Automating the Analysis of Quantitative Automata with QuAK[*]

Marek Chalupa[1] , Thomas A. Henzinger[1] , Nicolas Mazzocchi[2] , and
N. Ege Saraç[1]

[1] Institute of Science and Technology Austria (ISTA), Austria
`{mchalupa,tah,esarac}@ista.ac.at`
[2] Slovak University of Technology in Bratislava, Slovak Republic
`nicolas.mazzocchi@stuba.sk`

Abstract. Quantitative automata model beyond-boolean aspects of systems: every execution is mapped to a real number by incorporating weighted transitions and value functions that generalize acceptance conditions of boolean ω-automata. Despite the theoretical advances in systems analysis through quantitative automata, the first comprehensive software tool for quantitative automata (Quantitative Automata Kit, or QuAK) was developed only recently. QuAK implements algorithms for solving standard decision problems, e.g., emptiness and universality, as well as constructions for safety and liveness of quantitative automata. We present the architecture of QuAK, which reflects that all of these problems reduce to either checking inclusion between two quantitative automata or computing the highest value achievable by an automaton—its so-called top value. We improve QuAK by extending these two algorithms with an option to return, alongside their results, an ultimately periodic word witnessing the algorithm's output, as well as implementing a new safety-liveness decomposition algorithm that can handle nondeterministic automata, making QuAK more informative and capable.

Keywords: quantitative automata · automata-based analysis · quantitative safety · quantitative liveness

1 Introduction

Traditional system behavior analysis categorizes system behaviors as correct or incorrect. However, modern systems require more nuanced approaches to address performance and robustness criteria. Quantitative automata generalize boolean ω-automata by adding rational-valued weights to their transitions and using value functions (instead of acceptance conditions) that accumulate infinite weight sequences into single values. Common value functions include Inf, Sup, LimInf, and LimSup (respectively generalizing safety, reachability, co-Büchi and Büchi acceptance conditions), as well as DSum (discounted sum), LimInfAvg, and LimSupAvg (limit average a.k.a. mean payoff).

Decision problems for boolean automata extend naturally to quantitative automata. For example, a quantitative automaton $\mathcal{A}$ is nonempty with respect to a rational number v iff $\mathcal{A}$ maps some infinite word w to a value at least v [13]. These problems are closely related to game theory [21] and enable reasoning about quantitative system aspects. Although quantitative automata have been extensively studied from a theoretical perspective [16,5,8,34,35,4,26,24,25,7] and these works could significantly impact their practical verification, until recently, there was no general software tool for their analysis.

Quantitative Automata Kit (QuAK) [11] is the first general tool for quantitative automata analysis. It currently supports several automaton types (Inf, Sup, LimInf, LimSup, LimInfAvg, LimSupAvg) and provides decision procedures for fundamental problems such as emptiness, universality, inclusion, and safety.

We present an improved version of QuAK: (i) the safety-liveness decompositions are extended to handle nondeterministic automata, and (ii) the inclusion and top value algorithms are extended with capabilities to return a witness—an ultimately periodic word explaining the algorithm's output. For checking inclusion, i.e., whether $\mathcal{A}(w) \leq \mathcal{B}(w)$ for all words w, the witness $\hat{w}$ is a word such that $\mathcal{A}(\hat{w}) > \mathcal{B}(\hat{w})$. For computing top value, i.e., $\top_{\mathcal{A}} = \sup_{w \in \Sigma^{\omega}} \mathcal{A}(w)$, the witness $\widehat{w}$ is a word such that $\mathcal{A}(\hat{w}) = \top_{\mathcal{A}}$. Since all the other procedures are reduced to either inclusion checking or top value computation, these extensions significantly improve QuAK's informativeness for analyzing quantitative automata.

Several approaches extend system modeling beyond boolean aspects. One uses multi-valued truth domains [9,15], while another relies on weighted automata [36], where numerical weights are assigned to transitions and accumulated via semiring operations. Tools such as Vaucanson [32], Vcsn [18], and Awali [31] support weighted automata analysis. Other approaches address digital-analog interactions [2,1,22], with tools like UPPAAL [30] and HyTech [23]. Signal temporal logic [33] allows reasoning about specification satisfaction degrees, as implemented in Breach [19], S-TaLiRo [3], and RTAMT [37]. Probabilistic verification handles uncertainties, as implemented in PRISM [29] and STORM [17].

2   Quantitative Automata

Let Σ be a finite alphabet of letters. We denote by Σ^* (resp. Σ^{ω}) the set of all finite (resp. infinite) words over Σ. For $w \in \Sigma^{\omega}$ and $u \in \Sigma^*$, we write $u \prec w$ when u is a prefix of w. A *value domain* $\mathbb{D}$ is a nontrivial complete lattice. A *quantitative property* is a total function $\Phi : \Sigma^{\omega} \to \mathbb{D}$.

We study quantitative automata, which define a subset of quantitative properties on totally-ordered value domains. Formally, a *nondeterministic quantitative automaton* (or simply an *automaton*) is a tuple $\mathcal{A} = (\Sigma, Q, s, \delta)$ where Σ is a finite alphabet, Q is a finite nonempty set of states, $s \in Q$ is the initial state, and $\delta : Q \times \Sigma \to 2^{\mathbb{Q} \times Q}$ is a finite transition function over weight-state pairs [13]. A *transition* is a tuple $(q, \sigma, x, q') \in Q \times \Sigma \times \mathbb{Q} \times Q$ such that $(x, q') \in \delta(q, \sigma)$, denoted $q \xrightarrow{\sigma:x} q'$. The weight of a transition $t = (q, \sigma, x, q')$ is denoted by $\gamma(t) = x$. An automaton $\mathcal{A}$ is *deterministic* when $|\delta(q, a)| = 1$ for every $q \in Q$ and $a \in \Sigma$,
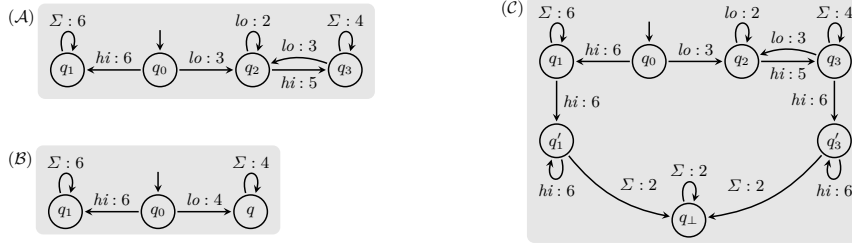
Fig. 1: A nondeterministic LimInfAvg-automaton $\mathcal{A}$ over the alphabet $\Sigma = \{hi, lo\}$, modeling the power consumption of a device where starting with high-power mode is not reversible, its safety closure $\mathcal{B}$, and its liveness component $\mathcal{C}$ in a corresponding decomposition [6].

and it is *total* (a.k.a. *complete*) when $|\delta(q,a)| \geq 1$ for every $q \in Q$ and $a \in \Sigma$. We require quantitative automata to be total.

A run of $\mathcal{A}$ on an infinite word $w = a_0 a_1 \ldots$ is an infinite sequence $\rho = q_0 \xrightarrow{a_0:x_0} q_1 \xrightarrow{a_1:x_1} q_2 \ldots$ of transitions where $q_0 = s$ and $(x_i, q_{i+1}) \in \delta(q_i, a_i)$ for each integer $i \geq 0$. Since each transition has a weight, a run $\rho = t_0 t_1 \ldots$ produces an infinite sequence $\gamma(\rho) = \gamma(t_0)\gamma(t_1) \ldots$ of weights. A value function $\mathsf{Val} : \mathbb{Q}^\omega \to \mathbb{R}$ maps infinite weight sequences to real values. A Val-automaton is a quantitative automaton equipped with the value function Val, i.e., where each run ρ is mapped to the value obtained by applying Val to its weight sequence $\gamma(\rho)$. Given a Val-automaton $\mathcal{A}$, the value of an infinite word w is $\mathcal{A}(w) = \sup\{\mathsf{Val}(\gamma(\rho)) \mid \rho \text{ is a run of } \mathcal{A} \text{ on } w\}$. The *top value* of an automaton $\mathcal{A}$ is $\top_{\mathcal{A}} = \sup_{w \in \Sigma^\omega} \mathcal{A}(w)$, and its *bottom value* is $\bot_{\mathcal{A}} = \inf_{w \in \Sigma^\omega} \mathcal{A}(w)$. We consider the below value functions over an infinite sequence $x = x_0 x_1 \ldots$ of rational weights.

- $\mathsf{Inf}(x) = \inf_{n \geq 0} x_n$
- $\mathsf{LimInf}(x) = \liminf_{n \to \infty} x_n$
- $\mathsf{LimInfAvg}(x) = \liminf_{n \to \infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} x_i \right)$

- $\mathsf{Sup}(x) = \sup_{n \geq 0} x_n$
- $\mathsf{LimSup}(x) = \limsup_{n \to \infty} x_n$
- $\mathsf{LimSupAvg}(x) = \limsup_{n \to \infty} \left(\frac{1}{n} \sum_{i=0}^{n-1} x_i \right)$

- For a discount factor $\lambda \in \mathbb{Q} \cap (0,1)$, $\mathsf{DSum}_\lambda(x) = \sum_{i \geq 0} \lambda^i x_i$

The automaton $\mathcal{A}$ in Figure 1 shows a LimInfAvg automaton modeling the long-term average power consumption of a device with two operating modes.

Quantitative Automata Problems We describe the standard decision problems of quantitative automata as well as the problems related to their safety and liveness. The complexity results are summarized in Table 1.

An automaton $\mathcal{A}$ is *nonempty* (resp. *universal*) with respect to a threshold $v \in \mathbb{Q}$ iff $\mathcal{A}(w) \geq v$ for some (resp. all) $w \in \Sigma^\omega$. Nonemptiness (resp. universality) is closely related to computing an automaton's top value (resp. bottom value): $\mathcal{A}$ is nonempty (resp. universal) with respect to $v \in \mathbb{Q}$ iff $\top_{\mathcal{A}} \geq v$ (resp. $\bot_{\mathcal{A}} \geq v$).

	Inf	Sup, LimInf, LimSup	LimInfAvg, LimSupAvg	DSum
Nonemptiness check	PTIME			
Universality check		PSPACE-complete	Undecidable	Open
Inclusion check		PSPACE-complete	Undecidable	Open
Equivalence check		PSPACE-complete	Undecidable	Open
Top value computation	PTIME			
Safety closure construction	$O(1)$	PTIME		$O(1)$
Safety-liveness decomposition	$O(1)$	PTIME		$O(1)$
Safety check	$O(1)$	PSPACE-complete	EXPSPACE; PSPACE-hard	$O(1)$
Liveness check	PSPACE-complete			
Constant-function check	PSPACE-complete			

Table 1: The complexity of performing the operations on the left column with respect to nondeterministic automata with the value function on the top. The decidability results in the top five rows are shown in [28,13] and undecidability in [16,12,27]. All the results in the bottom five rows are shown in [6]. All the operations are computable in PTIME for deterministic automata.

An automaton $\mathcal{A}$ is *included in* (resp. *equivalent to*) an automaton $\mathcal{B}$ iff $\mathcal{A}(w) \leq \mathcal{B}(w)$ (resp. $\mathcal{A}(w) = \mathcal{B}(w)$) for all $w \in \Sigma^\omega$. An automaton $\mathcal{A}$ is *constant* iff there exists $c \in \mathbb{R}$ such that $\mathcal{A}(w) = c$ for all $w \in \Sigma^\omega$. This problem is closely related to safety and liveness of quantitative automata, as we discuss below.

Quantitative safety generalizes the boolean view by considering membership hypotheses in the form of lower bound queries: a property is safe iff every wrong membership hypothesis has a finite witness for the violation. Formally, a quantitative property $\Phi : \Sigma^\omega \to \mathbb{D}$ is *safe* iff for every $w \in \Sigma^\omega$ and $v \in \mathbb{D}$ with $\Phi(w) \not\geq v$, there exists a finite prefix $u \prec w$ such that $\sup_{w' \in \Sigma^\omega} \Phi(uw') \not\geq v$ [25]. Moreover, an automaton $\mathcal{A}$ is safe iff the quantitative property defined by $\mathcal{A}$ is safe. Given a quantitative property $\Phi : \Sigma^\omega \to \mathbb{D}$, its *safety closure* is defined as $SafetyCl(\Phi)(w) = \inf_{u \prec w} \sup_{w' \in \Sigma^\omega} \Phi(uw')$ and is the least safety property that bounds Φ from above [25]. As expected, a property Φ is safe iff $\Phi(w) = SafetyCl(\Phi)(w)$ for all $w \in \Sigma^\omega$, and we can compute the safety closure of an automaton $\mathcal{A}$—the automaton $SafetyCl(\mathcal{A})$ that expresses the safety closure of the property defined by $\mathcal{A}$. While this characterization is useful for some classes of quantitative automata, the equivalence problem is undecidable for LimInfAvg and LimSupAvg automata. For these, the safety problem is still decidable by a reduction to their constant-function problem [7].

Quantitative liveness extends the membership-based view: a quantitative property $\Phi : \Sigma^\omega \to \mathbb{D}$ is live iff for every word (whose value is less than $\top = \sup \mathbb{D}$) there exists a wrong membership hypothesis without a finite witness for the violation. Formally, a quantitative property $\Phi : \Sigma^\omega \to \mathbb{D}$ is *live* iff for all $w \in \Sigma^\omega$, if $\Phi(w) < \top$, then there exists a value $v \in \mathbb{D}$ such that $\Phi(w) \not\geq v$ and for all prefixes $u \prec w$, we have $\sup_{w' \in \Sigma^\omega} \Phi(uw') \geq v$ [25]. Moreover, an automaton $\mathcal{A}$ is live iff the quantitative property defined by $\mathcal{A}$ is live. For the common classes of quantitative automata, deciding liveness reduces to
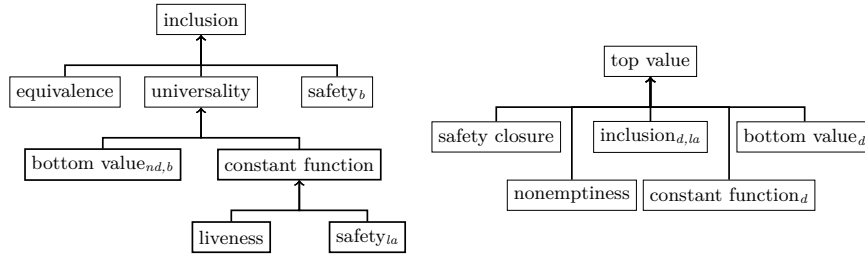
Fig. 2: Reductions of quantitative automata problems in QuAK. The subscript b stands for basic (i.e., $\mathsf{Val} \in \{\mathsf{Inf}, \mathsf{Sup}, \mathsf{LimInf}, \mathsf{LimSup}\}$), la for limit-average (i.e., $\mathsf{Val} \in \{\mathsf{LimInfAvg}, \mathsf{LimSupAvg}\}$), d for deterministic, and nd for nondeterministic. For example, checking safety of limit-average automata (safety$_{la}$) reduces to their constant-function problem, which reduces to universality of LimInf automata.

the constant-function problem: an automaton $\mathcal{A}$ is live iff *SafetyCl*$(\mathcal{A})$ is constant [7]. Just like every boolean property is the intersection of its safety closure and a liveness property, every quantitative property is the pointwise minimum of its safety closure and a liveness property [25]. Recently, it was proved that all the common classes of automata can be decomposed into its safety closure and a liveness property [6]. Consider the automaton $\mathcal{A}$, its safety closure $\mathcal{B}$, and its liveness part $\mathcal{C}$ as defined in Figure 1. In $\mathcal{B}$, each strongly connected component (SCC) of $\mathcal{A}$ is assigned the highest value achievable within the component, representing the greatest among the lower bound hypotheses that cannot be refuted by any finite prefix. The liveness part $\mathcal{C}$ consists of three components: the upper part is a copy of $\mathcal{A}$ (ensuring $\mathcal{C}$ can have runs with the same value as $\mathcal{A}$); the middle part contains a $\top_{\mathcal{A}}$-weighted copy of the highest-valued cycle in each SCC (enabling $\mathcal{C}$ to achieve high-valued runs when $\mathcal{A}$ and $\mathcal{B}$ agree); and the lower part includes a sink state looping with the lowest weight of $\mathcal{A}$ (allowing $\mathcal{C}$ to "escape" the middle part and realize a value using the upper part).

3 QuAK Overview and Usage

QuAK is written in C++ using the standard library, and the source code is available online [10]. It can be used both as a C++ library and a stand-alone tool through the command-line interface – see our project repository for instructions. For Inf, Sup, LimInf, LimSup, $\mathsf{LimInfAvg}$, and $\mathsf{LimSupAvg}$ automata, QuAK implements the operations listed in Table 1 and a monitoring procedure where the monitor maintains the maximal and minimal possible values or a running average. The problems handled in QuAK reduce to either the inclusion problem or the top value computation, as shown in Figure 2. For the details of these reductions, we refer the reader to [13,6,11].

We improve over QuAK's initial version [11] in two ways. First, a safety-liveness decomposition for nondeterministic automata is implemented, following a new result that provides a decomposition for nondeterministic automata with prefix-independent value functions, namely, LimInf, LimSup, $\mathsf{LimInfAvg}$, and

```
Word* witEmpt, witSafe;
Automaton* A = new Automaton("A.txt");
Automaton* B = safetyClosure(A, LimInfAvg);
Automaton* C = livenessComponent(A, LimInfAvg);
bool flagEmpt = A->isNonEmpty(LimInfAvg, 5, &witEmpt);
bool flagSafe = A->isSafe(LimInfAvg, &witSafe);
```

Fig. 3: An example usage of QuAK as a C++ library. The functions `isNonEmpty` and `isSafe` now take an additional (optional) parameter for storing the stem and the period of the ultimately periodic word witnessing the algorithms' outputs.

LimSupAvg automata [6, Thm. 9.11]. Second, the inclusion-checking and the top-value computation algorithms are extended with an option to return a witness (for negative instance of inclusion and all instances of top value). Since all other problems reduce to these two (see Figure 2), the ability to generate witnesses makes QuAK more informative for analyzing quantitative automata. Inclusion checking, a central component of QuAK, is implemented using an antichain-based algorithm, extending FORKLIFT for Büchi automata [20], with details and performance benefits discussed in [11]. As this algorithm systematically searches for counterexamples, it inherently supports witness construction for negative instances. Top value computation is based on graph-theoretic algorithms [13], with witness generation achieved via backtracking pointers. These improvements, along with other features of QuAK, were validated through unit testing, random testing, and cross-validation with existing implementations.

QuAK reads and constructs automata from text files. Each automaton is represented as a list of transitions of the format `a : v, q -> p` which encodes a transition from state q to state p with letter a and weight v. The initial state of the input automaton is the source state of the first transition in its text file.

Recall the nondeterministic limit-average automaton $\mathcal{A}$ and its safety-liveness decomposition from Figure 1. The first three lines of the code snippet in Figure 3 construct the automata $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ as presented in Figure 1. The nonemptiness check returns `false`, and `witEmpt` points to an array storing $u = hi$ and $v = hi$ as $\top_{\mathcal{A}} = \mathcal{A}(hi^{\omega}) = 6$. Similarly, the safety check returns `false` and `witSafe` points to an array storing $u = v = lo$ as $\mathcal{B}(lo^{\omega}) = 4$ and $\mathcal{A}(lo^{\omega}) = 2$.

4   Conclusion

We presented an improved version of QuAK, our software tool for automating quantitative automata analysis, which extends the functionality introduced in [11]. Future work aims to improve the tool's scalability and applicability while exploring more efficient verification methods. One promising avenue is the development of symbolic approaches to efficiently manage large state spaces. Another key direction involves extending the tool to support additional formalisms, such as various types of discounted-sum automata [4], mean-payoff automaton expressions [12], and nested quantitative automata [14]. In parallel with these efforts, developing novel verification methods specifically tailored to the safety fragments of expressive quantitative formalisms presents an exciting research direction.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994). https://doi.org/10.1016/0304-3975(94)90010-8
2. Alur, R., Henzinger, T.A.: Real-time logics: Complexity and expressiveness. Inf. Comput. **104**(1), 35–77 (1993). https://doi.org/10.1006/INCO.1993.1025
3. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-taliro: A tool for temporal logic falsification for hybrid systems. In: Abdulla, P.A., Leino, K.R.M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6605, pp. 254–257. Springer (2011). https://doi.org/10.1007/978-3-642-19835-9_21
4. Boker, U.: Discounted-sum automata with real-valued discount factors. In: Sobocinski, P., Lago, U.D., Esparza, J. (eds.) Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2024, Tallinn, Estonia, July 8-11, 2024. pp. 15:1–15:14. ACM (2024). https://doi.org/10.1145/3661814.3662090
5. Boker, U., Henzinger, T.A.: Exact and approximate determinization of discounted-sum automata. Log. Methods Comput. Sci. **10**(1) (2014). https://doi.org/10.2168/LMCS-10(1:10)2014
6. Boker, U., Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: Safety and liveness of quantitative properties and automata (2024), https://arxiv.org/abs/2307.06016
7. Boker, U., Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: Safety and liveness of quantitative automata. In: Pérez, G.A., Raskin, J. (eds.) 34th International Conference on Concurrency Theory, CONCUR 2023, September 18-23, 2023, Antwerp, Belgium. LIPIcs, vol. 279, pp. 17:1–17:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2023). https://doi.org/10.4230/LIPICS.CONCUR.2023.17
8. Boker, U., Henzinger, T.A., Otop, J.: The target discounted-sum problem. In: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015. pp. 750–761. IEEE Computer Society (2015). https://doi.org/10.1109/LICS.2015.74
9. Bruns, G., Godefroid, P.: Model checking partial state spaces with 3-valued temporal logics. In: Halbwachs, N., Peled, D.A. (eds.) Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1633, pp. 274–287. Springer (1999). https://doi.org/10.1007/3-540-48683-6_25
10. Chalupa, M., Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: QuAK - Quantitative Automata Kit (2024), maintained at https://github.com/ista-vamos/QuAK
11. Chalupa, M., Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: Quak: Quantitative automata kit (2024), https://arxiv.org/abs/2409.03569
12. Chatterjee, K., Doyen, L., Edelsbrunner, H., Henzinger, T.A., Rannou, P.: Mean-payoff automaton expressions. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010 - Concurrency Theory, 21th International Conference, CONCUR 2010, Paris, France, August 31-September 3, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6269, pp. 269–283. Springer (2010). https://doi.org/10.1007/978-3-642-15375-4_19
13. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. **11**(4), 23:1–23:38 (2010). https://doi.org/10.1145/1805950.1805953

14. Chatterjee, K., Henzinger, T.A., Otop, J.: Nested weighted automata. ACM Trans. Comput. Log. **18**(4), 31:1–31:44 (2017). https://doi.org/10.1145/3152769

15. Chechik, M., Gurfinkel, A., Devereux, B.: chi-chek: A multi-valued model-checker. In: Brinksma, E., Larsen, K.G. (eds.) Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2404, pp. 505–509. Springer (2002). https://doi.org/10.1007/3-540-45657-0_41

16. Degorre, A., Doyen, L., Gentilini, R., Raskin, J., Torunczyk, S.: Energy and mean-payoff games with imperfect information. In: Dawar, A., Veith, H. (eds.) Computer Science Logic, 24th International Workshop, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6247, pp. 260–274. Springer (2010). https://doi.org/10.1007/978-3-642-15205-4_22

17. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: A modern probabilistic model checker. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10427, pp. 592–600. Springer (2017). https://doi.org/10.1007/978-3-319-63390-9_31, https://doi.org/10.1007/978-3-319-63390-9_31

18. Demaille, A., Duret-Lutz, A., Lombardy, S., Sakarovitch, J.: Implementation concepts in vaucanson 2. In: Konstantinidis, S. (ed.) Implementation and Application of Automata - 18th International Conference, CIAA 2013, Halifax, NS, Canada, July 16-19, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7982, pp. 122–133. Springer (2013). https://doi.org/10.1007/978-3-642-39274-0_12

19. Donzé, A.: Breach, A toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P.B. (eds.) Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6174, pp. 167–170. Springer (2010). https://doi.org/10.1007/978-3-642-14295-6_17

20. Doveri, K., Ganty, P., Mazzocchi, N.: Forq-based language inclusion formal testing. In: Shoham, S., Vizel, Y. (eds.) Computer Aided Verification - 34th International Conference, CAV 2022, Haifa, Israel, August 7-10, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol. 13372, pp. 109–129. Springer (2022). https://doi.org/10.1007/978-3-031-13188-2_6

21. Fijalkow, N., Bertrand, N., Bouyer-Decitre, P., Brenguier, R., Carayol, A., Fearnley, J., Gimbert, H., Horn, F., Ibsen-Jensen, R., Markey, N., Monmege, B., Novotný, P., Randour, M., Sankur, O., Schmitz, S., Serre, O., Skomra, M.: Games on graphs. CoRR **abs/2305.10546** (2023). https://doi.org/10.48550/ARXIV.2305.10546, https://doi.org/10.48550/arXiv.2305.10546

22. Henzinger, T.A.: The theory of hybrid automata. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996. pp. 278–292. IEEE Computer Society (1996). https://doi.org/10.1109/LICS.1996.561342

23. Henzinger, T.A., Ho, P.: HYTECH: the cornell hybrid technology tool. In: Antsaklis, P.J., Kohn, W., Nerode, A., Sastry, S. (eds.) Hybrid Systems II, Proceedings of the Third International Workshop on Hybrid Systems, Ithaca, NY, USA, October 1994. Lecture Notes in Computer Science, vol. 999, pp. 265–293. Springer (1994). https://doi.org/10.1007/3-540-60472-3_14

24. Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: Abstract monitors for quantitative specifications. In: Dang, T., Stolz, V. (eds.) Runtime Verification - 22nd Inter-

national Conference, RV 2022, Tbilisi, Georgia, September 28-30, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13498, pp. 200–220. Springer (2022). https://doi.org/10.1007/978-3-031-17196-3_11

25. Henzinger, T.A., Mazzocchi, N., Saraç, N.E.: Quantitative safety and liveness. In: Kupferman, O., Sobocinski, P. (eds.) Foundations of Software Science and Computation Structures - 26th International Conference, FoSSaCS 2023, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2023, Paris, France, April 22-27, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13992, pp. 349–370. Springer (2023). https://doi.org/10.1007/978-3-031-30829-1_17

26. Henzinger, T.A., Saraç, N.E.: Quantitative and approximate monitoring. In: 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021. pp. 1–14. IEEE (2021). https://doi.org/10.1109/LICS52264.2021.9470547

27. Hunter, P., Pauly, A., Pérez, G.A., Raskin, J.: Mean-payoff games with partial observation. Theor. Comput. Sci. **735**, 82–110 (2018). https://doi.org/10.1016/J.TCS.2017.03.038, https://doi.org/10.1016/j.tcs.2017.03.038

28. Kupferman, O., Lustig, Y.: Lattice automata. In: Cook, B., Podelski, A. (eds.) Verification, Model Checking, and Abstract Interpretation, 8th International Conference, VMCAI 2007, Nice, France, January 14-16, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4349, pp. 199–213. Springer (2007). https://doi.org/10.1007/978-3-540-69738-1_14

29. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J.T., Harder, U. (eds.) Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, London, UK, April 14-17, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2324, pp. 200–204. Springer (2002). https://doi.org/10.1007/3-540-46029-2_13

30. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. Int. J. Softw. Tools Technol. Transf. **1**(1-2), 134–152 (1997). https://doi.org/10.1007/S100090050010

31. Lombardy, S., Marsault, V., Sakarovitch, J.: Awali, a library for weighted automata and transducers (version 2.3) (2022), software available at http://vaucanson-project.org/Awali/2.3/

32. Lombardy, S., Poss, R., Régis-Gianas, Y., Sakarovitch, J.: Introducing VAUCAN-SON. In: Ibarra, O.H., Dang, Z. (eds.) Implementation and Application of Automata, 8th International Conference, CIAA 2003, Santa Barbara, California, USA, July 16-18, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2759, pp. 96–107. Springer (2003). https://doi.org/10.1007/3-540-45089-0_10

33. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Lakhnech, Y., Yovine, S. (eds.) Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems, FORMATS 2004 and Formal Techniques in Real-Time and Fault-Tolerant Systems, FTRTFT 2004, Grenoble, France, September 22-24, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3253, pp. 152–166. Springer (2004). https://doi.org/10.1007/978-3-540-30206-3_12

34. Michaliszyn, J., Otop, J.: Approximate learning of limit-average automata. In: Fokkink, W.J., van Glabbeek, R. (eds.) 30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands. LIPIcs, vol. 140, pp. 17:1–17:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). https://doi.org/10.4230/LIPICS.CONCUR.2019.17

35. Michaliszyn, J., Otop, J.: Minimization of limit-average automata. In: Zhou, Z. (ed.) Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021. pp. 2819–2825. ijcai.org (2021). https://doi.org/10.24963/IJCAI.2021/388
36. Schützenberger, M.P.: On the definition of a family of automata. Inf. Control. **4**(2-3), 245–270 (1961). https://doi.org/10.1016/S0019-9958(61)80020-X
37. Yamaguchi, T., Hoxha, B., Nickovic, D.: RTAMT - runtime robustness monitors with application to CPS and robotics. Int. J. Softw. Tools Technol. Transf. **26**(1), 79–99 (2024). https://doi.org/10.1007/S10009-023-00720-3, https://doi.org/10.1007/s10009-023-00720-3