, M., Carnevali, L., Vicario, E.: The ORIS tool: quantitative evaluation of non-Markovian systems. IEEE Trans. Softw. Eng. **47**(6), 1211–1225 (June 2021)
18. Reddy, R., Almeida, L., Gaitán, M.G., Santos, P.M., Tovar, E.: Synchronous management of mixed traffic at signalized intersections towards sustainable road transportation. IEEE Access (2023)
19. SIRIO Library: `https://github.com/oris-tool/sirio` (2024)
20. Stephanopoulos, G., Michalopoulos, P.G., Stephanopoulos, G.: Modelling and analysis of traffic queue dynamics at signalized intersections. Transportation Research Part A: General **13**(5), 295–307 (1979)
21. Vicario, E., Sassoli, L., Carnevali, L.: Using stochastic state classes in quantitative evaluation of dense-time reactive systems. IEEE Trans. Softw. Eng. **35**(5), 703–719 (2009)
22. Webster, F.V.: Traffic signal settings. Road Research Technical Paper 39. Tech. rep. (1958)
23. Wei, H., Zheng, G., Gayah, V., Li, Z.: A survey on traffic signal control methods. arXiv preprint arXiv:1904.08117 (2019)
24. Yagar, S., Han, B., Greenough, J.: Real-time signal control. for mixed traffic and transit based on priority rules. In: Traffic Management. Proc. of the Eng. Foundation Conference (1992)