Approximate Determinization of Quantitative Automata[*]

Udi Boker[1,2] and Thomas A. Henzinger[2]

1 Hebrew University of Jerusalem
2 IST Austria

Abstract

Quantitative automata are nondeterministic finite automata with edge weights. They value a run by some function from the sequence of visited weights to the reals, and value a word by its minimal/maximal run. They generalize boolean automata, and have gained much attention in recent years. Unfortunately, important automaton classes, such as sum, discounted-sum, and limit-average automata, cannot be determinized. Yet, the quantitative setting provides the potential of approximate determinization. We define approximate determinization with respect to a distance function, and investigate this potential.

We show that *sum* automata cannot be determinized approximately with respect to any distance function. However, restricting to nonnegative weights allows for approximate determinization with respect to some distance functions.

Discounted-sum automata allow for approximate determinization, as the influence of a word's suffix is decaying. However, the naive approach, of unfolding the automaton computations up to a sufficient level, is shown to be doubly exponential in the discount factor. We provide an alternative construction that is singly exponential in the discount factor, in the precision, and in the number of states. We prove matching lower bounds, showing exponential dependency on each of these three parameters.

Average and *limit-average* automata are shown to prohibit approximate determinization with respect to any distance function, and this is the case even for two weights, 0 and 1.

1 Introduction

Quantitative automata have gained much attention in recent years. In the scope of formal verification, they allow a generalization of the traditionally boolean setting into a quantitative one (e.g. [5, 10, 13]). They are nondeterministic finite-state automata with edge weights, valuing a run by a function from the sequence of visited weights into $\mathbb{R}$. Unlike weighted automata that are based on a semi-ring [12] or lattice [15], quantitative automata do not limit the value function to certain algebraic properties. A quantitative automaton $\mathcal{A}$ assigns to each word w a real value $\mathcal{A}(w)$, which is the infimum or the supremum of all runs of the automaton on w.

Automata determinization is fundamental in formal methods, such as synthesis, which is based on game solving, and verification, which is based on the comparison of automata. Game solving requires automata that are essentially deterministic [14], and checking for automata equivalence and inclusion has good algorithms for deterministic automata, whereas it is expensive for boolean nondeterministic automata and even undecidable for some quantitative nondeterministic automata (such as limit-average automata [11]). Unfortunately, important quantitative automaton classes, such as sum, average, discounted-sum, LimSup, and limit-average automata, cannot be determinized [8].

[*] This work was supported in part by the Austrian Science Fund NFN RiSE (Rigorous Systems Engineering) and by the ERC Advanced Grant QUAREM (Quantitative Reactive Modeling).

The quantitative setting is directly related to approximation. Having real values for words, naturally suggests to approximate one automaton by another, e.g. a nondeterministic automaton by a deterministic one, requiring that the latter's values on all words are close enough to the former's values. With such approximately determinized automata, one may approximately compare between the original nondeterministic automata, as well as approximately solve games.

How should one define "close enough"? A general choice is some distance function on $\mathbb{R}$. We therefore define that an automaton $\mathcal{A}$ can be *determinized approximately* with respect to a distance function d if for every real precision $\varepsilon > 0$, there is a deterministic automaton $\mathcal{D}$ such that for every word w, $d(\mathcal{A}(w), \mathcal{D}(w)) \leq \varepsilon$.

One may check whether some automaton type can be determinized approximately with respect to a specific distance function, for example with respect to the absolute difference function ($|\mathcal{A}(w) - \mathcal{D}(w)|$). In addition, seeking a thorough picture of approximate determinization, one may try to show that a certain automaton type cannot be determinized approximately with respect to any distance function in a certain class. A central observation is that quantitative automata relate to the natural order on $\mathbb{R}$, as they use non-determinism for choosing the infimum or supremum of the run values. For that reason, an interesting and general class of distance functions (called *ordered* distance functions) contains those that respect the order on $\mathbb{R}$, meaning that for every $x \leq y \leq z \in \mathbb{R}$, it holds that $d(x, y) \leq d(x, z)$ and $d(y, z) \leq d(x, z)$.

The importance of approximate determinization is well known (e.g. [4, 7]), yet central questions are still open. In [4], they consider approximate determinization of sum automata with respect to ratio, showing that it is possible in cases that the nondeterministic automaton admits some property (called t-twins). In [7], they also consider sum automata, providing an efficient determinization algorithm, which, however, is not guaranteed to be within a certain distance from the nondeterministic automaton. In general, we are unaware of any work on the approximation of automata over *infinite* words (such as limit-average and discounted-sum automata), nor of any work on approximate determinization with respect to arbitrary distance functions.

We investigate the potential of determinizing quantitative-automata approximately with respect to the difference and ratio functions (which are most commonly mentioned in the literature), as well as with respect to arbitrary ordered distance functions. We pay special attention to discounted-sum automata, whose approximate determinization yields the most promising results. We also study, in Section 2.4, the problem of approximate automata comparison and approximate game solving, showing how they can be solved by approximate automata determinization.

The, possibly, simplest quantitative automata value a run by the minimal/maximal weight along it, and they can be determinized exactly [8]. The next level of quantitative automata generalizes the Büchi and co-Büchi conditions, valuing a run by the maximal (LimSup) or minimal (LimInf) value that occurs infinitely often. LimInf automata can be determinized exactly, while LimSup cannot [8]. It turns out that LimSup automata cannot even be determinized approximately with respect to any distance function. Yet, they allow for exact determinization into a stronger automaton class, analogously to determinizing Büchi automata into Rabin or parity automata.

The more involved quantitative automata, which are inherently different from boolean automata, value a run by accumulating the weights along it. The basic accumulation schemes are either summation or averaging. Another aspect, on top of the accumulation, is the influence of time. With sum and average automata, all weights along the run have the same influence. In some cases, one favors close events over far away ones, thus introducing discounting of the weights along time. Discounted-sum automata are the best-known representatives of this class. On the other hand, there are cases in which one is only interested in the long-run behavior, looking at the limit values. The well-known representatives of this class are limit-average automata. Both discounted-summation and limit-average (also called mean-payoff) get intensive attention (e.g. [1, 3, 16]), and are even argued to be the canonic representatives of a major class of quantitative objectives [9].

We show that *sum* automata do not allow for an effective approximate determinization with respect to any ordered distance function. [1] Furthermore, using the undecidability proof of their universality problem [2], we show that they cannot even be determinized approximately into any automaton class whose universality problem is decidable. On the positive side, restricting sum automata to *nonnegative* weights, while still not allowing for determinization, allows for approximate determinization with respect to some distance functions. We prove this by translating sum automata over nonnegative weights into product automata over weights in $[0, 1]$. The latter are shown to allow for approximate determinization with respect to the difference function.

Discounted-sum automata naturally allow for approximate determinization with respect to the difference function by unfolding the automaton computations up to a sufficient level. The smaller the required precision is, and the closer the discount-factor is to 1, the more expensive it is to determinize the automaton approximately. We represent the precision by $\varepsilon = 2^{-p}$ and the discount factor by $\lambda = 1 + 2^{-k}$. We analyze the unfolding approach to construct an automaton whose state space is exponential in the representation of the precision (p) and doubly exponential in the representation of the discount factor (k). We then provide an alternative construction that is singly exponential in k, in p, and in the number of states of the automaton. The construction generalizes the subset construction, keeping, for each state of the original automaton, its relative extra-cost ("gap"), within a fixed resolution. We complete the picture for discounted-sum automata by proving matching lower bounds, showing exponential dependency on each of these three parameters.

Average and *limit-average* automata use a more complicated accumulation scheme than summation. This gets an evident in approximate determinization, as they cannot be determinized approximately even if their weights are restricted to any pair of distinct values. It is left open whether there is a stronger automaton class with decidable properties into which average a limit-average automata can be determinized approximately.

For better readability, we only give the central ideas of the proofs in the main text, and provide the full proofs in the Appendix.

2 The Setting

We start with standard definitions of quantitative automata and distance functions, continue with our definition of approximate determinization, and conclude with describing how can one take advantage deterministic automata that approximate nondeterministic ones.

[1] Upfront, it might seem that approximate determinization and exact determinization are the same for sum automata, as the possible values of the original automaton are discrete. For example, the original automaton may only have integer values. However, this is not the case, since the corresponding deterministic automaton may have different weights. For instance, an automaton that always yields the value $\frac{1}{2}$ properly $\frac{1}{2}$-approximates, with respect to the difference function, every sum automaton with values in $\{0, 1\}$.

2.1 Quantitative Automata

A quantitative automaton is a weighted automaton, over finite or infinite words, valuing a run by a real number, and valuing a word by the infimum/supremum of the runs on it.

Formally, given an alphabet Σ, a *word* w over Σ is a finite or infinite sequence of letters in Σ, where $w[i]$ stands for the letter in position i, starting from 0. We denote the concatenation of a finite word u and a word w by $u{\cdot}w$, or simply by uw.

A *weighted automaton* is a tuple $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \gamma \rangle$ over a finite alphabet Σ, with a finite set of states Q, a subset of initial states $Q_{in} \subseteq Q$, a transition function $\delta \subseteq Q \times \Sigma \times Q$, and a weight function $\gamma : \delta \to \mathbb{Q}$. For a state q of $\mathcal{A}$, we denote by $\mathcal{A}^q$ the automaton that is identical to $\mathcal{A}$, except for having q as its initial state.

An automaton may have in general many possible transitions for each state and letter, and hence we say that $\mathcal{A}$ is *nondeterministic.* In the case where $|Q_{in}| = 1$ and for every $q \in Q$ and $\sigma \in \Sigma$, we have $|\{q' \mid (q, \sigma, q') \in \delta\}| \leq 1$, we say that $\mathcal{A}$ is *deterministic.* In the case where for every $q \in Q$ and $\sigma \in \Sigma$, we have $|\{q' \mid (q, \sigma, q') \in \delta\}| \geq 1$, we say that $\mathcal{A}$ is *complete.* Intuitively, a complete automaton cannot get stuck at some state. In this paper, we only consider complete automata.

A run of an automaton is a sequence of states and letters, $q_0, \sigma_1, q_1, \sigma_2, q_2, \ldots$, such that $q_0 \in Q_{in}$ and for every i, $(q_i, \sigma_{i+1}, q_{i+1}) \in \delta$. The length of a run, denoted $|r|$, is n for a finite run $r = q_0, \sigma_1, q_1, \ldots, \sigma_{n-1}, q_{n-1}$, and ∞ for an infinite run. A run r induces a sequence of weights, $\gamma_0, \gamma_1, \ldots$, where $\gamma_i = \gamma(q_i, \sigma_{i+1}, q_{i+1})$.

The value of a run r, denoted $\gamma(r)$, is a real number defined by a *value function* over its sequence of weights, depending on the automaton type. For an automaton $\mathcal{A}$, the value of a word w, denoted $\mathcal{A}(w)$, is either the infimum or the supremum of $\{\gamma(r) \mid r \text{ is a run of } \mathcal{A} \text{ on } w\}$, depending on $\mathcal{A}$'s type. A run r is *optimal* if $\gamma(r) = \mathcal{A}(w)$.

By the above definitions, an automaton $\mathcal{A}$ realizes a function from Σ^* or Σ^ω to $\mathbb{R}$. Two automata, $\mathcal{A}$ and $\mathcal{A}'$, are *equal* if they realize the same function.

Next, we define some types of quantitative automata, differing by their value function. We use below "inf of runs on w" as a shortcut for $\inf\{\gamma(r) \mid r \text{ is a run of } \mathcal{A} \text{ on } w\}$, and an analogous shortcut for the supremum.

Sum automata. Finite words; $\gamma(r) = \sum_{i=0}^{|r|-1} \gamma_i$; $\mathcal{A}(w) = \text{inf of runs on } w$.

Product automata. Finite words; $\gamma(r) = \prod_{i=0}^{|r|-1} \gamma_i$; $\mathcal{A}(w) = \text{sup of runs on } w$.

Average automata. Finite words; $\gamma(r) = \frac{1}{|r|} \sum_{i=0}^{|r|-1} \gamma_i$; $\mathcal{A}(w) = \text{sup of runs on } w$.

Discounted-sum automata. Operate over finite or infinite words, and are equipped with a rational discount factor $\lambda > 1$. [2] They value a run r by $\gamma(r) = \sum_{i=0}^{|r|-1} \frac{\gamma_i}{\lambda^i}$; and $\mathcal{A}(w) = \text{inf}$ of runs on w. We write NDA and DDA to denote nondeterministic and deterministic discounted-sum automata, respectively, as well as λ-NDA or λ-DDA to denote an NDA or DDA with a discount factor λ, for example $\frac{5}{2}$-NDA.

LimSup Automata. Infinite words; $\gamma(r) = \lim\limits_{n\to\infty} \sup\{\gamma_i \mid i \geq n\}$; $\mathcal{A}(w) = \text{sup of runs on } w$.

Limit-average Automata. Infinite words; $\gamma(r) = \lim\limits_{n\to\infty} \inf\{\frac{1}{i} \sum_{j=0}^{i-1} \gamma_j \mid i \geq n\}$, [3] $\mathcal{A}(w) = \text{sup}$ of runs on w.

[2] The discount factor λ is often used in the literature as a multiplying factor, rather than as a dividing factor, thus taking the role of $\frac{1}{\lambda}$, compared to our definitions.

[3] There is another version of limit-average automata, where $\gamma(r) = \lim\limits_{n\to\infty} \sup\{\frac{1}{i} \sum_{j=0}^{i-1} \gamma_j \mid i \geq n\}$. All of our results equally apply to both versions.

2.2 Distance Functions

We restrict our attention to distance functions over $\mathbb{R}$, yet the definitions equally apply to every ordered space.

A function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a *distance function* if for every $x, y, z \in \mathbb{R}$, it holds that:

1. $d(x, y) \geq 0$ (non-negative);
2. $d(x, y) = 0$ iff $x = y$ (identity of indiscernibles);
3. $d(x, y) = d(y, x)$ (symmetry); and
4. $d(x, y) + d(y, z) \geq d(x, z)$ (triangle inequality) .

We use distance functions for measuring the distance between two quantitative automata. Since these automata relate to the natural order on $\mathbb{R}$, using non-determinism for choosing the infimum or supremum of the run values, the relevant distance functions should also respect the order on $\mathbb{R}$. A distance function d is *ordered* if for every $x \leq y \leq z \in \mathbb{R}$, we have $d(x, y) \leq d(x, z)$ and $d(y, z) \leq d(x, z)$.

When we speak of the *difference function*, we refer to $d(x, y) = |x - y|$. Another common measure is *ratio*, however it is not a distance function, as it does not satisfy the triangle inequality. For that reason, when we speak of the *ratio distance function*, we refer to $d(x, y) = |\log x - \log y|$, which also measures the proportion between values.

2.3 Approximate Determinization

We define that an automaton can be determinized approximately if for every real precision $\varepsilon > 0$, there is a deterministic automaton such that the distance between their values on all words is less than or equal to ε. Formally,

▶ **Definition 1** (Approximation). The definitions below stand for approximation with respect to a distance function d.

- A quantitative automaton $\mathcal{A}'$ *ε-approximates* a quantitative automaton $\mathcal{A}$, for a real constant $\varepsilon > 0$, if for every word w, $d(\mathcal{A}(w), \mathcal{A}'(w)) \leq \varepsilon$.
- A quantitative automaton $\mathcal{A}$ can be *approximated* by an automaton class $\mathfrak{C}$ if for every real constant $\varepsilon > 0$ there is an automaton $\mathcal{A}' \in \mathfrak{C}$ that ε-approximates $\mathcal{A}$.
- A quantitative automaton $\mathcal{A}$ of a class $\mathfrak{C}$ can be *determinized approximately* if it can be approximated by the class of deterministic automata in $\mathfrak{C}$; the class $\mathfrak{C}$ can be *determinized approximately* if every automaton $\mathcal{A} \in \mathfrak{C}$ can.

2.4 Taking Advantage of Approximated Automata

Approximate determinization is useful for automata comparison, which is essential in formal verification, as well as for game solving, which is essential in synthesis.

Approximate automata comparison. Consider two nondeterministic quantitative automata $\mathcal{A}$ and $\mathcal{B}$. One can approximately solve the question of whether for all words w, $\mathcal{A}(w) \geq \mathcal{B}(w)$, with respect to the difference function and a precision $\varepsilon > 0$, by generating deterministic automata $\mathcal{A}'$ and $\mathcal{B}'$ that $\frac{\varepsilon}{2}$-approximate $\mathcal{A}$ and $\mathcal{B}$, respectively, and computing $m = \sup_w\{\mathcal{B}'(w) - \mathcal{A}'(w)\}$. If $m > \varepsilon$, it follows that $\mathcal{A}(w)$ is not always bigger than $\mathcal{B}(w)$. Otherwise, it follows that $\mathcal{A}(w)$ is always bigger than $\mathcal{B}(w)$, up to a possible mistake of 2ε. Equivalence and universality can be approximated similarly, up to any desired precision.

Note that one cannot solve the question of whether for all words w, $(\mathcal{B}(w) - \mathcal{A}(w)) \leq \varepsilon$, because it is as difficult as the question of whether $\mathcal{A} \geq \mathcal{B}$. Yet, one can reduce the uncertainty area to be arbitrarily small.

Approximate game solving. Consider a two-player game $\mathcal{G}$ whose winning condition is given by means of a nondeterministic quantitative automaton $\mathcal{A}$. For solving the game, one determinizes $\mathcal{A}$ into an automaton $\mathcal{D}$, takes the product of $\mathcal{G}$ and $\mathcal{D}$, and finds optimal strategies for the game $\mathcal{G}' = \mathcal{G} \times \mathcal{D}$. The value of the game is the value that $\mathcal{A}$ assigns to the trace of the game, once the two players follow their optimal strategies. Now, in the case that $\mathcal{D}$ is not equivalent to $\mathcal{A}$, but $\frac{\varepsilon}{2}$-approximates it, the value of $\mathcal{G}'$ is guaranteed to be up to ε-apart from the value of $\mathcal{G}$.

3 Sum Automata

Sum automata are the basic form of weighted automata that value a run by accumulating the weights along it. Having both positive and negative numbers allows for counting, which makes sum automata close enough, in some aspects, to counter machines. For that reason, their universality problem is undecidable [2]. Using this undecidability result, we show that sum automata cannot be effectively determinized approximately into any automaton class whose universality problem is decidable.

Restricting to *nonnegative* weights, the above undecidability result does not hold [2]. Indeed, we show that sum automata over nonnegative weights, while still not determinizable, allows for approximate determinization with respect to some distance functions.

We start with a lemma on the relation between sum and product automata.

▶ **Lemma 2.** *Consider a sum automaton $\mathcal{A}$, and construct from it a product automaton $\mathcal{B}$, by changing every weight x to 2^{-x}. Then for every word w, $\mathcal{B}(w) = 2^{-\mathcal{A}(w)}$.*

3.1 Unrestricted weights

We show that sum automata do not allow for an effective approximate determinization by using the undecidability proof of their universality problem.

▶ **Lemma 3** ([2]). *For every two-counter machine $\mathcal{M}$, there is a sum automaton $\mathcal{A}$ with weights in $\{-1, 0, 1\}$ such that M halts iff there is a word w such that $\mathcal{A}(w) > 0$.*

▶ **Theorem 4.** *Sum automata cannot be effectively determinized approximately with respect to any ordered distance function to any automaton class whose universality problem is decidable.*

Proof sketch. The key observation is that in order to solve the halting problem of two-counter machines, a deterministic automaton need not approximate the automaton $\mathcal{A}$ of Lemma 3 on all its possible values, but only on the values 0 and 1. Then, using the order property of the distance function, a close enough approximation will allow to deduce whether the accurate value is 0 or 1. ◀

As a special case of the above theorem, sum automata cannot be determinized approximately into deterministic sum automata, as their universality problem is equivalent to their emptiness problem, which is decidable (and polynomial).

By the relation between sum and product automata, we get the following corollary.

▶ **Corollary 5.** *Product automata over nonnegative weights cannot be effectively determinized approximately, with respect to any ordered distance function, to any automaton class whose universality problem is decidable.*

3.2 Nonnegative Weights

We show that sum automata over nonnegative weights cannot be determinized approximately with respect to difference, while they can be determinized approximately with respect to the distance function $d(x, y) = |2^{-x} - 2^{-y}|$. The usefulness of the latter distance function is arguable, yet it relates to two interesting observation. The first is the ability to determinize product automata approximately with respect to the natural difference function. The second is the conceptual difference between the sum and the average accumulation schemes, following from the inability to determinize approximately average automata even when restricting their weights (Section 5).

We start with the negative proof, using the standard sum automaton that computes the minimum between the number of a's and b's.

▶ **Theorem 6.** *Sum automata over nonnegative weights cannot be determinized approximately with respect to difference (even by sum automata with unrestricted weights).*

Proof sketch. We show that there is no deterministic sum automaton that can 1-approximate the automaton of Figure 5 (in the Appendix). The proof uses a sort of a pumping argument, showing, by way of contradiction, that the deterministic automaton must have two states, s_1 and s_2, such that s_1 has a 0-valued cycle over a word with only a's, and the same for s_2 for a word with only b's. ◀

▶ Remark. Sum automata over positive weights cannot be determinized approximately with respect to ratio, using the same arguments as in the proof of Theorem 6.

▶ **Corollary 7.** *Product automata over weights in* $(0, 1]$ *cannot be determinized exactly nor determinized approximately with respect to ratio.*

We now turn to the positive results.

▶ **Theorem 8.** *Product automata over weights in* $[0, 1]$ *can be determinized approximately with respect to difference.*

Proof sketch. The idea is to extend the subset construction with accumulated products, keeping for each state of the nondeterministic automaton the maximal value that can be accumulated for it. The automaton weights are between 0 and 1, implying that the accumulated products are monotonically decreasing. Therefore, once an accumulated product is below ε, it can be replaced with 0. Since there are finitely many different weights, there are also finitely many different accumulated weights above ε, guarantying the construction's termination. ◀

▶ Remark. Product automata over weights in $[-1, 1]$ can also be determinized approximately with respect to difference. The proof of Theorem 8 does not hold, because it uses the monotonicity of $[0, 1]$-multiplications. Yet, one can properly extend the construction of the proof of Theorem 8 by keeping for each state both the maximal and the minimal accumulated values.

Using the relation between sum and product automata (Lemma 2), we get the following result.

▶ **Theorem 9.** *Sum automata over nonnegative weights can be determinized approximately with respect to the distance function* $d(x, y) = |2^{-x} - 2^{-y}|$.

4 Discounted-Sum Automata

The naive approximation approach, of unfolding the automaton computations up to a sufficient level, is analyzed below to be doubly exponential in the representation of the discount factor. We then provide an alternative construction, based on keeping the relative extra-costs ("gaps") of the automaton states, and show that it is singly exponential in the representations of the precision, discount factor, and number of states. We conclude with proving matching lower bounds. In this section, when we speak of approximation, we mean approximation with respect to the difference function.

We start with the relation between discounted-sum automata over finite words and over infinite words, showing that approximation over finite words guarantees approximation over infinite words, but not vice versa.

▶ **Lemma 10.** *For every precision $\varepsilon > 0$ and discount factor $\lambda > 1$, if a λ-NDA ε-approximates another λ-NDA over finite words then it also ε-approximates it over infinite words. The converse need not hold.*

Proof sketch. Since the influence of word suffices is decaying, the distance between two automata cannot change "too much" after long enough prefixes. Hence, if the automata are close enough for every finite prefix, so they are for an entire infinite word. As for the converse, the distance between the automata might gradually decrease, only converging at the infinity. (See Figure 1.) ◀

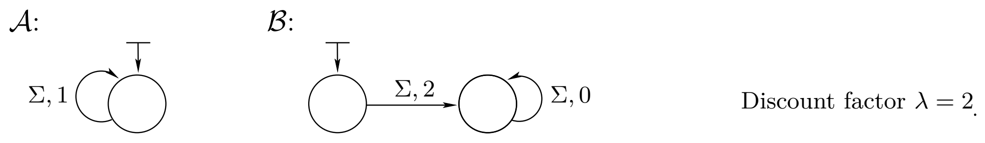


Figure 1 The DDA $\mathcal{B}$ ε-approximates $\mathcal{A}$ over *infinite* words for every precision ε (they are equivalent), while it does not ε-approximates $\mathcal{A}$ over *finite* words for any precision $\varepsilon < 1$.

Following Lemma 10, it is enough to prove the correctness of the constructions with respect to finite words, and the lower bounds with respect to infinite words.

4.1 Approximation by Unfolding

We formalize below the naive approach of unfolding the automaton computation up to a sufficient level.

The Construction. Given an NDA $\mathcal{A} = \langle \Sigma, Q, Q_{in}, \delta, \gamma, \lambda \rangle$ and a parameter $l \in \mathbb{N}$, we construct a DDA $\mathcal{D}$ that is the depth-l unfolding of $\mathcal{A}$. We later fix the value of l to obtain a DDA that approximates $\mathcal{A}$ with a desired precision ε.

The DDA is $\mathcal{D} = \langle \Sigma, Q', q'_{in}, \delta', \gamma', \lambda \rangle$ where:

- $Q' = \Sigma^l$; the set of words of length l.
- q'_{in} = the empty word.
- $\delta' = \{(w, \sigma, w \cdot \sigma) \bigm| |w| \leq l - 1 \wedge \sigma \in \Sigma\} \cup \{(w, \sigma, w) \bigm| |w| = l \wedge \sigma \in \Sigma\}$.
- For all $w \in \Sigma^{\leq l-1}$, and $\sigma \in \Sigma$, let $\gamma'(w, \sigma, w \cdot \sigma) = (\mathcal{A}(w \cdot \sigma) - \mathcal{A}(w))/\lambda^{|w|}$; for all $w \in \Sigma^l$, and $\sigma \in \Sigma$, let $\gamma'(w, \sigma, w) = \frac{v+V}{2}$ where v and V are the smallest and largest weights in $\mathcal{A}$, respectively.

◀

The above construction yields an automaton whose state space might be doubly exponential in the representation of the discount factor.

▶ **Theorem 11.** *Consider a precision $\varepsilon = 2^{-p}$ and an NDA $\mathcal{A}$ with a discount factor $\lambda = 1 + 2^{-k}$ and maximal weight difference of m. Then applying the unfolding construction on $\mathcal{A}$, for a precision ε, generates a DDA $\mathcal{D}$ that ε-approximates $\mathcal{A}$ with up to $2^{\Theta(2^k(k+p+\log m))}$ states.*

4.2 Approximation by Gap Rounding

We provide below an approximate-determinization construction that is singly exponential in the representations of the precision, in the discount factor, and in the number of states.

We start with defining the "cost" and the "gap" of a state, to be used in the determinization construction. The *cost* of reaching a state q of an NDA $\mathcal{A}$ over a finite word u is $\mathsf{cost}(q, u) = \min\{\gamma(r) \mid r \text{ is a run of } \mathcal{A} \text{ on } u \text{ ending in } q\}$, where $\min \emptyset = \infty$. The *gap* of q over a finite word u is $\mathsf{gap}(q, u) = \lambda^{|u|}(\mathsf{cost}(q, u) - \mathcal{A}(u))$. Note that when $\mathcal{A}$ operates over infinite words, we interpret $\mathcal{A}(u)$, for a finite word u, as if $\mathcal{A}$ was operating over finite words.

Intuitively, the gap of a state q over a word u stands for the weight that a run starting in q should save, compared to a run starting in u's optimal ending state, in order to make q's path optimal. A gap of a state q over a finite word u is said to be *recoverable* if there is a suffix that makes this path optimal; that is, if there is a word w, such that $\mathsf{cost}(q, u) + \frac{\mathcal{A}^q(w)}{\lambda^{|u|}} = \mathcal{A}(uw)$. The suffix w should be finite/infinite, depending on whether $\mathcal{A}$ operates over finite/infinite words.

The main idea in the determinization procedure of [6] is to extend the subset construction by keeping a recoverable-gap value to each element of the subset. It is shown [6] that for every integral factor the procedure is guaranteed to terminate, while for every non-integral rational factor it might not terminate. The problem with non-integral factors is that the recoverable-gaps might be arbitrarily dense, implying infinitely many gaps within the maximal bound of recoverable gaps.

Our approximation scheme generalizes the determinization procedure of [6], rounding the stored gaps to a fixed resolution. Since there is a bound on the maximal value of a recoverable gap, the fixed resolution guarantees the procedure's termination.

The question is, however, how an unbounded number of gap rounding allows for the required precision. The key observation is that the rounding is also discounted along the computation. For a λ-NDA, where $\lambda = 1 + 2^{-k}$, and a precision $\varepsilon = 2^{-p}$, we show that a resolution of $2^{-(p+k-1)}$ is sufficient. For an NDA whose maximal weight difference is m, the maximal recoverable gap is below $m2^{k+1}$. Hence, for an NDA with n states, the resulting DDA would have up to $2^{n(p+2k+\log m)}$ states.

The construction is formalized below, and illustrated with an example in Figure 2.)

The Construction. Consider a discount factor $\lambda = 1 + 2^{-k}$, with $k > 0$, and an NDA $\mathcal{A} = \langle \Sigma, Q = \langle q_1, \ldots, q_n \rangle, Q_{in}, \delta, \gamma, \lambda \rangle$, in which the maximal difference between the weights is m. For simplicity, we extend γ with $\gamma(\langle q_i, \sigma, q_j \rangle) = \infty$ for every $\langle q_i, \sigma, q_j \rangle \notin \delta$. Note that our discounted-sum automata do not have infinite weights; it is only used as an internal element of the construction.

For a precision $\varepsilon = 2^{-p}$, with $p > 0$, we construct a DDA $\mathcal{D} = \langle \Sigma, Q', q'_{in}, \delta', \gamma', \lambda \rangle$ that ε-approximates $\mathcal{A}$. We first define the set $G = \{i2^{-(p+k-1)} \mid i \in \mathbb{N} \text{ and } i \leq m2^{p+2k+1}\} \cup \{\infty\}$ of recoverable-gaps. The ∞ element denotes a non-recoverable gap, and behaves as the standard infinity element in the arithmetic operations that we will be using.

A state of $\mathcal{D}$ extends the standard subset construction by assigning a recoverable gap to each state of $\mathcal{A}$. That is, $Q' = \{\langle g_1, \dots, g_n\rangle \mid \text{for every } 1 \le h \le n, g_h \in G\}$.

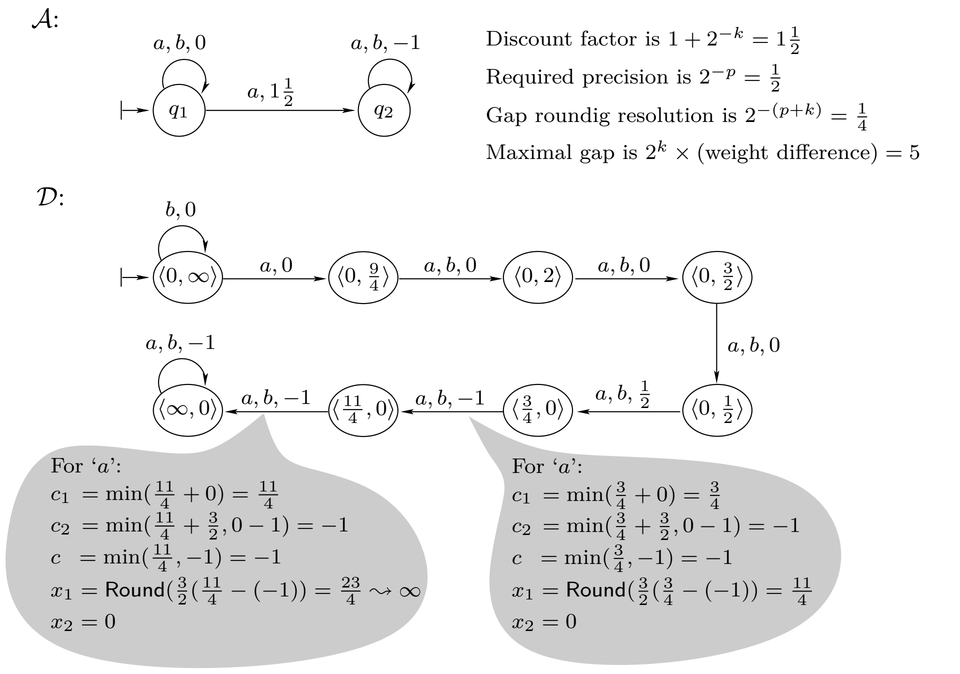
The initial state of $\mathcal{D}$ is $q'_{in} = \langle g_1, \dots, g_n\rangle$, where for every $1 \le i \le n$, $g_i = 0$ if $q_i \in Q_{in}$ and $g_i = \infty$ otherwise.

For bounding the number of possible states, the gaps are rounded to a resolution of $2^{-(p+k-1)}$. Formally, for every number $x \ge 0$, we define $\mathsf{Round}(x) = i2^{-(p+k-1)}$, such that $i \in \mathbb{N}$ and for every $j \in \mathbb{N}, |x - i2^{-(p+k-1)}| \le |x - j2^{-(p+k-1)}|$.

For every state $q' = \langle g_1, \dots, g_n\rangle \in Q'$ and letter $\sigma \in \Sigma$, we define the transition function $\delta(q', \sigma) = q'' = \langle x_1, \dots, x_n\rangle$, and the weight function $\gamma(\langle q', \sigma, q''\rangle) = c$ as follows.

- For every $1 \le h \le n$, we set $c_h := \min\{g_j + \gamma(\langle q_j, \sigma, q_h\rangle) \mid 1 \le j \le n\}$
- $c := \min\limits_{1 \le h \le n} (c_h)$
- For every $1 \le l \le n$,
  - $x_l := \mathsf{Round}(\lambda(c_h - c))$;
  - if $x_l \ge m2^{k+1}$ then $x_l := \infty$

◀


$\mathcal{A}$:
$a, b, 0$
$a, 1\frac{1}{2}$
$a, b, -1$
q_1
q_2
$\mathcal{D}$:
$b, 0$
$\langle 0, \infty\rangle$
$a, 0$
$\langle 0, \frac{9}{4}\rangle$
$a, b, 0$
$\langle 0, 2\rangle$
$a, b, 0$
$\langle 0, \frac{3}{2}\rangle$
$a, b, 0$
$\langle 0, \frac{1}{2}\rangle$
$a, b, \frac{1}{2}$
$\langle \frac{3}{4}, 0\rangle$
$a, b, -1$
$\langle \frac{11}{4}, 0\rangle$
$a, b, -1$
$\langle \infty, 0\rangle$
$a, b, -1$

Discount factor is $1 + 2^{-k} = 1\frac{1}{2}$
Required precision is $2^{-p} = \frac{1}{2}$
Gap roundig resolution is $2^{-(p+k)} = \frac{1}{4}$
Maximal gap is $2^k \times$ (weight difference) $= 5$

For 'a':
$c_1 = \min(\frac{11}{4} + 0) = \frac{11}{4}$
$c_2 = \min(\frac{11}{4} + \frac{3}{2}, 0 - 1) = -1$
$c = \min(\frac{11}{4}, -1) = -1$
$x_1 = \mathsf{Round}(\frac{3}{2}(\frac{11}{4} - (-1)) = \frac{23}{4} \rightsquigarrow \infty$
$x_2 = 0$

For 'a':
$c_1 = \min(\frac{3}{4} + 0) = \frac{3}{4}$
$c_2 = \min(\frac{3}{4} + \frac{3}{2}, 0 - 1) = -1$
$c = \min(\frac{3}{4}, -1) = -1$
$x_1 = \mathsf{Round}(\frac{3}{2}(\frac{3}{4} - (-1)) = \frac{11}{4}$
$x_2 = 0$

Figure 2 Determinizing the NDA $\mathcal{A}$ approximately into the DDA $\mathcal{D}$. The gray bubbles detail some of the intermediate calculations of the approximate-determinization construction.

The correctness and the state complexity of the construction is formalized below.

▶ **Theorem 12.** *Consider a discount factor $\lambda = 1 + 2^{-k}$ and a precision $\varepsilon = 2^{-p}$, for some positive numbers p and k. Then for every λ-NDA $\mathcal{A}$ with n states and weight difference of up to m, there is a λ-DDA that ε-approximates $\mathcal{A}$ with up to $2^{n(p+k+\log m)}$ states. The automata $\mathcal{A}$ and $\mathcal{D}$ may operate over finite words as well as over infinite words.*

Proof sketch. We show the correctness for finite words, implying, by Lemma 10, also the correctness for infinite words.

We start by proving the claim with respect to an *infinite-state* automaton that is constructed as above, except for not changing any gap to ∞.

We use the following notations: upon reading a word w, the constructed automaton yields the sequence $c_1, c_2, \dots, c_{|w|}$ of weights, and reaches a state $\langle g_{1,w}, \dots, g_{n,w} \rangle$. We denote half the gap resolution, namely $2^{-(p+k)}$, by r.

Intuitively, $\frac{g_{h,w}}{\lambda^{|w|}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}}$ stands for the approximated cost of reaching the state q_h upon reading the word w. We define for every word w and $1 \leq h \leq n$, the "mistake" in the approximated cost by $M(h,w) = \frac{g_{h,w}}{\lambda^{|w|}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_h, w)$, and (delicately) show by induction on the length of w that $|M(h,w)| \leq \sum_{i=1}^{|w|} \frac{r}{\lambda^i}$.

We conclude by showing that changing all gaps that exceed $m2^{k+1}$ to ∞ does not harm the correctness.

◀

4.3 Lower Bounds

The upper bound described in Section 4.2, for determinizing an NDA $\mathcal{A}$ approximately, exponentially depends on three parameters: n, denoting the number of states in $\mathcal{A}$; k, representing the proximity of $\mathcal{A}$'s discount factor to 1; and p, representing the precision. We show below that exponential dependency on these three factors is unavoidable.

For showing dependency on the precision (p), as well as on the discount factor (k), one can fix the other two parameters and show exponential dependency on the varying parameter. This is formalized in Theorems 13 and 14.

As for the number of states (n), there is no absolute dependency – one may approximate the non-deterministic automaton via the unfolding approach (Section 4.1), having no dependency on n. Yet, the trade off is a double-exponential dependency on k. Thus, one may check whether there is an exponential dependency on n, once p and k are fixed, and n remains below $O(2^k)$. This is indeed the case, as shown in Theorem 15.

▶ **Theorem 13.** *There is an NDA $\mathcal{A}$ with two states, such that for every $\varepsilon = 2^{-p} > 0$, every DDA (with any discount factor) that ε-approximates $\mathcal{A}$ has at least $2^{\lfloor \frac{p-2}{\log 3} \rfloor}$ states.*

▶ **Theorem 14.** *There is a family of NDAs, $\mathcal{A}_k$, for $k \geq 2$, with two states and discount factor $1+2^{-k}$ over an alphabet of two letters, such that every DDA (with any discount factor) that $\frac{1}{8}$-approximates $\mathcal{A}_k$ has at least 2^{k-1} states.*

▶ **Theorem 15.** *For every $k \geq 3$, there is a family of NDAs, $\mathcal{A}_n$, for $n \leq 2^k$, with $n+1$ states and discount factor $1+2^{-k}$ over an alphabet of size $2n+1$, such that every DDA (with any discount factor) that $\frac{1}{12}$-approximates $\mathcal{A}_n$ has at least 2^n states.*

Proof sketch for Theorems 13-15. Intuitively, if a state of an NDA has two recoverable gaps that are different enough, over two words that are short enough, then recovering them by the same suffix would yield two values that are also different enough. Two such words must lead to two different states in a deterministic automaton that properly approximates the NDA. Hence, the challenge is to figure out an NDA whose states have as many such different recoverable gaps as possible.

Each of the three lower bounds brings a different challenge, depending on the parameter that is not fixed (precision, discount factor, and number of states). A delicate analysis of the recoverable gaps in the automata of Figures 3, 6, and 7 (the latter two are in the Appendix), provides the proofs of Theorems 13-15, respectively. ◀

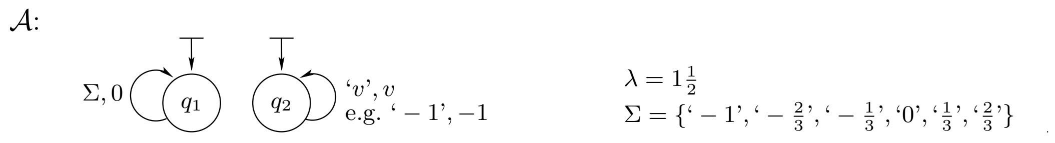

$\mathcal{A}$:
$\Sigma, 0$
q_1
q_2
'v', v
e.g. '-1', -1
$\lambda = 1\frac{1}{2}$
$\Sigma = \{`-1', `-\frac{2}{3}', `-\frac{1}{3}', `0', `\frac{1}{3}', `\frac{2}{3}'\}$

Figure 3 Every DDA that 2^{-p}-approximates the NDA $\mathcal{A}$ has at least $2^{\lfloor \frac{p-2}{log 3} \rfloor}$ states.

5 Average and Limit-Average Automata

There are two main differences between summation and averaging, leading to different results and proofs regarding sum and average/limit-average automata. On the one hand, averaging adds a computation layer on top of summation, for which reason average and limit-average automata cannot be determinized approximately with respect to any distance function, even if restricting their weights. On the other hand, average and limit-average automata yield dense sets of values. Therefore, the undecidability of their universality problem does not imply that they cannot be effectively determinized approximately into *another* automaton class with decidable properties. The question of whether they can, or cannot, is left open.

5.1 Average Automata

Average automata are inherently different from sum automata by not falling into the class of automata based on a semi-ring (the average function does not have an identity element). Their universality problem is undecidable, directly following from the undecidability of the universality problem of sum automata, as for every sequence π of numbers, $Sum(\pi) \leq 0$ if and only if $Avg(\pi) \leq 0$. Yet, it does not imply that average automata cannot be determinized approximately – the corresponding proof that relates to sum automata (Theorem 4) relies on the fixed minimal distance between two values of a sum automaton, which is not the case with average automata. Nevertheless, a different proof is used to show that average automata cannot be determinized approximately.

▶ **Theorem 16.** *Average automata over any set of weights (with at least two distinct weights) cannot be determinized approximately with respect to any ordered distance function.*

Proof sketch. Let $\mathcal{B}$ be an average automaton with the structure of the automaton $\mathcal{A}$ of Figure 5, but replacing the 0-weights with an arbitrary value x and replacing the 1-weights with some value $y > x$. We define $\varepsilon > 0$ to be small enough with respect to the distance (according to the required distance function) between the average of x and y and some values near it. We then show, by way of contradiction, that there is no deterministic average automaton that can ε-approximate $\mathcal{B}$.

In the proof, we use some sort of a pumping argument for constructing two words, w_1 and w_2, of about the same length with mostly a's and mostly b's, respectively, such that the value of the deterministic automaton on them is far enough from the average between x and y. The word that combines w_1 and w_2 provides a counter example for the assumption that the deterministic automaton properly approximates $\mathcal{B}$. ◀

5.2 Limit-Average Automata

As average automata, limit-average automata cannot be determinized approximately with respect to any distance function, and no weight restriction can overcome it. The proof is different from the one for average automata, providing a slightly stronger result, namely

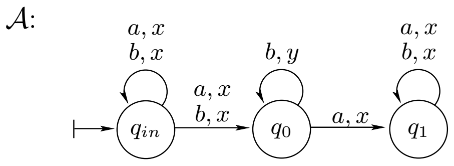


Figure 4 A limit-average automaton having the value y for a word with finitely many a's and x otherwise.

that limit-average automata cannot be determinized approximately even with respect to unordered distance functions.

▶ **Theorem 17.** *Limit-average automata over any set of weights (with at least two distinct weights) cannot be determinized approximately with respect to any distance function.*

Proof sketch. We use the automaton $\mathcal{A}$, defined in Figure 4, with arbitrary weights $x < y$, which generalizes the automaton used in [8], for showing that limit-average automata cannot be determinized exactly. The automaton realizes the function "$\mathcal{A}(w) = y$ if the word w contains finitely many a's, and $\mathcal{A}(w) = x$ otherwise".

Considering, by way of contradiction, a deterministic automaton that properly approximates it, we analyze the possible average weights of its cycles along words with only a's and only b's. We then construct a corresponding word in which the frequency of a's decays exponentially, and show that the value of the deterministic automaton for it is not close enough to the required value. ◀

6 Conclusions

Automata comparison and game solving are critical tasks in formal verification and synthesis. Both require, or are closely related to, automata determinization. Quantitative automata play a key role in the emerging area of quantitative formal methods, but, unfortunately, they usually cannot be determinized. For that reason, quantitative verification and synthesis are generally undecidable. It is thus natural to ask for approximate solutions, based on approximate automata determinization. Moreover, having automata with real values, rather than boolean ones, brings a natural potential for such approximations.

With discounted-sum automata, we gave a construction fulfilling this potential. It can be used for systems with inherent discounting, as well as for other systems, if willing to introduce some discounting. With automata that are based on an accumulation such as sum or product, weight restriction allows for approximate determinization with respect to some distance functions. On the other hand, automata that are based on averaging or limit-averaging cannot be determinized approximately with respect to any distance function, and no weight restriction (with at least two different weights) can overcome this.

Acknowledgment. We thank Laurent Doyen for great ideas and valuable help in analyzing discounted-sum automata.

References

1 L. de Alfaro, T. A. Henzinger, and R. Majumdar. Discounting the future in systems theory. In *Proc. 30th ICALP conference*, pages 1022–1037, 2003.
2 S. Almagor, U. Boker, and O. Kupferman. What's decidable about weighted automata? In *Proc. of ATVA11*, pages 482–491, 2011.

3 R. Alur and A. Trivedi. Relating average and discounted costs for quantitative analysis of timed systems. In *Proc. 11th EMSOFT conference*, pages 165–174, 2011.
4 B. Aminof, O. Kupferman, and R. Lampert. Rigorous approximated determinization of weighted automata. In *Proc. 26th LICS symposium*, pages 345–354, 2011.
5 R. Bloem, K. Chatterjee, T. A. Henzinger, and B. Jobstmann. Better quality in synthesis through quantitative objectives. In *Proc. 21st CAV conference*, pages 140–156, 2009.
6 U. Boker and T. A. Henzinger. Determinizing discounted-sum automata. In *Proc. 20th Annual Conf. of the European Association for Computer Science Logic*, 2011.
7 A. L. Buchsbaum, R. Giancarlo, and J. Westbrook. An approximate determinization algorithm for weighted finite-state automata. *Algorithmica*, 30(4):503–526, 2001.
8 K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. In *Proc. of CSL*, LNCS 5213, pages 385–400. Springer, 2008.
9 K. Chatterjee, L. Doyen, and R. Singh. On memoryless quantitative objectives. In *Proc. 18th FCT symposium*, pages 148–159, 2011.
10 K. Chatterjee, M. Randour, and J.F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. In *Proc. 23rd CONCUR conference*, 2012.
11 A. Degorre, L. Doyen, R. Gentilini, J. F. Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In *Proc. of CSL*, pages 260–274, 2010.
12 M. Droste, W. Kuich, and H. Vogler. *Handbook of Weighted Automata.* Springer Publishing Company, Incorporated, 2009.
13 C. von Essen and B. Jobstmann. Synthesizing systems with optimal average-case behavior for ratio objectives. In *Proc. of iWIGP workshop*, pages 17–32, 2011.
14 T. A. Henzinger and N. Piterman. Solving games without determinization. In *Proc. 15th Annual Conf. of the European Association for Computer Science Logic*, pages 395–410, 2006.
15 O. Kupferman and Y. Lustig. Lattice automata. In *proc. of 8th VMCAI conf.*, pages 199–213, 2007.
16 U. Zwick and M.S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.

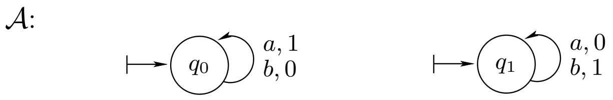


Figure 5 The sum automaton computing the minimum between a's and b's.

A Appendix – Proofs

A.1 Proof of Lemma 2

For every sequence $x_1, x_2, \ldots, x_n$ of weights, we have $\prod_{i=0}^{n} 2^{-x_i} = 2^{-\sum_{i=0}^{n} x_i}$. Hence, for every run r of $\mathcal{A}$ and $\mathcal{B}$ (recall that they are identical up to the transition weights), $\mathcal{B}$'s value for r is $2^{-(\mathcal{A}\text{'s value for } r)}$. Since the weight mapping, $x \mapsto 2^{-x}$, preserves the order on values (up to replacing between $\leq$ and $\geq$), the best (minimal) run of $\mathcal{A}$ is also the best (maximal) run of $\mathcal{B}$. Therefore, for every word w, $\mathcal{B}(w) = 2^{-\mathcal{A}(w)}$. ◀

A.2 Proof of Theorem 4

Assume, by contradiction, that sum automata can be determinized approximately with respect to an ordered distance function d into a class $\mathfrak{C}$ of automata whose universality problem is decidable.

Consider a two-counter machine $\mathcal{M}$. By Lemma 3, there is a sum automaton $\mathcal{A}$ with weights in $\{-1, 0, 1\}$ such that M halts iff there is a word w such that $\mathcal{A}(w) \geq 1$. Let $\varepsilon = \min(d(0, \frac{1}{2}), d(\frac{1}{2}, 1))$. By the assumption, there is an automaton $\mathcal{C} \in \mathfrak{C}$ such that for every word w, $d(\mathcal{A}(w), \mathcal{C}(w)) \leq \varepsilon$.

Now, assume a negative answer to $\mathcal{C}'s$ universality question, implying that there is a word w such that $\mathcal{C}(w) > \frac{1}{2}$. Since d is an ordered distance function and $d(\mathcal{A}(w), \mathcal{C}(w)) < d(0, \frac{1}{2})$, it follows that $\mathcal{A}(w)$ cannot be smaller than or equal to 0. Hence, $\mathcal{A}(w) = 1$ and $\mathcal{M}$ halts. Analogously, a negative answer to $\mathcal{C}'s$ universality question implies that for every word w, $\mathcal{C}(w) \leq \frac{1}{2}$. Since d is an ordered distance function and for every w, $d(\mathcal{A}(w), \mathcal{C}(w)) < d(\frac{1}{2}, 1)$, it follows that for every w, $\mathcal{A}(w)$ cannot be equal to 1. Hence, $\mathcal{M}$ does not halt. We have reached a contradiction, showing the decidability of the two-counter machine halting problem. ◀

A.3 Proof of Theorem 6

Consider the automaton $\mathcal{A}$ defined in Figure 5. Assume, by contradiction, a deterministic sum automaton $\mathcal{D}$ that 1-approximates $\mathcal{A}$ with respect to difference. There must be states s_1 and s_2 in $\mathcal{D}$ and four words, u_1, u_2, u_3, and u_4, such that u_2 has only a's, u_4 has only b's, $\mathcal{D}$ reaches s_1 reading u_1, $\mathcal{D}^{s_1}$ reaches s_1 reading u_2, $\mathcal{D}^{s_1}$ reaches s_2 reading u_3, and $\mathcal{D}^{s_2}$ reaches s_2 reading u_4.

We claim that the accumulated sum of the weights on the path from s_1 back to itself reading u_2 must be 0. Indeed, if it is negative there is a number $z \in \mathbb{N}$ such that $\mathcal{D}(u_1 \cdot {u_2}^z) < -1 < \mathcal{A}(u_1 \cdot {u_2}^z) - 1$, while if it is positive there is a number $z \in \mathbb{N}$ such that $\mathcal{D}(u_1 \cdot {u_2}^z) >$ (number of b's in u_1) $+ 1 = \mathcal{A}(u_1 \cdot {u_2}^z) + 1$. Analogously, the accumulated sum of the weights on the path from s_2 back to itself reading u_4 must be 0.

Let x be the accumulated sum of the weights on the path from the initial state to s_2 reading $u_1 \cdot u_3$. We get that for every number $z \in \mathbb{N}$, $\mathcal{D}(u_1 \cdot {u_2}^z \cdot u_3 \cdot {u_4}^z) = x$, while $\mathcal{A}(u_1 \cdot {u_2}^z \cdot u_3 \cdot {u_4}^z) \geq z$, implying that $\mathcal{D}$ does not 1-approximates $\mathcal{A}$. ◀

A.4 Proof of Corollary 7

It is a consequence of Theorem 6 and Lemma 2, using the same arguments as in the proof of Theorem 9. ◀

A.5 Proof of Theorem 8

Consider a precision $\varepsilon > 0$, and a product automaton $\mathcal{A} = \langle \Sigma, Q = \langle q_1, \dots, q_n \rangle, Q_{in}, \delta, \gamma \rangle$ over weights in $[0,1]$. For simplicity, we extend γ with $\gamma(\langle q_i, \sigma, q_j \rangle) = 0$ for every $\langle q_i, \sigma, q_j \rangle \notin \delta$.

We inductively construct a deterministic product automaton $\mathcal{D} = \langle \Sigma, Q', q'_{in}, \delta', \gamma' \rangle$ that ε-approximates $\mathcal{A}$, via the intermediate automata $\mathcal{D}_i = \langle \Sigma, Q'_i, q'_{in}, \delta'_i, \gamma'_i \rangle$.

The initial state of all $\mathcal{D}_i$s is $q'_{in} = \langle p_1, \dots, p_n \rangle$, where for every $1 \le i \le n$, $p_i = 1$ if $q_i \in Q_{in}$ and $p_i = 0$ otherwise.

We start with $\mathcal{D}_1$, in which $Q'_1 = \{q'_{in}\}$, $\delta'_1 = \emptyset$ and $\gamma'_1 = \emptyset$, and proceed from $\mathcal{D}_i$ to $\mathcal{D}_{i+1}$, such that $Q'_i \subseteq Q'_{i+1}$, $\delta'_i \subseteq \delta'_{i+1}$ and $\gamma'_i \subseteq \gamma'_{i+1}$. The construction is completed once $\mathcal{D}_i = \mathcal{D}_{i+1}$, finalizing the desired deterministic automaton $\mathcal{D} = \mathcal{D}_i$.

In the induction step, $\mathcal{D}_{i+1}$ extends $\mathcal{D}_i$ by (possibly) adding, for every state $q' = \langle p_1, \dots, p_n \rangle \in Q'_i$ and letter $\sigma \in \Sigma$, a state $q'' = \langle p''_1, \dots, p''_n \rangle$, a transition $\langle q', \sigma, q'' \rangle$ and a weight $\gamma'_{i+1}(\langle q', \sigma, q'' \rangle) = c$, as follows:

- For every $1 \le h \le n$, $p''_h := \max\{p_j \cdot \gamma'_i(\langle q_j, \sigma, q_h \rangle) \bigm| 1 \le j \le n\}$
- For every $1 \le h \le n$, if $p''_h < \varepsilon$ then $p''_h := 0$
- $Q'_{i+1} := Q'_{i+1} \cup q''$
- $\delta'_{i+1} := \delta'_{i+1} \cup \langle q', \sigma, q'' \rangle$
- $\gamma'_{i+1}(\langle q', \sigma, q'' \rangle) = \frac{\max\{p_j \bigm| 1 \le j \le n\}}{\max\{p''_h \bigm| 1 \le h \le n\}}$, taking 0 in the case that the denominator is zero.

We claim that the construction always terminates, as there are finitely many possible values for $\langle p_1, \dots, p_n \rangle$ in the state representation. Indeed, let t be the set of weights in $\mathcal{A}$ that differ from 1. Then a sequence of t-weights multiplications whose value is bigger than ε cannot be longer than $\log_{\max(t)} \varepsilon$. Thus, each element in the above tuple can have up to $|t|^{\log_{\max(t)} \varepsilon}$ different values.

◀

A.6 Proof of Theorem 9

Consider a sum automaton $\mathcal{A}$ over nonnegative weights and a precision ε. We construct from $\mathcal{A}$ a product automaton $\mathcal{B}$ over weights in $[0,1]$, by changing every weight x to 2^{-x}. By Theorem 8, there is a deterministic product automaton $\mathcal{B}'$ over $[0,1]$ such that for every word w, $|\mathcal{B}(b) - \mathcal{B}'(w)| < \varepsilon$. We now construct from $\mathcal{B}'$ a deterministic sum automaton $\mathcal{A}'$, by changing every weight y to $(-\log y)$.

By Lemma 2, for every word w, $\mathcal{B}(w) = 2^{-\mathcal{A}(w)}$ and $\mathcal{B}'(w) = 2^{-\mathcal{A}'(w)}$. Thus, for every word w, $d(\mathcal{A}(w), \mathcal{A}'(w)) = |2^{-\mathcal{A}(w)} - 2^{-\mathcal{A}'(w)}| = |\mathcal{B}(w) - \mathcal{B}'(w)| < \varepsilon$. Therefore, $\mathcal{A}'$ ε-approximates $\mathcal{A}$. ◀

A.7 Proof of Lemma 10

Assume, by contradiction, a precision $\varepsilon > 0$, a discount factor $\lambda > 1$, and two λ-NDAs, $\mathcal{A}$ and $\mathcal{B}$, such that $\mathcal{B}$ ε-approximates $\mathcal{A}$ with respect to finite words but not with respect to infinite words.

Then there is an infinite word w, such that $|\mathcal{A}(w) - \mathcal{B}(w)| - \varepsilon = c > 0$. Let m be the maximal difference between a weight in $\mathcal{A}$ and a weight in $\mathcal{B}$. Since for every $1 < \lambda$,

$\sum_{i=0}^{\infty}(\frac{1}{\lambda^i}) = \frac{1}{1-\frac{1}{\lambda}} = \frac{\lambda}{\lambda-1}$, it follows that the difference between the values that $\mathcal{A}$ and $\mathcal{B}$ assign to any (finite or infinite) word is smaller or equal to $\frac{m\lambda}{\lambda-1}$. Hence, the difference between the values of their runs on suffixes of w, starting at a position p, is smaller or equal to $\frac{m\lambda}{(\lambda-1)\lambda^p}$.

Now, since $\mathcal{B}$ ε-approximates $\mathcal{A}$ over finite words, it follows that they have optimal runs over every prefix of w, such that their difference is smaller than or equal to ε. Thus, after a long enough prefix, of length p such that $\frac{m\lambda}{(\lambda-1)\lambda^p} < c$, the difference between the values of $\mathcal{A}$'s and $\mathcal{B}$'s optimal runs on w must be smaller than c, leading to a contradiction.

A counter example for the converse is provided in Figure 1.

◀

A.8 Proof of Theorem 11

We start with an interesting observation on discounting and half life time: for every $K > 1$, the half life time with respect to the discount factor $1 + \frac{1}{K}$, meaning the number of time units before the discounting gets to 2, is roughly K. More precisely, as K tends to infinity, $(1 + \frac{1}{K})^K$ is exactly $e\,(\approx 2.72)$. Note that we can take advantage of this property, as we represent the discount factor by $1 + 2^{-k}$, which equals to $1 + \frac{1}{K}$, for $K = 2^k$. For our purposes, we show in Lemma 18 below that $(1 + \frac{1}{K})^K$ is always between 2 and 3, as well as a corresponding bound for $\log(1 + \frac{1}{K})$.

▶ **Lemma 18.** *For every $K \geq 2$, we have:*
1. $1 < K\log(1 + \frac{1}{K}) < \frac{3}{2}$.
2. $2 < (1 + \frac{1}{K})^K < 3$.

Proof.
1. We use the Mercator series, which is the Taylor series for the natural logarithm, stating that for every $-1 < x \leq 1$, $\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \ldots$. Setting $x = \frac{1}{K}$, we get that $\ln(1 + \frac{1}{K}) = \frac{1}{K} - \frac{1}{2K^2} + \frac{1}{3K^3} - \frac{1}{4K^4} + \ldots$. Thus,
$$K\ln(1 + \frac{1}{K}) = 1 - \frac{1}{2K} + \frac{1}{3K^2} - \frac{1}{4K^3} + \frac{1}{5K^4} - \frac{1}{6K^5} + \ldots.$$
Since for every positive integer i, $(-\frac{1}{iK^{(i-1)}} + \frac{1}{(i+1)K^i}) < 0$, it follows that the above series is smaller than 1. Analogously, since for every positive integer i, $\frac{1}{iK^{(i-1)}} - \frac{1}{(i+1)K^i} > 0$, it follows that the above series is bigger than $1 - \frac{1}{2K}$. Hence, $1 - \frac{1}{2K} < K\ln(1 + \frac{1}{K}) < 1$. Therefore, as k tends to infinity, $K\log(1 + \frac{1}{K})$ converges to $\log e$, where e is Euler's constant. Specifically, for every $K \geq 2$, we have $1 < K\log(1 + \frac{1}{K}) < \frac{3}{2}$.
2. Let $z = \log(1 + \frac{1}{K})$. we have $(1 + \frac{1}{K})^K = (1 + \frac{1}{K})^{\frac{zK}{z}} = ((1 + \frac{1}{K})^{\frac{1}{z}})^{zK} = 2^{zK}$.
From the first part of the lemma, we know that $\frac{1}{K} < z < \frac{3}{2K}$. Thus, $2 = 2^{\frac{K}{K}} < 2^{zK} < 2^{\frac{3K}{2K}} < 3$. Hence, $2 < (1 + \frac{1}{K})^K < 3$.

◀

We continue with the theorem's proof.

Proof of Theorem 11. Let l be the depth of $\mathcal{A}$'s unfolding that is used for generating $\mathcal{D}$. Then, for all words $w \in \Sigma^{\leq l}$, the automata $\mathcal{A}$ and $\mathcal{D}$ agree, by definition, on the value of w, that is $\mathcal{A}(w) = \mathcal{D}(w)$. For longer, or infinite, words $w \in \Sigma^{>l} \cup \Sigma^{\omega}$, we have:

$$\mathcal{D}(w) = \mathcal{A}(w[0\ldots l-1]) + \frac{v+V}{2}\sum_{i=l}^{|w|}\frac{1}{\lambda^i}.$$

As $v \cdot \sum_{i=l}^{|w|} \frac{1}{\lambda^i} \leq \mathcal{A}(w) - \mathcal{A}(w[0 \ldots l-1]) \leq V \cdot \sum_{i=l}^{|w|} \frac{1}{\lambda^i}$, we obtain the following:

$$|\mathcal{A}(w) - \mathcal{D}(w)| \leq \frac{V-v}{2} \sum_{i=l}^{|w|} \frac{1}{\lambda^i} \leq \frac{V-v}{2} \sum_{i=l}^{\infty} \frac{1}{\lambda^i} = \frac{V-v}{2} \frac{1}{\lambda^l} \sum_{i=0}^{\infty} \frac{1}{\lambda^i} = \frac{m}{2\lambda^{l-1}(\lambda-1)} ,$$

where $m = V - v$ is the largest weight difference in $\mathcal{A}$.

Note that the above inequality is tight, in the sense that there is an automaton $\mathcal{A}$ and (an infinite) word w, such that $|\mathcal{D}(w) - \mathcal{A}(w)| = \frac{m}{2\lambda^{l-1}(\lambda-1)}$.

In order to compute the minimal unfolding depth l that guarantees a precision $\varepsilon = 2^{-p}$ when determinizing an automaton with a discount factor $\lambda = 1 + 2^{-k}$, we should solve the following inequality $\frac{m}{2\lambda^{l-1}(\lambda-1)} = \frac{m2^{k-1}}{\lambda^{l-1}} = \frac{m2^{k-1}}{(1+2^{-k})^{l-1}} \leq 2^{-p}$.

Hence, $m2^{k+p-1} \leq (1 + 2^{-k})^{l-1}$. Therefore, $(l-1)\log(1 + 2^{-k}) \geq k + p + \log m - 1$, yielding that $l \geq \frac{k+p+\log m-1}{\log(1+2^{-k})} + 1$.

By Lemma 18, we have $\log(1 + 2^{-k})$ is linear in 2^{-k}. Hence, $l \geq \Theta(2^k(k + p + \log m))$.

The unfolding construction of $\mathcal{D}$ generates up to Σ^l states, implying that the deterministic automaton has up to $2^{\Theta(2^k(p+\log m))}$ states. ◀

A.9 Proof of Theorem 12

For every such NDA $\mathcal{A}$, we construct the DDA $\mathcal{D}$ as defined in the construction of Section 4.2. We show below that for every finite word w, $|\mathcal{A}(w) - \mathcal{D}(w)| \leq \varepsilon$. By Lemma 10, it also implies the correctness for infinite words.

We start by showing the correctness with respect to an *infinite*-state automaton $\mathcal{D}'$ that is constructed as $\mathcal{D}$, except for not changing any gap to ∞. That is, the state space of $\mathcal{D}'$ is $\{\langle g_1, \ldots, g_n\rangle \mid \text{for every } 1 \leq h \leq n,\ g_h = i2^{-(p+k-1)} \text{ for some } i \in \mathbb{N}\}$. Afterwards, we shall argue that changing all gaps that exceed $m2^{k+1}$ to ∞ does not harm the correctness.

We use the following notations: upon reading a word w, the automaton $\mathcal{D}'$ yields the sequence $c_1, c_2, \ldots, c_{|w|}$ of weights, and reaches a state $\langle g_{1,w}, \ldots, g_{n,w}\rangle$. We denote half the gap resolution, namely $2^{-(p+k)}$, by r.

Intuitively, $\frac{g_{h,w}}{\lambda^{|w|}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}}$ stands for the approximated cost of reaching the state q_h upon reading the word w. We define for every word w and $1 \leq h \leq n$, the "mistake" in the approximated cost by $M(h, w) = \frac{g_{h,w}}{\lambda^{|w|}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_h, w)$, and show by induction on the length of w that $|M(h, w)| \leq \sum_{i=1}^{|w|} \frac{r}{\lambda^i}$.

The assumptions obviously hold for the initial step, where w is the empty word and all values are 0. As for the induction step, we assume they hold for w and show that for every $\sigma \in \Sigma$, they hold for $w{\cdot}\sigma$.

We first handle the case that $M(x, w{\cdot}\sigma) \geq 0$.

Recall that for every $1 \leq x \leq n$, the actual cost of reaching q_x is $\mathsf{cost}(q_x, w \cdot \sigma) = \mathsf{cost}(q_h, w) + \frac{\gamma(\langle q_h,\sigma,q_x\rangle)}{\lambda^{|w|}}$, for some $1 \leq h \leq n$. By the construction, we have $g_{x,w\sigma} \leq \lambda(g_{h',w} + \gamma(\langle q_{h'}, \sigma, q_x\rangle) - c_{|w|+1}) + r$, for some $1 \leq h' \leq n$. Thus, $M(x, w{\cdot}\sigma) = \frac{g_{x,w\sigma}}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|+1} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_x, w{\cdot}\sigma) \leq \frac{\lambda(g_{h',w}+\gamma(\langle q_{h'},\sigma,q_x\rangle)-c_{|w|+1})+r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|+1} \frac{c_i}{\lambda^{i-1}} - (\mathsf{cost}(q_h, w) + \frac{\gamma(\langle q_h,\sigma,q_x\rangle)}{\lambda^{|w|}})$.

By the construction, $g_{h',w} + \gamma(\langle q_{h'}, \sigma, q_x\rangle) \leq g_{h,w} + \gamma(\langle q_h, \sigma, q_x\rangle)$. Therefore,

$$\begin{aligned}
M(x, w{\cdot}\sigma) &\leq \frac{\lambda(g_{h,w} + \gamma(\langle q_h, \sigma, q_x\rangle) - c_{|w|+1}) + r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|+1} \frac{c_i}{\lambda^{i-1}} - \\
&\quad - \left(\mathsf{cost}(q_h, w) + \frac{\gamma(\langle q_h, \sigma, q_x\rangle)}{\lambda^{|w|}}\right) = \\
&= \frac{g_{h,w}}{\lambda^{|w|}} + \frac{r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_h, w) = \\
&= M(h, w) + \frac{r}{\lambda^{|w|+1}} \leq \sum_{i=1}^{|w|} \frac{r}{\lambda^i} + \frac{r}{\lambda^{|w|+1}} = \sum_{i=1}^{|w|+1} \frac{r}{\lambda^i}\ .
\end{aligned}$$

We continue with the second case, where $M(x, w{\cdot}\sigma) \leq 0$. Then,

$$\begin{aligned}
M(x, w{\cdot}\sigma) &\geq \frac{\lambda(g_{h',w} + \gamma(\langle q_{h'}, \sigma, q_x\rangle) - c_{|w|+1}) - r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|+1} \frac{c_i}{\lambda^{i-1}} - \\
&\quad - \left(\mathsf{cost}(q_h, w) + \frac{\gamma(\langle q_h, \sigma, q_x\rangle)}{\lambda^{|w|}}\right)\ .
\end{aligned}$$

Since $\mathsf{cost}(q_h, w) + \frac{\gamma(\langle q_h,\sigma,q_x\rangle)}{\lambda^{|w|}} \leq \mathsf{cost}(q_{h'}, w) + \frac{\gamma(\langle q_{h'},\sigma,q_x\rangle)}{\lambda^{|w|}}$, we have

$$\begin{aligned}
M(x, w{\cdot}\sigma) &\geq \frac{\lambda(g_{h',w} + \gamma(\langle q_{h'}, \sigma, q_x\rangle) - c_{|w|+1}) - r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|+1} \frac{c_i}{\lambda^{i-1}} - \\
&\quad - \left(\mathsf{cost}(q_{h'}, w) + \frac{\gamma(\langle q_{h'}, \sigma, q_x\rangle)}{\lambda^{|w|}}\right) = \\
&= \frac{g_{h',w}}{\lambda^{|w|}} - \frac{r}{\lambda^{|w|+1}} + \sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_{h'}, w) = \\
&= M(h', w) - \frac{r}{\lambda^{|w|+1}} \geq -\sum_{i=1}^{|w|} \frac{r}{\lambda^i} - \frac{r}{\lambda^{|w|+1}} = -\sum_{i=1}^{|w|+1} \frac{r}{\lambda^i}\ .
\end{aligned}$$

We can now show that $|\mathcal{D}'(w) - \mathcal{A}(w)| \leq \varepsilon$. Indeed, upon reading w, $\mathcal{D}'$ reaches some state q' in which $g_{h,w} = 0$, for some $1 \leq h \leq n$. Thus, $|\sum_{i=1}^{|w|} \frac{c_i}{\lambda^{i-1}} - \mathsf{cost}(q_h, w)| \leq \varepsilon$. Assume that $\mathcal{A}(w) > \mathcal{D}'(w)$. Then, since $\mathcal{A}(w) \leq \mathsf{cost}(q_h, w)$, it follows that $0 \leq \mathcal{A}(w) - \mathcal{D}'(w) \leq \mathsf{cost}(q_h, w) - \mathcal{D}'(w) \leq \varepsilon$. Analogously, assume that $\mathcal{D}'(w) > \mathcal{A}(w)$. Then, since $\mathcal{A}(w) \leq \mathsf{cost}(q_h, w)$, it follows that $0 \leq \mathcal{D}'(w) - \mathcal{A}(w) \leq \mathcal{D}'(w) - \mathsf{cost}(q_h, w) \leq \varepsilon$.

It is left to show that $|\mathcal{D}(w) - \mathcal{A}(w)| \leq \varepsilon$. The only difference between the construction of $\mathcal{D}$ and of $\mathcal{D}'$ is that the latter changes all gaps above $m(2^{k+1})$ to ∞. Upon reading w, $\mathcal{D}'$ ends in some state q' in which $g_{h,w} = 0$, for some $1 \leq h \leq n$. By the construction of $\mathcal{D}'$, there is a sequence of gaps $g_1, \ldots, g_{|w|} = g_{h,w} = 0$, such that for every i, $g_{i+1} \geq \lambda(g_i + x) - r$, where $|x| \leq m$. We claim that $\mathcal{D}$ also contains this sequence of gaps. Indeed, assume, by contradiction, that $g_i \geq m(2^{k+1})$, for some $i < |w|$. Then, $g_{i+1} \geq \lambda(g_i - m - r) = (1+2^{-k})(g_i - m - r) = g_i - m - r + (2^{-k})(g_i - m - r) \geq g_i - m - r + (2^{-k})(m(2^{k+1}) - m - r) = g_i - m - r + 2m - (2^{-k})(m + r) > g_i$. Hence, the sequence of gaps is growing from position i onwards, contradicting the assumption that the last gap in the sequence is 0. ◀

A.10 Proof of Theorem 13

We start with a lemma, analyzing the recoverable gaps of the NDA on which we will show the lower bound.

▶ **Lemma 19.** *Consider the NDA described in Figure 3. Then, for every $i, l \in \mathbb{N}$, where $i \leq 2^l$, there is a word $u_{l,i} \in \Sigma^l$ such that q_2 has the recoverable gap of $\frac{i}{2^l}$ over $u_{l,i}$.*

Proof. We prove the claim by induction on l. For $l = 1$, we have the words $u_{1,0} = $ '0', $u_{1,1} = $ '$\frac{1}{3}$', and $u_{1,2} = $ '$\frac{2}{3}$', for which q_2's gaps are $\frac{3}{2} \times 0 = \frac{0}{2}$, $\frac{3}{2} \times \frac{1}{3} = \frac{1}{2}$, and $\frac{3}{2} \times \frac{2}{3} = \frac{2}{2}$, respectively. For the induction step, consider a number $j \leq 2^{l+1}$, and let $r \in \{0, 1, 2\}$ be the remainder of dividing j by 3.

When $r = 0$, we have the word $u_{l+1,j} = u_{l,j/3} \cdot$ '0', as $\mathsf{gap}(q_2, u_{l+1,j}) = \frac{3}{2}(\mathsf{gap}(q_2, u_{l,j/3}) - 0) = \frac{3}{2}\frac{j}{3\times 2^l} = \frac{j}{2^{l+1}}$.

When $r \neq 0$, we split between three disjoint cases, depending on whether $j \leq 2^l$, $2^l < j \leq 4^l$, or $j > 4^l$. In the first case, if the remainder of dividing 2^l by 3 is different from r then we have the word $u_{l+1,j} = u_{l,(j+2^l)/3} \cdot$ '$-\frac{1}{3}$', otherwise the remainder of dividing 2^{l+1} by 3 is different from r, and we have the word $u_{l+1,j} = u_{l,(j+4^l)/3} \cdot$ '$-\frac{2}{3}$'. The gap of q_2 on $u_{l+1,j}$ is $\frac{3}{2}(\frac{j+2^l}{3\times 2^l} - \frac{1}{3}) = \frac{j}{2^{l+1}}$, and $\frac{3}{2}(\frac{j+4^l}{3\times 2^l} - \frac{2}{3}) = \frac{j}{2^{l+1}}$, respectively.

In the second case, having $2^l < j \leq 4^l$, if the remainder of dividing 2^l by 3 is different from r then we have the word $u_{l+1,j} = u_{l,(j+2^l)/3} \cdot$ '$-\frac{1}{3}$', otherwise we have the word $u_{l+1,j} = u_{l,(j-2^l)/3} \cdot$ '$\frac{1}{3}$'.

In the third case, having $j > 4^l$, if the remainder of dividing 2^l by 3 is equal to r then we have the word $u_{l+1,j} = u_{l,(j-2^l)/3} \cdot$ '$\frac{1}{3}$', otherwise we have the word $u_{l+1,j} = u_{l,(j-4^l)/3} \cdot$ '$\frac{2}{3}$'.

All these gaps of q_2 do not exceed 1, thus they can be recovered by adding '-1' at the end of the words. ◀

We continue with the theorem's proof.

Proof of Theorem 13. Consider the NDA $\mathcal{A}$ described in Figure 3, and let $\mathcal{D}$ be a DDA (with any discount factor) that ε-approximates $\mathcal{A}$, where $\varepsilon = 2^{-p}$. We claim that $\mathcal{D}$ has at least $2^{\lfloor \frac{p-2}{\log 3} \rfloor}$ states.

Intuitively, if q_2 has different recoverable gaps over two short enough words, then recovering them would yield two values that are more than ε apart. Two such words must lead to two different states in a deterministic automaton that ε-approximates $\mathcal{A}$.

Formally, let $l = \lceil \frac{p-2}{\log 3} \rceil - 1$. By Lemma 19, for every $i \leq 2^l$, there is a word $u_{l,i}$, such that $\mathsf{gap}(q_2, u_{l,i}) = \frac{i}{2^l}$. We show below that for every $i, j \leq 2^l$, such that $i \neq j$, $\mathcal{D}$ reaches two different states upon reading $u_{l,i}$ and $u_{l,j}$, which implies that $\mathcal{D}$ has at least $2^{(\lceil \frac{p-2}{\log 3} \rceil - 1)}$ states.

Assume, by contradiction, two different words, $u = u_{l,i}$ and $u' = u_{l,j}$, such that $\mathcal{D}$ reaches the same state s upon reading them. Let $w = u \cdot$ '0'$^\omega$ and $w' = u' \cdot$ '0'$^\omega$. Since $\mathcal{A}(w) = \mathcal{A}(w') = 0$ and $\mathcal{D}$ 2^{-p}-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(w) - \mathcal{D}(w')| \leq 2 \times 2^{-p}$. By the determinism of $\mathcal{D}$, we have $\mathcal{D}(w) = \mathcal{D}(u) + \frac{\mathcal{D}^s(\text{'0'}^\omega)}{\lambda^l}$ and $\mathcal{D}(w') = \mathcal{D}(u') + \frac{\mathcal{D}^s(\text{'0'}^\omega)}{\lambda^l}$. Hence, $|\mathcal{D}(u) - \mathcal{D}(u')| \leq 2 \times 2^{-p} = 2^{(-p+1)}$.

Now, consider the words $z = u \cdot$ '-1'$^\omega$ and $z' = u' \cdot$ '-1'$^\omega$. Since both $\mathsf{gap}(q_2, u)$ and $\mathsf{gap}(q_2, u')$ are recoverable by concatenating '-1'$^\omega$, it follows that the best run of $\mathcal{A}$ over both z and z' goes via q_2. Thus, $|\mathcal{A}(z) - \mathcal{A}(z')| = \frac{|\mathsf{gap}(q_2,u) - \mathsf{gap}(q_2,u')|}{\lambda^l} \geq \frac{1}{2^l} \times \frac{2^l}{3^l} = \frac{1}{3^l} > 2^{-(p-2)}$. Since $\mathcal{D}$ 2^{-p}-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(z) - \mathcal{D}(z')| > 2^{-(p-2)} - 2 \times 2^{-p} = 2^{-(p-1)}$. By the determinism of $\mathcal{D}$, we have $|\mathcal{D}(u) - \mathcal{D}(u')| > 2^{(-p+1)}$.

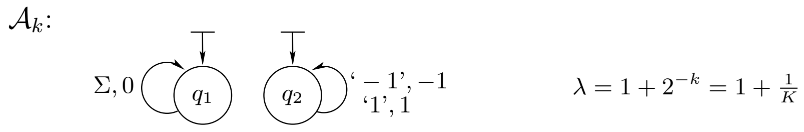


Figure 6 The family of NDAs, such that every DDA (with any discount factor) that $\frac{1}{8}$-approximates $\mathcal{A}_k$ has at least 2^{k-1} states.

We have shown that $|\mathcal{D}(u) - \mathcal{D}(u')|$ is both smaller and bigger than $2^{(-p+1)}$, contradicting the assumption that $\mathcal{D}$ reaches the same state upon reading two different words $u_{l,i}$ and $u_{l,j}$. Hence, $\mathcal{D}$ has at least $2^{(\lceil \frac{p-2}{\log 3} \rceil - 1)}$ states. ◀

A.11 Proof of Theorem 14

For convenience, we set $K = 2^k$. For every $k \geq 2$, consider the NDA $\mathcal{A}_k$, with discount factor $\lambda = 1 + 2^{-k} = 1 + \frac{1}{K}$, described in Figure 6, and let $\mathcal{D}$ be a DDA (with any discount factor) that $\frac{1}{8}$-approximates $\mathcal{A}_k$. We claim that $\mathcal{D}$ has at least 2^{k-1} states.

Intuitively, if q_2 has two recoverable gaps that are different enough, over two words that are short enough, then recovering them would yield two values that are also different enough. Two such words must lead to two different states in a deterministic automaton that $\frac{1}{8}$-approximates $\mathcal{A}_k$.

Formally, for every $1 \leq i \leq \frac{K}{2}$, consider the words $u_i = \text{'1'}^i \cdot \text{'0'}^{(K/2)-i}$. We claim that q_2 has recoverable gaps over all these words, and that the difference between the costs of reaching q_2 over each two words is at least $\frac{1}{2}$. Indeed, $\mathsf{gap}(q_2, u_i) = \sum_{j=1}^{i} \lambda^j = \sum_{j=1}^{i} (1 + \frac{1}{K})^j$. By Lemma 18, we have $(1 + \frac{1}{K})^K < 3$, implying that $(1 + \frac{1}{K})^{K/2} < \sqrt{3}$. Thus, $\mathsf{gap}(q_2, u_i) < \sum_{j=1}^{K/2} (1 + \frac{1}{K})^{K/2} < (K/2)\sqrt{3} < K$. The maximal recoverable gap of q_2 is $\sum_{i=0}^{\infty} (\frac{1}{\lambda^i}) = \frac{\lambda}{\lambda - 1} = \frac{1 + \frac{1}{K}}{\frac{1}{K}} = K + 1$, implying that q_2 has a recoverable gap over u_i. As for the costs of reaching q_2 over different words, consider the words u_i and u_j, where $i \neq j$. We have $|\mathsf{cost}(q_2, u_i) - \mathsf{cost}(q_2, u_j)| \geq \frac{1}{(1 + \frac{1}{K})^{K/2}} > \frac{1}{\sqrt{3}} > \frac{1}{2}$.

We continue with analyzing the runs of $\mathcal{D}$ on these words. Assume, by contradiction, two different words, u_i and u_j, such that $\mathcal{D}$ reaches the same state s upon reading them. Let $w = u_i \cdot \text{'0'}^\omega$ and $w' = u_j \cdot \text{'0'}^\omega$. Since $\mathcal{A}(w) = \mathcal{A}(w') = 0$ and $\mathcal{D}$ $\frac{1}{8}$-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(w) - \mathcal{D}(w')| \leq 2 \times \frac{1}{8} = \frac{1}{4}$. By the determinism of $\mathcal{D}$, we have $\mathcal{D}(w) = \mathcal{D}(u_i) + \frac{\mathcal{D}^s(\text{'0'}^\omega)}{\lambda^l}$ and $\mathcal{D}(w') = \mathcal{D}(u_j) + \frac{\mathcal{D}^s(\text{'0'}^\omega)}{\lambda^l}$. Hence, $|\mathcal{D}(u_i) - \mathcal{D}(u_j)| \leq \frac{1}{4}$.

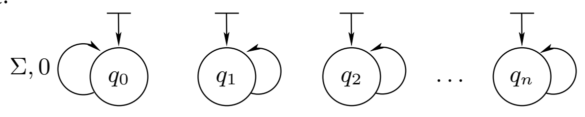
Now, consider the words $z = u_i \cdot \text{'}-1\text{'}^\omega$ and $z' = u_j \cdot \text{'}-1\text{'}^\omega$. Since both $\mathsf{gap}(q_2, u_i)$ and $\mathsf{gap}(q_2, u_j)$ are recoverable by concatenating '-1'$^\omega$, it follows that the best run of $\mathcal{A}$ over both z and z' goes via q_2. Thus, $|\mathcal{A}(z) - \mathcal{A}(z')| = |\mathsf{cost}(q_2, u_i) - \mathsf{cost}(q_2, u_j)| > \frac{1}{2}$. Since $\mathcal{D}$ $\frac{1}{8}$-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(z) - \mathcal{D}(z')| > \frac{1}{2} - 2 \times \frac{1}{8} = \frac{1}{4}$. By the determinism of $\mathcal{D}$, we have $|\mathcal{D}(u) - \mathcal{D}(u')| > \frac{1}{4}$.

We have shown that $|\mathcal{D}(u_i) - \mathcal{D}(u_j)|$ is both smaller and bigger than $\frac{1}{4}$, contradicting the assumption that $\mathcal{D}$ reaches the same state upon reading two different words u_i and u_j. Hence, $\mathcal{D}$ has at least $2^{K/2} = 2^{p-1}$ states. ◀

A.12 Proof of Theorem 15

For convenience, we set $K = 2^k$. For every $k \geq 2$ and $n \leq 2^k$, consider the NDA $\mathcal{A}_n$, with discount factor $\lambda = 1 + 2^{-k} = 1 + \frac{1}{K}$, described in Figure 7, and let $\mathcal{D}$ be a DDA (with any discount factor) that $\frac{1}{12}$-approximates $\mathcal{A}_n$. We claim that $\mathcal{D}$ has at least 2^n states.

$\mathcal{A}$:



$\lambda = 1 + 2^{-k}$
$\Sigma = \{\text{`}0\text{'}, \text{`}-1_i\text{'}, \text{`}1_i\text{'} \mid i \in \{1, \ldots, n\}\}$

For every $i, j \geq 1$: $\gamma(q_i, \text{`}1_j\text{'}, q_i) = 1$ if $i = j$ and 0 otherwise
$\gamma(q_i, \text{`}-1_j\text{'}, q_i) = -1$ if $i = j$ and 0 otherwise
$\gamma(q_i, \text{`}0\text{'}, q_i) = 0$

Figure 7 The family of NDAs, such that every DDA (with any discount factor) that $\frac{1}{12}$-approximates $\mathcal{A}_n$ has at least 2^n states.

Intuitively, we describe 2^n words of length n, encoding all binary combinations of n bits. The cost of reaching a state q_i over a word u is above $\frac{1}{3}$ if the ith bit in u is 1, and zero otherwise. These words are recoverable for all states, implying that a deterministic automaton that $\frac{1}{12}$-approximates $\mathcal{A}$ should reach a different state upon reading each of the 2^n words.

Formally, for every binary word $b \in \{0,1\}^n$ of length n, we define the word $u_b \in \Sigma^n$, by setting the ith letter of u_b to '1_i' if the ith letter of b is 1, and to '0' otherwise. We denote this set of Σ^n-words by U.

For every $1 \leq i \leq n$ and $u \in U$, we have $\mathsf{cost}(q_i, u) = 0$ if $u[i] = \text{`}0\text{'}$ and $\frac{1}{\lambda^i}$ otherwise. By Lemma 18, we have $(1 + \frac{1}{K})^K < 3$. Since, $i \leq n \leq K$, it follows that $\mathsf{cost}(q_i, u) > \frac{1}{3}$ if $u[i] \neq \text{`}0\text{'}$. In addition, we have that for every $1 \leq i \leq n$ and $u \in U$, the gap of q_i over u is recoverable by concatenating '-1_i'$^\omega$, as $\mathsf{gap}(q_i, u) \leq \lambda^n \leq (1 + \frac{1}{K})^K < 3$ and the value of concatenating '-1_i'$^\omega$ from the nth position is $\frac{1}{\lambda^n} \sum_{j=1}^{\infty} \lambda^j \geq \frac{1}{3}(K+1) \geq \frac{1}{3}(2^3+1) = 3$.

We continue with analyzing the runs of $\mathcal{D}$ on these words. Assume, by contradiction, two different words, $u, u' \in U$, such that $\mathcal{D}$ reaches the same state s upon reading them. Let $w = u \cdot \text{`}0\text{'}^\omega$ and $w' = u' \cdot \text{`}0\text{'}^\omega$. Since $\mathcal{A}(w) = \mathcal{A}(w') = 0$ and $\mathcal{D}$ $\frac{1}{12}$-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(w) - \mathcal{D}(w')| \leq 2 \times \frac{1}{12} = \frac{1}{6}$. By the determinism of $\mathcal{D}$, we have $\mathcal{D}(w) = \mathcal{D}(u) + \frac{\mathcal{D}^s(\text{`}0\text{'}^\omega)}{\lambda^l}$ and $\mathcal{D}(w') = \mathcal{D}(u') + \frac{\mathcal{D}^s(\text{`}0\text{'}^\omega)}{\lambda^l}$. Hence, $|\mathcal{D}(u_i) - \mathcal{D}(u_j)| \leq \frac{1}{6}$.

Now, since $u \neq u'$, there is some index $1 \leq i \leq n$, such that $u[i] = \text{`}0\text{'}$ and $u'[i] = \text{`}1_i\text{'}$, or vice versa. Consider the words $z = u \cdot (-1_i)^\omega$ and $z' = u' \cdot \text{`}-1_i\text{'}^\omega$. One can observe that for every $0 \leq j \leq n$, such that $j \neq i$, we have $\mathsf{cost}(q_j, u) \geq 0$, $\mathsf{cost}(q_j, u') \geq 0$, and $\mathcal{A}^{q_j}(\text{`}-1\text{'}^\omega) = 0$. On the other hand, $\mathcal{A}(z) < 0$ and $\mathcal{A}(z') < 0$, going via q_i. Thus, $|\mathcal{A}(z) - \mathcal{A}(z')| = |\mathsf{cost}(q_i, u) - \mathsf{cost}(q_i, u')| > \frac{1}{3}$. Since $\mathcal{D}$ $\frac{1}{12}$-approximates $\mathcal{A}$, it follows that $|\mathcal{D}(z) - \mathcal{D}(z')| > \frac{1}{3} - 2 \times \frac{1}{12} = \frac{1}{6}$. By the determinism of $\mathcal{D}$, we have $|\mathcal{D}(u) - \mathcal{D}(u')| > \frac{1}{6}$.

We have shown that $|\mathcal{D}(u) - \mathcal{D}(u')|$ is both smaller and bigger than $\frac{1}{6}$, contradicting the assumption that $\mathcal{D}$ reaches the same state upon reading two different words u and u'. Hence, $\mathcal{D}$ has at least 2^n states. ◀

A.13 Proof of Theorem 16

Let $\mathcal{B}$ be an average automaton with the structure of the automaton $\mathcal{A}$ of Figure 5, but replacing the 0-weights with an arbitrary value x and replacing the 1-weights with some value $y > x$. Assume, by contradiction, that $\mathcal{B}$ can be determinized approximately with respect to some ordered distance function d.

We define $\varepsilon > 0$ to be small enough with respect to the distance between the average of x and y and some values near it; formally, $\varepsilon = \min(d(x, \frac{x+y}{2}), d(\frac{x+y}{2}, \frac{x+2y}{3}),$

$d(\frac{x+2y}{3},\frac{x+3y}{4}), d(\frac{x+3y}{4},\frac{x+4y}{5}), d(\frac{x+4y}{5},\frac{x+5y}{6}))$.

Let $\mathcal{D}$ be a deterministic average automaton with n states that ε-approximates $\mathcal{B}$. There must be states s_1 and s_2 in $\mathcal{D}$ and four words, u_1, u_2, u_3, and u_4, such that u_2 has only a's, u_4 has only b's, $\mathcal{D}$ reaches s_1 reading u_1, $\mathcal{D}^{s_1}$ reaches s_1 reading u_2, $\mathcal{D}^{s_1}$ reaches s_2 reading u_3, and $\mathcal{D}^{s_2}$ reaches s_2 reading u_4.

We construct two words, w_1 and w_2, of about the same length with mostly a's and mostly b's, respectively, such that the value of $\mathcal{D}$ for them is far from the average between x and y. Formally, $w_1 = u_1 \cdot {u_2}^{10n|u_4|}$ and $w_2 = u_1 \cdot u_3 \cdot {u_4}^{10n|u_2|}$. Note that $\mathcal{B}(w_1) \geq \frac{x+5y}{6}$ and $\mathcal{B}(w_2) \geq \frac{x+5y}{6}$. Since $\mathcal{D}$ ε-approximates $\mathcal{B}$ with respect to d, which is an ordered distance function, we have $\mathcal{D}(w_1) > \frac{x+4y}{5}$ and $\mathcal{D}(w_2) > \frac{x+4y}{5}$.

We shall now combine the words w_1 and w_2 into a word w, such that the value of $\mathcal{B}$ for w is close to the average between x and y, while the value of $\mathcal{D}$ must be far away. Formally, $w = u_1 {\cdot} {u_2}^{10n|u_4|} {\cdot} u_3 {\cdot} {u_4}^{10n|u_2|}$. We have $\mathcal{B}(w) \leq \frac{x+2y}{3}$, while $\mathcal{D}(w) = \frac{|w_1|\mathcal{D}(w_1)+|w_2|\mathcal{D}(w_2)-|u_1|\mathcal{D}(u_1)}{|w_1|+|w_2|-|u_1|} \geq (20\frac{x+4y}{5} - x)/19 \geq \frac{x+3y}{4}$. Hence, $d(\mathcal{B}(w),\mathcal{D}(w)) \geq d(\frac{x+3y}{4},\frac{x+4y}{5}) \geq \varepsilon$, implying that $\mathcal{D}$ does not ε approximates $\mathcal{B}$. ◀

A.14 Proof of Theorem 17

In [8], it is shown that the automaton $\mathcal{A}$, defined in Figure 4, cannot be determinized. We generalize the result, showing that it can neither be determinized approximately with respect to any distance function.

We start with a lemma, formalizing the intuition that inserting infinitely, but negligibly, many constant values to a sequence of numbers does not change its limit-average value.

▶ **Lemma 20.** *Consider an infinite sequence $\pi = \pi_1, \pi_2, \dots$ and a finite sequence $\mu = \mu_1, \mu_2, \dots, \mu_k$ of numbers absolutely bounded by a constant c (that is, $|\pi_i| \leq c$ and $|\mu_j| \leq c$ for all $i \geq 1$ and $1 \leq j \leq k$). Let π' be the infinite sequence obtained from π by inserting μ at positions $\{2^i \mid i \in \mathbb{N}\}$. Then, $\lim_{n\to\infty} \inf\{\frac{1}{n}\sum_{i=0}^{n-1}\pi_i \mid i \geq n\} = \lim_{n\to\infty} \inf\{\frac{1}{n}\sum_{i=0}^{n-1}\pi'_i \mid i \geq n\}$.*

Proof. For showing that π and π' have the same limit-average value, we define a surjective mapping ρ between the positions of π' and π, such that $\rho(j_1) \geq \rho(j_2)$ iff $j_1 \geq j_2$, and show that $\lim_{j\to\infty} |\frac{\sum_0^j \pi'_j}{j} - \frac{\sum_0^{\rho(j)} \pi_{\rho(j)}}{\rho(j)}|$ converges to 0.

We denote the range of a function f by $\mathsf{range}(f)$ and define the functions $\mathsf{Move} : \mathbb{N} \to \mathbb{N}$, $\mathsf{Next} : \mathbb{N} \to \mathbb{N}$, and $\rho : \mathbb{N} \to \mathbb{N}$ as follows.

$$\begin{aligned} \mathsf{Move}(j) &= j + k \cdot |\{2^i \mid i \in \mathbb{N} \text{ and } 2^i \leq j\}|; \\ \mathsf{Next}(j) &= \min\{i \mid i \in \mathsf{range}(\mathsf{Move}) \text{ and } i \geq j\}; \text{ and} \\ \rho(j) &= \mathsf{Move}^{-1}(\mathsf{Next}(j)). \end{aligned}$$

Intuitively, every position of π' that originated in π is mapped by ρ to its original position in π, while a position of π' that originated in μ is treated as the next position of π' that originated in π.

Now, for every $j \in \mathbb{N}$ we have $|\frac{\sum_0^j \pi'_j}{j} - \frac{\sum_0^{\rho(j)} \pi_{\rho(j)}}{\rho(j)}| \leq |\frac{c(j-\rho(j))}{\rho(j)}|$. Hence, $\lim_{j\to\infty} |\frac{\sum_0^j \pi'_j}{j} - \frac{\sum_0^{\rho(j)} \pi_{\rho(j)}}{\rho(j)}| \leq \lim_{j\to\infty} |\frac{c(j-\rho(j))}{\rho(j)}| = 0$, as required. ◀

We continue with the theorem's proof.

Proof of Theorem 17. Consider the automaton $\mathcal{A}$, defined in Figure 4, with arbitrary weights $x < y$. It realizes the function "$\mathcal{A}(w) = y$ if the word w contains finitely many a's, and $\mathcal{A}(w) = x$ otherwise". Assume, by contradiction, that $\mathcal{A}$ can be determinized approximately with respect to a distance function d and a precision $\varepsilon = \frac{d(x,y)}{2}$. We have, by the above assumption, a deterministic limit-average automaton $\mathcal{B}$, such that for every word w, $d(\mathcal{A}(w) - \mathcal{B}(w)) < \varepsilon$.

There must be a state s in $\mathcal{B}$ and three finite words, u_1, u_2, and u_3, such that u_2 has only b's, u_3 starts with an a, $\mathcal{B}$ reaches s reading u_1, and $\mathcal{B}^s$ reaches s reading each of u_2 and u_3.

Let v be the average weight of the path from s back to itself reading u_2. Since $\mathcal{B}(u_1 u_2^\omega) = v$, $\mathcal{A}(u_1 \cdot u_2^\omega) = y$ and $d(\mathcal{B}(u_1 \cdot u_2^\omega), \mathcal{A}(u_1 \cdot u_2^\omega)) < \varepsilon$, it follows that $d(v, y) < \varepsilon$.

Now, consider the word $w = u_1 \cdot u_2^2 \cdot u_3 \cdot u_2^4 \cdot u_3 \cdot u_2^8 \cdot u_3 \ldots$, in which the frequency of a's decays exponentially. By Lemma 20, we have $\mathcal{B}(w) = v$. Since $\mathcal{A}(w) = x$ and $d(\mathcal{A}(w), \mathcal{B}(w)) < \varepsilon$, it follows that $d(x, v) < \varepsilon$. However, we get by the triangle inequality of the distance function that $2\varepsilon = d(x, y) \leq d(x, v) + d(v, y) < 2\varepsilon$, leading to a contradiction. ◀