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.

1998 ACM Subject Classification F.1.1 Automata; D.2.4 Formal methods

Keywords and phrases Quantitative; Automata; Determinization; Approximation

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2012.362

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]). 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 [14], 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 [13], and checking for automata

* 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).

32nd Int'l Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2012).
Editors: D. D'Souza, J. Radhakrishnan, and K. Telikepalli; pp. 362–373

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

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].

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, 15]), 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. 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.

Due to lack of space, the proofs are omitted in this version and can be found in the full version, at https://repository.ist.ac.at/102.

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.

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

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

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

Discounted-sum automata. Operate over finite or infinite words, and are equipped with a rational discount factor $\lambda > 1$. [1] They value a run r by $\gamma(r) = \sum_{i=0}^{|r|-1} \frac{\gamma_i}{\lambda^i}$; and $\mathcal{A}(w) = \inf$ 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; $\gamma(r) = \lim_{n\to\infty} \sup\{\gamma_i \mid i \geq n\}$; $\mathcal{A}(w) = \sup$ of runs on w.

Limit-average Automata. Infinite words; $\gamma(r) = \lim_{n\to\infty} \inf\{\frac{1}{i} \sum_{j=0}^{i-1} \gamma_j \mid i \geq n\}$, [2] $\mathcal{A}(w) = \sup$ of runs on w.

[1] 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.

[2] There is another version of limit-average automata, where $\gamma(r) = \lim_{n\to\infty} \sup\{\frac{1}{i} \sum_{j=0}^{i-1} \gamma_j \mid i \geq n\}$. 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).*

▶ 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.*

▶ 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.

▶ **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.*

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.
- $\delta' = \{(w, \sigma, w \cdot \sigma) \mid |w| \leq l - 1 \wedge \sigma \in \Sigma\} \cup \{(w, \sigma, w) \mid |w| = l \wedge \sigma \in \Sigma\}$.
- 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 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 1.)

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, $Q' = \{\langle g_1, \ldots, g_n \rangle \mid \text{ for every } 1 \leq h \leq n, g_h \in G\}$.

The initial state of $\mathcal{D}$ is $q'_{in} = \langle g_1, \ldots, g_n \rangle$, where for every $1 \leq i \leq n$, $g_i = 0$ 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 $x \geq 0$, we define $\mathsf{Round}(x) = i2^{-(p+k-1)}$, such that $i \in \mathbb{N}$ and for every $j \in \mathbb{N}, |x - i2^{-(p+k-1)}| \leq |x - j2^{-(p+k-1)}|$.

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

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

◀

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 ∞, and argue afterwards 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 constructed automaton 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 (delicately) show by induction on the length of w that $|M(h, w)| \leq \sum_{i=1}^{|w|} \frac{r}{\lambda^i}$.

◀

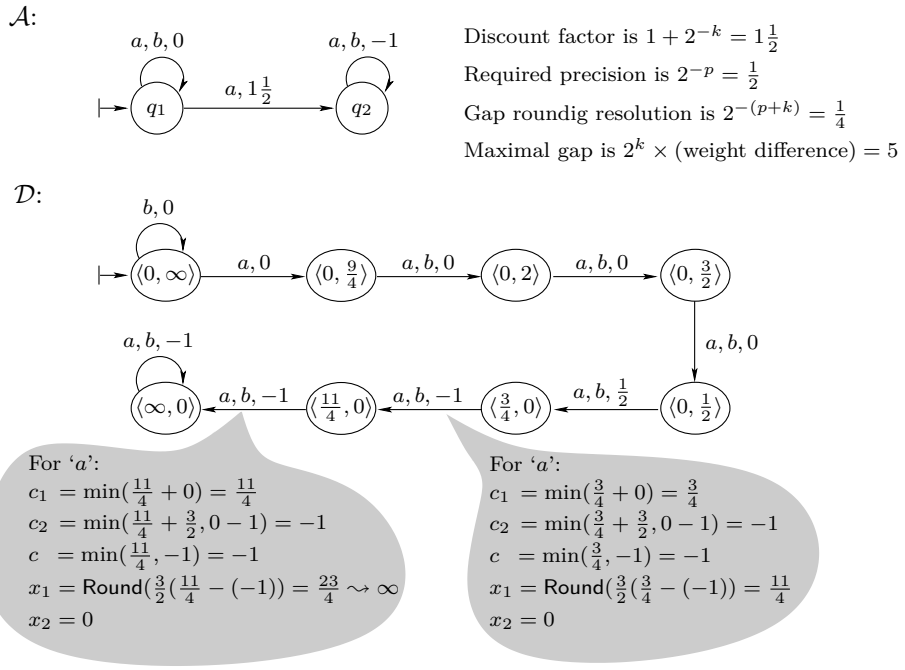

Figure 1 Determinizing the NDA $\mathcal{A}$ approximately into the DDA $\mathcal{D}$. The gray bubbles detail some of the intermediate calculations of the approximate-determinization construction.

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.*

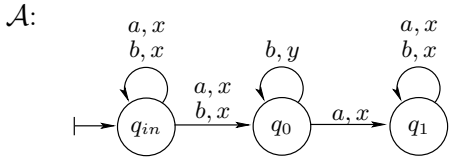

Figure 2 A limit-average automaton assigning y to words with finitely many a's and x otherwise.

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.*

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 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 2, with arbitrary weights $x < y$. Considering 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 deterministic automaton misvalues it. ◀

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 T. A. Henzinger and N. Piterman. Solving games without determinization. In *proc. of 15th EACSL conf.*, pages 395–410, 2006.

14 O. Kupferman and Y. Lustig. Lattice automata. In *8th VMCAI conf.*, pages 199–213, 2007.

15 U. Zwick and M.S. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158:343–359, 1996.