Between Deterministic and Nondeterministic Quantitative Automata

Udi Boker
Reichman University, Herzliya, Israel

Abstract

There is a challenging trade-off between deterministic and nondeterministic automata, where the former suit various applications better, however at the cost of being exponentially larger or even less expressive. This gave birth to many notions in between determinism and nondeterminism, aiming at enjoying, sometimes, the best of both worlds. Some of the notions are yes/no ones, for example initial nondeterminism (restricting nondeterminism to allowing several initial states), and some provide a measure of nondeterminism, for example the ambiguity level.

We analyze the possible generalization of such notions from Boolean to quantitative automata, and suggest that it depends on the following key characteristics of the considered notion N– whether it is syntactic or semantic, and if semantic, whether it is word-based or language-based.

A syntactic notion, such as initial nondeterminism, applies as is to a quantitative automaton $\mathcal{A}$, namely $\mathsf{N}(\mathcal{A})$. A word-based semantic notion, such as unambiguity, applies as is to a Boolean automaton t-$\mathcal{A}$ that is derived from $\mathcal{A}$ by accompanying it with some threshold value $t \in \mathbb{R}$, namely $\mathsf{N}(t\text{-}\mathcal{A})$. A language-based notion, such as history determinism, also applies as is to t-$\mathcal{A}$, while in addition, it naturally generalizes into two different notions with respect to $\mathcal{A}$ itself, by either: i) taking the supremum of $\mathsf{N}(t\text{-}\mathcal{A})$ over all thresholds t, denoted by $\mathsf{Threshold\text{-}N}(\mathcal{A})$; or ii) generalizing the basis of the notion from a language to a function, denoted simply by $\mathsf{N}(\mathcal{A})$. While in general $\mathsf{N}(\mathcal{A}) \Rightarrow \mathsf{Threshold\text{-}N}(\mathcal{A}) \Rightarrow \mathsf{N}(t\text{-}\mathcal{A})$, we have for some notions $\mathsf{N}(\mathcal{A}) \equiv \mathsf{Threshold\text{-}N}(\mathcal{A})$, and for some not. (For measure notions, $\Rightarrow$ stands for $\geq$ with respect to the nondeterminism level.)

We classify numerous notions known in the Boolean setting according to their characterization above, generalize them to the quantitative setting and look into relations between them. The generalized notions open new research directions with respect to quantitative automata, and provide insights on the original notions with respect to Boolean automata.

2012 ACM Subject Classification Theory of computation → Logic and verification

Keywords and phrases Quantitative Automata, Measure of Nondeterminism, Determinism

Digital Object Identifier 10.4230/LIPIcs.CSL.2022.1

Category Invited Talk

Acknowledgements We thank Karoliina Lehtinen for stimulating discussions on prelimiary versions of the paper, and specifically for suggesting the all-thresholds generalization of the notions.

1 Introduction

The tension between determinism and nondeterminism is at the core of computer science, most notably evident in the P vs. NP epic. In automata theory, determinism often suits certain applications, such as synthesis, better and allows for efficient decision algorithms, while nondeterminism allows for exponential succinctness, and sometimes higher expressiveness.

As a result, there is intensive research in automata theory on notions in between determinism and nondeterminism, starting in the 1970s [41, 40], and actively continuing until these days. (See, for example, the recent breakthrough on a superpolynomial lower bound for the state complexity involved in complementing unambiguous finite automata [50, 30]). Numerous such notions have developed over the years, many of which we classify in Section 3.

Some of them are yes/no ones, for example *determinizability by pruning*, and some provide a measure of nondeterminism, for example *deviation*. Deterministic automata have in the former case value "yes", and in the latter case value 0 or 1, depending on the notion.

We shall discuss the notions with respect to automata on finite and infinite words, while they often apply also to automata over richer input structures, such as trees and graphs. Further, while the notions are usually considered with respect to nondeterministic automata, they often also apply to alternating automata, setting limitations on both nondeterministic and universal transitions. (See, for example, [20, 13, 39])

Over the past decade, there is a growing interest in quantitative verification and synthesis (e.g., [2, 9, 24, 18, 25, 6, 15, 3, 36, 27]), and in *quantitative*[1] *automata* [17] as their underlying computational model, for addressing the requirements of contemporary systems – the traditional Boolean perspective of verification and synthesis falls short of many aspects of contemporary systems, concerning performance, robustness, and resource-constraints, according to which one system is often preferred over another, even though they are both correct, since one is, for example, faster than the other, or, if they are both incorrect, one misbehaves less frequently than the other.

Since quantitative automata keep the *choice* meaning of nondeterminism, notions of limited nondeterminism can be naturally generalized from Boolean to quantitative automata, which is indeed starting to take place in recent years [19, 4, 36, 26, 14]. In [14], the current author and Karoliina Lehtinen have considered three such notions – *determinizability by pruning*, *history determinism*, and *good-for-gameness* – and suggested to look into three different generalizations of each of them to the setting of quantitative automata.

In this paper, we look into the question of how to generalize to the quantitative setting the numerous limited-nondeterminism notions that exist for Boolean automata. We suggest that it depends on the following key characteristics of the considered notion N – whether it is syntactic or semantic, and if semantic, whether it is word-based or language-based.

A syntactic notion (Sections 3.1 and 4.1) applies as is to a quantitative automaton $\mathcal{A}$, namely $\mathsf{N}(\mathcal{A})$ – it only relates to the automaton's structure, indifferent to whether the semantics of the automaton is based on a Boolean acceptance condition or a value function. Among these notions are initial nondeterminism, tree width, structural ambiguity, and determinism in the limit.

A word-based semantic notion (Sections 3.2 and 4.2) applies as is to a Boolean automaton $t\text{-}\mathcal{A}$ that is derived from the quantitative automaton $\mathcal{A}$ by accompanying it with some threshold value $t \in \mathbb{R}$, namely $\mathsf{N}(t\text{-}\mathcal{A})$. Among these notions are ambiguity, branching, guessing, amount of nondeterminism, trace, and deviation.

A language-based notion (Sections 3.3 and 4.3) allows for the three natural generalizations suggested in [14]: i) applying it as is to $t\text{-}\mathcal{A}$; ii) applying it to $\mathcal{A}$ itself by taking the supremum of $\mathsf{N}(t\text{-}\mathcal{A})$ over all thresholds t, denoted by Threshold-$\mathsf{N}(\mathcal{A})$; and iii) applying it to $\mathcal{A}$ itself by generalizing the language basis of the notion to a function basis, denoted simply by $\mathsf{N}(\mathcal{A})$.

Interestingly, while in general $\mathsf{N}(\mathcal{A}) \Rightarrow$ Threshold-$\mathsf{N}(\mathcal{A}) \Rightarrow \mathsf{N}(t\text{-}\mathcal{A})$, it might be that $\mathsf{N}(\mathcal{A}) \not\Leftarrow$ Threshold-$\mathsf{N}(\mathcal{A})$, even though Threshold-$\mathsf{N}(\mathcal{A})$ is defined with respect to *all* thresholds $t \in \mathbb{R}$, as is the case with history determinism (see Section 4.3.1). (For measure notions, $\Rightarrow$ stands for $\geq$ with respect to the nondeterminism level.)

[1] Quantitative automata generally suit verification needs better than weighted automata, since they keep the *choice* meaning of nondeterminism and realize functions to the totally-ordered domain of real numbers. (See Section 2 for the definition of nondeterministic quantitative automata and [8] for an analysis of the differences between quantitative and weighted automata.)

Among the language-based notions are residuality, history determinism, good-for-gameness, good-for-MDPness, determinizability by pruning, and k-tokens games.

We provide in Section 3 partially-formal definitions of numerous notions known in the Boolean setting, classified according to the above characterization. In Section 4 we analyze their generalization to the quantitative setting, and in Section 5 we look into their interrelations, observing that they refine the interrelations known for the Boolean setting.

The notion generalization provides novel definitions of automata classes, and opens up many research directions on limited nondeterminism in each of the various quantitative automata types (limit-average automata, discounted-sum automata, etc.): What advantages of determinism are preserved (complexity of decision problems, suitability for various applications, etc.)? What advantages of nondeterminism are preserved (expressiveness, succinctness, etc.)? What are the closure properties of the resulting class of automata? What is the complexity of its membership check? etc.

Furthermore, the generalized notions provide insights on the original notions with respect to Boolean automata. For example, the refined notion interrelations, whereby history determinism and good-for-gameness are not equivalent, suggests that the two notions, which are known to be equivalent with respect to ω-regular automata, might not be equivalent with respect to other types of Boolean automata. Another example is the implication between history determinism and pre-residuality, which is known to hold for ω-regular automata, but turns out to depend on the suffix monotonicity of the acceptance condition.

2 Preliminaries

Words. An *alphabet* Σ is a finite nonempty set of letters. A finite (resp. infinite) *word* $u = \sigma_0 \dots \sigma_k \in \Sigma^*$ (resp. $w = \sigma_0\sigma_1 \dots \in \Sigma^\omega$) is a finite (resp. infinite) sequence of letters from Σ. A *language* is a set of words.

Boolean automata. Many notions of limited nondeterminism apply to various types of Boolean automata, namely to automata that define a *language* by accepting or rejecting words, such as finite automata on finite words (NFAs), pushdown automata, ω-regular automata, two-way automata, and more.

In our partially-formal presentation of the notions, we relate to a nondeterministic Boolean automaton as a structure $\mathcal{A} = (\Sigma, Q, I, \delta, \alpha)$, where Σ is an alphabet; Q is a finite nonempty set of states; $I \subseteq Q$ is a set of initial states; $\delta\colon Q \times \Sigma \to 2^Q$ is a transition function, and α is an acceptance condition. This basic structure can be extended for various automata types.

We denote by $\mathcal{A}^q$ the automaton that is derived from $\mathcal{A}$ by changing its initial set of states to be $\{q\}$. An automaton is deterministic if I is a singleton, and for every state q and letter σ, we have $|\delta(q, \sigma)| \leq 1$.

A run of the automaton is a path over its states, starting with an initial state and continuing along the transition function. A run is accepting if it satisfies the acceptance condition; a word is accepted by $\mathcal{A}$ if there is an accepting run on it; and the language that $\mathcal{A}$ recognizes, denoted by $L(\mathcal{A})$, is the set of words that it accepts. Two automata $\mathcal{A}$ and $\mathcal{A}'$ are equivalent, denoted by $\mathcal{A} \equiv \mathcal{A}'$, if they recognize the same language.

In the case of NFAs, which define regular languages, the acceptance condition α is a subset of Q, and a run is accepting if it ends in a state in α. In the case of Büchi automata, which define ω-regular languages, α is also a subset of Q, and a run is accepting if it visits a state in α infinitely often. (More on acceptance conditions of ω-regular automata can be found, for example, in [7].)

Quantitative Automata. A *nondeterministic quantitative*[2] *automaton* on words is a tuple $\mathcal{A} = (\Sigma, Q, I, \delta)$, where Σ is an alphabet; Q is a finite nonempty set of states; $I \subseteq Q$ is a set of initial states; and $\delta\colon Q \times \Sigma \to 2^{(\mathbb{R}\times Q)}$ is a transition function over weight-state pairs.

A *transition* is a tuple $(q, \sigma, x, q') \in Q\times\Sigma\times\mathbb{R}\times Q$, also written $q \xrightarrow{\sigma:x} q'$. (Note that there might be several transitions with different weights over the same letter between the same pair of states[3].) We write $\gamma(t) = x$ for the weight of a transition $t = (q, \sigma, x, q')$.

We require that the automaton $\mathcal{A}$ is *complete*, namely that for every state $q \in Q$ and letter $\sigma \in \Sigma$, there is at least one state q' and a transition $q \xrightarrow{\sigma:x} q'$.

A *run* of the automaton on a word w is a sequence $\rho = q_0 \xrightarrow{w[0]:x_0} q_1 \xrightarrow{w[1]:x_1} q_2 \dots$ of transitions where $q_0 \in I$ and $q_{i+1} \in \delta(q_i, w[i])$. As each transition t_i carries a weight $\gamma(t_i) \in \mathbb{R}$, the sequence ρ provides a weight sequence $\gamma(\pi) = \gamma(t_0)\gamma(t_1)\gamma(t_2)\dots$.

A Val automaton (for example a Sum automaton) is one equipped with a *value function* $\mathsf{Val} : \mathbb{R}^* \to \mathbb{R}$ or $\mathsf{Val} : \mathbb{R}^\omega \to \mathbb{R}$, which assigns real values to runs of $\mathcal{A}$. The value of a run π is $\mathsf{Val}(\gamma(\pi))$. The value of $\mathcal{A}$ on a word w is the supremum of $\mathsf{Val}(\pi)$ over all runs π of $\mathcal{A}$ on w.

Automata $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent*, denoted by $\mathcal{A} \equiv \mathcal{A}'$, if they realize the same function.

Value functions. We list below some of the common value functions used for quantitative automata on finite/infinite words, defined over sequences of real weights.

For finite sequences $v = v_0 v_1 \dots v_{n-1}$:

- $\mathsf{Sum}(v) = \sum_{i=0}^{n-1} v_i$
- $\mathsf{Avg}(v) = \frac{1}{n}\sum_{i=0}^{n-1} v_i$
- $\mathsf{Prod}(v) = \prod_{i=0}^{n-1} v_i$

For finite and infinite sequences $v = v_0 v_1 \dots$:

- $\mathsf{Inf}(v) = \inf\{v_n \mid n \geq 0\}$
- $\mathsf{Sup}(v) = \sup\{v_n \mid n \geq 0\}$
- For a discount factor $\lambda \in \mathbb{Q} \cap (0,1)$, $\lambda\text{-}\mathsf{DSum}(v) = \sum_{i\geq 0} \lambda^i v_i$

For infinite sequences $v = v_0 v_1 \dots$:

- $\mathsf{LimInf}(v) = \lim_{n\to\infty} \inf\{v_i \mid i \geq n\}$
- $\mathsf{LimSup}(v) = \lim_{n\to\infty} \sup\{v_i \mid i \geq n\}$
- $\mathsf{LimInfAvg}(v) = \mathsf{LimInf}(\mathsf{Avg}(v_0), \mathsf{Avg}(v_0, v_1), \mathsf{Avg}(v_0, v_1, v_2), \dots)$
- $\mathsf{LimSupAvg}(v) = \mathsf{LimSup}(\mathsf{Avg}(v_0), \mathsf{Avg}(v_0, v_1), \mathsf{Avg}(v_0, v_1, v_2), \dots)$

($\mathsf{LimInfAvg}$ and $\mathsf{LimSupAvg}$ are also called $\underline{\mathsf{MeanPayoff}}$ and $\overline{\mathsf{MeanPayoff}}$.)

3 Notion Classification

We consider notions that limit the nondeterminism of a Boolean automaton $\mathcal{A}$. The notions may either be yes/no ones (always having "yes" for deterministic automata) or provide a measure of nondeterminism, having a fixed value $k \in \mathbb{N}$, dependency on $\mathcal{A}$'s size (e.g., polynomial), *finite*, or *infinite* value (and always have value 0 or 1 for deterministic automata).

[2] We speak of "quantitative" rather than "weighted" automata, following the distinction made in [8] between the two.

[3] This extra flexibility of "parallel" transitions with different weights is often omitted (e.g., in [16]) since it is redundant for some value functions while important for others.

We classify the notions into syntactic and semantic ones, then classify the semantic ones into word-based and language-based, and finally further classify the language-based ones.

Due to the extent of the different notions, we define them only partially formally.

3.1 Syntactic Notions

These notions only relate to the structure of $\mathcal{A}$, unrelated to the language it recognizes.

- Initially nondeterministic($\mathcal{A}$) holds if $\mathcal{A}$ is deterministic, except for possibly having several initial states. Considering NFAs, such an automaton is called "deterministic finite automaton with multiple initial states" (MDFA). See [37, 33]. (Such a Büchi automaton is called in [52, Section 2] "semi deterministic", however this name is currently more in use for a "deterministic in the limit" automaton, as defined below.)
- Tree width$(\mathcal{A}, w)$ = number of runs of $\mathcal{A}$ on a word w; Tree width$(\mathcal{A})$ = sup{tree width$(\mathcal{A}, w)$} over all words w. It is also called leaf size. See [35].
- Structural ambiguity$(\mathcal{A}) = k$ if there is a single initial state q_0 in $\mathcal{A}$, and for every word w and state q of $\mathcal{A}$, there are at most k runs of $\mathcal{A}$ over w that end in q. When $k = 1$, $\mathcal{A}$ is structurally unambiguous. See [45].
- General structural ambiguity$(\mathcal{A}) = k$ if for every word w and states q and q' of $\mathcal{A}$, there are at most k paths from q to q' over w. When $k = 1$, $\mathcal{A}$ is generally structurally unambiguous. See [45].

We also consider in this class notions that do not relate to $\mathcal{A}$'s language, but do relate to its state labeling or transition labeling.

- Deterministic in the limit$(\mathcal{A})$ holds if $\mathcal{A}^q$ is deterministic for every accepting state q of $\mathcal{A}$ (considering only the transitions and states reachable from q). See [23, page 34]. It is also called limit determinism and semi determinism.

3.2 Semantic Word-Based Notions

A notion N in this class is derived from a word measure $\mathsf{N}(\mathcal{A}, w)$, by setting its value for an automaton $\mathcal{A}$ to be $\mathsf{N}(\mathcal{A}) = \sup\{\mathsf{N}(\mathcal{A}, w) \mid w \in L(\mathcal{A})\}$, namely the supremum over all words that the automaton *accepts*. (When it is clear from the context what $\mathcal{A}$ is, we write $\mathsf{N}(w)$ instead of $\mathsf{N}(\mathcal{A}, w)$.)

The most common such notion is of *ambiguity*, defined by

- Ambiguity(w) = the number of accepting runs of $\mathcal{A}$ on w. When ambiguity$(\mathcal{A}) = 1$, $\mathcal{A}$ is said to be unambiguous. See a survey in [21].

There are various additional notions in the class, whose measure is based on *aggregating the nondeterminism along runs*. They differ by the aggregation function, and by whether they consider all runs, the best run, or the worst run on the word w.

- Amount of nondeterminism(w) = the minimal number of nondeterministic choices that occur in an accepting run of $\mathcal{A}$ on w. See [42].
- Branching(w) = the minimum branching of an accepting run of $\mathcal{A}$ on w, where branching of a run r is the product of the nodes' degrees along r in the computation tree of $\mathcal{A}$ on w. See [29].
- Guessing$(w) = \log($branching$(w))$, namely summing up the logarithm of the nodes' degrees rather than multiplying the degrees. See [29].
- Trace(w) = the maximum branching of an accepting run of $\mathcal{A}$ on w. See [49].
- Deviation(w) is almost the same as guessing(w), just considering for each node its degree minus one rather than its degree. See [28].

3.3 Semantic Language-Based Notions

Notions within this class are not based on a word measure, but rather relate, explicitly or implicitly, to language equality. We further classify them into the following subclasses.

3.3.1 Determinism Embedding

A nondeterministic such automaton embeds an equivalent deterministic automaton, which can be derived from it by just pruning transitions or by more complicated manipulations, such as merging and pruning. A well known example of merging is the standard minimization procedure of DFAs, merging states within the same Myhill Nerode equivalence class.

Consider automata $\mathcal{A} = (\Sigma, Q, I, \delta, \alpha)$ and $\mathcal{A}' = (\Sigma, Q', I', \delta', \alpha')$. We say that $\mathcal{A}'$ is derived from $\mathcal{A}$ by *pruning* if $Q' \subseteq Q$, $I' \subseteq I$ and $\delta' \subseteq \delta$. We further say that $\mathcal{A}'$ is derived from $\mathcal{A}$ by *merging and pruning* if $Q' \subseteq Q$ and there exists a partition $P \subseteq 2^Q$ of Q and a mapping $\mu : P \to Q$, such that i) for every $S \in P, \mu(S) \in S$; and ii) for every transition $p' \xrightarrow{\sigma:x} q' \in \delta'$ we have states $p \in \mu^{-1}(p')$ and $q \in \mu^{-1}(q')$, such that there is a transition $p \xrightarrow{\sigma:x} q \in \delta$. (We've already added transition weights, which can be ignored when irrelevant.)

- Determinizable by pruning($\mathcal{A}$) holds if there exists an automaton $\mathcal{A}'$ that is equivalent to $\mathcal{A}$ and derived from it by pruning. See [4, 10].
- Determinizable by merging and pruning($\mathcal{A}$) holds if there exists an automaton $\mathcal{A}'$ that is equivalent to $\mathcal{A}$ and derived from it by merging and pruning. See [47, Section 2.1].

3.3.2 Language Similarity between States

The most common such notion requires all target states of transitions from the same state over the same letter to have the same residual language. Pre-residuality loosens it by requiring the existence of a target state whose residual language contains the languages of all other target states.

- Residual($\mathcal{A}$) holds if for every state q, letter σ, and states $q', q'' \in \delta(q, \sigma)$, we have $L(\mathcal{A}^{q'}) = L(A^{q''})$. See [5, Definition 15]. It is called semantic determinism in [1].
- Pre-residual($\mathcal{A}$) holds if $\mathcal{A}$ can be pruned into an equivalent residual automaton. See [5, Definition 15].

3.3.3 Restricted Choice Function

A nondeterministic automaton has, in each of its choices, "unlimited view" of the entire history of the word prefix read so far and of the entire future of the word suffix to be read. Two natural restrictions are to only allow a (partial) view of the history or to only allow a (partial) view of the future. An automaton is *history deterministic* if it can only view the history of the word, and it has k *lookahead* if it can only see the next k letters. History determinism is refined or made coarse by two orthogonal measures, the first considering the size of the memory required to store the relevant part of the history, and the second allowing to maintain several runs, one of which should always be correct.

- History deterministic($\mathcal{A}$) holds if there exists a strategy $s : Q \times \Sigma^* \to \delta$ to resolve $\mathcal{A}$'s nondeterminism. See [19, 13]. It is formally defined via a *letter game* played on $\mathcal{A}$ (see Section 3.3.4), whereby $\mathcal{A}$ is history deterministic if Eve has a winning strategy in it.
- History deterministic with memory M($\mathcal{A}$) holds if there exists a history-determinism strategy for $\mathcal{A}$ that uses memory M, where M can be a fixed size, dependent on the size of $\mathcal{A}$, *finite*, or *infinite*. Notice that when $M = 0$, $\mathcal{A}$ is determinizable by pruning (see Section 3.3.1).

- History determinism$(\mathcal{A}) = k$ if there exist k strategies that can be used in parallel to resolve $\mathcal{A}$'s nondeterminism. (When $k = 1$, $\mathcal{A}$ is history deterministic.) It is formally defined via a *k-runs letter game*, in which Eve should have a winning strategy (see Section 3.3.4). The notion is originally defined in [48], where it is called the width of $\mathcal{A}$.
- Lookahead$(\mathcal{A}) = k$ if there exists a strategy $s : Q \times \Sigma^{\leq k+1} \to \delta$ to resolve $\mathcal{A}$'s nondeterminism, where $\Sigma^{\leq k+1}$ stands for the current and next up to k letters of the word read. See [51, 46].

3.3.4 Winning Letter Games on $\mathcal{A}$

The *letter game* formalizes the history determinism notion (see Section 3.3.3). In the game, Adam produces a word w letter by letter, and Eve should resolve the nondeterminism. Eve wins a play if her run is accepting or $w \notin L(\mathcal{A})$. Then, $\mathcal{A}$ is history deterministic if Eve has a winning strategy in the letter game. In the *k-runs letter game*, Eve has k runs to maintain, one of which should be accepting if $w \in L(\mathcal{A})$.

Notice that resolving the winner of a letter game, for deciding whether or not a given automaton is history deterministic, is generally a difficult task, due to the complicated winning condition of the game – it depends on w's membership in $L(\mathcal{A})$, which relates to all (possibly uncountably many) runs of $\mathcal{A}$ on w. In an attempt to overcome this difficulty, Bangol and Kuperberg defined in [5] the *k-token game*, which replaces the $w \in L(\mathcal{A})$ part of the winning condition by a requirement from Adam to provide an evidence for the membership of w in $\mathcal{A}$ – Adam should also resolve the nondeterminism along k runs, and Eve wins a play if her run is accepting or all of Adam's runs are rejecting.

If Eve wins the letter game on $\mathcal{A}$ then she obviously wins the k-tokens game on $\mathcal{A}$ for any k. As for the converse, the 1-token game is generally easier for Eve than the letter game [5], but it is open whether or not the 2-tokens game is equivalent to the letter game, known as the "G2 conjecture" [5, Conclusions]. It was proved correct so far for Büchi and coBüchi automata [5, 11], and it is still open for stronger acceptance conditions.

There are various variants of the k-tokens game in the literature, depending on how exactly Adam is allowed to conduct his runs, one of which is the *joker game* [43].

- Letter game$(\mathcal{A})$: In each round of a play, Adam chooses a letter $\sigma \in \Sigma$ and then Eve chooses a corresponding transition $t \in \delta$. In the limit, a play consists of a word w generated by Adam and a run r on w generated by Eve. Eve wins if r is accepting or $w \notin L(\mathcal{A})$. See [32, 13].
- k-runs letter game$(\mathcal{A})$: As the letter game, except that Eve maintains k runs, by choosing at each round k corresponding transitions, and she wins if at least one of her runs is accepting or $w \notin L(\mathcal{A})$. See [48].
- k-tokens game$(\mathcal{A})$: In each round of a play, Adam chooses a letter $\sigma \in \Sigma$, then Eve chooses a corresponding transition $t \in \delta$, and then Adam chooses k corresponding transitions. In the limit, a play consists of a word w generated by Adam, a run r on w generated by Eve, and k runs on w generated by Adam. Eve wins if r is accepting or all of Adam's runs are rejecting. See [5].

3.3.5 Good For (composition with) Entities

An automaton $\mathcal{A}$ is intuitively good for some entities, e.g., games or Markov decision processes (MDPs), if the composition of $\mathcal{A}$ with every entity E whose definition is based on $\mathcal{A}$'s language yields an entity $E \times \mathcal{A}$ that is equivalent to E.

- Good for games($\mathcal{A}$) holds if for every game G with Σ-labeled transitions and winning condition $L(\mathcal{A})$, we have that G and $G \times \mathcal{A}$ have the same winner. See [32, 13].
- Good for small (n) games($\mathcal{A}$) holds if $\mathcal{A}$ is good for games with up to n positions. See [22, Definition 8] and [44].
- Good for trees($\mathcal{A}$) is equivalent to $\mathcal{A}$ being good for one-player games [13, Lemma 15]. See [10, Section 2.3] for the original definition.
- Good for automata($\mathcal{A}$) holds if for every alternating automaton $\mathcal{B}$ with Σ-labeled transitions (or states) and acceptance condition $L(\mathcal{A})$, we have $\mathcal{B} \times \mathcal{A}$ is equivalent to $\mathcal{B}$. See [20] and [13, Definition 6].
- Good for MDPs($\mathcal{A}$) holds if for every MDP M with Σ-labeled transitions and the objective that its runs generate words in $L(\mathcal{A})$, we have that M and $M \times \mathcal{A}$ have the same value to their optimal strategies. See [31, Definition 1].

4 Generalizing the Notions to the Quantitative Setting

We follow below the structure of Section 3, which classified the different notions, and explain how the notions of each class can be generalized to the quantitative setting. When relevant, we provide the (partially formal) definition of each of the considered generalized notions.

4.1 Syntactic Notions

Notions unrelated to the automaton's language and acceptance labeling, as described in Section 3.1, apply as is to quantitative automata – they only relate to the automaton's structure, which is indifferent to whether the semantics of the automaton relates to Boolean or quantitative values. This is the case, in particular, with initial nondeterminsm, tree width, structural ambiguity, and general structural ambiguity.

As for determinisim in the limit, which does not relate to a language, but does relate to labeling of states or transitions, it can analogously apply to quantitative automata, requiring for example that an automaton is deterministic after every transition with some designated weight. It might be relevant only to quantitative automata of certain types, as it is only relevant to Büchi automata among the standard ω-regular automata.

4.2 Semantic Word-Based Notions

A quantitative automaton $\mathcal{A}$ can be naturally turned into a Boolean one, by accompanying it with a threshold value $t \in \mathbb{R}$, and defining that a word w is accepted by the *threshold automaton* t-$\mathcal{A}$ iff $\mathcal{A}(w) \geq t$ (or $\mathcal{A}(w) > t$).

Semantic word-based notions, as described in Section 3.2, measure for each *accepted* word the level of nondeterminism in its runs. Since these measures only consider the nondeterminism involved in the runs, they apply as is to threshold automata – Once an acceptance criterion is set, the semantic word-based measures are analogous to the syntactic measures considered in Section 3.1.

Looking for generalized notions that apply to $\mathcal{A}$ itself, and not only to t-$\mathcal{A}$, as will be defined in Section 4.3 for language-based notions, we currently do not see natural such generalizations: i) Taking the supremum of the notion's value on t-$\mathcal{A}$ over all thresholds $t \in \mathbb{R}$ will eliminate the semantic meaning of the notion, making it just a syntactic notion as in Section 3.1. For example, ambiguity will be the same as tree width. ii) Replacing the Boolean acceptance criterion with some function that relates to the value of the automaton on a word might be an interesting notion for some types of quantitative automata, but it changes the original meaning of the notion and does not seem to provide a general notion between determinism and nondeterminism.

4.3 Semantic Language-Based Notions

A notion N that is based, explicitly or implicitly, on language equality, as described in Section 3.3, can be naturally generalized to the quantitative setting in the following three ways.

Approach I. $\mathsf{N}(t\text{-}\mathcal{A})$. Applying the notion as is to a threshold automaton t-$\mathcal{A}$, as described in Section 4.2 above for word-based notions.

Approach II. Threshold-$\mathsf{N}(\mathcal{A})$. Defining the notion's value on $\mathcal{A}$ to be the worst (supremum) value of $\mathsf{N}(t\text{-}\mathcal{A})$ over all thresholds $t \in \mathbb{R}$.

Approach III. $\mathsf{N}(\mathcal{A})$. Directly generalizing the notion to apply to $\mathcal{A}$ by replacing the language basis of N with a function basis. For example, rather then requiring that $\mathcal{A}$ and some related automaton $\mathcal{A}'$ recognize the same language, we require that the quantitative automata $\mathcal{A}$ and $\mathcal{A}'$ realize the same function.

By definition, if Threshold-$\mathsf{N}(\mathcal{A})$ holds for some automaton $\mathcal{A}$ than $\mathsf{N}(t\text{-}\mathcal{A})$ holds with respect to every threshold $t \in \mathbb{R}$. (For a notion that provides a nondeterminism measure rather than a yes/no value, we have Threshold-$\mathsf{N}(\mathcal{A}) \geq \mathsf{N}(t\text{-}\mathcal{A})$).

We also generally have $\mathsf{N}(\mathcal{A}) \Rightarrow$ Threshold-$\mathsf{N}(\mathcal{A})$ (or $\mathsf{N}(\mathcal{A}) \geq$ Threshold-$\mathsf{N}(\mathcal{A})$), since morally the former requires exact equality and the latter requires equality with respect to thresholds. Interestingly, it might be that $\mathsf{N}(\mathcal{A}) \not\Leftarrow$ Threshold-$\mathsf{N}(\mathcal{A})$, even though Threshold-$\mathsf{N}(\mathcal{A})$ is defined with respect to *all* thresholds $t \in \mathbb{R}$, as is the case with history determinism (see Section 4.3.1).

Notice that when speaking of $\mathsf{N}(t\text{-}\mathcal{A})$, we consider the same definition of N as in the Boolean case. Further, the definition of Threshold-N is directly derived from it. Yet, the definition of N in $\mathsf{N}(\mathcal{A})$, for a quantitative automaton $\mathcal{A}$, is more general than its definition for a Boolean automaton, as its basis is generalized from languages to functions.

For each notion N listed in Section 3.3, we provide below its generalization $\mathsf{N}(\mathcal{A})$ to a quantitative automaton $\mathcal{A}$. Observe that if N is explicitly based on automata equivalence or containment, then it can be directly generalized to quantitative automata, in which setting automata equivalence stands for realizing the same function and an automaton $\mathcal{A}'$ is contained in $\mathcal{A}$ if for every word w, $\mathcal{A}'(w) \leq \mathcal{A}(w)$. However, if N is only implicitly based on language equality, its generalization requires some subtlety.

Determinism Embedding. Such notions, and in particular determinizability by pruning and determinizability by merging and pruning, consider the *equivalence* of $\mathcal{A}$ and an embedded automaton $\mathcal{A}'$, and are thus directly generalized to quantitative automata.

Language Similarity between States. These notions, and in particular residuality and pre-residuality also directly speak of automata equivalence or containment, and can thus directly apply to quantitative automata. Yet, it is natural to extend them in two aspects:

I. Since in quantitative automata there are weights on transitions, the first transition can make a difference and should be considered. Thus, the definition should be that $\mathsf{Residual}(\mathcal{A})$ holds, for a Val automaton $\mathcal{A}$, if for every state q, letter σ, and transition $t = q \xrightarrow{\sigma:x} q'$, we have that $\sup\{\mathsf{Val}(\pi) \mid \pi \text{ is a run of } \mathcal{A}^q \text{ on } w \text{ starting with } t\} = \mathcal{A}^q(w)$.

II. The intended meaning of residual transitions is that it is "safe" to follow them. However, while this is the case when the value function only depends on the future, like in ω-regular automata, or when it is monotonic with respect to suffixes (cf. "suffix-monotonic functions" [14, Definition 18]), it need not be the case with general value functions. (See more about it in Section 5.) For making the notion relevant also to more general value functions, one can define transitions to be "safe" if taking them allows to get the best value to every suffix, following *every prefix* (cf. "cautious strategies" [14, Definition 14]).

Restricted choice function. These notions, and in particular lookahead and the different variants of history determinism, put limitations on the transition function of $\mathcal{A}$, which stand equally for Boolean and quantitative automata, and require the automaton $\mathcal{A}'$ that is obtained from $\mathcal{A}$ by these limitations to be equivalent to $\mathcal{A}$, which again directly generalizes to quantitative automata.

Notice that history determinism can still be formally defined via the letter game, whose generalization to the quantitative setting is described below.

Winning Letter Games on $\mathcal{A}$. These notions do not explicitly speak of language equality, and thus need a bit more delicate handling. The games are played exactly as in the Boolean setting and the only change is in the definition of Eve's winning condition – In the letter game, instead of requiring her run to accept if the generated word w is in $L(\mathcal{A})$, we require that the *value* of her run on w is exactly $\mathcal{A}(w)$; In the k-tokens game, instead of requiring her run to accept if some of Adam's runs are accepting, we require that the *value* of her run is at least as high as all of Adam's runs. See [14, Definition 2] (where the notions are formally defined also with respect to alternating automata).

Good For (composition with) Entities. These notions, differently from all of the previously discussed notions, consider also external entities – A "good for some entities" notion N holds for a Boolean automaton $\mathcal{A}$ if $\mathcal{A}$ properly composes with every entity whose definition is based on $\mathcal{A}$'s language.

Hence, for generalizing N to apply to quantitative automata, which realize functions from words to real numbers, we should first see if there is a relevant generalization of the external entities to be based on functions to real numbers rather than on languages.

- Good-for-gameness. The generalization of the notion relates to composition with *zero-sum games*, which generalize the win-lose games used for Boolean automata. A quantitative automaton $\mathcal{A}$ is good for games iff for every *zero-sum* game G whose transitions are Σ-labeled and the *payoff* (value) of a play generating a word w is $\mathcal{A}(w)$, we have that G and $G \times \mathcal{A}$ have the same *value*. See [14, Definition 1].
- Good-for-automataness. The generalization of the notion relates to composition with *alternating quantitative automata*, which generalize alternating Boolean ones.
  Notice that in the composition $\mathcal{B} \times \mathcal{A}$ of (Boolean or quantitative) automata $\mathcal{A}$ and $\mathcal{B}$, the transitions or states of $\mathcal{B}$ should be labeled with the alphabet of $\mathcal{A}$. (See [20] and [13, Definition 2]). A quantitative automaton $\mathcal{B}$ formally has transition labels (weights) in $\mathbb{R}$ (usually in $\mathbb{Q}$), implying that the alphabet Σ of $\mathcal{A}$ should be contained in $\mathbb{R}$. Yet, as there are finitely many weight labels in $\mathcal{B}$, we may consider an arbitrary alphabet Σ for $\mathcal{A}$, where each letter $\sigma \in \Sigma$ stands for a "name" to a weight from $\mathcal{B}$'s labeling.
  Then, a quantitative automaton $\mathcal{A}$ is good for automata iff for every alternating quantitative automaton $\mathcal{B}$ with Σ-labeled transitions (or states) and *value function* $\mathcal{A}$, we have that $\mathcal{B} \times \mathcal{A}$ is equivalent to $\mathcal{B}$.
- Good-for-MDPness. The generalization is with respect to MDPs whose objective is to maximize $\mathcal{A}$'s value on the generated word; that is, instead of having an objective function "$f(w) = 1$ if $w \in L(\mathcal{A})$ and 0 otherwise", the MDP has the objective function "$f(w) = \mathcal{A}(w)$". Then, a quantitative automaton $\mathcal{A}$ is good for MDPs iff for every MDP M with Σ-labeled transitions and the objective function $\mathcal{A}$, we have that M and $M \times \mathcal{A}$ have the same value to their optimal strategies. (In the product $M \times \mathcal{A}$ with a Val automaton $\mathcal{A}$, the transitions are labeled with the weight of the corresponding transition in $\mathcal{A}$, and the objective of the MDP is to maximize $\mathsf{Val}(r)$, where r is the sequence of weights generated along a run of the MDP.)

4.3.1 Notions vs. Threshold-Notions

While generally we have Notion $\Rightarrow$ Threshold-Notion for the language-based notions, it depends on the notion whether or not there is also a converse implication.

When Notions $\equiv$ Threshold-Notions. This is the case with notions that are "based on direct values", for example residuality: If residual($\mathcal{A}$) does not hold, there is a transition $t = q \xrightarrow{\sigma:x} q'$ and a word w, such that $v_1 = \sup\{\mathsf{Val}(\pi) \mid \pi \text{ is a run of } \mathcal{A}^q \text{ on } w \text{ starting with } t\} < \mathcal{A}^q(w) = v_2$. Then the threshold automaton t-$\mathcal{A}$, for $t = (v_1 - v_2)/2$, is not residual.

Notice that this is, however, not the case with pre-residuality, which is not based on direct values, as it allows for a pruning phase.

Good-for-some-entities notions can also have equivalence between the two generalized versions of the notion, since intuitively if an automaton $\mathcal{A}$ is not good for some entities then there is some specific entity E having a specific value v with respect to which $\mathcal{A}$ is not good, implying that t-$\mathcal{A}$, with $t = v$, is also not good for E. This is indeed the case with good-for-gamenesss [14, Lemma 5] and good-for-automataness (see Section 5).

When Notions $\not\equiv$ Threshold-Notions. The two variants are generally different for notions that are based on strategies, since Notion requires the same strategy for all values, while Threshold-Notion allows for a different strategy to each threshold.

It is shown in [14, Lemma 10] and demonstrated in [14, Figure 3], which we repeat in Figure 1, to be the case for history determinism and determinizability by pruning, and the same reasoning applies also to determinizability by merging and pruning, pre-residuality, lookahead, letter game, k-runs letter game, and k-tokens game.

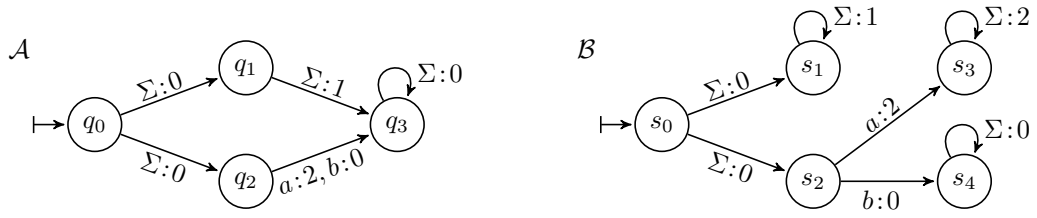

Figure 1 From [14, Figure 3]. Nondeterministic automata that demonstrate the case of Notion $\not\equiv$ Threshold-Notion with respect to the notions listed above. The automaton $\mathcal{A}$ demonstrates it with respect, for example, to the Sum/DSum/Sup value functions, and $\mathcal{B}$ with respect, for example, to the Avg/LimSup/LimInf/LimSupAvg/LimInfAvg value functions. Consider, for instance, determinizability by pruning – $\mathcal{A}$ is not determinizable by pruning, but for every threshold t it is: going only from q_0 to q_1 for a threshold $t \leq 1$ and going only from q_0 to q_2 for a threshold $t > 1$.

5 Relations between Notions

Some of the notions are related to each other by means of implication or equivalence. In some cases, their generalization to the quantitative setting does not change their interrelations, while in other cases it strictly refines them.

Syntactic notions and word-based notions. The syntactic notions do not change when generalized to the quantitative setting, and the semantic word-based notions also remain as is, just relating to threshold automata. Hence, the basic relations between these notions are as

in the Boolean setting. (See, for example, [34, 49, 38].) In addition, relations between notions that depend on the automaton type can be further researched with respect to threshold automata of various quantitative automata types.

All-Thresholds generalization of language-based notions. Considering Threshold-$\mathsf{N}(\mathcal{A})$ notions, they *morally* relate to each other as in the Boolean setting. For example, notions N_1 and N_2 that are equivalent with respect to Boolean automata (i.e., for every Boolean automaton $\mathcal{A}$, we have $\mathsf{N}_1(\mathcal{A}) = \mathsf{N}_2(\mathcal{A})$) are also equivalent with respect to threshold automata (i.e., for every quantitative automaton $\mathcal{A}$ and threshold $t \in \mathbb{R}$, we have $\mathsf{N}_1(t\text{-}\mathcal{A}) = \mathsf{N}_2(t\text{-}\mathcal{A})$, as the latter are also Boolean automata. Thus, we also have Threshold-$\mathsf{N}_1(\mathcal{A})$ = Threshold-$\mathsf{N}_2(\mathcal{A})$).

Yet, observe that general Boolean automata need not be regular or ω-regular, and notions that are equivalent with respect to (ω-)regular automata need not be equivalent with respect to Boolean automata that are derived from quantitative automata. For example, over ω-regular automata, history determinism implies pre-residuality, while the implication does not hold for all Boolean automata nor for all threshold quantitative automata. (See Section 5).

Basis Generalization of Language-Based Notions

Generalizing the basis of notions from language equality to function equality refines the relations between them: Notions that are different with respect to Boolean (ω-)regular automata are also different with respect to quantitative automata, as the former are special cases of the latter, however notions "known to be equivalent" might turn inequivalent in the quantitative setting, as is the case for example with history determinism and good-for-gameness.

As in the Boolean setting.

- Determinizability by pruning $\Rightarrow^1$ history determinism $\Rightarrow^2$ good-for-gameness $\equiv^3$ good-for-automataness.

  1. Determinizability by pruning is a special case of history determinism, restricting the history-determinism strategy to be positional.
  2. A history-determinism strategy for an automaton $\mathcal{A}$ can be combined with an optimal strategy in a game G, giving an optimal strategy in the game $G \times \mathcal{A}$ [14, Theorem 4].
  3. A game is a special case of an alternating automaton $\mathcal{B}$, giving the implication from right to left, while for every specific word w, as in the Boolean setting [13, Theorem 16], the value of $\mathcal{B}$ on w is viewed as a game, giving the implication from left to right.

- History determinism $\Rightarrow$ good-for-MDPness. As in the Boolean setting [31], a history-determinism strategy for an automaton $\mathcal{A}$ can be combined with an optimal strategy for an MDP M, ensuring an optimal strategy in $M \times \mathcal{A}$.
- The G2 conjecture. Two-token games (G2) characterize history determinism for Büchi and coBüchi automata [5, 11], and it is conjectured in [5, Conclusions] that this characterization also holds for parity automata (and thus for all ω-regular automata). It is shown in [12] that G2 also characterizes history determinism for various quantitative automata, and it is open whether this is the case for all quantitative automata.

Different from the Boolean setting.

- Good-for-gameness $\not\Rightarrow$ history determinism. While the two notions are equivalent and often mixed in the ω-regular setting, they turn out to be inequivalent in the quantitative setting, having good-for-gameness $\equiv$ Threshold-history-determinism $\not\rightleftarrows$ history determinism [14, Theorem 4]. (The left equivalence assumes that the letter game on $\mathcal{A}$ is determined [14, Lemma 7].) For example, the automata in Figure 1 are good for games but not history deterministic.
- History determinism $\not\Rightarrow$ pre-residuality. In the ω-regular setting, the implication holds [43, Lemma 18] (and there is no back implication [1, Theorem 1]). Yet, the implication is based on the implicit assumption that the value function depends on suffixes: a history deterministic automaton $\mathcal{A}$ all of whose transitions are used by some history deterministic strategy is guaranteed, in this case, to be residual, as otherwise once Eve chooses a non-residual transition, Adam will choose a suffix that witnesses its non-residuality, leading to Eve's lose. However, this need not be the case if the value function also depends on prefixes, as demonstrated in Figure 2.

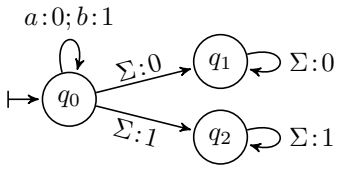

Figure 2 A Val automaton $\mathcal{A}$ on finite words with the value function $\mathsf{Val}(\rho) = 1$ if ρ has both even and odd values, and 0 otherwise. Notice that $\mathcal{A}$ is history deterministic but not pre-residual.

6 Conclusions

We have analyzed how notions of limited nondeterminism can be generalized from Boolean to quantitative automata; first observing that it depends on whether the notion is syntactic, semantic word-based, or semantic language-based, and then analyzing the possible notion generalizations within each of these three classes.

Our analysis results with generalized notions that define novel classes of quantitative automata, and opens up new research directions, which can contribute to the advance of quantitative verification and synthesis. For example, results on history deterministic and good for games quantitative automata directly relate to possible new solutions to quantitative synthesis [14].

In addition, the interrelation between the notions in the quantitative setting is shown to refine their known interrelation with respect to ω-regular automata, providing better understanding of the notions also with respect to Boolean automata whose behavior need not be regular.

References

1 Bader Abu Radi, Orna Kupferman, and Ofer Leshkowitz. A hierarchy of nondeterminism. In *Proc. of MFCS*, volume 202 of *LIPIcs*, pages 85:1–85:21, 2021.
2 Shaull Almagor, Udi Boker, and Orna Kupferman. Formalizing and reasoning about quality. *Journal of the ACM*, 63(3):24:1–24:56, 2016.
3 Shaull Almagor, Orna Kupferman, Jan Oliver Ringert, and Yaron Velner. Quantitative assume guarantee synthesis. In *Proc. of CAV*, pages 353–374. Springer, 2017.

4 Benjamin Aminof, Orna Kupferman, and Robby Lampert. Reasoning about online algorithms with weighted automata. *ACM Transactions on Algorithms*, 6(2):28:1–28:36, 2010.

5 Marc Bagnol and Denis Kuperberg. Büchi good-for-games automata are efficiently recognizable. In *Proc. of FSTTCS*, page 16, 2018.

6 Roderick Bloem, Krishnendu Chatterjee, Thomas A. Henzinger, and Barbara Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. of CAV*, pages 140–156, 2009.

7 Udi Boker. Why these automata types? In *Proc. of LPAR*, pages 143–163, 2018.

8 Udi Boker. Quantitative vs. weighted automata. In *Proc. of RP*, pages 1–16, 2021.

9 Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014.

10 Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Proc. of ICALP*, pages 89–100, 2013.

11 Udi Boker, Denis Kuperberg, Karoliina Lehtinen, and Michał Skrzypczak. On succinctness and recognisability of alternating good-for-games automata. *arXiv:2002.07278*, 2020.

12 Udi Boker and Karoliina Lehtinen. Token games and history deterministic quantitative automata. Submitted.

13 Udi Boker and Karoliina Lehtinen. Good for games automata: From nondeterminism to alternation. In *Proc. of CONCUR*, pages 19:1–19:16, 2019.

14 Udi Boker and Karoliina Lehtinen. History determinism vs. good for gameness in quantitative automata. In *Proc. of FSTTCS*, pages 35:1–35:20, 2021.

15 Romain Brenguier, Lorenzo Clemente, Paul Hunter, Guillermo A Pérez, Mickael Randour, Jean-François Raskin, Ocan Sankur, and Mathieu Sassolas. Non-zero sum games for reactive synthesis. In *Proc. of LATA*, pages 3–23, 2016.

16 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In *Proceedings of FCT*, pages 3–13, 2009.

17 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4):23:1–23:38, 2010.

18 Krishnendu Chatterjee, Thomas A Henzinger, Barbara Jobstmann, and Rohit Singh. Quasy: Quantitative synthesis tool. In *Proc. of TACAS*, pages 267–271. Springer, 2011.

19 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Proc. of ICALP*, pages 139–150, 2009.

20 Thomas Colcombet. Fonctions régulières de coût. *Habilitation à diriger les recherches, École Doctorale de Sciences Mathématiques de Paris Centre*, 2013.

21 Thomas Colcombet. Unambiguity in automata theory. In *Proc. of DCFS*, pages 3–18, 2015.

22 Thomas Colcombet and Nathanaël Fijalkow. Universal graphs and good for games automata: New tools for infinite duration games. In *Proc. of FOSSACS*, pages 1–26, 2019.

23 Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *J. ACM*, 42(4):857–907, 1995.

24 Laurent Doyen. Games and automata: From Boolean to quantitative verification. Habilitation Thesis, École Normale Supérieure de Cachan, 2011.

25 Marco Faella, Axel Legay, and Mariëlle Stoelinga. Model checking quantitative linear time logic. *Electr. Notes Theor. Comput. Sci.*, 220(3):61–77, 2008.

26 Emmanuel Filiot, Ismaël Jecker, Nathan Lhote, Guillermo A. Pérez, and Jean-François Raskin. On delay and regret determinization of max-plus automata. In *LICS*, pages 1–12, 2017.

27 Emmanuel Filiot, Christof Löding, and Sarah Winter. Synthesis from weighted specifications with partial domains over finite words. In *FSTTCS*, pages 46:1–46:16, 2020.

28 Daniel Goč and Kai Salomaa. Computation width and deviation number. In *Descriptional Complexity of Formal Systems*, pages 150–161, 2014.

29 Jonathan Goldstine, Chandra M. R. Kintala, and Detlef Wotschke. On measuring nondeterminism in regular languages. *Inf. Comput.*, 86(2):179–194, 1990.

30 Mika Göös and Stefan Kiefer. Lower bounds on unambiguous automata complementation and separation via communication complexity. *CoRR*, abs/2109.09155, 2021. `arXiv:2109.09155`.

31 Ernst Hahn, Mateo Perez, Sven Schewe, Fabio Somenzi, Ashutosh Trivedi, and Dominik Wojtczak. Good-for-MDPs automata for probabilistic analysis and reinforcement learning. In *Proc. of TACAS*, pages 306–323, 2020.

32 Thomas Henzinger and Nir Piterman. Solving games without determinization. In *Proc. of CSL*, pages 395–410, 2006.

33 Markus Holzer, Kai Salomaa, and Sheng Yu. On the state complexity of k-entry deterministic finite automata. *J. Autom. Lang. Comb.*, 6(4):453–466, 2001.

34 Juraj Hromkovič, Juhani Karhumäki, Hartmut Klauck, Georg Schnitger, and Sebastian Seibert. Measures of nondeterminism in finite automata. In *Proc. of ICALP*, pages 199–210, 2000.

35 Juraj Hromkovic, Sebastian Seibert, Juhani Karhumäki, Hartmut Klauck, and Georg Schnitger. Communication complexity method for measuring nondeterminism in finite automata. *Inf. Comput.*, 172(2):202–217, 2002.

36 Paul Hunter, Guillermo A. Pérez, and Jean-François Raskin. Reactive synthesis without regret. *Acta Informatica*, 54(1):3–39, 2017.

37 Martin Kappes. Descriptional complexity of deterministic finite automata with multiple initial states. *Journal of Automata, Languages and Combinatorics*, 5:269–278, January 2000.

38 Chris Keeler and Kai Salomaa. Branching measures and nearly acyclic NFAs. *Int. J. Found. Comput. Sci.*, 30(6-7):1135–1155, 2019.

39 Chris Keeler and Kai Salomaa. Alternating finite automata with limited universal branching. In *Proc. of LATA*, pages 196–207, 2020.

40 Chandra M. R. Kintala. Refining nondeterminism in context-free languages. *Math. Syst. Theory*, 12:1–8, 1978.

41 Chandra M. R. Kintala and Patrick C. Fischer. Computations with a restricted number of nondeterministic steps (extended abstract). In *Proc. of STOC*, pages 178–185, 1977.

42 Chandra M. R. Kintala and Detlef Wotschke. Amounts of nondeterminism in finite automata. *Acta Informatica*, 13:199–204, 1980.

43 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *Proc. of ICALP*, pages 299–310, 2015.

44 Karoliina Lehtinen and Udi Boker. Register games. *Log. Methods Comput. Sci.*, 16(2), 2020.

45 Hing Leung. Structurally unambiguous finite automata. In *Proc. of CIAA*, pages 198–207, 2006.

46 Christof Löding and Stefan Repke. Decidability results on the existence of lookahead delegators for NFA. In *Proc. of FSTTCS*, pages 327–338, 2013.

47 Christof Löding and Andreas Tollkötter. State space reduction for parity automata. In *Proc. of CSL*, pages 27:1–27:16, 2020.

48 Anirban Majumdar and Denis Kuperberg. Computing the width of non-deterministic automata. *Logical Methods in Computer Science*, 15, 2019.

49 Alexandros Palioudakis, Kai Salomaa, and Selim Akl. Comparisons between measures of nondeterminism on finite automata. In *Proc. of DCFS*, pages 217–228, 2013.

50 Mikhail A. Raskin. A superpolynomial lower bound for the size of non-deterministic complement of an unambiguous automaton. In *Proc. of ICALP*, pages 138:1–138:11, 2018.

51 Bala Ravikumar and Nicolae Santean. On the existence of lookahead delegators for NFA. *Int. J. Found. Comput. Sci.*, 18(5):949–973, 2007.

52 M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification (preliminary report). In *Proc. of LICS*, pages 332–344, 1986.