Discounted-Sum Automata with Multiple Discount Factors

Udi Boker
The Interdisciplinary Center, Herzliya, Israel
udiboker@idc.ac.il

Guy Hefetz
The Interdisciplinary Center, Herzliya, Israel
ghefetz@gmail.com

Abstract

Discounting the influence of future events is a key paradigm in economics and it is widely used in computer-science models, such as games, Markov decision processes (MDPs), reinforcement learning, and automata. While a single game or MDP may allow for several different discount factors, discounted-sum automata (NDAs) were only studied with respect to a single discount factor. For every integer $\lambda \in \mathbb{N} \setminus \{0,1\}$, as opposed to every $\lambda \in \mathbb{Q} \setminus \mathbb{N}$, the class of NDAs with discount factor λ (λ-NDAs) has good computational properties: it is closed under determinization and under the algebraic operations min, max, addition, and subtraction, and there are algorithms for its basic decision problems, such as automata equivalence and containment.

We define and analyze discounted-sum automata in which each transition can have a different integral discount factor (integral *NMDAs*). We show that integral NMDAs with an arbitrary choice of discount factors are not closed under determinization and under algebraic operations. We then define and analyze a restricted class of integral NMDAs, which we call *tidy NMDAs*, in which the choice of discount factors depends on the prefix of the word read so far. Tidy NMDAs are as expressive as deterministic integral NMDAs with an arbitrary choice of discount factors, and some of their special cases are NMDAs in which the discount factor depends on the action (alphabet letter) or on the elapsed time.

We show that for every function θ that defines the choice of discount factors, the class of θ-NMDAs enjoys all of the above good properties of integral NDAs, as well as the same complexities of the required decision problems. To this end, we also improve the previously known complexities of the decision problems of integral NDAs, and present tight bounds on the size blow-up involved in algebraic operations on them.

All our results hold equally for automata on finite words and for automata on infinite words.

2012 ACM Subject Classification Theory of computation → Formal languages and automata theory

Keywords and phrases Automata, Discounted-sum, Quantitative verification, NMDA, NDA

Digital Object Identifier 10.4230/LIPIcs.CSL.2021.23

Related Version A full version of the paper is available at `https://www.idc.ac.il/en/schools/cs/research/documents/thesis-guy-hefetz.pdf` ([24]).

Funding *Udi Boker*: Israel Science Foundation grant 1373/16

1 Introduction

Discounted summation is a central valuation function in various computational models, such as games (e.g., [39, 2, 17]), Markov decision processes (e.g, [23, 30, 15]), reinforcement learning (e.g, [34, 37]), and automata (e.g, [19, 11, 13, 14]), as it formalizes the concept that an immediate reward is better than a potential one in the far future, as well as that a potential problem (such as a bug in a reactive system) in the far future is less troubling than a current one.

A Nondeterministic Discounted-sum Automaton (NDA) is an automaton with rational weights on the transitions, and a fixed rational discount factor $\lambda > 1$. The value of a (finite or infinite) run is the discounted summation of the weights on the transitions, such that the weight in the ith position of the run is divided by λ^i. The value of a (finite or infinite) word is the minimal value of the automaton runs on it. An NDA $\mathcal{A}$ realizes a function from words to real numbers, and we write $\mathcal{A}(w)$ for the value of $\mathcal{A}$ on a word w.

In the Boolean setting, where automata realize languages, closure under the basic Boolean operations of union, intersection, and complementation is desirable, as it allows to use automata in formal verification, logic, and more. In the quantitative setting, where automata realize functions from words to numbers, the above Boolean operations are naturally generalized to algebraic ones: union to min, intersection to max, and complementation to multiplication by -1 (depending on the function's co-domain). Likewise, closure under these algebraic operations, as well as under addition and subtraction, is desirable for quantitative automata, serving for quantitative verification. Determinization is also very useful in automata theory, as it gives rise to many algorithmic solutions, and is essential for various tasks, such as synthesis and probabilistic model checking[1].

NDAs cannot always be determinized [14], they are not closed under basic algebraic operations [6], and basic decision problems on them, such as universality, equivalence, and containment, are not known to be decidable and relate to various longstanding open problems [7]. However, restricting NDAs to an integral discount factor $\lambda \in \mathbb{N}$ provides a robust class of automata that is closed under determinization and under the algebraic operations, and for which the decision problems of universality equivalence, and containment are decidable [6].

Various variants of NDAs are studied in the literature, among which are *functional*, *k-valued*, *probabilistic*, and more [21, 20, 12]. Yet, to the best of our knowledge, all of these models are restricted to have a single discount factor in an automaton. This is a significant restriction of the general discounted-summation paradigm, in which multiple discount factors are considered. For example, Markov decision processes and discounted-sum games allow for multiple discount factors within the same entity [23, 2].

A natural extension to NDAs is to allow for different discount factors over the transitions, providing the ability to model systems in which each action (alphabet letter in the automaton) causes a different discounting, systems in which the discounting changes over time, and more. As integral NDAs provide robust automata classes, whereas non-integral NDAs do not, we look into extending integral NDAs into integral *NMDAs* (Definition 1), allowing multiple integral discount factors in a single automaton.

As automata are aimed at modeling systems, NMDAs significantly extend the system behaviors that can be modeled with discounted-sum automata. For an intuitive example, consider how the value of used vehicles changes over time: It decreases a lot in the first year, slightly less rapidly in the next couple of years, and significantly less rapidly in further years. An NDA cannot model such a behavior, as the discount factor cannot change over time, whereas an NMDA provides the necessary flexibility of the discount factor.

On a more formal level, NMDAs may allow to enhance formal verification of reinforcement learning applications. In the reinforcement learning process, the expected return value is the discounted-summation of the accumulated future rewards. In classic reinforcement learning, the discounted summation uses a single discount factor, whereas novel approaches in reinforcement learning study how to enhance the process to allow multiple discount factors

[1] In some cases, automata that are "almost deterministic", such as limit-deterministic [36] or good-for-games automata [25, 8] suffice.

[28, 22, 37, 32, 27]. This enhancement of reinforcement learning parallels our extension of discounted-sum automata to support multiple discount factors.

We start with analyzing NMDAs in which the integral discount factors can be chosen arbitrarily. Unfortunately, we show that this class of automata does not allow for determinization and is not closed under the basic algebraic operations.

For more restricted generalizations of integral NDAs, in which the discount factor depends on the transition's letter (*letter-oriented* NMDAs) or on the elapsed time (*time-oriented* NMDAs), we show that the corresponding automata classes do enjoy all of the good properties of integral NDAs, while strictly extending their expressiveness.

We further analyze a rich class of integral NMDAs that extends both letter-oriented and time-oriented NMDAs, in which the choice of discount factor depends on the word-prefix read so far (*tidy* NMDAs). We show that their expressiveness is as of deterministic integral NMDAs with an arbitrary choice of discount factors and that for every choice function $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0, 1\}$, the class of θ-NMDAs enjoys all of the good properties of integral NDAs. (See Figure 1.)

Considering closure under algebraic operations, we further provide tight bounds on the size blow-up involved in the different operations (Table 1). To this end, we provide new lower bounds also for the setting of NDAs, by developing a general scheme to convert every NFA to a corresponding NDA of linearly the same size, and to convert some specific NDAs back to corresponding NFAs.

As for the decision problems of tidy NMDAs, we provide a PTIME algorithm for emptiness and PSPACE algorithms for the other problems of exact-value, universality, equivalence, and containment. The complexities are with respect to the automaton (or automata) size, which is considered as the maximum between the number of transitions and the maximal binary representation of any discount factor or weight in it. These new algorithms also improve the complexities of the previously known algorithms for solving the decision problems of NDAs, which were PSPACE with respect to unary representation of the weights.

As general choice functions need not be finitely represented, it might upfront limit the usage of tidy NMDAs. Yet, we show that finite transducers suffice, in the sense that they allow to represent every choice function θ that can serve for a θ-NMDA. We provide a PTIME algorithm to check whether a given NMDA is tidy, as well as if it is a $\mathcal{T}$-NMDA for a given transducer $\mathcal{T}$.

We show all of our results for both automata on finite words and automata on infinite words. Whenever possible, we provide a single proof for both settings. Due to lack of space, some of the full proofs appear in the appendix, while the rest can be found in [24].

2 Discounted-Sum Automata with Multiple Integral Discount Factors

We define a discounted-sum automaton with arbitrary discount factors, abbreviated NMDA, by adding to an NDA a discount factor in each of its transitions. An NMDA is defined on either finite or infinite words. The formal definition is given in Definition 1, and an example in Figure 2.

An *alphabet* Σ is an arbitrary finite set, and a *word* over Σ is a finite or infinite sequence of letters in Σ, with ε for the empty word. We denote the concatenation of a finite word u and a finite or infinite word w by $u \cdot w$, or simply by uw. We define Σ^+ to be the set of all finite words except the empty word, i.e., $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. For a word $w = w(0)w(1)w(2)\ldots$, we denote the sequence of its letters starting at index i and ending at index j as $w[i..j] = w(i)w(i+1)\ldots w(j)$.

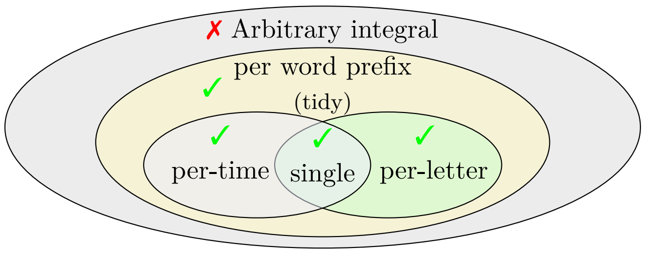

Figure 1 Classes of integral NMDAs, defined according to the flexibility of choosing the discount factors. The class of NMDAs with arbitrary integral factors is not closed under algebraic operations and under determinization. The other classes (for a specific choice function) are closed under both algebraic operations and determinization. Tidy NMDAs are as expressive as deterministic NMDAs with arbitrary integral discount factors.

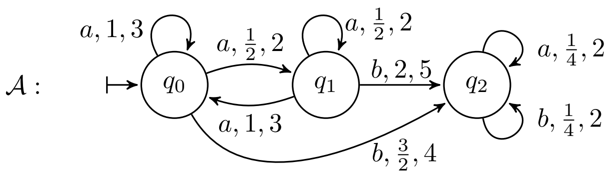

Figure 2 An NMDA $\mathcal{A}$. The labeling on the transitions indicate the alphabet letter, the weight of the transition, and its discount factor.

▶ **Definition 1.** *A nondeterministic discounted-sum automaton with multiple discount factors (NMDA), on finite or infinite words, is a tuple $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, \gamma, \rho \rangle$ over an alphabet Σ, with a finite set of states Q, an initial set of states $\iota \subseteq Q$, a transition function $\delta \subseteq Q \times \Sigma \times Q$, a weight function $\gamma : \delta \to \mathbb{Q}$, and a discount-factor function $\rho : \delta \to \mathbb{Q} \cap (1, \infty)$, assigning to each transition its discount factor, which is a rational greater than one.* [2]

- *A* walk *in $\mathcal{A}$ from a state p_0 is a sequence of states and letters, $p_0, \sigma_0, p_1, \sigma_1, p_2, \cdots$, such that for every i, $(p_i, \sigma_i, p_{i+1}) \in \delta$. For example, $\psi = q_1, a, q_1, b, q_2$ is a walk of the NMDA $\mathcal{A}$ of Figure 2 on the word ab from the state q_1 .*
- *A run of $\mathcal{A}$ is a walk from an initial state.*
- *The length of a walk ψ, denoted by $|\psi|$, is n for a finite walk $\psi = p_0, \sigma_0, p_1, \cdots, \sigma_{n-1}, p_n$, and ∞ for an infinite walk.*
- *The i-th transition of a walk $\psi = p_0, \sigma_0, p_1, \sigma_1, \cdots$ is denoted by $\psi(i) = (p_i, \sigma_i, p_{i+1})$.*
- *The* value *of a finite or an infinite walk ψ is $\mathcal{A}(\psi) = \sum_{i=0}^{|\psi|-1} \left(\gamma\big(\psi(i)\big) \cdot \prod_{j=0}^{i-1} \frac{1}{\rho\big(\psi(j)\big)} \right)$. For example, the value of the walk $r_1 = q_0, a, q_0, a, q_1, b, q_2$ (which is also a run) of $\mathcal{A}$ from Figure 2 is $\mathcal{A}(r_1) = 1 + \frac{1}{2} \cdot \frac{1}{3} + 2 \cdot \frac{1}{2 \cdot 3} = \frac{3}{2}$.*
- *The* value *of $\mathcal{A}$ on a finite or infinite word w is $\mathcal{A}(w) = \inf\{\mathcal{A}(r) \mid r \text{ is a run of } \mathcal{A} \text{ on } w\}$.*
- *In the case where $|\iota| = 1$ 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*, denoted by DMDA, and view δ as a function to states.*
- *When all the discount factors are integers, we say that $\mathcal{A}$ is an* integral *NMDA.*

In the case where for every $q \in Q$ and $\sigma \in \Sigma$, we have $|\{q' \mid (q, \sigma, q') \in \delta\}| \geq 1$, intuitively meaning that $\mathcal{A}$ cannot get stuck, we say that $\mathcal{A}$ is *complete*. It is natural to assume that

[2] Discount factors are sometimes defined in the literature as numbers between 0 and 1, under which setting weights are multiplied by these factors rather than divided by them.

discounted-sum automata are complete, and we adopt this assumption, as dead-end states, which are equivalent to states with infinite-weight transitions, break the property of the decaying importance of future events.

Automata $\mathcal{A}$ and $\mathcal{A}'$ are *equivalent*, denoted by $\mathcal{A} \equiv \mathcal{A}'$, if for every word w, $\mathcal{A}(w) = \mathcal{A}'(w)$.

For every finite (infinite) walk $\psi = p_0, \sigma_0, p_1, \sigma_1, p_2, \cdots, \sigma_{n-1}, p_n$ ($\psi = p_0, \sigma_0, p_1, \cdots$), and all integers $0 \leq i \leq j \leq |\psi| - 1$ ($0 \leq i \leq j$), we define the finite sub-walk from i to j as $\psi[i..j] = p_i, \sigma_i, p_{i+1}, \cdots, \sigma_j, p_{j+1}$. For an infinite walk, we also define $\psi[i..\infty] = p_i, \sigma_i, p_{i+1}, \cdots$, namely the infinite suffix from position i. For a finite walk, we also define the target state as $\delta(\psi) = p_n$ and the accumulated discount factor as $\rho(\psi) = \prod_{i=0}^{n-1} \rho\big(\psi(i)\big)$.

We extend the transition function δ to finite words in the regular manner: For a word $u \in \Sigma^*$ and a letter $\sigma \in \Sigma$, $\delta(\varepsilon) = \iota$; $\delta(u \cdot \sigma) = \bigcup_{q \in \delta(u)} \delta(q, \sigma)$. 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 single initial state.

An NMDA may have rational weights, yet it is often convenient to consider an analogous NMDA with integral weights, achieved by multiplying all weights by their common denominator.

▶ **Proposition 2.** *For all constant $0 < m \in \mathbb{Q}$, NMDA $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, \gamma, \rho \rangle$, NMDA $\mathcal{A}' = \langle \Sigma, Q, \iota, \delta, m \cdot \gamma, \rho \rangle$ obtained from $\mathcal{A}$ by multiplying all its weights by m, and a finite or infinite word w, we have $\mathcal{A}'(w) = m \cdot \mathcal{A}(w)$.*

Size. We define the size of $\mathcal{A}$, denoted by $|\mathcal{A}|$, as the maximum between the number of transitions and the maximal binary representation of any discount factor or weight in it. For rational weights, we assume all of them to have the same denominator. The motivation for a common denominator stems from the determinization algorithm (Theorem 8). Omitting this assumption will still result in a deterministic automaton whose size is only single exponential in the size of the original automaton, yet storing its states will require a much bigger space, changing our PSPACE algorithms (Section 4) into EXPSPACE ones.

Algebraic operations. Given automata $\mathcal{A}$ and $\mathcal{B}$ over the same alphabet and a non-negative scalar $c \in \mathbb{Q}$, we define an algebraic operation $op \in \{\min, \max, +, -\}$ on $\mathcal{A}$ and $\mathcal{B}$ as $\mathcal{C} \equiv op(\mathcal{A}, \mathcal{B})$ iff $\forall w. \mathcal{C}(w) = op\big(\mathcal{A}(w), \mathcal{B}(w)\big)$, and $op \in \{c\cdot, -\}$ as $\mathcal{C} \equiv op(\mathcal{A})$ iff $\forall w. \mathcal{C}(w) = op\big(\mathcal{A}(w)\big)$.

Decision problems. Given automata $\mathcal{A}$ and $\mathcal{B}$ and a threshold $\nu \in \mathbb{Q}$, we consider the following properties, with strict (or non-strict) inequalities: *Nonemptiness:* There exists a word w, s.t. $\mathcal{A}(w) < \nu$ (or $\mathcal{A}(w) \leq \nu$); *Exact-value:* There exists a word w, s.t. $\mathcal{A}(w) = \nu$; *Universality:* For all words w, $\mathcal{A}(w) < \nu$ (or $\mathcal{A}(w) \leq \nu$); *Equivalence:* For all words w, $\mathcal{A}(w) = \mathcal{B}(w)$; *Containment:* For all words w, $\mathcal{A}(w) > \mathcal{B}(w)$ (or $\mathcal{A}(w) \geq \mathcal{B}(w)$). [3]

Finite and infinite words. Results regarding NMDAs on finite words that refer to the existence of an equivalent automaton ("positive results") can be extended to NMDAs on infinite words due to Lemma 3 below. Likewise, results that refer to non-existence of an equivalent automaton ("negative results") can be extended from NMDAs on infinite words to NMDAs on finite words. Accordingly, if not stated otherwise, we prove the positive results for automata on finite words and the negative results for automata on infinite words, getting the results for both settings.

[3] Considering quantitative containment as a generalization of language containment, and defining the "acceptance" of a word w as having a small enough value on it, we define that $\mathcal{A}$ is contained in $\mathcal{B}$ if for every word w, $\mathcal{A}$'s value on w is at least as big as $\mathcal{B}$'s value. (Observe the $>$ and $\geq$ signs in the definition.)

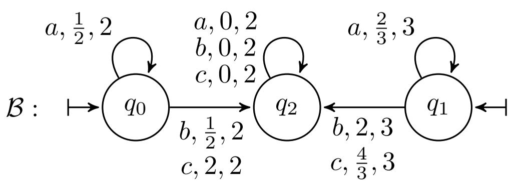

Figure 3 An integral NMDA $\mathcal{B}$ on infinite words that cannot be determinized.

▶ **Lemma 3.** *For all NMDAs $\mathcal{A}$ and $\mathcal{B}$, if for all finite word $u \in \Sigma^+$, we have $\mathcal{A}(u) = \mathcal{B}(u)$, then also for all infinite word $w \in \Sigma^\omega$, we have $\mathcal{A}(w) = \mathcal{B}(w)$.*

The proof is a simple extension of the proof of a similar lemma in [6] with respect to NDAs.

Notice that the converse does not hold, namely there are automata equivalent w.r.t. infinite words, but not w.r.t. finite words. (See an example in Figure 4.)

3 Arbitrary Integral NMDAs

Unfortunately, we show that the family of integral NMDAs in which discount factors can be chosen arbitrarily is not closed under determinization and under basic algebraic operations.

▶ **Theorem 4.** *There exists an integral NMDA that no integral DMDA is equivalent to.*

Proof sketch. Consider the integral NMDA $\mathcal{B}$ depicted in Figure 3. We show that for every $n \in \mathbb{N}$, $\mathcal{B}(a^n b^\omega) = 1 - \frac{1}{2^{n+1}}$ and $\mathcal{B}(a^n c^\omega) = 1 + \frac{1}{3^{n+1}}$.

An integral DMDA $\mathcal{D}$ that is equivalent to $\mathcal{B}$ will intuitively need to preserve an accumulated discount factor Π_n and an accumulated weight W_n on every a^n prefix, such that both suffixes of b^ω and c^ω will match the value of $\mathcal{B}$. Since the difference between the required value of each pair $\langle a^n b^\omega, a^n c^\omega \rangle$ is "relatively large", Π_n must have "many" small discount factors of 2 to compensate this difference. But too many discount factors of 2 will not allow to achieve the "delicate" values of $1 + \frac{1}{3^{n+1}}$. In the full proof, we formally analyze the mathematical properties of Π_n, showing that its prime-factor decomposition must indeed contain mostly 2's, "as well as" mostly 3's, leading to a contradiction. ◀

In the following proof that integral NMDAs are not closed under algebraic operations, we cannot assume toward contradiction a candidate deterministic automaton, and thus, as opposed to the proof of Theorem 4, we cannot assume a specific accumulative discount factor for each word prefix. Yet, we analyze the behavior of a candidate nondeterministic automaton on an infinite series of words, and build on the observation that there must be a state that appears in "the same position of the run" in infinitely many optimal runs of the automaton on these words.

▶ **Theorem 5.** *There exist integral NMDAs (even deterministic integral NDAs) $\mathcal{A}$ and $\mathcal{B}$ over the same alphabet, such that no integral NMDA is equivalent to $\max(\mathcal{A}, \mathcal{B})$, and no integral NMDA is equivalent to $\mathcal{A} + \mathcal{B}$.*

Proof. Consider the NMDAs $\mathcal{A}$ and $\mathcal{B}$ depicted in Figure 4, and assume towards contradiction that there exists an integral NMDA $\mathcal{C}'$ such that for every $n \in \mathbb{N}$,

$$\mathcal{C}'(a^n b^\omega) = \max(\mathcal{A}, \mathcal{B})(a^n b^\omega) = \Big(\mathcal{A} + \mathcal{B}\Big)(a^n b^\omega) = \begin{cases} \frac{1}{2^n} & n \text{ is odd} \\ \frac{1}{3^n} & n \text{ is even} \end{cases}$$

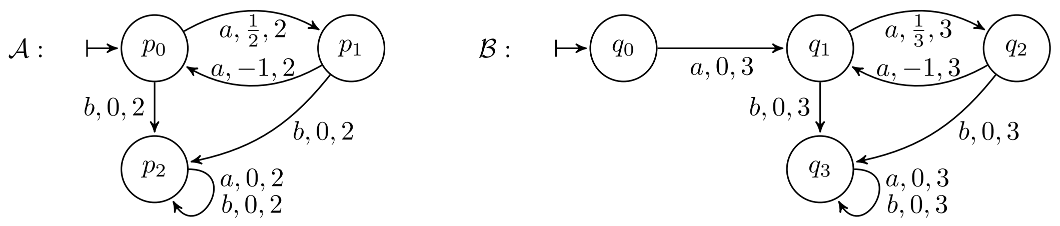

Figure 4 Deterministic integral NDAs that no integral NMDA is equivalent to their max or addition.

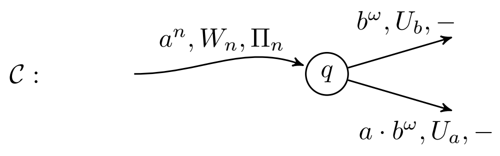

Figure 5 The state q and the notations from the proof of Theorem 5, for two different even $n \in \mathbb{N}$ such that $\delta(r_n[1..n]) = q$. The labels on the walks indicate the input word and the accumulated weight and discount factors.

Let $d \in \mathbb{N}$ be the least common denominator of the weights in $\mathcal{C}'$, and consider the NMDA $\mathcal{C} = \langle \Sigma, Q, \iota, \delta, \gamma, \rho \rangle$ created from $\mathcal{C}'$ by multiplying all its weights by d. Observe that all the weights in $\mathcal{C}$ are integers. According to Proposition 2, for every $n \in \mathbb{N}$, we have $\mathcal{C}(a^n b^\omega) = d \cdot \mathcal{C}'(a^n b^\omega) = \begin{cases} \frac{d}{2^n} & n \text{ is odd} \\ \frac{d}{3^n} & n \text{ is even} \end{cases}$

For every even $n \in \mathbb{N}$, let $w_n = a^n b^\omega$, and r_n a run of $\mathcal{C}$ on w_n that entails the minimal value of $\frac{d}{3^n}$. Since $\mathcal{C}$ is finite, there exists a state $q \in Q$ such that for infinitely many even $n \in \mathbb{N}$, the target state of r_n after n steps is q, i.e, $\delta(r_n[0..n-1]) = q$. We now show that the difference between $U_b = \mathcal{C}^q(b^\omega)$ and $U_a = \mathcal{C}^q(a \cdot b^\omega)$, the weights of the b^ω and $a \cdot b^\omega$ suffixes starting at q, discounted by $\Pi_n = \rho(r_n[0..n-1])$, which is the accumulated discount factor of the prefix of r_n up to q, is approximately $\frac{1}{2^n}$ (See Figure 5 for the notations). Since the weights of the prefixes are constant, for large enough n we will conclude that $m_1 \cdot 2^n \geq \Pi_n$ for some positive constant m_1.

For every such $n \in \mathbb{N}$, let $W_n = \mathcal{C}(r_n[0..n-1])$, and since $\mathcal{C}(r_n) = \frac{d}{3^n}$, we have

$$W_n + \frac{U_b}{\Pi_n} = \frac{d}{3^n} \tag{1}$$

Since the value of every run of $\mathcal{C}$ on $a^{n+1}b^\omega$ is at least $\frac{d}{2^{n+1}}$, we have $W_n + \frac{U_a}{\Pi_n} \geq \frac{d}{2^{n+1}}$. Hence, $\frac{d}{3^n} - \frac{U_b}{\Pi_n} + \frac{U_a}{\Pi_n} \geq \frac{d}{2^{n+1}}$ resulting in $\frac{U_a - U_b}{\Pi_n} \geq d \cdot \left(\frac{1}{2^{n+1}} - \frac{1}{3^n}\right)$. But for large enough n, we have $3^n > 2^{n+2}$, hence we get $\frac{1}{2^{n+1}} - \frac{1}{3^n} > \frac{1}{2^{n+1}} - \frac{1}{2^{n+2}} = \frac{1}{2^{n+2}}$, resulting in $\frac{U_a - U_b}{d} \cdot 2^{n+2} \geq \Pi_n$. And indeed, there exists a positive constant $m_1 = \frac{U_a - U_b}{d} \cdot 2^2$ such that $m_1 \cdot 2^n \geq \Pi_n$.

Now, U_b is a rational constant, otherwise Equation (1) cannot hold, as the other elements are rationals. Hence, there exist $x \in \mathbb{Z}$ and $y \in \mathbb{N}$ such that $U_b = \frac{x}{y}$, and $\frac{1}{3^n} = \frac{W_n \cdot \Pi_n + U_b}{d \cdot \Pi_n} = \frac{W_n \cdot \Pi_n + \frac{x}{y}}{d \cdot \Pi_n} = \frac{W_n \cdot \Pi_n \cdot y + x}{d \cdot y \cdot \Pi_n}$. Since the denominator and the numerator of the right-hand side are integers, we conclude that there exists a positive constant $m_2 = d \cdot y$, such that $m_2 \cdot \Pi_n \geq 3^n$. Eventually, we get $m_1 \cdot m_2 \cdot 2^n \geq 3^n$, for some positive constants m_1 and m_2, and for infinitely many $n \in \mathbb{N}$. But this stands in contradiction with $\lim_{n \to \infty} \left(\frac{2}{3}\right)^n = 0$. ◀

4 Tidy NMDAs

We present the family of "tidy NMDAs" and show that it is as expressive as deterministic NMDAs with arbitrary integral discount factors. Intuitively, an integral NMDA is tidy if the choice of discount factors depends on the word prefix read so far. We further show that for every choice function θ, the class of all θ-NMDAs is closed under determinization and algebraic operations, and enjoys decidable algorithms for its decision problems.

The family of tidy NMDAs contains various other natural subfamilies, such as integral NMDAs in which the discount factors are chosen per letter (action) or per the elapsed time, on which we elaborate at the end of this section. Each of these subfamilies strictly extends the expressive power of integral NDAs.

▶ **Definition 6.** *An integral NMDA $\mathcal{A}$ over an alphabet Σ and with discount-factor function ρ is* tidy *if there exists a function $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0,1\}$, such that for every finite word $u = \sigma_1 \dots \sigma_n \in \Sigma^+$, and every run $q_0, \sigma_1, \cdots, q_n$ of $\mathcal{A}$ on u, we have $\rho(q_{n-1}, \sigma_n, q_n) = \theta(u)$.*

In this case we say that $\mathcal{A}$ is a θ-NMDA.

▶ **Definition 7.** *For an alphabet Σ, a function $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0,1\}$ is a* choice function *if there exists an integral NMDA that is a θ-NMDA.*

For choice functions θ_1 and θ_2, the classes of θ_1-NMDAs and of θ_2-NMDAs are *equivalent* if they express the same functions, namely if for every θ_1-NMDA $\mathcal{A}$, there exists a θ_2-NMDA $\mathcal{B}$ equivalent to $\mathcal{A}$ and vice versa.

For every tidy NMDA $\mathcal{A}$ and finite word u, all the runs of $\mathcal{A}$ on u entail the same accumulated discount factor. We thus use the notation $\rho(u)$ to denote $\rho(r)$, where r is any run of $\mathcal{A}$ on u.

Observe that a general function $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0,1\}$ might require an infinite representation. Yet, we will show in Theorem 9 that every choice function has a finite representation.

Determinizability

We determinize a tidy NMDA by generalizing the determinization algorithm presented in [6] for NDAs. The basic idea in that algorithm is to extend the subset construction, by not only storing in each state of the deterministic automaton whether or not each state q of the original automaton $\mathcal{A}$ is reachable, but also the "gap" that q has from the currently optimal state q' of $\mathcal{A}$. This gap stands for the difference between the accumulated weights for reaching q and for reaching q', multiplied by the accumulated discounted factor. Since we consider tidy NMDAs, we can generalize this view of gaps to the setting of multiple discount factors, as it is guaranteed that the runs to q and to q' accumulated the same discount factor.

▶ **Theorem 8.** *For every choice function θ and a θ-NMDA $\mathcal{A}$, there exists a θ-DMDA $\mathcal{D} \equiv \mathcal{A}$ of size in $2^{O(|\mathcal{A}|)}$. Every state of $\mathcal{D}$ can be represented in space polynomial in $|\mathcal{A}|$.*

Representing Choice Functions

We show that, as opposed to the case of a general function $f : \Sigma^+ \to \mathbb{N} \setminus \{0,1\}$, every choice function θ can be finitely represented by a transducer.

A transducer $\mathcal{T}$ (Mealy machine) is a 6-tuple $\langle P, \Sigma, \Gamma, p_0, \delta, \rho \rangle$, where P is a finite set of states, Σ and Γ are finite sets called the input and output alphabets, $p_0 \in P$ is the initial state, $\delta : P \times \Sigma \to P$ is the total transition function and $\rho : P \times \Sigma \to \Gamma$ is the total output function.

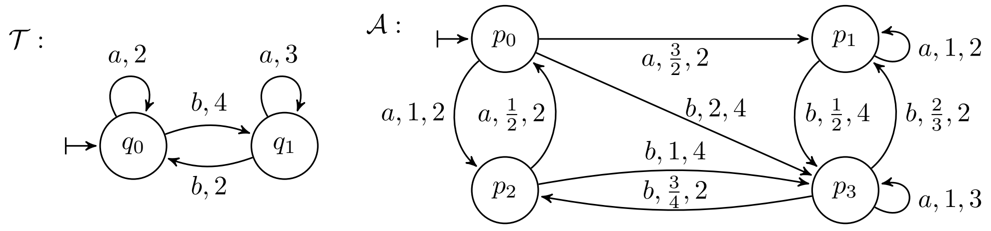

Figure 6 A transducer $\mathcal{T}$ and a $\mathcal{T}$-NMDA.

A transducer $\mathcal{T}$ represents a function, to which for simplicity we give the same name $\mathcal{T} : \Sigma^+ \to \Gamma$, such that for every word w, the value $\mathcal{T}(w)$ is the output label of the last transition taken when running $\mathcal{T}$ on w. The size of $\mathcal{T}$, denoted by $|\mathcal{T}|$, is the maximum between the number of transitions and the maximal binary representation of any output in the range of ρ.

Since in this work we only consider transducers in which the output alphabet Γ is the natural numbers $\mathbb{N}$, we omit Γ from their description, namely write $\langle P, \Sigma, p_0, \delta, \rho\rangle$ instead of $\langle P, \Sigma, \mathbb{N}, p_0, \delta, \rho\rangle$. An example of a transducer $\mathcal{T}$ and a $\mathcal{T}$-NMDA is given in Figure 6.

▶ **Theorem 9.** *For every function $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0, 1\}$, θ is a choice function, namely there exists a θ-NMDA, if and only if there exists a transducer $\mathcal{T}$ such that $\theta \equiv \mathcal{T}$.*

Closure under Algebraic Operations

▶ **Theorem 10.** *For every choice function θ, the set of θ-NMDAs is closed under the operations of min, max, addition, subtraction, and multiplication by a rational constant.*

Proof. Consider a choice function θ and θ-NMDAs $\mathcal{A}$ and $\mathcal{B}$.

- *Multiplication by constant $c \geq 0$*: A θ-NMDA for $c \cdot \mathcal{A}$ is straightforward from Proposition 2.
- *Multiplication by -1*: A θ-NMDA for $-\mathcal{A}$ can be achieved by first determinizing $\mathcal{A}$, as per Theorem 8, into a θ-DMDA $\mathcal{D}$ and then multiplying all the weights in $\mathcal{D}$ by -1.
- *Addition*: Considering $\mathcal{A} = \langle \Sigma, Q_1, \iota_1, \delta_1, \gamma_1, \rho_1\rangle$ and $\mathcal{B} = \langle \Sigma, Q_2, \iota_2, \delta_2, \gamma_2, \rho_2\rangle$, a θ-NMDA for $\mathcal{A} + \mathcal{B}$ can be achieved by constructing the product automaton $\mathcal{C} = \langle \Sigma, Q_1 \times Q_2, \iota_1 \times \iota_2, \delta, \gamma, \rho\rangle$ such that $\delta = \big\{\big((q_1, q_2), \sigma, (p_1, p_2)\big) \bigm| (q_1, \sigma, p_1) \in \delta_1 \text{ and } (q_2, \sigma, p_2) \in \delta_2\big\}$, $\gamma\big((q_1, q_2), \sigma, (p_1, p_2)\big) = \gamma_1(q_1, \sigma, p_1) + \gamma_2(q_2, \sigma, p_2)$, $\rho\big((q_1, q_2), \sigma, (p_1, p_2)\big) = \rho_1(q_1, \sigma, p_1) = \rho_2(q_2, \sigma, p_2)$. The latter must hold since both ρ_1 and ρ_2 are compliant with θ.
- *Subtraction*: A θ-NMDA for $\mathcal{A} - \mathcal{B}$ can be achieved by i) Determinizing $\mathcal{B}$ to $\mathcal{B}'$; ii) Multiplying $\mathcal{B}'$ by -1, getting $\mathcal{B}''$; and iii) Constructing a θ-NMDA for $\mathcal{A} + \mathcal{B}''$.
- *min*: A θ-NMDA for $\min(\mathcal{A}, \mathcal{B})$ is straightforward by the nondeterminism on their union.
- *max*: A θ-NMDA for $\max(\mathcal{A}, \mathcal{B})$ can be achieved by i) Determinizing $\mathcal{A}$ and $\mathcal{B}$ to $\mathcal{A}'$ and $\mathcal{B}'$, respectively; ii) Multiplying $\mathcal{A}'$ and $\mathcal{B}'$ by -1, getting $\mathcal{A}''$ and $\mathcal{B}''$, respectively; iii) Constructing a θ-NMDA $\mathcal{C}''$ for $\min(\mathcal{A}'', \mathcal{B}'')$; iv) Determinizing $\mathcal{C}''$ into a θ-DMDA $\mathcal{D}$; and v) Multiplying $\mathcal{D}$ by -1, getting θ-NMDA $\mathcal{C}$, which provides $\max(\mathcal{A}, \mathcal{B})$.

◀

We analyze next the size blow-up involved in algebraic operations. Most results in Table 1 are straightforward from the constructions presented in the proof of Theorem 10, however the size blow-up of the max operation is a little more involved. At a first glance, determinizing back and forth might look like requiring a double-exponential blow-up, however

$c \cdot \mathcal{A}$ (for $c \geq 0$)	$\min(\mathcal{A}, \mathcal{B})$	$\mathcal{A} + \mathcal{B}$	$-\mathcal{A}$	$\max(\mathcal{A}, \mathcal{B})$	$\mathcal{A} - \mathcal{B}$
Linear		Quadratic	Single Exponential		

Table 1 The size blow-up involved in algebraic operations on tidy NMDAs.

in this case an optimized procedure for the second determinization can achieve an overall single-exponential blow-up: Determinizing a tidy NMDA that is the union of two DMDAs, in which the transition weights are polynomial in the number of states, is shown to only involve a polynomial size blow-up.

▶ **Theorem 11.** *The size blow-up involved in the* max *operation on tidy NMDAs is at most single-exponential.*

We are not aware of prior lower bounds on the size blow-up involved in algebraic operations on NDAs. For achieving such lower bounds, we develop a general scheme to convert every NFA to a λ-NDA of linearly the same size that defines the same language with respect to a threshold value 0, and to convert some specific λ-NDAs back to corresponding NFAs.

The conversion of an NFA to a corresponding λ-NDA is quite simple. It roughly uses the same structure of the original NFA, and assigns four different transitions weights, depending on whether each of the source and target states is accepting or rejecting.

▶ **Lemma 12.** *For every $\lambda \in \mathbb{N} \setminus \{0, 1\}$ and NFA $\mathcal{A}$ with n states, there exists a λ-NDA $\tilde{\mathcal{A}}$ with $n + 2$ states, such that for every word $u \in \Sigma^+$, we have $u \in L(\mathcal{A})$ iff $\tilde{\mathcal{A}}(u) < 0$. That is, the language defined by $\mathcal{A}$ is equivalent to the language defined by $\tilde{\mathcal{A}}$ and the threshold 0.*

Converting an NDA to a corresponding NFA is much more challenging, since a general NDA might have arbitrary weights. We develop a conversion scheme, whose correctness proof is quite involved, from every NDA $\dot{\mathcal{B}}$ that is equivalent to $-\tilde{\mathcal{A}}$, where $\tilde{\mathcal{A}}$ is generated from an arbitrary NFA as per Lemma 12, to a corresponding NFA $\mathcal{B}$. Notice that the assumption that $\dot{\mathcal{B}} \equiv -\tilde{\mathcal{A}}$ gives us some information on $\dot{\mathcal{B}}$, yet $\dot{\mathcal{B}}$ might a priori still have arbitrary transition weights. Using this scheme, we provide an exponential lower bound on the size blow-up involved in multiplying an NDA by (-1). The theorem holds with respect to both finite and infinite words.

▶ **Theorem 13.** *For every $n \in \mathbb{N}$ and $\lambda \in \mathbb{N} \setminus \{0, 1\}$, there exists a λ-NDA $\mathcal{A}$ with n states over a fixed alphabet, such that every λ-NDA that is equivalent to $-\mathcal{A}$, w.r.t. finite or infinite words, has $\Omega(2^n)$ states.*

Proof sketch. NFA complementation is known to impose an exponential state blow-up [33, 26]. Hence, a conversion of an NFA $\mathcal{A}$ to a λ-NDA $\tilde{\mathcal{A}}$ as per Lemma 12, and a polynomial conversion of every λ-NDA $\dot{\mathcal{B}} \equiv -\tilde{\mathcal{A}}$ to a corresponding NFA $\mathcal{B}$, will show the required lower bound.

We provide such a back-conversion for NDAs whose values on the input words converge to some threshold as the words length grow to infinity, which is the case with every such $\dot{\mathcal{B}}$.

We first construct from $\dot{\mathcal{B}}$ a similar equivalent λ-NDA $\mathcal{B}'$ whose initial states have no incoming transitions. This eliminates the possibility that one run is a suffix of another, allowing to simplify some of our arguments. We then define $\hat{\delta}$ to be the transitions of $\mathcal{B}'$ that participate in some minimal run of $\mathcal{B}'$ on a word whose value is smaller than 0, and $\hat{\hat{\delta}} \subseteq \hat{\delta}$ to have those of them that are the last transition of such runs.

We define the NFA $\mathcal{B}$ to have the states of $\mathcal{B}'$, but only the transitions from $\hat{\delta}$. Its accepting states are clones of the target states of transitions in $\hat{\hat{\delta}}$, but without outgoing transitions. We then prove that $\mathcal{B}$ accepts a word u iff $\mathcal{B}'(u) = -\frac{1}{\lambda^{|u|}}$.

The first direction is easy: if $\mathcal{B}'(u) = -\frac{1}{\lambda^{|u|}}$, we get that all the transitions of a minimal run of $\mathcal{B}'$ on u are in $\hat{\delta}$, and its final transition is in $\hat{\hat{\delta}}$, hence there exists a run of $\mathcal{B}$ on u ending at an accepting state.

For the other direction, we assume towards contradiction that there exists a word u, such that $\mathcal{B}'(u) \neq -\frac{1}{\lambda^{|u|}}$, while there is an accepting run r_u of $\mathcal{B}$ on u. We define the "normalized value" of a run r' of $\mathcal{B}'$ as the value of $\mathcal{B}'$ multiplied by the accumulated discount factor, i.e., $\mathcal{B}'(r') \cdot \lambda^{|r'|}$. According to the special values assigned by $\mathcal{B}'$, whenever the normalized value reaches -1, we have an "accepting" run. We show that r_u and the structure of $\mathcal{B}$ imply the existence of two "accepting" runs $r'_1, r'_2 \in R^-$ that intersect in some state q, such that taking the prefix of r'_1 up to q results in a normalized value $\lambda^k W_1$ that is strictly smaller than the normalized value $\lambda^j W_2$ of the prefix of r'_2 up to q. Since r'_2 is an "accepting" run, the suffix of r'_2 reduces $\lambda^j W_2$ to -1 and therefore it will reduce $\lambda^k W_1$ to a value strictly smaller than -1, and the total value of the run to a value strictly smaller than $-\frac{1}{\lambda^n}$, which is not a possible value of $\mathcal{B}'$.

For showing the lower bound for NDAs that run on infinite words, we properly adjust the proof to consider words of the form $u \cdot \#^\omega$, for a fresh letter #, rather than finite words. ◀

Basic Subfamilies

Tidy NMDAs constitute a rich family that also contains some basic subfamilies that are still more expressive than integral NDAs. Two such subfamilies are integral NMDAs in which the discount factors depend on the transition letter or on the elapsed time.

Notice that closure of tidy NMDAs under determinization and under algebraic operations is related to a specific choice function θ, namely every class of θ-NMDAs enjoys these closure properties (Theorems 8 and 10). Since the aforementioned subfamilies of tidy NMDAs also consist of θ-NMDA classes, their closure under determinization and under algebraic operations follows. For example, the class of NMDAs that assigns a discount factor of 2 to the letter 'a' and of 3 to the letter 'b' enjoys these closure properties.

Letter-Oriented Discount Factors

Allowing each action (letter) to carry its own discount factor is a basic extension of discounted summation, used in various models, such as Markov decision processes [29, 38].

A θ-NMDA over an alphabet Σ is *letter oriented* if all transitions over the same alphabet letter share the same discount factor; that is, if $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0,1\}$ coincides with a function $\Lambda : \Sigma \to \mathbb{N} \setminus \{0,1\}$, in the sense that for every finite word u and letter σ, we have $\theta(u\sigma) = \Lambda(\sigma)$. (See an example in Figure 7.) Notice that every choice function θ for a letter-oriented θ-NMDA can be defined via a simple transducer of a single state, having a self loop over every letter with its assigned discount factor.

We show that letter-oriented NMDAs indeed add expressiveness over NDAs.

▶ **Theorem 14.** *There exists a letter-oriented NMDA that no integral NDA is equivalent to.*

Proof sketch. In the proof we consider the NMDA $\mathcal{A}$ depicted in Figure 7, and assume towards contradiction that there exists an integral λ-DDA $\mathcal{B}$ which is equivalent to $\mathcal{A}$. Since $\mathcal{B}$ is deterministic and has finitely many states, after reading only a letters some cycle will be eventually reached. We analyze the runs on words of the form $a^n b^\omega$ for values of n such

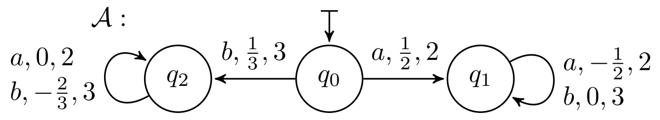

Figure 7 A letter-oriented discounted-sum automaton, for the discount factor function $\Lambda(a) = 2$; $\Lambda(b) = 3$, that no integral NDA is equivalent to.

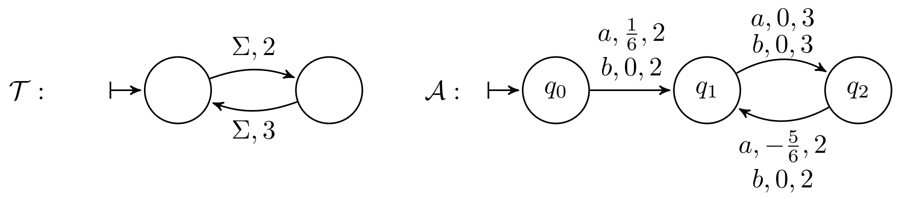

Figure 8 A time-oriented NMDA that no integral NDA is equivalent to, and a transducer that defines its choice function.

that the cycle in $\mathcal{B}$ is not taken at all, such that it is taken once, and such that it is taken twice. Since the accumulated discount factor added by the cycle is a constant equals to λ^i, where i is the length of the cycle, in order for $\mathcal{B}$ to have values of $\mathcal{B}(a^n b^\omega) = \frac{1}{2^n}$ on these words, we conclude that λ must equal 2. We now apply similar analysis regarding words of the form $b^n a^\omega$, for which $\mathcal{B}$ should have values of $\mathcal{B}(b^n a^\omega) = \frac{1}{3^n}$, and a cycle for the b letter, to conclude that λ must equal 3, and reach a contradiction. ◀

Time-Oriented Discount Factors

Allowing the discount factor to change over time is a natural extension of discounted summation, used in various disciplines, such as reinforcement learning [28, 22].

A θ-NMDA over an alphabet Σ is *time oriented* if the discount factor on a transition is determined by the distance of the transition from an initial state; that is, if $\theta : \Sigma^+ \to \mathbb{N} \setminus \{0, 1\}$ coincides with a function $\Lambda : \mathbb{N} \setminus \{0\} \to \mathbb{N} \setminus \{0, 1\}$, in the sense that for every finite word u, we have $\theta(u) = \Lambda\big(|u|\big)$.

For example, the NMDA $\mathcal{A}$ of Figure 8 is time-oriented, as all transitions taken at odd steps, in any run, have discount factor 2, and those taken at even steps have discount factor 3. The transducer $\mathcal{T}$ in Figure 8 represents its choice function.

Time-oriented NMDAs extend the expressiveness of NDAs, as proved for the time-oriented NMDA depicted in Figure 8.

▶ **Theorem 15.** *There exists a time-oriented NMDA that no integral NDA is equivalent to.*

5 Tidy NMDAs – Decision Problems

We show that all of the decision problems of tidy NMDAs are in the same complexity classes as the corresponding problems for discounted-sum automata with a single integral discount factor. That is, the nonemptiness problem is in PTIME, and the exact-value, universality, equivalence, and containment problems are in PSPACE (see Table 2). In the equivalence and containment problems, we consider θ-NMDAs with the same choice function θ. In addition, the problem of checking whether a given NMDA is tidy, as well as whether it is a

	Finite words	Infinite words
Non-emptiness ($<$)	PTIME (Theorem 19)	PTIME (Theorem 18)
Non-emptiness ($\leq$)	PTIME (Theorem 20)	
Containment ($>$)	PSPACE-complete (Theorem 24)	PSPACE (Theorem 26)
Containment ($\geq$)		PSPACE-complete (Theorem 25)
Equivalence	PSPACE-complete (Corollary 27)	
Universality ($<$)	PSPACE-complete (Theorem 28)	PSPACE (Theorem 28)
Universality ($\leq$)		PSPACE-complete (Theorem 28)
Exact-value	PSPACE-complete (Theorem 29)	PSPACE (Theorem 29)

Table 2 The complexities of the decision problems of tidy NMDAs.

θ-NMDA, for a given choice function θ, is decidable in PTIME. The complexities are w.r.t. the automata size (as defined in Section 2), and when considering a threshold ν, w.r.t. its binary representation.

Tidiness

Given an NMDA $\mathcal{A}$, one can check in PTIME whether $\mathcal{A}$ is tidy. The algorithm follows by solving a reachability problem in a Cartesian product of $\mathcal{A}$ with itself, to verify that for every word, the last discount factors are identical in all runs.

▶ **Theorem 16.** *Checking if a given NMDA $\mathcal{A}$ is tidy is decidable in time $O\big(|\mathcal{A}|^2\big)$.*

Given also a transducer $\mathcal{T}$, one can check in polynomial time whether $\mathcal{A}$ is a $\mathcal{T}$-NMDA.

▶ **Theorem 17.** *Checking if a given NMDA $\mathcal{A}$ is a $\mathcal{T}$-NMDA, for a given transducer $\mathcal{T}$, is decidable in time $O\big(|\mathcal{A}| \cdot |\mathcal{T}|\big)$.*

Proof sketch. We construct a nondeterministic weighted automaton that resembles the input NMDA and a deterministic weighted automaton that resembles the input transducer, replacing the original discount factors with weights. We then construct the product of the two automata, setting the transition weights to be the difference between the corresponding weights in the two automata, and check whether the weights on all reachable transitions are zero. ◀

Nonemptiness

We start with nonemptiness with respect to infinite words, for which there is a simple reduction to one-player discounted-payoff games. Notice that it applies to arbitrary NMDAs, not only to tidy ones.

▶ **Theorem 18.** *The nonemptiness problem of NMDAs w.r.t. infinite words is in PTIME for both strict and non-strict inequalities.*

For nonemptiness with respect to finite words, we cannot directly use the aforementioned game solution, as it relies on the convergence of the values in the limit. However, for nonemptiness with respect to strict inequality, we can reduce the finite-words case to the infinite-words case: If there exists an infinite word w such that $\mathcal{A}(w)$ is strictly smaller than the threshold, the distance between them cannot be compensated in the infinity, implying the existence of a finite prefix that also has a value smaller than the threshold; As for the other direction, we add to every state a 0-weight self loop, causing a small-valued finite word to also imply a small-valued infinite word.

▶ **Theorem 19.** *The nonemptiness problem of NMDAs w.r.t. finite words and strict inequality is in PTIME.*

For nonemptiness with respect to finite words and non-strict inequality, we cannot use the construction used in the proof of Theorem 19, since its final part is inadequate: It is possible to have an infinite word with value that equals the threshold, while every finite prefix of it has a value strictly bigger than the threshold. Yet, when considering *integral* NMDAs, we can use a different approach for resolving the problem, applying linear programming to calculate the minimal value of a finite run ending in every state.

▶ **Theorem 20.** *The nonemptiness problem of integral NMDAs w.r.t. finite words and non-strict inequality is in PTIME.*

Exact-Value, Universality, Equivalence, and Containment

We continue with the PSPACE-complete problems, to which we first provide hardness proofs, by reductions from the universality problem of NFAs, known to be PSPACE-complete [31]. Notice that the provided hardness results already stand for integral NDAs, not only to tidy NMDAs.

PSPACE-hardness of the containment problem for NDAs with respect to infinite words and non-strict inequalities is shown in [3]. We provide below more general hardness results, considering the equivalence problem, first with respect to finite words and then with respect to infinite words, as well as the exact-value, universality($\leq$) and universality($<$) problems with respect to finite words.

▶ **Lemma 21.** *The equivalence and universality($\leq$) problems of integral NDAs w.r.t. finite words are PSPACE-hard.*

Proof sketch. Given an NFA A, we construct in polynomial time an NDA $\mathcal{B}$ such that $\mathcal{A}$ is universal if and only if $\mathcal{B}$ gets a value of 0 on all finite words. $\mathcal{B}$ has the same structure as $\mathcal{A}$, and we assign weights on the transitions to guarantee that the value of $\mathcal{B}$ on every word u is at most $\frac{1}{2^{|u|}}$. In addition, we have in $\mathcal{B}$ a new "good" state q_{acc}, and for every original transition to an accepting state q of $\mathcal{A}$, we add in $\mathcal{B}$ a new "good" transition to q_{acc}, such that its weight allows $\mathcal{B}$ to have a value of 0 on a word that reaches q in $\mathcal{A}$. Finally, we add a "bad" transition out of q_{acc}, such that its weight ensures a total positive value, in the case that $\mathcal{B}$ continues the run out of q_{acc}. ◀

▶ **Lemma 22.** *The equivalence and universality($\leq$) problems of integral NDAs w.r.t. infinite words are PSPACE-hard.*

Proof sketch. The reduction from the universality problem of an NFA $\mathcal{A}$ is similar to the one provided in the proof of Lemma 21, with intuitively the following adaptations of the constructed NDA $\mathcal{B}$ to the case of infinite words: We add a new letter # to the alphabet, low-weighted #-transitions from the accepting states, and high-weighted #-transitions from the non-accepting states.

By this construction, the value of $\mathcal{B}$ on an infinite word $u \cdot \# \cdot w$, where u does not contain #, will be 0 if and only if $\mathcal{A}$ accepts u.

Notice that the value of $\mathcal{B}$ on an infinite word that does not contain # is also 0, as it is $\lim_{n\to\infty} \frac{1}{2^n}$. ◀

▶ **Lemma 23.** *The universality($<$) and exact-value problems of integral NDAs w.r.t. finite words are PSPACE-hard.*

We continue with the PSPACE upper bounds. The containment problem of NDAs was proved in [3] to be in PSPACE, using comparators to reduce the problem to language inclusion between Büchi automata. Our approach for the containment problem of NMDAs is different, and it also improves the complexity provided in [3] for NDAs (having a single discount factor), as we refer to binary representation of weights, while [3] assumes unary representation.[4]

Our algorithm for solving the containment problem between θ-NMDAs $\mathcal{A}$ and $\mathcal{B}$ is a nondeterministic polynomial space algorithm that determines the opposite, meaning whether there exists a word w such that $\mathcal{A}(w) - \mathcal{B}(w) < 0$ for containment($\geq$) or $\mathcal{A}(w) - \mathcal{B}(w) \leq 0$ for containment($>$), to conclude that the problems are in co-NPSPACE and hence in PSPACE. We perform the determinization of $\mathcal{B}$ on-the-fly into a DMDA $\mathcal{D}$, and simulate on the fly a θ-NMDA for the difference between $\mathcal{A}$ and $\mathcal{D}$. We then non-deterministically guess a run r that witnesses a negative value of the difference automaton, while ensuring that the entire process only uses space polynomial in the size of the input automata. For meeting this space requirement, after each step of the run r, the algorithm maintains a *local data* consisting of the current state of $\mathcal{A}$, the current state of $\mathcal{D}$ and a "normalized difference" between the values of the runs of $\mathcal{A}$ and $\mathcal{D}$ on the word generated so far. When the normalized difference goes below 0, we have that the generated word w is a witness for $\mathcal{A}(w) < \mathcal{D}(w)$, when it gets to 0 we have a witness for $\mathcal{A}(w) = \mathcal{D}(w)$, and when it exceeds a certain *maximal recoverable difference*, which is polynomial in $|\mathcal{A}| + |\mathcal{B}|$, no suffix can be added to w for getting a witness.

▶ **Theorem 24.** *For every choice function θ, the containment problem of θ-NMDAs w.r.t. finite words is PSPACE-complete for both strict and non-strict inequalities.*

The algorithm for determining containment($\geq$) in the infinite-words settings is similar to the one presented for finite words, with the difference that rather than witnessing a finite word w, such that $\mathcal{A}(w) - \mathcal{B}(w) < 0$, we witness a finite prefix u (of an infinite word w), such that the normalized difference between $\mathcal{A}(u)$ and $\mathcal{B}(u)$ (taking into account the accumulated discount factor on u) is bigger than some fixed threshold.

▶ **Theorem 25.** *For every choice function θ, the containment problem of θ-NMDAs w.r.t. infinite words and non-strict inequality is PSPACE-complete.*

To find a witness for strict non-containment in the infinite-words setting, we adapt the proof of Theorem 25 by adding an accept condition for detecting convergence of the difference between the two automata values to the threshold value, which is the existence of a cycle with the same normalized difference.

▶ **Theorem 26.** *For every choice function θ, the containment problem of θ-NMDAs w.r.t. infinite words and strict inequality is in PSPACE.*

A PSPACE algorithm for equivalence directly follows from the fact that $\mathcal{A} \equiv \mathcal{B}$ if and only if $\mathcal{A} \geq \mathcal{B}$ and $\mathcal{B} \geq \mathcal{A}$.

▶ **Corollary 27.** *The equivalence problem of tidy NMDAs is PSPACE-complete.*

[4] Rational weights are assumed to have a common denominator, both by us and by [3], where in the latter it is stated implicitly, by providing the complexity analysis with respect to transition weights that are natural numbers.

We continue with the universality problems which are special cases of the containment problems.

▶ **Theorem 28.** *The universality problems of tidy NMDAs are in PSPACE. The universality($<$) w.r.t. finite words, universality($\leq$) w.r.t. finite words, and universality($\leq$) w.r.t. infinite words are PSPACE-complete.*

▶ **Theorem 29.** *The exact-value problem of tidy NMDAs is in PSPACE (and PSPACE-complete w.r.t. finite words).*

6 Conclusions

The measure functions most commonly used in the field of quantitative verification, whether for describing system properties [9, 17, 30], automata valuation schemes [5, 6, 14, 3], game winning conditions [2, 18, 39], or temporal specifications [1, 4, 16, 35], are the limit-average (mean payoff) and the discounted-sum functions.

Limit-average automata cannot always be determinized [14] and checking their (non-strict) universality is undecidable [18]. Therefore, the tendency is to only use deterministic such automata, possibly with the addition of algebraic operations on them [10].

Discounted-sum automata with arbitrary rational discount factors also cannot always be determinized [14] and are not closed under algebraic operations [6]. Yet, with integral discount factors, they do enjoy all of these closure properties and their decision problems are decidable [6]. They thus provide a very interesting automata class for quantitative verification. Yet, they have a main drawback of only allowing a single discount factor.

We define a rich class of discounted-sum automata with multiple integral factors (tidy NMDAs) that strictly extends the expressiveness of automata with a single factor, while enjoying all of the good properties of the latter, including the same complexity of the required decision problems. We thus believe that tidy NMDAs can provide a natural and useful generalization of integral discounted-sum automata in all fields, and especially in quantitative verification of reinforcement learning applications, as novel approaches in this field extend the single discount factor that is used in the calculation of the expected return value to multiple ones [28, 22, 32].

References

1 Shaull Almagor, Udi Boker, and Orna Kupferman. Discounting in LTL. In *proceedings of TACAS*, volume 8413 of *LNCS*, pages 424–439, 2014. doi:10.1007/978-3-642-54862-8_37.

2 Daniel Andersson. An improved algorithm for discounted payoff games. In *proceedings of ESSLLI Student Session*, pages 91–98, 2006.

3 Suguman Bansal, Swarat Chaudhuri, and Moshe Y. Vardi. Comparator automata in quantitative verification. In *proceedings of FoSSaCS*, volume 10803 of *LNCS*, pages 420–437, 2018. doi:10.1007/978-3-319-89366-2_23.

4 Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. *ACM Trans. Comput. Log.*, 15(4):27:1–27:25, 2014. doi:10.1145/2629686.

5 Udi Boker and Thomas A. Henzinger. Approximate determinization of quantitative automata. In *proceedings of FSTTCS*, volume 18 of *LIPIcs*, pages 362–373, 2012. doi:10.4230/LIPIcs.FSTTCS.2012.362.

6 Udi Boker and Thomas A. Henzinger. Exact and approximate determinization of discounted-sum automata. *Log. Methods Comput. Sci.*, 10(1), 2014. doi:10.2168/LMCS-10(1:10)2014.

7 Udi Boker, Thomas A. Henzinger, and Jan Otop. The target discounted-sum problem. In *proceedings of LICS*, pages 750–761, 2015. doi:10.1109/LICS.2015.74.

8 Udi Boker, Orna Kupferman, and Michal Skrzypczak. How deterministic are good-for-games automata? In *proceedings of FSTTCS*, volume 93 of *LIPIcs*, pages 18:1–18:14, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.18.

9 Krishnendu Chatterjee. Markov decision processes with multiple long-run average objectives. In *proceedings of FSTTCS*, volume 4855 of *LNCS*, pages 473–484. Springer, 2007. doi:10.1007/978-3-540-77050-3_39.

10 Krishnendu Chatterjee, Laurent Doyen, Herbert Edelsbrunner, Thomas A. Henzinger, and Philippe Rannou. Mean-payoff automaton expressions. In *proceedings of CONCUR*, volume 6269 of *LNCS*, pages 269–283, 2010. doi:10.1007/978-3-642-15375-4_19.

11 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Alternating weighted automata. In *proceedings of FCT*, volume 5699 of *LNCS*, pages 3–13, 2009. doi:10.1007/978-3-642-03409-1_2.

12 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Probabilistic weighted automata. In *proceedings of CONCUR*, volume 5710 of *LNCS*, pages 244–258, 2009. doi:10.1007/978-3-642-04081-8_17.

13 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Expressiveness and closure properties for quantitative languages. *Log. Methods Comput. Sci.*, 6(3), 2010. URL: http://arxiv.org/abs/1007.4018.

14 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log.*, 11(4):23:1–23:38, 2010. doi:10.1145/1805950.1805953.

15 Krishnendu Chatterjee, Vojtech Forejt, and Dominik Wojtczak. Multi-objective discounted reward verification in graphs and mdps. In *proceedings of LPAR*, volume 8312 of *LNCS*, pages 228–242, 2013. doi:10.1007/978-3-642-45221-5_17.

16 Luca de Alfaro, Marco Faella, Thomas A. Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. *Theor. Comput. Sci.*, 345(1):139–170, 2005. doi:10.1016/j.tcs.2005.07.033.

17 Luca de Alfaro, Thomas A. Henzinger, and Rupak Majumdar. Discounting the future in systems theory. In *proceedings of ICALP*, volume 2719, pages 1022–1037, 2003. doi:10.1007/3-540-45061-0_79.

18 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Torunczyk. Energy and mean-payoff games with imperfect information. In *proceedings of CSL*, volume 6247 of *LNCS*, pages 260–274, 2010. doi:10.1007/978-3-642-15205-4_22.

19 Manfred Droste and Dietrich Kuske. Skew and infinitary formal power series. *Theor. Comput. Sci.*, 366(3):199–227, 2006. doi:10.1016/j.tcs.2006.08.024.

20 Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Finite-valued weighted automata. In *proceedings of FSTTCS*, volume 29 of *LIPIcs*, pages 133–145, 2014. doi:10.4230/LIPIcs.FSTTCS.2014.133.

21 Emmanuel Filiot, Raffaella Gentilini, and Jean-François Raskin. Quantitative languages defined by functional automata. *Log. Methods Comput. Sci.*, 11(3), 2015. doi:10.2168/LMCS-11(3:14)2015.

22 Vincent François-Lavet, Raphaël Fonteneau, and Damien Ernst. How to discount deep reinforcement learning: Towards new dynamic strategies. *CoRR*, 2015. URL: http://arxiv.org/abs/1512.02011.

23 Hugo Gimbert and Wieslaw Zielonka. Limits of multi-discounted markov decision processes. In *proceedings of LICS*, pages 89–98, 2007. doi:10.1109/LICS.2007.28.

24 Guy Hefetz. Discounted-sum automata with multiple discount factors. Master's thesis, IDC, Herzliya, Israel, 2020. URL: https://www.idc.ac.il/en/schools/cs/research/documents/thesis-guy-hefetz.pdf.

25 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *proceedings of CSL*, volume 4207 of *LNCS*, pages 395–410, 2006. doi:10.1007/11874683_26.

26 Galina Jirásková. State complexity of some operations on binary regular languages. *Theor. Comput. Sci.*, 330(2):287–298, 2005. doi:10.1016/j.tcs.2004.04.011.

27 Yafim Kazak, Clark W. Barrett, Guy Katz, and Michael Schapira. Verifying deep-rl-driven systems. In *proceedings of NetAI@SIGCOMM*, pages 83–89, 2019. doi:10.1145/3341216.3342218.

28 Tor Lattimore and Marcus Hutter. Time consistent discounting. In *proceedings of ALT*, volume 6925 of *LNCS*, pages 383–397, 2011. doi:10.1007/978-3-642-24412-4_30.

29 Fernando Luque-Vásquez and J. Adolfo Minjárez-Sosa. *Iteration Algorithms in Markov Decision Processes with State-Action-Dependent Discount Factors and Unbounded Costs*, chapter 4, pages 55–69. Operations Research: the Art of Making Good Decisions. IntechOpen, 2017. doi:10.5772/65044.

30 Omid Madani, Mikkel Thorup, and Uri Zwick. Discounted deterministic markov decision processes and discounted all-pairs shortest paths. *ACM Trans. Algorithms*, 6(2):33:1–33:25, 2010. doi:10.1145/1721837.1721849.

31 Albert R. Meyer and Larry J. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. In *proceedings of 13th IEEE Symp. on Switching and Automata Theory*, pages 125–129, 1972. doi:10.1109/SWAT.1972.29.

32 Chris Reinke, Eiji Uchibe, and Kenji Doya. Average reward optimization with multiple discounting reinforcement learners. In *proceedings of ICONIP*, volume 10634 of *LNCS*, pages 789–800, 2017. doi:10.1007/978-3-319-70087-8_81.

33 William J. Sakoda and Michael Sipser. Nondeterminism and the size of two way finite automata. In *proceedings of STOC*, pages 275–286, 1978. doi:10.1145/800133.804357.

34 Richard S. Sutton and Andrew G.Barto. *Introduction to Reinforcement Learning.* MIT Press, 1998. URL: http://dl.acm.org/doi/book/10.5555/551283.

35 Takashi Tomita, Shin Hiura, Shigeki Hagihara, and Naoki Yonezaki. A temporal logic with mean-payoff constraints. In *proceedings of ICFEM*, volume 7635 of *LNCS*, pages 249–265. Springer, 2012. doi:10.1007/978-3-642-34281-3_19.

36 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *proceedings of FOCS*, pages 327–338, 1985. doi:10.1109/SFCS.1985.12.

37 Yufei Wang, Qiwei Ye, and Tie-Yan Liu. Beyond exponentially discounted sum: Automatic learning of return function. *CoRR*, 2019. URL: http://arxiv.org/abs/1905.11591.

38 Xiao Wu and Xianping Guo. Convergence of Markov decision processes with constraints and state-action dependent discount factors. *Sci. China Math.*, 63:167–182, 2020. doi:10.1007/s11425-017-9292-1.

39 Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.*, 158:343–359, 1996. doi:10.1016/0304-3975(95)00188-3.

A Selected Proofs

This appendix presents some of the omitted proofs. All the full proofs can be found in [24].

Proof of Lemma 12. Given an NFA $\mathcal{A} = \langle \Sigma, Q, \iota, \delta, F\rangle$ and a discount factor $\lambda \in \mathbb{N} \setminus \{0,1\}$, we construct a λ-NDA $\tilde{\mathcal{A}} = \langle \Sigma, Q', \{p_0\}, \delta', \gamma'\rangle$ for which there exists a bijection f between the runs of $\mathcal{A}$ and the runs of $\tilde{\mathcal{A}}$ such that for every run r of $\tilde{\mathcal{A}}$ on a word u,

- r is an accepting run of $\mathcal{A}$ iff $f(r)$ is a run of $\tilde{\mathcal{A}}$ on u with the value $\tilde{\mathcal{A}}\big(f(r)\big) = -\frac{1}{\lambda^{|r|}}$.
- r is a non-accepting run of $\mathcal{A}$ iff $f(r)$ is a run of $\tilde{\mathcal{A}}$ on u with the value $\tilde{\mathcal{A}}\big(f(r)\big) = \frac{1}{\lambda^{|r|}}$.

We first transform $\mathcal{A}$ to an equivalent NFA $\mathcal{A}' = \langle \Sigma, Q', \{p_0\}, \delta', F\rangle$ that is complete and in which there are no transitions entering its initial state, and later assign weights to its transitions to create $\tilde{\mathcal{A}}$.

To construct $\mathcal{A}'$ we add two states to Q, having $Q' = Q \cup \{p_0, q_{hole}\}$, duplicate all the transitions from ι to start from p_0, and add a transition from every state to q_{hole}, namely $\delta' = \delta \cup \big\{(p_0,\sigma,q) \;\big|\; \exists p \in \iota, (p,\sigma,q) \in \delta\big\} \cup \big\{(q,\sigma,q_{hole}) \;\big|\; q \in Q', \sigma \in \Sigma\big\}$. Observe that $|Q'| = |Q| + 2$, and $L(\mathcal{A}) = L(\mathcal{A}')$. Next, we assign the following transition weights:

- For every $t = (p_0,\sigma,q) \in \delta'$, $\gamma'(t) = -\frac{1}{\lambda}$ if $q \in F$ and $\gamma'(t) = \frac{1}{\lambda}$ if $q \notin F$.
- For every $t = (p,\sigma,q) \in \delta'$ such that $p \neq p_0$, $\gamma'(t) = \frac{\lambda-1}{\lambda}$ if $p,q \in F$; $\gamma'(t) = \frac{\lambda+1}{\lambda}$ if $p \in F$ and $q \notin F$; $\gamma'(t) = -\frac{\lambda+1}{\lambda}$ if $p \notin F$ and $q \in F$; and $\gamma'(t) = -\frac{\lambda-1}{\lambda}$ if $p,q \notin F$.

By induction on the length of the runs on an input word u, one can show that for every $u \in \Sigma^+$, $\tilde{\mathcal{A}}(u) = -\frac{1}{\lambda^{|u|}}$ if $u \in L(\mathcal{A})$ and $\tilde{\mathcal{A}}(u) = \frac{1}{\lambda^{|u|}}$ if $u \notin L(\mathcal{A})$. ◀

Proof of Theorem 13.

Consider $n \in \mathbb{N}$ and $\lambda \in \mathbb{N} \setminus \{0,1\}$. By [33, 26] there exists an NFA $\mathcal{A}$ with n states over a fixed alphabet of two letters, such that any NFA for the complement language $\overline{L(\mathcal{A})}$ has at least 2^n states.

Finite words.

Let $\tilde{\mathcal{A}}$ be a λ-NDA that is correlated to $\mathcal{A}$ as per Lemma 12, and assume towards contradiction that there exists a λ-NDA $\dot{\mathcal{B}} = \langle \Sigma, Q_{\dot{\mathcal{B}}}, \iota_{\dot{\mathcal{B}}}, \delta_{\dot{\mathcal{B}}}, \gamma_{\dot{\mathcal{B}}}\rangle$ with less than $\frac{2^n}{4}$ states such that $\dot{\mathcal{B}} \equiv -\tilde{\mathcal{A}}$.

We provide below a conversion opposite to Lemma 12, leading to an NFA for $\overline{L(\mathcal{A})}$ with less than 2^n states, and therefore to a contradiction. The conversion of $\dot{\mathcal{B}}$ back to an NFA builds on the specific values that $\dot{\mathcal{B}}$ is known to assign to words, as opposed to the construction of Lemma 12, which works uniformly for every NFA, and is much more challenging, since $\dot{\mathcal{B}}$ might have arbitrary transition weights. This conversion scheme can only work for λ-NDAs whose values on the input words converge to some threshold as the words length grow to infinity.

For simplification, we do not consider the empty word, since one can easily check if the input NFA accepts it, and set the complemented NFA to reject it accordingly.

By Lemma 12 we have that for every word $u \in \Sigma^+$, $\tilde{\mathcal{A}}(u) = -\frac{1}{\lambda^{|u|}}$ if $u \in L(\mathcal{A})$ and $\tilde{\mathcal{A}}(u) = \frac{1}{\lambda^{|u|}}$ if $u \notin L(\mathcal{A})$. Hence, $\dot{\mathcal{B}}(u) = -\frac{1}{\lambda^{|u|}}$ if $u \notin L(\mathcal{A})$ and $\dot{\mathcal{B}}(u) = \frac{1}{\lambda^{|u|}}$ if $u \in L(\mathcal{A})$. We will show that there exists an NFA $\mathcal{B}$, with less than 2^n states, such that $u \in L(\mathcal{B})$ iff $\dot{\mathcal{B}}(u) = -\frac{1}{\lambda^{|u|}}$, implying that $L(B) = \overline{L(\mathcal{A})}$.

We first construct a λ-NDA $\mathcal{B}' = \langle \Sigma, Q_{\mathcal{B}'}, \iota, \delta, \gamma\rangle$ that is equivalent to $\dot{\mathcal{B}}$, but has no transitions entering its initial states. This construction eliminates the possibility that one run is a suffix of another, allowing to simplify some of our arguments. Formally, $Q_{\mathcal{B}'} = Q_{\dot{\mathcal{B}}} \cup \iota$, $\iota = \iota_{\dot{\mathcal{B}}} \times \{1\}$, $\delta = \delta_{\dot{\mathcal{B}}} \cup \big\{\big((p,1),\sigma,q\big) \;\big|\; (p,\sigma,q) \in \delta_{\dot{\mathcal{B}}}\big\}$, and weights $\gamma(t) = \gamma_{\dot{\mathcal{B}}}(t)$ if $t \in \delta_{\dot{\mathcal{B}}}$ and $\gamma\big((p,1),\sigma,q\big) = \gamma_{\dot{\mathcal{B}}}(p,\sigma,q)$ otherwise.

Let R^- be the set of all the runs of $\mathcal{B}'$ that entail a minimal value which is less than 0, i.e., $R^- = \{r \mid r \text{ is a minimal run of } \mathcal{B}' \text{ on some word and } \mathcal{B}'(r) < 0\}$. Let $\hat{\delta} \subseteq \delta$ be the set of all the transitions that take part in some run in R^-, meaning $\hat{\delta} = \{r(i) \mid r \in R^- \text{ and } 0 \leq i < |r|\}$, and $\hat{\hat{\delta}} \subseteq \delta$ the set of all transitions that are the last transition of those runs, meaning $\hat{\hat{\delta}} = \big\{r\big(|r|-1\big) \mid r \in R^-\big\}$.

We construct next the NFA $\mathcal{B} = \langle \Sigma, Q_{\mathcal{B}}, \iota, \delta_{\mathcal{B}}, F_{\mathcal{B}}\rangle$. Intuitively, $\mathcal{B}$ has the states of $\mathcal{B}'$, but only the transitions from $\hat{\delta}$. Its accepting states are clones of the target states of the transitions in $\hat{\hat{\delta}}$, but without outgoing transitions. We will later show that the only runs of $\mathcal{B}$ that reach these clones are those that have an equivalent run in R^-. Formally, $Q_{\mathcal{B}} = Q'_{\mathcal{B}} \cup F_{\mathcal{B}}$, $F_{\mathcal{B}} = \big\{(q,1) \mid \exists p, q \in Q'_{\mathcal{B}} \text{ and } (p,\sigma,q) \in \hat{\hat{\delta}}\big\}$, and $\delta_{\mathcal{B}} = \hat{\delta} \cup \big\{\big(p,\sigma,(q,1)\big) \mid (p,\sigma,q) \in \hat{\hat{\delta}}\big\}$.

Observe that the number of states in $\mathcal{B}$ is at most 3 times the number of states in $\dot{\mathcal{B}}$, and thus less than 2^n. We will now prove that for every word u, $\mathcal{B}$ accepts u iff $\mathcal{B}'(u) = -\frac{1}{\lambda^{|u|}}$.

The first direction is easy: if $\mathcal{B}'(u) = -\frac{1}{\lambda^{|u|}}$, we get that all the transitions of a minimal run of $\mathcal{B}'$ on u are in $\hat{\delta}$, and its final transition is in $\hat{\hat{\delta}}$, hence there exists a run of $\mathcal{B}$ on u ending at an accepting state.

For the other direction, assume towards contradiction that there exists a word u, such that $\mathcal{B}'(u) = \frac{1}{\lambda^{|u|}}$, while there is an accepting run r_u of $\mathcal{B}$ on u.

Intuitively, we define the "normalized value" of a run r' of $\mathcal{B}'$ as the value of $\mathcal{B}'$ multiplied by the accumulated discount factor, i.e., $\mathcal{B}'(r') \cdot \lambda^{|r'|}$. Whenever the normalized value reaches -1, we have an "accepting" run. We will show that r_u and the structure of $\mathcal{B}$ imply the existence of two "accepting" runs $r'_1, r'_2 \in R^-$ that intersect in some state q, such that taking the prefix of r'_1 up to q results in a normalized value $\lambda^k W_1$ that is strictly smaller than the normalized value $\lambda^j W_2$ of the prefix of r'_2 up to q. Since r'_2 is an "accepting" run, the suffix of r'_2 reduces $\lambda^j W_2$ to -1 and therefore it will reduce $\lambda^k W_1$ to a value strictly smaller than -1, and the total value of the run to a value strictly smaller than $-\frac{1}{\lambda^n}$, which is not a possible value of $\mathcal{B}'$.

Formally, let $r_u(|u|-1) = \big(p', u(|u|-1), (q',1)\big)$ be the final transition of r_u. We replace it with the transition $t' = \big(p', u(|u|-1), q'\big)$. The resulting run $r'_u = r_u[0..|u|-2] \cdot t$ is a run of $\mathcal{B}'$ on u, and therefore $\mathcal{B}'(r'_u) \geq \frac{1}{\lambda^{|u|}}$. Since $(q',1)$ is an accepting state, we get by the construction of $\mathcal{B}$ that t' is in $\hat{\hat{\delta}}$. Consider a run $r'_1 \in R^-$ that shares the maximal suffix with r'_u, meaning that if there exist $r' \in R^-$ and $x > 0$ such that $r'[|r'|-x..|r'|-1] = r'_u[|u|-x..|u|-1]$ then also $r'_1[|r'_1|-x..|r'_1|-1] = r'_u[|u|-x..|u|-1]$.

Recall that all the initial states of $\mathcal{B}'$ have no transitions entering them and $\mathcal{B}'(r'_1) \neq \mathcal{B}'(r'_u)$, hence r'_1 is not a suffix of r'_u and r'_u is not a suffix of r'_1. Let i be the maximal index of r'_u such that $r'_u[i..|u|-1]$ is a suffix of r'_1, but $r'_u[i-1..|u|-1]$ is not a suffix of r'_1. Let k be the index in r'_1 such that $r'_1[k..|r'_1|-1] = r_u[i..|u|-1]$, and let $x = |r'_1| - k$ (see Figure 9).

Since $r'_u(i-1) \in \hat{\delta}$, there exists $r'_2 \in R^-$ and index j such that $r'_2(j-1) = r'_u(i-1)$. Let $y = |r'_2| - j$ (see Figure 9). Consider the run $r'_3 = r'_2[0..j-1] \cdot r'_u[i..|u|-1]$, starting with the prefix of r'_2 up to the shared transition with r'_u, and then continuing with the suffix of r'_u. Observe that $\mathcal{B}'(r'_3) > -\frac{1}{\lambda^{|r'_3|}}$ as otherwise $r'_3 \in R^-$ and has a larger suffix with r'_u than r'_1 has.

Let $W_1 = \mathcal{B}'\big(r'_1[0..k-1]\big)$, $W_2 = \mathcal{B}'\big(r'_2[0..j-1]\big)$, $X = \mathcal{B}'\big(r'_1[k..k+x-1]\big)$ (which is also $\mathcal{B}'\big(r'_u[i..|u|-1]\big)$), and $Y = \mathcal{B}'\big(r'_2[j..j+y-1]\big)$ (see Figure 9). The following must hold:

1. $W_1 + \frac{X}{\lambda^k} = \mathcal{B}'(r'_1) = -\frac{1}{\lambda^{k+x}}$. Hence, $\lambda^k W_1 = -\frac{1}{\lambda^x} - X$.
2. $W_2 + \frac{X}{\lambda^j} = \mathcal{B}'(r'_3) > -\frac{1}{\lambda^{j+x}}$. Hence, $\lambda^j W_2 > -\frac{1}{\lambda^x} - X$, and after combining with the previous equation, $\lambda^j W_2 > \lambda^k W_1$.
3. $W_2 + \frac{Y}{\lambda^j} = \mathcal{B}'(r'_2) = -\frac{1}{\lambda^{j+y}}$. Hence, $\lambda^j W_2 + Y = -\frac{1}{\lambda^y}$

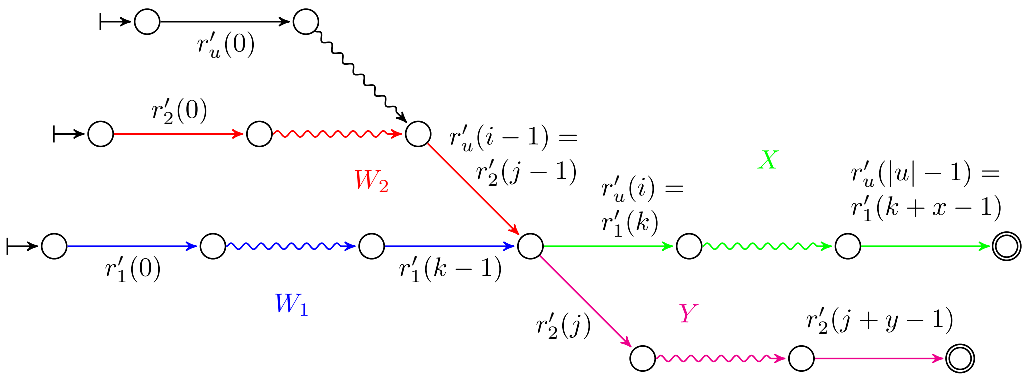

Figure 9 The runs and notations used in the proof of Theorem 13.

Consider now the run $r'_4 = r'_1[0..k-1] \cdot r'_2[j..j+y-1]$, and combine Items 2 and 3 above to get that $\lambda^k W_1 + Y < -\frac{1}{\lambda^y}$. But this leads to $\mathcal{B}'(r'_4) = W_1 + \frac{Y}{\lambda^k} < -\frac{1}{\lambda^{k+y}} = -\frac{1}{\lambda^{|r'_4|}}$, and this means that there exists a word w of length $k+y$ such that $\mathcal{B}'(w) < -\frac{1}{\lambda^{k+y}}$, contradicting the assumption that $\mathcal{B}' \equiv \dot{\mathcal{B}} \equiv -\tilde{\mathcal{A}}$.

Infinite words.

For showing the lower bound for the state blow-up involved in multiplying an NDA by (-1) w.r.t. infinite words, we add a new letter # to the alphabet, and correlate every finite word u to an infinite word $u \cdot \#^\omega$. The proof is similar, applying the following modifications:

- The scheme presented in the proof of Lemma 12 now constructs a λ-NDA $\tilde{\mathcal{A}}$ over the alphabet $\Sigma \cup \{\#\}$, adding a 0-weighted transition from every state of $\tilde{\mathcal{A}}$ to q_{hole}. The function f that correlates between the runs of $\mathcal{A}$ and $\tilde{\mathcal{A}}$ is still a bijection, but with a different co-domain, correlating every run r of $\mathcal{A}$ on a finite word $u \in \Sigma^+$ to the run $f(r)$ of $\tilde{\mathcal{A}}$ on the word $u \cdot \#^\omega$.
- With this scheme, we get that $\dot{\mathcal{B}}(u \cdot \#^\omega) = -\frac{1}{\lambda^{|u|}}$ if $u \notin L(A)$ and $\dot{\mathcal{B}}(u \cdot \#^\omega) = \frac{1}{\lambda^{|u|}}$ if $u \in L(A)$, hence replacing all referencing to $\mathcal{B}'(u)$ with referencing to $\mathcal{B}'(u \cdot \#^\omega)$.
- R^- is defined with respect to words of the form $u \cdot \#^\omega$, namely $R^- = \{r \mid u \in \Sigma^+, r$ is a minimal run of $\mathcal{B}'$ on $u \cdot \#^\omega$ and $\mathcal{B}'(r) < 0\}$.
- R^-_p is a new set of all the maximal (finite) prefixes of the runs of R^- without any transitions for the # letter, meaning $R^-_p = \{r[0..i-1] \mid r \in R^-, r(i-1) = (p,\sigma,q)$ for some $\sigma \in \Sigma$, and $r(i) = (q,\#,s)\}$. $\hat{\delta}$ and $\hat{\hat{\delta}}$ are defined with respect to R^-_p instead of R^-.
- Defining r'_u, we consider a run $r'_t \in R^-$ that is a witness for $t' \in \hat{\hat{\delta}}$, meaning there exists $i \in \mathbb{N}$ for which $r'_t(i) = t'$, and $r'_t(i+1)$ is a transition for the # letter. Then $r'_u = r_u[0..|u|-2] \cdot t \cdot r'[i+1..\infty] = r_u[0..|u|-2] \cdot r'[i..\infty]$, is a run of $\mathcal{B}'$ on $u \cdot \#^\omega$.
- For choosing r'_1 that "shares the maximal suffix" with r'_u, we take $r'_1 \in R^-$ such that for every $r' \in R^-$ and $x > 0$, if $r'_u[i..\infty]$ is a suffix of r' then it is also a suffix of r'_1.
- For the different runs and their parts, we set $X = \mathcal{B}'(r'_1[k..\infty])$, $Y = \mathcal{B}'(r'_2[j..\infty])$, $r'_3 = r'_2[0..j-1] \cdot r'_u[i..\infty]$ and $r'_4 = r'_1[0..k-1] \cdot r'_2[j..\infty]$.

◀

Proof of Theorem 24. PSPACE hardness directly follows from Lemmas 21 and 23.

We provide a PSPACE upper bound. Consider a choice function θ, and θ-NMDAs $\mathcal{A} = \langle \Sigma, Q_\mathcal{A}, \iota, \delta_\mathcal{A}, \gamma_\mathcal{A}, \rho_\mathcal{A} \rangle$ and $\mathcal{B}$. We have that

$$\forall w.\mathcal{A}(w) > \mathcal{B}(w) \Leftrightarrow \nexists w.\mathcal{A}(w) \le \mathcal{B}(w) \Leftrightarrow \nexists w.\mathcal{A}(w) - \mathcal{B}(w) \le 0$$

and

$$\forall w.\mathcal{A}(w) \geq \mathcal{B}(w) \Leftrightarrow \nexists w.\mathcal{A}(w) < \mathcal{B}(w) \Leftrightarrow \nexists w.\mathcal{A}(w) - \mathcal{B}(w) < 0$$

We present a nondeterministic algorithm that determines the converse of containment, namely whether there exists a word w such that $\mathcal{A}(w) - \mathcal{B}(w) \leq 0$ for continament($>$) or $\mathcal{A}(w) - \mathcal{B}(w) < 0$ for continament($\geq$), while using polynomial space w.r.t. $|\mathcal{A}|$ and $|\mathcal{B}|$, to conclude that the problems are in co-NPSPACE and hence in PSPACE.

Let $\mathcal{D} = \langle \Sigma, Q_{\mathcal{D}}, \{p_0\}, \delta_{\mathcal{D}}, \gamma_{\mathcal{D}}, \rho_{\mathcal{D}} \rangle$ be a θ-DMDA equivalent to $\mathcal{B}$, as per Theorem 8. Observe that the size of $\mathcal{D}$ can be exponential in the size of $\mathcal{B}$, but we do not save it all, but rather simulate it on the fly, and thus only save a single state of $\mathcal{D}$ at a time. We will later show that indeed the intermediate data we use in each iteration of the algorithm only requires a space polynomial in $|\mathcal{A}|$ and $|\mathcal{B}|$.

We consider separately non-strict containment ($\geq$) and strict containment ($>$).

Non-strict containment($\geq$).

For providing a word $w \in \Sigma^+$, such that $\mathcal{A}(w) - \mathcal{B}(w) < 0$, we nondeterministically generate on the fly a word w, a run r_w of $\mathcal{A}$ on w, and the single run of $\mathcal{D}$ on w, such that $\mathcal{A}(r_w) - \mathcal{B}(w) = \mathcal{A}(r_w) - \mathcal{D}(w) < 0$. Observe that $\mathcal{A}(w) \leq \mathcal{A}(r_w)$, hence the above condition is equivalent to $\mathcal{A}(w) - \mathcal{B}(w) < 0$.

Let $M_{\mathcal{A}}$, $M_{\mathcal{B}}$, and $M_{\mathcal{D}}$ be the maximal absolute weights in $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{D}$, respectively.

We start by guessing an initial state q_{in} of $\mathcal{A}$ and setting a *local data* storage of $\langle q_{in}, p_0, 0\rangle$. The local data will maintain the current state of $\mathcal{A}$ and $\mathcal{D}$ respectively, and a "normalized difference" between the value of the run in $\mathcal{A}$ generated so far and the value of $\mathcal{D}$ on the word generated so far, as formalized below. The algorithm iteratively guesses, given a local data $\langle q, p, d\rangle$, a letter $\sigma \in \Sigma$ and a transition $t = (q, \sigma, q') \in \delta_{\mathcal{A}}(q, \sigma)$, and calculates the *normalized difference* $d' = \rho_{\mathcal{A}}(t)\big(d + \gamma_{\mathcal{A}}(t) - \gamma_{\mathcal{D}}(p, \sigma)\big)$ between the values $\mathcal{A}(r_w)$ and $\mathcal{B}(w)$, w.r.r. the word w and the run r_w generated so far. If d' is bigger than the *maximal recoverable difference* $2S$, where $S = M_{\mathcal{A}} + 3M_{\mathcal{B}}$, we abort, if $d' < 0$, we have that the generated word w indeed witnesses that $\mathcal{A}(w) < \mathcal{D}(w)$ (the *accept condition* holds), and otherwise we continue and update the local data to $\langle q', \delta(p, \sigma), d'\rangle$. Observe that by the construction in the proof of Theorem 8, for every weight W in $\mathcal{D}$ we have that $|W| \leq 2T + M_{\mathcal{B}} \leq 3M_{\mathcal{B}}$, where T is the maximal difference between the weights in $\mathcal{B}$. Hence $S > M_{\mathcal{A}} + M_{\mathcal{D}}$ is polynomial w.r.t. $|\mathcal{A}|$ and $|\mathcal{B}|$, and can be calculated in polynomial space w.r.t. $|\mathcal{A}|$ and $|\mathcal{B}|$.

We show by induction on the length of the word w that whenever a word w and a run r_w are generated, the value d in the corresponding local data $\langle q, p, d\rangle$ indeed stands for the normalized difference between $\mathcal{A}(r_w)$ and $\mathcal{D}(w)$, namely

$$d = \rho_{\mathcal{A}}(r_w)\big(\mathcal{A}(r_w) - \mathcal{D}(w)\big) \tag{2}$$

For the base case we have a single-letter word $w = \sigma$, and a single-transition run $r_w = t$. Hence, $d' = \rho_{\mathcal{A}}(t)\big(d + \gamma_{\mathcal{A}}(t) - \gamma_{\mathcal{D}}(p, \sigma)\big) = \rho_{\mathcal{A}}(r_w)\big(0 + \mathcal{A}(r_w) - \mathcal{D}(w)\big) = \rho_{\mathcal{A}}(r_w)\big(\mathcal{A}(r_w) - \mathcal{D}(w)\big)$.

For the induction step, consider an iteration whose initial local data is $\langle q, p, d\rangle$, for a generated word w and run r_w, that guessed the next letter σ and transition t, and calculated the next local data $\langle q', p', d'\rangle$. Then we have $d' = \rho_{\mathcal{A}}(t)\big(d + \gamma_{\mathcal{A}}(t) - \gamma_{\mathcal{D}}(p, \sigma)\big)$. By the induction assumption, we get:

$$\begin{aligned} d' &= \rho_{\mathcal{A}}(t)\Big(\rho_{\mathcal{A}}(r_w)\big(\mathcal{A}(r_w) - \mathcal{D}(w)\big) + \gamma_{\mathcal{A}}(t) - \gamma_{\mathcal{D}}(p,\sigma)\Big) \\ &= \rho_{\mathcal{A}}(r_w)\rho_{\mathcal{A}}(t)\Big(\mathcal{A}(r_w) + \frac{\gamma_{\mathcal{A}}(t)}{\rho_{\mathcal{A}}(r_w)} - \mathcal{D}(w) - \frac{\gamma_{\mathcal{D}}(p,\sigma)}{\rho_{\mathcal{A}}(r_w)}\Big) \\ &= \rho_{\mathcal{A}}(r_w \cdot t)\Big(\mathcal{A}(r_w \cdot t) - \Big(\mathcal{D}(w) + \frac{\gamma_{\mathcal{D}}(p,\sigma)}{\rho_{\mathcal{A}}(r_w)}\Big)\Big), \end{aligned}$$

and since the discount-factor functions of $\mathcal{A}$ and $\mathcal{D}$ both agree with θ, we have

$$d' = \rho_{\mathcal{A}}(r_w \cdot t)\Big(\mathcal{A}(r_w \cdot t) - \Big(\mathcal{D}(w) + \frac{\gamma_{\mathcal{D}}(p,\sigma)}{\rho_{\mathcal{D}}(w)}\Big)\Big) = \rho_{\mathcal{A}}(r_w \cdot t)\big(\mathcal{A}(r_w \cdot t) - \mathcal{D}(w \cdot \sigma)\big),$$

which provides the required result of the induction claim.

Next, we show that the accept condition holds iff there exist a finite word w and run r_w of $\mathcal{A}$ on w such that $\mathcal{A}(r_w) - \mathcal{D}(w) < 0$. Since for every finite word w we have $\rho_{\mathcal{A}}(w) > 0$, we conclude from Equation (2) that if $d' < 0$ was reached for a generated word w and a run r_w, we have that $\mathcal{A}(r_w) - \mathcal{D}(w) < 0$. For the other direction, assume toward contradiction that there exist finite word w and run r_w of $\mathcal{A}$ on w such that $\mathcal{A}(r_w) - \mathcal{D}(w) < 0$, but the algorithm aborts after generating some prefixes $w[0..i]$ and $r_w[0..i]$. Meaning that $\rho_{\mathcal{A}}(r_w[0..i])\big(\mathcal{A}(r_w[0..i]) - \mathcal{D}(w[0..i])\big) > 2M_{\mathcal{A}} + 2M_{\mathcal{D}}$. Let $W_1 = \mathcal{A}(r_w[i+1..|r_w|-1])$ and $W_2 = \mathcal{D}^{\delta_{\mathcal{D}}(w[0..i])}(w[i+1..|r_w|-1])$. Observe that

$$\begin{aligned} 0 &> \mathcal{A}(r_w) - \mathcal{D}(w) > \rho_{\mathcal{A}}\big(r_w[0..i]\big)\big(\mathcal{A}(r_w) - \mathcal{D}(w)\big) \\ &= \rho_{\mathcal{A}}\big(r_w[0..i]\big)\mathcal{A}\big(r_w[0..i]\big) + W_1 - \big(\rho_{\mathcal{A}}(r_w[0..i])\mathcal{D}(w[0..i]) + W2\big) \\ &> 2M_{\mathcal{A}} + 2M_{\mathcal{D}} + W_1 - W_2 \end{aligned}$$

But since all the discount factors applied by θ are greater or equal to 2, we have that $|W_1| \leq 2M_{\mathcal{A}}$ and $|W_2| \leq 2M_{\mathcal{B}}$, leading to a contradiction.

To see that the algorithm indeed only uses space polynomial in $|\mathcal{A}|$ and $|\mathcal{B}|$, observe that the first element of the data storage is a state of $\mathcal{A}$, only requiring a space logarithmic in $|\mathcal{A}|$, the second element is a state of $\mathcal{D}$, requiring by Theorem 8 a space polynomial in $\mathcal{B}$, and the third element is a non-negative rational number bounded by $2S$, whose denominator is the multiplication of the denominators of the weights in $\mathcal{A}$ and $\mathcal{D}$, and as shown in the proof of Theorem 8, also of the multiplication of the denominators of the weights in $\mathcal{A}$ and $\mathcal{B}$, thus requires a space polynomial in $|\mathcal{A}|$ and $|\mathcal{B}|$. Finally, in order to compute this third element, we calculated a weight of a transition in $\mathcal{D}$, which only requires, by the proof of Theorem 8, a space polynomial in $|\mathcal{B}|$.

Strict Containment(>).

The algorithm is identical to the one used for the containment($\geq$) problem with changing the accept condition $d' < 0$ to $d' \leq 0$. This condition is met iff there exists a finite word w such that $\mathcal{A}(w) - \mathcal{B}(w) \leq 0$. The proof is identical while modifying "< 0" to "≤ 0" in all of the equations. ◀