

On the Comparison of Discounted-Sum Automata with Multiple Discount Factors [★]

Udi Boker^{**[0000-0003-4322-8892]} and Guy Hefetz^[0000-0002-4451-6581]

Reichman University, Herzliya, Israel
udiboker@runi.ac.il, ghefetz@gmail.com

Abstract. We look into the problems of comparing nondeterministic discounted-sum automata on finite and infinite words. That is, the problems of checking for automata \mathcal{A} and \mathcal{B} whether or not it holds that for all words w , $\mathcal{A}(w) = \mathcal{B}(w)$, $\mathcal{A}(w) \leq \mathcal{B}(w)$, or $\mathcal{A}(w) < \mathcal{B}(w)$.

These problems are known to be decidable when both automata have the same single integral discount factor, while decidability is open in all other settings: when the single discount factor is a non-integral rational; when each automaton can have multiple discount factors; and even when each has a single integral discount factor, but the two are different.

We show that it is undecidable to compare discounted-sum automata with multiple discount factors, even if all are integrals, while it is decidable to compare them if each has a single, possibly different, integral discount factor. To this end, we also provide algorithms to check for given nondeterministic automaton \mathcal{N} and deterministic automaton \mathcal{D} , each with a single, possibly different, rational discount factor, whether or not $\mathcal{N}(w) = \mathcal{D}(w)$, $\mathcal{N}(w) \geq \mathcal{D}(w)$, or $\mathcal{N}(w) > \mathcal{D}(w)$ for all words w .

Keywords: Discounted-sum Automata · Comparison · Containment.

1 Introduction

Equivalence and containment checks of Boolean automata, namely the checks of whether $L(\mathcal{A}) = L(\mathcal{B})$, $L(\mathcal{A}) \subseteq L(\mathcal{B})$, or $L(\mathcal{A}) \subset L(\mathcal{B})$, where $L(\mathcal{A})$ and $L(\mathcal{B})$ are the languages that \mathcal{A} and \mathcal{B} recognize, are central in the usage of automata theory in diverse areas, and in particular in formal verification (e.g, [33,25,16,32,34,27]). Likewise, comparison of quantitative automata, which extends the equivalence and containment checks by asking whether $\mathcal{A}(w) = \mathcal{B}(w)$, whether $\mathcal{A}(w) \leq \mathcal{B}(w)$, or whether $\mathcal{A}(w) < \mathcal{B}(w)$ for all words w , are essential for harnessing quantitative-automata theory to the service of diverse fields and in particular to the service of quantitative formal verification (e.g, [14,13,20,10,26,3,5,21]).

Discounted summation is a common valuation function in quantitative automata theory (e.g, [18,11,13,14]), as well as in various other computational models, such as games (e.g., [36,4,1]), Markov decision processes (e.g, [22,28,15]), and

[★] This is the full version of a chapter with the same title that appears in the FoSSaCS 2023 conference proceedings.

^{**} Research supported by the Israel Science Foundation grant 2410/22.

reinforcement learning (e.g. [31,35]), 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) has 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 i th transition of the run is divided by λ^i . The value of a (finite or infinite) word is the infimum value of the automaton runs on it. An NDA thus realizes a function from words to real numbers.

NDA's cannot always be determinized [14], they are not closed under basic algebraic operations [7], and their comparison is not known to be decidable, relating to various longstanding open problems [8]. However, restricting NDA's to have an integral discount factor $\lambda \in \mathbb{N} \setminus \{0, 1\}$ provides a robust class of automata that is closed under determinization and under algebraic operations, and for which comparison is decidable [7].

Various variants of NDA's are studied in the literature, among which are *functional*, *k-valued*, *probabilistic*, and more [20,19,12]. Yet, until recently, all of these models were restricted to have a single discount factor. 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 multiple discount factors within the same entity [22,4]. In [6], NDA's were extended to NMDA's, allowing for multiple discount factors, where each transition can have a different one. Special attention was given to integral NMDA's, namely to those with only integral discount factors, analyzing whether they preserve the good properties of integral NDA's. It was shown that they are generally not closed under determinization and under algebraic operations, while a restricted class of them, named tidy-NMDA's, in which the choice of discount factors depends on the prefix of the word read so far, does preserve the good properties of integral NDA's.

While comparison of tidy-NMDA's with the same choice function is decidable in PSPACE [6], it was left open whether comparison of general integral NMDA's \mathcal{A} and \mathcal{B} is decidable. It is even open whether comparison of two integral NDA's with different (single) discount factors is decidable.

We show that it is undecidable to resolve for given NMDA \mathcal{N} and deterministic NMDA (DMDA) \mathcal{D} , even if both have only integral discount factors, on both finite and infinite words, whether $\mathcal{N} \equiv \mathcal{D}$ and whether $\mathcal{N} \leq \mathcal{D}$, and on finite words also whether $\mathcal{N} < \mathcal{D}$. We prove the undecidability result by reduction from the halting problem of two-counter machines. The general scheme follows similar reductions, such as in [17,2], yet the crux is in simulating a counter by integral NMDA's. Upfront, discounted summation is not suitable for simulating counters, since a current increment has, in the discounted setting, a much higher influence than of a far-away decrement. However, we show that multiple discount factors allow in a sense to eliminate the influence of time, having automata in which no matter where a letter appears in the word, it will have the same influence

on the automaton value. (See Lemma 1 and Fig. 3). Another main part of the proof is in showing how to nondeterministically adjust the automaton weights and discount factors in order to “detect” whether a counter is at a current value 0. (See Figs. 5, 6, 8 and 9.)

On the positive side, we provide algorithms to decide for given NDA \mathcal{N} and deterministic NDA (DDA) \mathcal{D} , with arbitrary, possibly different, rational discount factors, whether $\mathcal{N} \equiv \mathcal{D}$, $\mathcal{N} \geq \mathcal{D}$, or $\mathcal{N} > \mathcal{D}$ (Theorem 4). Our algorithms work on both finite and infinite words, and run in PSPACE when the automata weights are represented in binary and their discount factors in unary. Since integral NDAs can always be determinized [7], our method also provides an algorithm to compare two integral NDAs, though not necessarily in PSPACE, since determinization might exponentially increase the number of states. (Even though determinization of NDAs is in PSPACE [7,6], the exponential number of states might require an exponential space in our algorithms of comparing NDAs with different discount factors.)

The challenge with comparing automata with different discount factors comes from the combination of their different accumulations, which tends to be intractable, resulting in the undecidability of comparing integral NMDAs, and in the open problems of comparing rational NDAs and of analyzing the representation of numbers in a non-integral basis [29,23,24,8]. Yet, the main observation underlying our algorithm is that when each automaton has a single discount factor, we may unfold the combination of their computation trees only up to some level k , after which we can analyze their continuation separately, first handling the automaton with the lower (slower decreasing) discount factor and then the other one. The idea is that after level k , since the accumulated discounting of the second automaton is already much more significant, even a single non-optimal transition of the first automaton cannot be compensated by a continuation that is better with respect to the second automaton. We thus compute the optimal suffix words and runs of the first automaton from level k , on top which we compute the optimal runs of the second automaton.

2 Preliminaries

Words. 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 $w = \sigma_0\sigma_1\sigma_2 \cdots$ and indexes $i \leq j$, we denote the *letter at index i* as $w[i] = \sigma_i$, and the *sub-word from i to j* as $w[i..j] = \sigma_i\sigma_{i+1} \cdots \sigma_j$.

For a finite word w and letter $\sigma \in \Sigma$, we denote the number of occurrences of σ in w by $\#(\sigma, w)$, and for a set $S \subseteq \Sigma$, we denote $\sum_{\sigma \in S} \#(\sigma, w)$ by $\#(S, w)$.

For a finite or infinite word w and a letter $\sigma \in \Sigma$, we define the *prefix of w up to σ* , $\text{PREFIX}_\sigma(w)$, as the minimal prefix of w that contains a σ letter if

there is a σ letter in w or w itself if it does not contain any σ letters. Formally,

$$\text{PREF}_\sigma(w) = \begin{cases} w[0..\min\{i \mid w[i] = \sigma\}] & \exists i \mid w[i] = \sigma \\ w & \text{otherwise} \end{cases}$$

Automata. A nondeterministic discounted-sum automaton (NDA) [14] is an automaton with rational weights on the transitions, and a fixed rational discount factor $\lambda > 1$. A nondeterministic discounted-sum automaton with multiple discount factors (NMDA) [6] is similar to an NDA, but with possibly a different discount factor on each of its transitions. They are formally defined as follows:

Definition 1 ([6]). *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 $\gamma : \delta \rightarrow \mathbb{Q}$, and a discount-factor function $\rho : \delta \rightarrow \mathbb{Q} \cap (1, \infty)$, assigning to each transition its discount factor, which is a rational greater than one.*¹

- A run of \mathcal{A} is a sequence of states and alphabet letters, $p_0, \sigma_0, p_1, \sigma_1, p_2, \dots$, such that $p_0 \in \iota$ is an initial state, and for every i , $(p_i, \sigma_i, p_{i+1}) \in \delta$.
- The length of a run r , denoted by $|r|$, is n for a finite run $r = p_0, \sigma_0, p_1, \dots, \sigma_{n-1}, p_n$, and ∞ for an infinite run.
- For an index $i < |r|$, we define the i -th transition of r as $r[i] = (p_i, \sigma_i, p_{i+1})$, and the prefix run with i transitions as $r[0..i] = p_0, \sigma_0, p_1, \dots, \sigma_i, p_{i+1}$.
- The value of a finite/infinite run r is $\mathcal{A}(r) = \sum_{i=0}^{|r|-1} \left(\gamma(r[i]) \cdot \prod_{j=0}^{i-1} \frac{1}{\rho(r[j])} \right)$.
For example, the value of the run $r_1 = q_0, a, q_0, a, q_1, b, q_2$ of \mathcal{A} from Fig. 1 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 $\mathcal{A}(w) = \inf\{\mathcal{A}(r) \mid r \text{ is a run of } \mathcal{A} \text{ on } w\}$.
- For every finite run $r = p_0, \sigma_0, p_1, \dots, \sigma_{n-1}, p_n$, we define the target state as $\delta(r) = p_n$ and the accumulated discount factor as $\rho(r) = \prod_{i=0}^{n-1} \rho(r[i])$.
- When all discount factors are integers, we say that \mathcal{A} is an integral NMDA.
- 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 DMMA, and view δ as a function from words to states.
- When the discount factor function ρ is constant, $\rho \equiv \lambda \in \mathbb{Q} \cap (1, \infty)$, we say that \mathcal{A} is a nondeterministic discounted-sum automaton (NDA) [14] with discount factor λ (a λ -NDA). If \mathcal{A} is deterministic, it is a λ -DDA.
- For a state $q \in Q$, we write \mathcal{A}^q for the NMDA $\mathcal{A}^q = \langle \Sigma, Q, \{q\}, \delta, \gamma, \rho \rangle$.

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

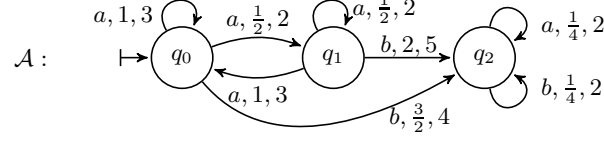


Fig. 1. An NMDA \mathcal{A} . The labeling on the transitions indicate the alphabet letter, the weight of the transition, and its discount factor.

Counter machines. A two-counter machine [30] \mathcal{M} is a sequence (l_1, \dots, l_n) of commands, for some $n \in \mathbb{N}$, involving two counters x and y . We refer to $\{1, \dots, n\}$ as the *locations* of the machine. For every $i \in \{1, \dots, n\}$ we refer to l_i as the *command in location i* . There are five possible forms of commands:

$$\text{INC}(c), \text{DEC}(c), \text{GOTO } l_k, \text{ IF } c=0 \text{ GOTO } l_k \text{ ELSE GOTO } l_{k'}, \text{ HALT},$$

where $c \in \{x, y\}$ is a counter and $1 \leq k, k' \leq n$ are locations. For not decreasing a zero-valued counter $c \in \{x, y\}$, every $\text{DEC}(c)$ command is preceded by the command $\text{IF } c=0 \text{ GOTO } \langle \text{CURRENT_LINE} \rangle \text{ ELSE GOTO } \langle \text{NEXT_LINE} \rangle$, and there are no other direct goto-commands to it. The counters are initially set to 0. An example of a two-counter machine is given in Fig. 2.

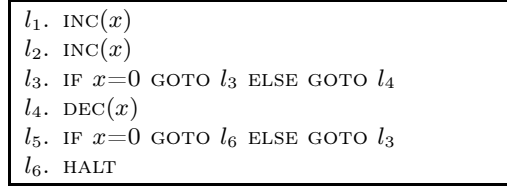


Fig. 2. An example of a two-counter machine.

Let L be the set of possible commands in \mathcal{M} , then a *run* of \mathcal{M} is a sequence $\psi = \psi_1, \dots, \psi_m \in (L \times \mathbb{N} \times \mathbb{N})^*$ such that the following hold:

1. $\psi_1 = \langle l_1, 0, 0 \rangle$.
2. For all $1 < i \leq m$, let $\psi_{i-1} = (l_j, \alpha_x, \alpha_y)$ and $\psi_i = (l', \alpha'_x, \alpha'_y)$. Then, the following hold.
 - If l_j is an INC(x) command (resp. INC(y)), then $\alpha'_x = \alpha_x + 1$, $\alpha'_y = \alpha_y$ (resp. $\alpha_y = \alpha_y + 1$, $\alpha'_x = \alpha_x$), and $l' = l_{j+1}$.
 - If l_j is DEC(x) (resp. DEC(y)) then $\alpha'_x = \alpha_x - 1$, $\alpha'_y = \alpha_y$ (resp. $\alpha_y = \alpha_y - 1$, $\alpha'_x = \alpha_x$), and $l' = l_{j+1}$.
 - If l_j is GOTO l_k then $\alpha'_x = \alpha_x$, $\alpha'_y = \alpha_y$, and $l' = l_k$.
 - If l_j is IF $x=0$ GOTO l_k ELSE GOTO $l_{k'}$ then $\alpha'_x = \alpha_x$, $\alpha'_y = \alpha_y$, and $l' = l_k$ if $\alpha_x = 0$, and $l' = l_{k'}$ otherwise.
 - If l_j is IF $y=0$ GOTO l_k ELSE GOTO $l_{k'}$ then $\alpha'_x = \alpha_x$, $\alpha'_y = \alpha_y$, and $l' = l_k$ if $\alpha_y = 0$, and $l' = l_{k'}$ otherwise.
 - If l' is HALT then $i = m$, namely a run does not continue after HALT.

If, in addition, we have that $\psi_m = \langle l_j, \alpha_x, \alpha_y \rangle$ such that l_j is a HALT command, we say that ψ is a *halting run*. We say that a machine \mathcal{M} 0-halts if its run is halting and ends in $\langle l, 0, 0 \rangle$. We say that a sequence of commands $\tau \in L^*$ fits a run ψ , if τ is the projection of ψ on its first component.

The *command trace* $\pi = \sigma_1, \dots, \sigma_m$ of a halting run $\psi = \psi_1, \dots, \psi_m$ describes the flow of the run, including a description of whether a counter c was equal to 0 or larger than 0 in each occurrence of an IF $c=0$ GOTO l_k ELSE GOTO $l_{k'}$ command. It is formally defined as follows. $\sigma_m = \text{HALT}$ and for every $1 < i \leq m$, we define σ_{i-1} according to $\psi_{i-1} = \langle l_j, \alpha_x, \alpha_y \rangle$ in the following manner:

- $\sigma_{i-1} = l_j$ if l_j is not of the form IF $c=0$ GOTO l_k ELSE GOTO $l_{k'}$.
- $\sigma_{i-1} = (\text{GOTO } l_k, c = 0)$ for $c \in \{x, y\}$, if $\alpha_c = 0$ and the command l_j is of the form IF $c=0$ GOTO l_k ELSE GOTO $l_{k'}$.
- $\sigma_{i-1} = (\text{GOTO } l_{k'}, c > 0)$ for $c \in \{x, y\}$, if $\alpha_c > 0$ and the command l_j is of the form IF $c=0$ GOTO l_k ELSE GOTO $l_{k'}$.

For example, the command trace of the halting run of the machine in Fig. 2 is $\text{INC}(x), \text{INC}(x), (\text{GOTO } l_4, x > 0), \text{DEC}(x), (\text{GOTO } l_3, x > 0), (\text{GOTO } l_4, x > 0), \text{DEC}(x), (\text{GOTO } l_6, x = 0), \text{HALT}$.

Deciding whether a given counter machine \mathcal{M} halts is known to be undecidable [30]. Deciding whether \mathcal{M} halts with both counters having value 0, termed the *0-halting problem*, is also undecidable. Indeed, the halting problem can be reduced to the latter by adding some commands that clear the counters, before every HALT command.

3 Comparison of NMDAs

We show that comparison of (integral) NMDAs is undecidable by reduction from the halting problem of two-counter machines. Notice that our NMDAs only use integral discount factors, while they do have non-integral weights. Yet, weights can be easily changed to integers as well, by multiplying them all by a common denominator and making the corresponding adjustments in the calculations.

We start with a lemma on the accumulated value of certain series of discount factors and weights. Observe that by the lemma, no matter where the pair of discount-factor $\lambda \in \mathbb{N} \setminus \{0, 1\}$ and weight $w = \frac{\lambda-1}{\lambda}$ appear along the run, they will have the same effect on the accumulated value. This property will play a key role in simulating counting by NMDAs.

Lemma 1. *For every sequence $\lambda_1, \dots, \lambda_m$ of integers larger than 1 and weights w_1, \dots, w_m such that $w_i = \frac{\lambda_i-1}{\lambda_i}$, we have $\sum_{i=1}^m (w_i \cdot \prod_{j=1}^{i-1} \frac{1}{\lambda_j}) = 1 - \frac{1}{\prod_{j=1}^m \lambda_j}$.*

Proof. We show the claim by induction on m .

The base case, i.e. $m = 1$, is trivial. For the induction step we have

$$\begin{aligned}
 \sum_{i=1}^{m+1} \left(w_i \cdot \prod_{j=1}^{i-1} \frac{1}{\lambda_j} \right) &= \sum_{i=1}^m \left(w_i \cdot \prod_{j=1}^{i-1} \frac{1}{\lambda_j} \right) + w_{m+1} \cdot \prod_{j=1}^m \frac{1}{\lambda_j} \\
 &= 1 - \frac{1}{\prod_{j=1}^m \lambda_j} + \frac{\lambda_{m+1} - 1}{\lambda_{m+1}} \cdot \prod_{j=1}^m \frac{1}{\lambda_j} \\
 &= 1 - \frac{\lambda_{m+1}}{\prod_{j=1}^{m+1} \lambda_j} + \frac{\lambda_{m+1} - 1}{\prod_{j=1}^{m+1} \lambda_j} = 1 - \frac{1}{\prod_{j=1}^{m+1} \lambda_j}
 \end{aligned}$$

□

3.1 The Reduction

We turn to our reduction from the halting problem of two-counter machines to the problem of NMDA containment. We provide the construction and the correctness lemma with respect to automata on finite words, and then show in Section 3.2 how to use the same construction also for automata on infinite words.

Given a two-counter machine \mathcal{M} with the commands (l_1, \dots, l_n) , we construct an integral DMDA \mathcal{A} and an integral NMDA \mathcal{B} on finite words, such that \mathcal{M} 0-halts iff there exists a word $w \in \Sigma^+$ such that $\mathcal{B}(w) \geq \mathcal{A}(w)$ iff there exists a word $w \in \Sigma^+$ such that $\mathcal{B}(w) > \mathcal{A}(w)$.

The automata \mathcal{A} and \mathcal{B} operate over the following alphabet Σ , which consists of $5n + 5$ letters, standing for the possible elements in a command trace of \mathcal{M} :

$$\begin{aligned}
 \Sigma^{\text{INCDEC}} &= \{ \text{INC}(x), \text{DEC}(x), \text{INC}(y), \text{DEC}(y) \} \\
 \Sigma^{\text{GOTO}} &= \{ \text{GOTO } l_k : k \in \{1, \dots, n\} \} \cup \\
 &\quad \{ (\text{GOTO } l_k, c = 0) : k \in \{1, \dots, n\}, c \in \{x, y\} \} \cup \\
 &\quad \{ (\text{GOTO } l_{k'}, c > 0) : k' \in \{1, \dots, n\}, c \in \{x, y\} \} \\
 \Sigma^{\text{NOHALT}} &= \Sigma^{\text{INCDEC}} \cup \Sigma^{\text{GOTO}} \\
 \Sigma &= \Sigma^{\text{NOHALT}} \cup \{ \text{HALT} \}
 \end{aligned}$$

When \mathcal{A} and \mathcal{B} read a word $w \in \Sigma^+$, they intuitively simulate a sequence of commands τ_u that induces the command trace $u = \text{PREF}_{\text{HALT}}(w)$. If τ_u fits the actual run of \mathcal{M} , and this run 0-halts, then the minimal run of \mathcal{B} on w has a value strictly larger than $\mathcal{A}(w)$. If, however, τ_u does not fit the actual run of \mathcal{M} , or it does fit the actual run but it does not 0-halt, then the violation is detected by \mathcal{B} , which has a run on w with value strictly smaller than $\mathcal{A}(w)$.

In the construction, we use the following partial discount-factor functions $\rho_p, \rho_d : \Sigma^{\text{NOHALT}} \rightarrow \mathbb{N}$ and partial weight functions $\gamma_p, \gamma_d : \Sigma^{\text{NOHALT}} \rightarrow \mathbb{Q}$.

$$\rho_p(\sigma) = \begin{cases} 5 & \sigma = \text{INC}(x) \\ 4 & \sigma = \text{DEC}(x) \\ 7 & \sigma = \text{INC}(y) \\ 6 & \sigma = \text{DEC}(y) \\ 15 & \text{otherwise} \end{cases} \quad \rho_d(\sigma) = \begin{cases} 4 & \sigma = \text{INC}(x) \\ 5 & \sigma = \text{DEC}(x) \\ 6 & \sigma = \text{INC}(y) \\ 7 & \sigma = \text{DEC}(y) \\ 15 & \text{otherwise} \end{cases}$$

$\gamma_p(\sigma) = \frac{\rho_p(\sigma)-1}{\rho_p(\sigma)}$, and $\gamma_d(\sigma) = \frac{\rho_d(\sigma)-1}{\rho_d(\sigma)}$. We say that ρ_p and γ_p are the *primal* discount-factor and weight functions, while ρ_d and γ_d are the *dual* functions. Observe that for every $c \in \{x, y\}$ we have that

$$\rho_p(\text{INC}(c)) = \rho_d(\text{DEC}(c)) > \rho_p(\text{DEC}(c)) = \rho_d(\text{INC}(c)) \quad (1)$$

Intuitively, we will use the primal functions for \mathcal{A} 's discount factors and weights, and the dual functions for identifying violations. Notice that if changing the primal functions to the dual ones in more occurrences of $\text{INC}(c)$ letters than of $\text{DEC}(c)$ letters along some run, then by Lemma 1 the run will get a value lower than the original one.

We continue with their formal definitions. $\mathcal{A} = \langle \Sigma, \{q_{\mathcal{A}}, q_{\mathcal{A}}^h\}, \{q_{\mathcal{A}}\}, \delta_{\mathcal{A}}, \gamma_{\mathcal{A}}, \rho_{\mathcal{A}} \rangle$ is an integral DMDA consisting of two states, as depicted in Fig. 3. Observe that the initial state $q_{\mathcal{A}}$ has self loops for every alphabet letter in Σ^{NOHALT} with weights and discount factors according to the primal functions, and a transition $(q_{\mathcal{A}}, \text{HALT}, q_{\mathcal{A}}^h)$ with weight of $\frac{14}{15}$ and a discount factor of 15.

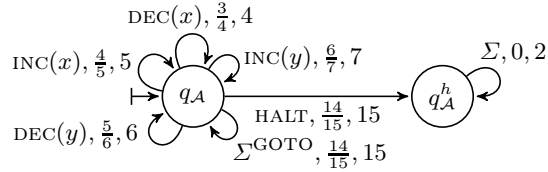


Fig. 3. The DMDA \mathcal{A} constructed for the proof of Lemma 2.

The integral NMDA $\mathcal{B} = \langle \Sigma, Q_{\mathcal{B}}, \iota_{\mathcal{B}}, \delta_{\mathcal{B}}, \gamma_{\mathcal{B}}, \rho_{\mathcal{B}} \rangle$ is the union of the following eight gadgets (checkers), each responsible for checking a certain type of violation in the description of a 0-halting run of \mathcal{M} . It also has the states $q_{\text{freeze}}, q_{\text{halt}} \in Q_{\mathcal{B}}$ such that for all $\sigma \in \Sigma$, there are 0-weighted transitions $(q_{\text{freeze}}, \sigma, q_{\text{freeze}}) \in \delta_{\mathcal{B}}$ and $(q_{\text{halt}}, \sigma, q_{\text{halt}}) \in \delta_{\mathcal{B}}$ with an arbitrary discount factor. Observe that in all of \mathcal{B} 's gadgets, the transition over the letter HALT to q_{halt} has a weight higher than the weight of the corresponding transition in \mathcal{A} , so that when no violation is detected, the value of \mathcal{B} on a word is higher than the value of \mathcal{A} on it.

1. Halt Checker. This gadget, depicted in Fig. 4, checks for violations of non-halting runs. Observe that its initial state q_{HC} has self loops identical to those

of \mathcal{A} 's initial state, a transition to q_{halt} over HALT with a weight higher than the corresponding weight in \mathcal{A} , and a transition to the state q_{last} over every letter that is not HALT, “guessing” that the run ends without a HALT command.

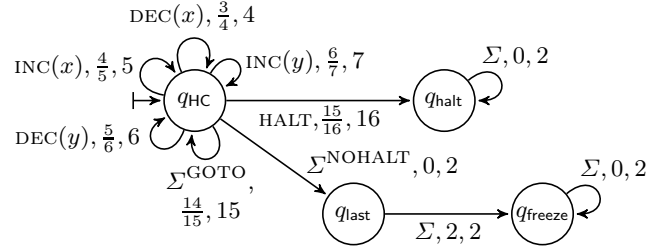


Fig. 4. The Halt Checker in the NMDA \mathcal{B} .

2. Negative-Counters Checker. The second gadget, depicted in Fig. 5, checks that the input prefix u has no more DEC(c) than INC(c) commands for each counter $c \in \{x, y\}$. It is similar to \mathcal{A} , however having self loops in its initial states that favor DEC(c) commands when compared to \mathcal{A} .

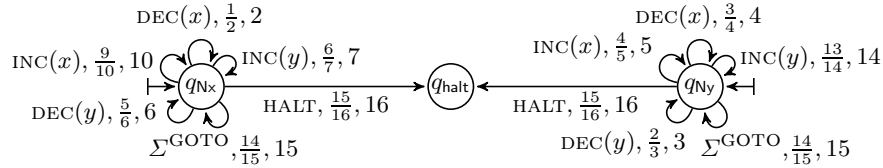


Fig. 5. The negative-counters checker, on the left for x and on the right for y , in the NMDA \mathcal{B} .

3. Positive-Counters Checker. The third gadget, depicted in Fig. 6, checks that for every $c \in \{x, y\}$, the input prefix u has no more INC(c) than DEC(c) commands. It is similar to \mathcal{A} , while having self loops in its initial state according to the dual functions rather than the primal ones.

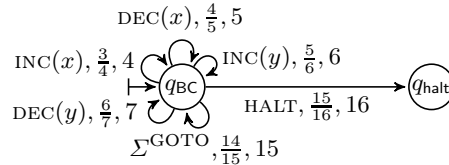


Fig. 6. The Positive-Counters Checker in the NMDA \mathcal{B} .

4. Command Checker. The next gadget checks for local violations of successive commands. That is, it makes sure that the letter w_i represents a command

that can follow the command represented by w_{i-1} in \mathcal{M} , ignoring the counter values. For example, if the command in location l_2 is $\text{INC}(x)$, then from state q_2 , which is associated with l_2 , we move with the letter $\text{INC}(x)$ to q_3 , which is associated with l_3 . The test is local, as this gadget does not check for violations involving illegal jumps due to the values of the counters. An example of the command checker for the counter machine in Fig. 2 is given in Fig. 7.

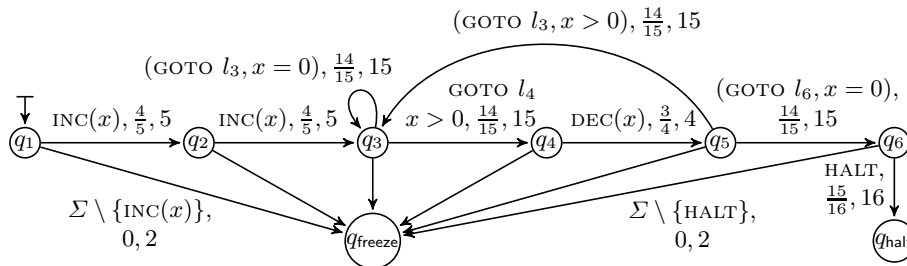


Fig. 7. The command checker that corresponds to the counter machine in Fig. 2.

The command checker, which is a DMDA, consists of states q_1, \dots, q_n that correspond to the commands l_1, \dots, l_n , and the states q_{halt} and q_{freeze} . For two locations j and k , there is a transition from q_j to q_k on the letter σ iff l_k can *locally follow* l_j in a run of \mathcal{M} that has σ in the corresponding location of the command trace. That is, either l_j is a $\text{GOTO } l_k$ command (meaning $l_j = \sigma = \text{GOTO } l_k$), k is the next location after j and l_j is an INC or a DEC command (meaning $k = j + 1$ and $l_j = \sigma \in \Sigma^{\text{INCDEC}}$), l_j is an $\text{IF } c=0 \text{ GOTO } l_k \text{ ELSE GOTO } l_{k'}$ command with $\sigma = (\text{GOTO } l_k, c = 0)$, or l_j is an $\text{IF } c=0 \text{ GOTO } l_s \text{ ELSE GOTO } l_k$ command with $\sigma = (\text{GOTO } l_k, c > 0)$. The weights and discount factors of the Σ^{NOHALT} transitions mentioned above are according to the primal functions γ_p and ρ_p respectively. For every location j such that $l_j = \text{HALT}$, there is a transition from q_j to q_{halt} labeled by the letter HALT with a weight of $\frac{15}{16}$ and a discount factor of 16. Every other transition that was not specified above leads to q_{freeze} with weight 0 and some discount factor.

5,6. Zero-Jump Checkers. The next gadgets, depicted in Fig. 8, check for violations in conditional jumps. In this case, we use a different checker instance for each counter $c \in \{x, y\}$, ensuring that for every $\text{IF } c=0 \text{ GOTO } l_k \text{ ELSE GOTO } l_{k'}$ command, if the jump $\text{GOTO } l_k$ is taken, then the value of c is indeed 0.

Intuitively, q_{zc}^c profits from words that have more $\text{INC}(c)$ than $\text{DEC}(c)$ letters, while q_c continues like \mathcal{A} . If the move to q_c occurred after a balanced number of $\text{INC}(c)$ and $\text{DEC}(c)$, as it should be in a real command trace, neither the prefix word before the move to q_c , nor the suffix word after it result in a profit. Otherwise, provided that the counter is 0 at the end of the run (as guaranteed by the negative- and positive-counters checkers), both prefix and suffix words get profits, resulting in a smaller value for the run.

7,8. Positive-Jump Checkers. These gadgets, depicted in Fig. 9, are dual to the zero-jump checkers, checking for the dual violations in conditional jumps.

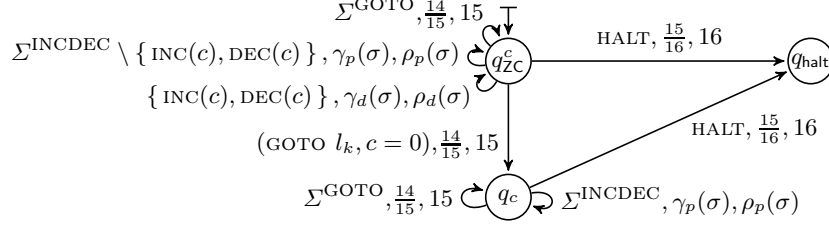


Fig. 8. The Zero-Jump Checker (for a counter $c \in \{x, y\}$) in the NMDA \mathcal{B} .

Similarly to the zero-jump checkers, we have a different instance for each counter $c \in \{x, y\}$, ensuring that for every IF $c=0$ GOTO l_k ELSE GOTO $l_{k'}$ command, if the jump GOTO $l_{k'}$ is taken, then the value of c is indeed greater than 0.

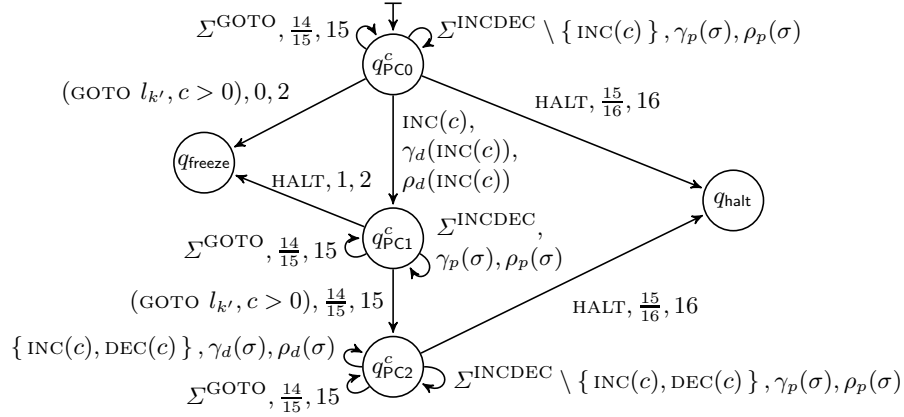


Fig. 9. The Positive-Jump Checker (for a counter c) in the NMDA \mathcal{B} .

Intuitively, if the counter is 0 on a (GOTO $l_{k'}, c > 0$) command when there was no INC(c) command yet, the gadget benefits by moving from q_{PC0}^c to q_{freeze} . If there was an INC(c) command, it benefits by having the dual functions on the move from q_{PC0}^c to q_{PC1}^c over INC(c) and the primal functions on one additional self loop of q_{PC1}^c over DEC(c).

Lemma 2. *Given a two-counter machine \mathcal{M} , we can compute an integral DMDA \mathcal{A} and an integral NMDA \mathcal{B} on finite words, such that \mathcal{M} 0-halts iff there exists a word $w \in \Sigma^+$ such that $\mathcal{B}(w) \geq \mathcal{A}(w)$ iff there exists a word $w \in \Sigma^+$ such that $\mathcal{B}(w) > \mathcal{A}(w)$.*

Proof. Given a two-counter machine \mathcal{M} , consider the DMDA \mathcal{A} and the NMDA \mathcal{B} constructed in Section 3.1, and an input word w . Let $u = \text{PREF}_{\text{HALT}}(w)$.

We prove the claim by showing that I) if u correctly describes a 0-halting run of \mathcal{M} then $\mathcal{B}(w) > \mathcal{A}(w)$, and II) if u does not fit the actual run of \mathcal{M} , or

if it does fit it, but the run does not 0-halt, then the violation is detected by \mathcal{B} , in the sense that $\mathcal{B}(w) < \mathcal{A}(w)$.

I. We start with the case that u correctly describes a 0-halting run of \mathcal{M} , and show that $\mathcal{B}(w) > \mathcal{A}(w)$.

Observe that in all of \mathcal{B} 's checkers, the transition over the HALT command to the q_{halt} state has a weight higher than the weight of the corresponding transition in \mathcal{A} . Thus, if a checker behaves like \mathcal{A} over u , namely uses the primal functions, it generates a value higher than that of \mathcal{A} .

We show below that each of the checkers generates a value higher than the value of \mathcal{A} on u (which is also the value of \mathcal{A} on w), also if it nondeterministically “guesses a violation”, behaving differently than \mathcal{A} .

1. Halt Checker. Since u does have the HALT command, the run of the halt checker on u , if guessing a violation, will end in the pair of transitions from q_{HC} to q_{last} to q_{freeze} with discount factor 2 and weights 0 and 2, respectively.

Let D be the accumulated discount factor in the gadget up to these pair of transitions. According to Lemma 1, the accumulated weight at this point is $1 - \frac{1}{D}$, hence the value of the run will be $1 - \frac{1}{D} + \frac{1}{D} \cdot 0 + \frac{1}{2D} \cdot 2 = 1$, which is, according to Lemma 1, larger than the value of \mathcal{A} on any word.

2,3. Negative- and Positive-Counters Checkers. Since u has the same number of INC(c) and DEC(c) letters, by Eq. (1) and Lemma 1, these gadgets and \mathcal{A} will have the same value on the prefix of u until the last transition, on which the gadgets will have a higher weight.

4. Command Checker. As this gadget is deterministic, it cannot “guess a violation”, and its value on u is larger than $\mathcal{A}(u)$ due to the weight on the HALT command.

5,6. Zero-Jump Checkers. Consider a counter $c \in \{x, y\}$ and a run r of the gadget on u . If r did not move to q_c , we have $\mathcal{B}(r) > \mathcal{A}(w)$, similarly to the analysis in the negative- and positive-counters checkers. Otherwise, denote the transition that r used to move to q_c as t . Observe that since u correlates to the actual run of \mathcal{M} , we have that t was indeed taken when $c = 0$. In this case the value of the run will not be affected, since before t we have the same number of INC(c) and DEC(c) letters, and after t we also have the same number of INC(c) and DEC(c) letters. Hence, due to the last transition over the HALT command, we have $\mathcal{B}(r) > \mathcal{A}(u)$.

7,8. Positive-Jump Checkers. Consider a counter $c \in \{x, y\}$ and a run r of the gadget on u . If r never reaches q_{PC1}^c , it has the same sequence of weights and discount factors as \mathcal{A} , except for the higher-valued HALT transition. If r reaches q_{PC1}^c but never reaches q_{PC2}^c , since u ends with a HALT letter, we have that r ends with a transition to q_{freeze} that has a weight of 1, hence $\mathcal{B}(r) = 1 > \mathcal{A}(w)$.

If r reaches q_{PC2}^c , let $u = y \cdot \text{INC}(c) \cdot z \cdot v$ where y has no INC(c) letters, $t = r[|y|+1+|z|]$ is the first transition in r targeted at q_{PC2}^c , and $\alpha_c \geq 1$ is the value of the counter c when t is taken. We have that $1 + \#(\text{INC}(c), z) = \#(\text{DEC}(c), z) + \alpha_c$. Since u is balanced, we also have that $\#(\text{DEC}(c), v) = \#(\text{INC}(c), v) + \alpha_c$. For the

first $\text{INC}(c)$ letter, r gets a discount factor of $\rho_d(\text{INC}(c)) = \rho_p(\text{DEC}(c))$. All the following $\text{INC}(c)$ and $\text{DEC}(c)$ letters contribute discount factors according to ρ_p in z and according to ρ_d in v . Hence, r gets the discount factor $\rho_p(\text{DEC}(c))$ a total of

$$\begin{aligned} 1 + \#(\text{DEC}(c), z) + \#(\text{INC}(c), v) &= 1 + 1 + \#(\text{INC}(c), z) - \alpha_c + \#(\text{INC}(c), v) \\ &= \#(\text{INC}(c), u) + 1 - \alpha_c \\ &\leq \#(\text{INC}(c), u) = \#(\text{DEC}(c), u) \end{aligned}$$

times, and the discount factor $\rho_p(\text{INC}(c))$ a total of

$$\begin{aligned} \#(\text{INC}(c), z) + \#(\text{DEC}(c), v) &= \#(\text{INC}(c), z) + \#(\text{INC}(c), v) + \alpha_c \\ &= \#(\text{INC}(c), u) - 1 + \alpha_c \geq \#(\text{INC}(c), u) \end{aligned}$$

times.

Therefore, the value of r is at least as big as the value of \mathcal{A} on the prefix of u until the HALT transition, and due to the higher weight of r on the latter, we have $\mathcal{B}(r) > \mathcal{A}(u)$.

II. We continue with the case that u does not correctly describe a 0-halting run of \mathcal{M} , and show that $\mathcal{B}(w) < \mathcal{A}(w)$. Observe that the incorrectness must fall into one of the following cases, each of which results in a lower value of one of \mathcal{B} 's gadgets on u , compared to the value of \mathcal{A} on u :

- *The word u has no HALT command.* In this case the minimal-valued run of the halt checker on u will be the same as of \mathcal{A} until the last transition, on which the halt checker will have a 0 weight, compared to a strictly positive weight in \mathcal{A} .
- *The word u does not describe a run that ends up with value 0 in both counters.* Then there are the following sub-cases:

- *The word u has more $\text{DEC}(c)$ than $\text{INC}(c)$ letters for some counter $c \in \{x, y\}$.* For $c = x$, in the negative-counters checker, more discount factors were changed from 4 to 2 than those changed from 5 to 10, compared to their values in \mathcal{A} , implying that the total value of the gadget until the last letter will be lower than of \mathcal{A} on it. For $c = y$, we have a similar analysis with respect to the discount factors 6; 3, and 7; 14.
- *The word u has more $\text{INC}(c)$ than $\text{DEC}(c)$ letters for some counter $c \in \{x, y\}$.* By Eq. (1) and Lemma 1, the value of the positive-counters checker until the last transition will be lower than of \mathcal{A} until the last transition.

Observe, though, that the weight of the gadgets on the HALT transition (16) is still higher than that of \mathcal{A} on it (15). Nevertheless, since a “violation detection” results in replacing at least one discount factor from 4 to 2, from 6 to 3, from 5 to 4, or from 7 to 6 (and replacing the corresponding weights, for preserving the $\frac{\rho-1}{\rho}$ ratio), and the ratio difference between 16 and 15 is less significant than between the other pairs of weights, we have that the gadget’s value and therefore \mathcal{B} ’s value on u is smaller than $\mathcal{A}(u)$. Indeed, by

Lemma 1 $\mathcal{A}(u) = 1 - \frac{1}{D_{\mathcal{A}}}$, where $D_{\mathcal{A}}$ is the multiplication of the discount factors along \mathcal{A} 's run, and $\mathcal{B}(u) \leq 1 - (\frac{1}{D_{\mathcal{A}}} \cdot \frac{7}{6} \cdot \frac{15}{16}) < 1 - \frac{1}{D_{\mathcal{A}}} = \mathcal{A}(u)$.

– *The word u does not correctly describe the run of \mathcal{M} .* Then there are the following sub-cases:

- *The incorrect description does not relate to conditional jumps.* Then the command-checker has the same weights and discount factors as \mathcal{A} on the prefix of u until the incorrect description, after which it has 0 weights, compared to strictly positive weights in \mathcal{A} .

- *The incorrect description relates to conditional jumps.* Then there are the following sub-sub-cases:

- * *A counter $c > 0$ at a position i of \mathcal{M} 's run, while $u[i] = \text{GOTO } l_k, c = 0$.* Let $v = u[0..i-1]$ and $u = v \cdot v'$, and consider the run r of the zero-jump checker on u that moves to q_c after v . Then $\#(\text{INC}(c), v) > \#(\text{DEC}(c), v)$ and $\#(\text{INC}(c), v') < \#(\text{DEC}(c), v')$. (We may assume that the total number of $\text{INC}(c)$ and $\text{DEC}(c)$ letters is the same, as otherwise one of the previous checkers detects it.)

All the $\text{INC}(c)$ and $\text{DEC}(c)$ transitions in $r[0..i-1]$ have weights and discount factors according to the dual functions, and those transitions in $r[i..|w|-1]$ have weights and discount factors according to the primal functions. Therefore, compared to \mathcal{A} , more weights changed from $\gamma_p(\text{INC}(c))$ to $\gamma_d(\text{INC}(c)) = \gamma_p(\text{DEC}(c))$ than weights changed from $\gamma_p(\text{DEC}(c))$ to $\gamma_d(\text{DEC}(c)) = \gamma_p(\text{INC}(c))$, resulting in a lower total value of r than of \mathcal{A} on u . (As shown for the negative- and positive-counters checkers, the higher weight of the HALT transition is less significant than the lower values above.)

- * *A counter $c = 0$ at a position i of \mathcal{M} 's run, while $u[i] = \text{GOTO } l_k, c > 0$.* Let r be a minimal-valued run of the positive-jump checker on u . If there are no $\text{INC}(c)$ letters in u before position i , r will have the same weights and discount factors as \mathcal{A} until the i 's letter, on which it will move from q_{PC1}^c to q_{freeze} , continuing with 0-weight transitions, compared to strictly positive ones in \mathcal{A} .

Otherwise, we have that the first $\text{INC}(c)$ letter of u takes r from q_{PC0}^c to q_{PC1}^c with a discount factor of $\rho_d(\text{INC}(c))$. Then in q_{PC1}^c we have more $\text{DEC}(c)$ transitions than $\text{INC}(c)$ transitions, and in q_{PC2}^c we have the same number of $\text{DEC}(c)$ and $\text{INC}(c)$ transitions. (We may assume that u passed the previous checkers, and thus has the same total number of $\text{INC}(c)$ and $\text{DEC}(c)$ letters.) Hence, we get two more discount factors of $\rho_d(\text{INC}(c))$ than $\rho_p(\text{INC}(c))$, resulting in a value smaller than $\mathcal{A}(u)$. (As in the previous cases, the higher value of the HALT transition is less significant.)

□

3.2 Undecidability of Comparison

For finite words, the undecidability result directly follows from Lemma 2 and the undecidability of the 0-halting problem of counter machines [30].

Theorem 1. *Strict and non-strict containment of (integral) NMDAs on finite words are undecidable. More precisely, the problems of deciding for given integral NMDA \mathcal{N} and integral DMDA \mathcal{D} whether $\mathcal{N}(w) \leq \mathcal{D}(w)$ for all finite words w and whether $\mathcal{N}(w) < \mathcal{D}(w)$ for all finite words w .*

For infinite words, undecidability of non-strict containment also follows from the reduction given in Section 3.1, as the reduction considers prefixes of the word until the first HALT command. We leave open the question of whether strict containment is also undecidable for infinite words. The problem with the latter is that a HALT command might never appear in an infinite word w that incorrectly describes a halting run of the two-counter machine, in which case both automata \mathcal{A} and \mathcal{B} of the reduction will have the same value on w . On words w that have a HALT command but do not correctly describe a halting run of the two-counter machine we have $\mathcal{B}(w) < \mathcal{A}(w)$, and on a word w that does correctly describe a halting run we have $\mathcal{B}(w) > \mathcal{A}(w)$. Hence, the reduction only relates to whether $\mathcal{B}(w) \leq \mathcal{A}(w)$ for all words w , but not to whether $\mathcal{B}(w) < \mathcal{A}(w)$ for all words w .

Theorem 2. *Non-strict containment of (integral) NMDAs on infinite words is undecidable. More precisely, the problem of deciding for given integral NMDA \mathcal{N} and integral DMDA \mathcal{D} whether $\mathcal{N}(w) \leq \mathcal{D}(w)$ for all infinite words w .*

Proof. The automata \mathcal{A} and \mathcal{B} in the reduction given in Section 3.1 can operate as is on infinite words, ignoring the Halt-Checker gadget of \mathcal{B} which is only relevant to finite words.

Since the values of both \mathcal{A} and \mathcal{B} on an input word w only relate to the prefix $u = \text{PREFIX}_{\text{HALT}(w)}$ of w until the first HALT command, we still have that $\mathcal{B}(w) > \mathcal{A}(w)$ if u correctly describes a halting run of the two-counter machine \mathcal{M} and that $\mathcal{B}(w) < \mathcal{A}(w)$ if u is finite and does not correctly describe a halting run of \mathcal{M} .

Yet, for infinite words there is also the possibility that the word w does not contain the HALT command. In this case, the value of both \mathcal{A} and the command checker of \mathcal{B} will converge to 1, getting $\mathcal{A}(w) = \mathcal{B}(w)$.

Hence, if \mathcal{M} 0-halts, there is a word w , such that $\mathcal{B}(w) > \mathcal{A}(w)$ and otherwise, for all words w , we have $\mathcal{B}(w) \leq \mathcal{A}(w)$. \square

Observe that for NMDAs, equivalence and non-strict containment are interreducible.

Theorem 3. *Equivalence of (integral) NMDAs on finite as well as infinite words is undecidable. That is, the problem of deciding for given integral NMDAs \mathcal{A} and \mathcal{B} on finite or infinite words whether $\mathcal{A}(w) = \mathcal{B}(w)$ for all words w .*

Proof. Assume toward contradiction the existence of a procedure for equivalence check of \mathcal{A} and \mathcal{B} . We can use the nondeterminism to obtain an automaton $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$, having $\mathcal{C}(w) \leq \mathcal{A}(w)$ for all words w . We can then check whether \mathcal{C} is equivalent to \mathcal{A} , which holds if and only if $\mathcal{A}(w) \leq \mathcal{B}(w)$ for all words w . Indeed, if $\mathcal{A}(w) \leq \mathcal{B}(w)$ then $\mathcal{A}(w) \leq \min(\mathcal{A}(w), \mathcal{B}(w)) = \mathcal{C}(w)$, while if there exists a word w , such that $\mathcal{B}(w) < \mathcal{A}(w)$, we have $\mathcal{C}(w) = \min(\mathcal{A}(w), \mathcal{B}(w)) < \mathcal{A}(w)$, implying that \mathcal{C} and \mathcal{A} are not equivalent. Thus, such a procedure contradicts the undecidability of non-strict containment, shown in Theorems 1 and 2. \square

4 Comparison of NDAs with Different Discount Factors

We present below our algorithm for the comparison of NDAs with different discount factors. We start with automata on infinite words, and then show how to solve the case of finite words by reduction to the case of infinite words.

The algorithm is based on our main observation that, due to the difference between the discount factors, we only need to consider the combination of the automata computation trees up to some level k , after which we can consider first the best/worst continuation of the automaton with the smaller discount factor, and on top of it the worst/best continuation of the second automaton.

For an NDA \mathcal{A} , we define its *lowest* (resp. *highest*) *infinite run value* by $\text{LOWRUN}(\mathcal{A})$ (resp. $\text{HIGHRUN}(\mathcal{A}) = \min$ (resp. \max) $\{\mathcal{A}(r) \mid r \text{ is an infinite run of } \mathcal{A} \text{ (on some word } w \in \Sigma^\omega)\}$.

Observe that we can use \min and \max (rather than \inf and \sup) since the infimum and supremum values are indeed attainable by specific infinite runs of the NDA (cf. [9, Proof of Theorem 9]). Notice that $\text{LOWRUN}(\mathcal{A})$ and $\text{HIGHRUN}(\mathcal{A})$ can be calculated in PTIME by a simple reduction to one-player discounted-payoff games [4].

Considering word values, we also refer to the *lowest* (resp. *highest*) *word value* of \mathcal{A} , defined by $\text{LOWWORD}(\mathcal{A})$ (resp. $\text{HIGHWORD}(\mathcal{A}) = \min$ (resp. \max) $\{\mathcal{A}(w) \mid w \in \Sigma^\omega\}$. Observe that $\text{LOWWORD}(\mathcal{A}) = \text{LOWRUN}(\mathcal{A})$, $\text{HIGHWORD}(\mathcal{A}) \leq \text{HIGHRUN}(\mathcal{A})$, and for deterministic automaton, $\text{HIGHWORD}(\mathcal{A}) = \text{HIGHRUN}(\mathcal{A})$.

For an NMDA \mathcal{A} with states Q , we define the *maximal difference between suffix runs* of \mathcal{A} as $\text{MAXDIFF}(\mathcal{A}) = \max \{\text{HIGHRUN}(\mathcal{A}^q) - \text{LOWRUN}(\mathcal{A}^q) \mid q \in Q\}$. Notice that $\text{MAXDIFF}(\mathcal{A}) \geq 0$ and that $\mathcal{A}^q(w)$ is bounded as follows.

$$\text{LOWRUN}(\mathcal{A}^q) \leq \mathcal{A}^q(w) \leq \text{LOWRUN}(\mathcal{A}^q) + \text{MAXDIFF}(\mathcal{A}) \quad (2)$$

Lemma 3. *There is an algorithm that computes for every input discount factors $\lambda_A, \lambda_D \in \mathbb{Q} \cap (1, \infty)$, λ_A -NDA \mathcal{A} and λ_D -DDA \mathcal{D} on infinite words the value of $\min\{\mathcal{A}(w) - \mathcal{D}(w) \mid w \in \Sigma^\omega\}$.*

Proof. Consider an alphabet Σ , discount factors $\lambda_A, \lambda_D \in \mathbb{Q} \cap (1, \infty)$, a λ_A -NDA $\mathcal{A} = \langle \Sigma, Q_{\mathcal{A}}, \iota_{\mathcal{A}}, \delta_{\mathcal{A}}, \gamma_{\mathcal{A}} \rangle$ and a λ_D -DDA $\mathcal{D} = \langle \Sigma, Q_{\mathcal{D}}, \iota_{\mathcal{D}}, \delta_{\mathcal{D}}, \gamma_{\mathcal{D}} \rangle$. When $\lambda_A = \lambda_D$, we can generate a λ_A -NDA $\mathcal{C} \equiv \mathcal{A} - \mathcal{D}$ over the product of \mathcal{A} and \mathcal{D} and compute $\text{LOWWORD}(\mathcal{C})$.

When $\lambda_A \neq \lambda_D$, **we consider first the case that $\lambda_A < \lambda_D$.**

Our algorithm unfolds the computation trees of \mathcal{A} and \mathcal{D} , up to a level in which only the minimal-valued suffix words of \mathcal{A} remain relevant – Due to the massive difference between the accumulated discount factor in \mathcal{A} compared to the one in \mathcal{D} , any “penalty” of not continuing with a minimal-valued suffix word in \mathcal{A} , defined below as $m_{\mathcal{A}}$, cannot be compensated even by the maximal-valued word of \mathcal{D} , which “profit” is at most as high as $\text{MAXDIFF}(\mathcal{D})$. Hence, at that level, it is enough to look among the minimal-valued suffixes of \mathcal{A} for the one that implies the highest value in \mathcal{D} .

For every transition $t = (q, \sigma, q') \in \delta_{\mathcal{A}}$, let $\text{MINVAL}(q, \sigma, q') = \gamma_{\mathcal{A}}(q, \sigma, q') + \frac{1}{\lambda_A} \cdot \text{LOWWORD}(\mathcal{A}^{q'})$ be the best (minimal) value that \mathcal{A}^q can get by taking t as

the first transition. We say that t is *preferred* if it starts a minimal-valued infinite run of \mathcal{A}^q , namely $\delta_{pr} = \{t = (q, \sigma, q') \in \delta_{\mathcal{A}} \mid \text{MINVAL}(t) = \text{LOWWORD}(\mathcal{A}^q)\}$ is the set of preferred transitions of \mathcal{A} . Observe that an infinite run of \mathcal{A}^q that takes only transitions from δ_{pr} , has a value equal to $\text{LOWRUN}(\mathcal{A}^q)$ (cf. [9, Proof of Theorem 9]).

If all the transitions of \mathcal{A} are preferred, \mathcal{A} has the same value on all words, and then $\min\{\mathcal{A}(w) - \mathcal{D}(w) \mid w \in \Sigma^\omega\} = \text{LOWRUN}(\mathcal{A}) - \text{HIGHWORD}(\mathcal{D})$. (Recall that since \mathcal{D} is deterministic, we can easily compute $\text{HIGHWORD}(\mathcal{D})$.) Otherwise, let $m_{\mathcal{A}}$ be the minimal penalty for not taking a preferred transition in \mathcal{A} , meaning $m_{\mathcal{A}} = \min\left\{\text{MINVAL}(t') - \text{MINVAL}(t'') \mid \begin{array}{l} t' = (q, \sigma', q') \in \delta_{\mathcal{A}} \setminus \delta_{pr}, \\ t'' = (q, \sigma'', q'') \in \delta_{pr} \end{array}\right\}$. Observe that $m_{\mathcal{A}} > 0$.

Considering the connection between $m_{\mathcal{A}}$ and $\text{MAXDIFF}(\mathcal{D})$, notice first that if $\text{MAXDIFF}(\mathcal{D}) = 0$, \mathcal{D} has the same value on all words, and then we have $\min\{\mathcal{A}(w) - \mathcal{D}(w) \mid w \in \Sigma^\omega\} = \text{LOWRUN}(\mathcal{A}) - \text{LOWRUN}(\mathcal{D})$. Otherwise, meaning $\text{MAXDIFF}(\mathcal{D}) > 0$, we unfold the computation trees of \mathcal{A} and \mathcal{D} for the first k levels, until the maximal difference between suffix runs in \mathcal{D} , divided by the accumulated discount factor of \mathcal{D} , is smaller than the minimal penalty for not taking a preferred transition in \mathcal{A} , divided by the accumulated discount factor of \mathcal{A} . Meaning, k is the minimal integer such that

$$\frac{\text{MAXDIFF}(\mathcal{D})}{\lambda_{\mathcal{D}}^k} < \frac{m_{\mathcal{A}}}{\lambda_{\mathcal{A}}^k} \quad (3)$$

Starting at level k , the penalty gained by taking a non-preferred transition of \mathcal{A} cannot be compensated by a higher-valued word of \mathcal{D} .

At level k , we consider separately every run ψ of \mathcal{A} on some prefix word u . We should look for a suffix word w , that minimizes

$$\mathcal{A}(uw) - \mathcal{D}(uw) = \mathcal{A}(\psi) + \frac{1}{\lambda_{\mathcal{A}}^k} \cdot \mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w) - \mathcal{D}(u) - \frac{1}{\lambda_{\mathcal{D}}^k} \cdot \mathcal{D}^{\delta_{\mathcal{D}}(u)}(w) \quad (4)$$

A central point of the algorithm is that every word that minimizes $\mathcal{A} - \mathcal{D}$ must take only preferred transitions of \mathcal{A} starting at level k (see Lemma 4). As all possible remaining continuations after level k yield the same value in \mathcal{A} , we can choose among them the continuation that yields the highest value in \mathcal{D} .

Let \mathcal{B} be the partial automaton with the states of \mathcal{A} , but only its preferred transitions δ_{pr} . (We ignore words on which \mathcal{B} has no runs.) We shall use the automata product $\mathcal{B}^{\delta_{\mathcal{A}}(\psi)} \times \mathcal{D}^{\delta_{\mathcal{D}}(u)}$ to force suffix words that only take preferred transitions of \mathcal{A} , while calculating among them the highest value in \mathcal{D} .

Let $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))} = \langle \Sigma, Q_{\mathcal{A}} \times Q_{\mathcal{D}}, \{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))\}, \delta_{pr} \times \delta_{\mathcal{D}}, \gamma_{\mathcal{C}} \rangle$ be the partial $\lambda_{\mathcal{D}}$ -NDA that is generated by the product of $\mathcal{B}^{\delta_{\mathcal{A}}(\psi)}$ and $\mathcal{D}^{\delta_{\mathcal{D}}(u)}$, while only considering the weights (and discount factor) of \mathcal{D} , meaning $\gamma_{\mathcal{C}}((q, p), \sigma, (q', p')) = \gamma_{\mathcal{D}}(p, \sigma, p')$.

A word w has a run in $\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}$ that uses only preferred transitions iff w has a run in $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}$. Also, observe that the nondeterminism in \mathcal{C} is only related to the nondeterminism in \mathcal{A} , and the weight function of \mathcal{C} only depends on the

weights of \mathcal{D} , hence all the runs of $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}$ on the same word result in the same value, which is the value of that word in \mathcal{D} . Combining both observations, we get that a word w has a run in $\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}$ that uses only preferred transitions iff w has a run r in $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}$ such that $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}(r) = \mathcal{D}^{\delta_{\mathcal{D}}(u)}(w)$. Hence, after taking the k -sized run ψ of \mathcal{A} , and under the notations defined in Eq. (4), a suffix word w that can take only preferred transitions of \mathcal{A} , and maximizes $\mathcal{D}^{\delta_{\mathcal{D}}(u)}(w)$, has a value of $\mathcal{D}^{\delta_{\mathcal{D}}(u)}(w) = \text{HIGHRUN}(\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))})$. This leads to

$$\begin{aligned} & \min \{ \mathcal{A}(v) - \mathcal{D}(v) \mid v \in \Sigma^\omega \} = \\ & \min \left\{ \mathcal{A}(\psi) + \frac{\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w)}{\lambda_{\mathcal{A}}^k} - \mathcal{D}(u) - \frac{\mathcal{D}^{\delta_{\mathcal{D}}(u)}(w)}{\lambda_{\mathcal{D}}^k} \mid \psi \text{ is a run of } \mathcal{A} \text{ on } u \right\} = \\ & \min_{\psi} \left\{ \mathcal{A}(\psi) + \frac{\text{LOWRUN}(\mathcal{A}^{\delta_{\mathcal{A}}(\psi)})}{\lambda_{\mathcal{A}}^k} - \mathcal{D}(u) - \frac{\text{HIGHRUN}(\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))})}{\lambda_{\mathcal{D}}^k} \mid \begin{array}{l} u \in \Sigma^k, \\ \psi \text{ is a run} \\ \text{of } \mathcal{A} \text{ on } u \end{array} \right\} \end{aligned}$$

and it is only left to calculate this value for every k -sized run of \mathcal{A} , meaning for every leaf in the computation tree of \mathcal{A} .

The case of $\lambda_{\mathcal{A}} > \lambda_{\mathcal{D}}$ is analogous, with the following changes:

- For every transition of \mathcal{D} , we compute $\text{MAXVAL}(p, \sigma, p') = \gamma_{\mathcal{D}}(p, \sigma, p') + \frac{1}{\lambda_{\mathcal{D}}} \cdot \text{HIGHWORD}(\mathcal{D}^{p'})$, instead of $\text{MINVAL}(q, \sigma, q')$.
- The preferred transitions of \mathcal{D} are the ones that start a maximal-valued infinite run, that is $\delta_{pr} = \{ t = (p, \sigma', p') \in \delta_{\mathcal{D}} \mid \text{MAXVAL}(t) = \text{HIGHRUN}(\mathcal{D}^p) \}$, and the minimal penalty $m_{\mathcal{D}}$ is

$$m_{\mathcal{D}} = \min \left\{ \text{MAXVAL}(t'') - \text{MAXVAL}(t') \mid \begin{array}{l} t'' = (p, \sigma'', p'') \in \delta_{pr}, \\ t' = (p, \sigma', p') \in \delta_{\mathcal{D}} \setminus \delta_{pr} \end{array} \right\}$$
- k should be the minimal integer such that $\frac{\text{MAXDIFF}(\mathcal{A})}{\lambda_{\mathcal{A}}^k} < \frac{m_{\mathcal{D}}}{\lambda_{\mathcal{D}}^k}$.
- We define \mathcal{B} to be the restriction of \mathcal{D} to its preferred transitions, and $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}$ as a partial $\lambda_{\mathcal{A}}$ -NDA on the product of $\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}$ and $\mathcal{B}^{\delta_{\mathcal{D}}(u)}$ while considering the weights of \mathcal{A} .
- We calculate $\text{LOWRUN}(\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))})$ for every k -sized run of \mathcal{A} , ψ , and conclude that $\min \{ \mathcal{A} - \mathcal{D} \}$ is equal to

$$\min_{\psi} \left\{ \mathcal{A}(\psi) + \frac{\text{LOWRUN}(\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))})}{\lambda_{\mathcal{A}}^k} - \mathcal{D}(u) - \frac{\text{HIGHRUN}(\mathcal{D}_{\delta_{\mathcal{D}}(u)})}{\lambda_{\mathcal{D}}^k} \right\}$$

Observe that in this case, it might not hold that all runs of $\mathcal{C}^{(\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u))}$ on the same word have the same value, but such property is not required, since we look for the minimal run value (which is the minimal word value).

□

Notice that the algorithm of Lemma 3 does not work if switching the direction of containment, namely if considering a deterministic \mathcal{A} and a nondeterministic \mathcal{D} . The determinism of \mathcal{D} is required for finding the maximal value of a valid word in $\mathcal{B}^{\delta_{\mathcal{A}}(\psi)} \times \mathcal{D}^{\delta_{\mathcal{D}}(u)}$. If \mathcal{D} is not deterministic, the maximal-valued run of

$\mathcal{B}^{\delta_{\mathcal{A}}(\psi)} \times \mathcal{D}^{\delta_{\mathcal{D}}(u)}$ on some word w equals the value of some run of \mathcal{D} on w , but not necessarily the value of \mathcal{D} on w . We also need \mathcal{D} to be deterministic for computing $\text{HIGHWORD}(\mathcal{D}^p)$ in the case that $\lambda_{\mathcal{A}} > \lambda_{\mathcal{D}}$.

To show the correctness of Lemma 3, we present the following claim.

Lemma 4. *For every input discount factors $\lambda_{\mathcal{A}}, \lambda_{\mathcal{D}} \in \mathbb{Q} \cap (1, \infty)$ such that $\lambda_{\mathcal{A}} < \lambda_{\mathcal{D}}$, $\lambda_{\mathcal{A}}$ -NDA \mathcal{A} and $\lambda_{\mathcal{D}}$ -DDA \mathcal{D} , every infinite word w that minimizes $\mathcal{A}(w) - \mathcal{D}(w)$ must take a preferred transition of \mathcal{A} at every level k for which $\frac{\text{MAXDIFF}(\mathcal{D})}{\lambda_{\mathcal{D}}^k} < \frac{m_{\mathcal{A}}}{\lambda_{\mathcal{A}}^k}$.*

Proof. Consider discount factors $\lambda_{\mathcal{A}}, \lambda_{\mathcal{D}} \in \mathbb{Q} \cap (1, \infty)$ such that $\lambda_{\mathcal{A}} < \lambda_{\mathcal{D}}$, $\lambda_{\mathcal{A}}$ -NDA \mathcal{A} , $\lambda_{\mathcal{D}}$ -DDA \mathcal{D} , and k the minimal integer such that

$$\frac{\text{MAXDIFF}(\mathcal{D})}{\lambda_{\mathcal{D}}^k} < \frac{m_{\mathcal{A}}}{\lambda_{\mathcal{A}}^k}$$

Assume toward contradiction the existence of a word v that minimizes $\mathcal{A} - \mathcal{D}$, while a minimal-valued run $\psi_{\mathcal{A}}$ of \mathcal{A} on v does not take a preferred transition at some level $n \geq k$. Let u be the n -sized prefix of v , w the corresponding suffix (meaning $v = u \cdot w$), ψ the prefix run of $\psi_{\mathcal{A}}$ on u , and w' some minimal-valued word of $\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}$. The first transition taken by $\psi_{\mathcal{A}}$ when continuing with w is not preferred, meaning

$$\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w) \geq \text{LOWWORD}(\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}) + m_{\mathcal{A}} = \mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w') + m_{\mathcal{A}} \quad (5)$$

Hence,

$$\begin{aligned} \mathcal{A}(v) - \mathcal{D}(v) &\stackrel{(4)}{=} \mathcal{A}(\psi) + \frac{\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w)}{\lambda_{\mathcal{A}}^n} - \mathcal{D}(u) - \frac{\mathcal{D}^{\delta_{\mathcal{D}}(u)}(w)}{\lambda_{\mathcal{D}}^n} \\ &\stackrel{(5),(2)}{\geq} \mathcal{A}(\psi) + \frac{\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w') + m_{\mathcal{A}}}{\lambda_{\mathcal{A}}^n} - \mathcal{D}(u) - \frac{\text{LOWRUN}(\mathcal{D}^{\delta_{\mathcal{D}}(u)}) + \text{MAXDIFF}(\mathcal{D})}{\lambda_{\mathcal{D}}^n} \\ &\stackrel{(3)}{>} \mathcal{A}(\psi) + \frac{\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w')}{\lambda_{\mathcal{A}}^n} - \mathcal{D}(u) - \frac{\text{LOWRUN}(\mathcal{D}^{\delta_{\mathcal{D}}(u)})}{\lambda_{\mathcal{D}}^n} \\ &\stackrel{(2)}{\geq} \mathcal{A}(\psi) + \frac{\mathcal{A}^{\delta_{\mathcal{A}}(\psi)}(w')}{\lambda_{\mathcal{A}}^n} - \mathcal{D}(u) - \frac{\mathcal{D}^{\delta_{\mathcal{D}}(u)}(w')}{\lambda_{\mathcal{D}}^n} \\ &\stackrel{(4)}{=} \mathcal{A}(u \cdot w') - \mathcal{D}(u \cdot w') \end{aligned}$$

leading to a contradiction. \square

Moving to automata on finite words, we reduce the problem to the corresponding problem with respect to automata on infinite words, by adding to the alphabet a new letter that represents the end of the word, and making some required adjustments.

Lemma 5. *There is an algorithm that computes for every input discount factors $\lambda_{\mathcal{A}}, \lambda_{\mathcal{D}} \in \mathbb{Q} \cap (1, \infty)$, $\lambda_{\mathcal{A}}$ -NDA \mathcal{A} and $\lambda_{\mathcal{D}}$ -DDA \mathcal{D} on finite words the value of $\inf \{ \mathcal{A}(u) - \mathcal{D}(u) \mid u \in \Sigma^+ \}$, and determines if there exists a finite word u for which $\mathcal{A}(u) - \mathcal{D}(u)$ equals that value.*

Proof. Without loss of generality, we assume that initial states of automata have no incoming transitions. (Every automaton can be changed in linear time to an equivalent automaton with this property.)

We convert, as described below, an NDA \mathcal{N} on finite words to an NDA $\hat{\mathcal{N}}$ on infinite words, such that $\hat{\mathcal{N}}$ intuitively simulates the finite runs of \mathcal{N} . For an alphabet Σ , a discount factor $\lambda \in \mathbb{Q} \cap (1, \infty)$, and a λ -NDA (DDA) $\mathcal{N} = \langle \Sigma, Q_{\mathcal{N}}, \iota_{\mathcal{N}}, \delta_{\mathcal{N}}, \gamma_{\mathcal{N}} \rangle$ on finite words, we define the λ -NDA (DDA) $\hat{\mathcal{N}} = \langle \hat{\Sigma}, Q_{\mathcal{N}} \cup \{q_{\tau}\}, \iota_{\mathcal{N}}, \delta_{\hat{\mathcal{N}}}, \gamma_{\hat{\mathcal{N}}}\rangle$ on infinite words. The new alphabet $\hat{\Sigma} = \Sigma \cup \{\tau\}$ contains a new letter $\tau \notin \Sigma$ that indicates the end of a finite word. The new state q_{τ} has 0-valued self loops on every letter in the alphabet, and there are 0-valued transitions from every non-initial state to q_{τ} on the new letter τ . Formally, $\delta_{\hat{\mathcal{N}}} = \delta_{\mathcal{N}} \cup \{(q_{\tau}, \sigma, q_{\tau} \mid \sigma \in \hat{\Sigma})\} \cup \{(q, \tau, q_{\tau} \mid q \in Q_{\mathcal{N}} \setminus \iota_{\mathcal{N}})\}$, and

$$\gamma_{\hat{\mathcal{N}}}(t) = \begin{cases} \gamma_{\mathcal{N}}(t) & t \in \delta_{\mathcal{N}} \\ 0 & \text{otherwise} \end{cases}$$

Observe that for every state $q \in Q_{\mathcal{N}}$, the following hold.

1. For every finite run $r_{\mathcal{N}}$ of \mathcal{N}^q , there is an infinite run $r_{\hat{\mathcal{N}}}$ of $\hat{\mathcal{N}}^q$, such that $\hat{\mathcal{N}}^q(r_{\hat{\mathcal{N}}}) = \mathcal{N}^q(r_{\mathcal{N}})$, and $r_{\hat{\mathcal{N}}}$ takes some τ transitions. ($r_{\hat{\mathcal{N}}}$ can start as $r_{\mathcal{N}}$ and then continue with only τ transitions.)
2. For every infinite run $r_{\hat{\mathcal{N}}}$ of $\hat{\mathcal{N}}^q$ that has a τ transition, there is a finite run $r_{\mathcal{N}}$ of \mathcal{N}^q , such that $\hat{\mathcal{N}}^q(r_{\hat{\mathcal{N}}}) = \mathcal{N}^q(r_{\mathcal{N}})$. ($r_{\mathcal{N}}$ can be the longest prefix of $r_{\hat{\mathcal{N}}}$ up to the first τ transition).
3. For every infinite run $r_{\hat{\mathcal{N}}}$ of $\hat{\mathcal{N}}^q$ that has no τ transition, there is a series of finite runs of \mathcal{N}^q , such that the values of the runs in \mathcal{N}^q converge to $\hat{\mathcal{N}}^q(r_{\hat{\mathcal{N}}})$. (For example, the series of all prefixes of $r_{\hat{\mathcal{N}}}$).

Hence, for every $q \in Q_{\mathcal{N}}$ we have $\inf \{\mathcal{N}^q(r) \mid r \text{ is a run of } \mathcal{N}^q\} = \text{LOWRUN}(\hat{\mathcal{N}}^q)$ and $\sup \{\mathcal{N}^q(r) \mid r \text{ is a run of } \mathcal{N}^q\} = \text{HIGHRUN}(\hat{\mathcal{N}}^q)$. (For a non-initial state q , we also consider the “run” of \mathcal{N}^q on the empty word, and define its value to be 0.) Notice that the infimum (supremum) run value of \mathcal{N}^q is attained by an actual run of \mathcal{N}^q iff there is an infinite run of $\hat{\mathcal{N}}^q$ that gets this value and takes a τ transition.

For every state $q \in Q_{\hat{\mathcal{N}}}$, we can determine, as follows, whether $\text{LOWRUN}(\hat{\mathcal{N}}^q)$ is attained by an infinite run taking a τ transition. We calculate $\text{LOWRUN}(\hat{\mathcal{N}}^q)$ for all states, and then start a process that iteratively marks the states of $\hat{\mathcal{N}}$, such that at the end, $q \in Q_{\hat{\mathcal{N}}}$ is marked iff $\text{LOWRUN}(\hat{\mathcal{N}}^q)$ can be achieved by a run with a τ transition. We start with q_{τ} as the only marked state. In each iteration we further mark every state q from which there exists a preferred transition $t = (q, \sigma, q') \in \delta_{pr}$ to some marked state q' . The process terminates when an iteration has no new states to mark. Analogously, we can determine whether $\text{HIGHRUN}(\hat{\mathcal{N}}^q)$ is attained by a run that goes to q_{τ} .

Consider discount factors $\lambda_A, \lambda_D \in \mathbb{Q} \cap (1, \infty)$, a λ_A -NDA \mathcal{A} and a λ_D -DDA \mathcal{D} on finite words. When $\lambda_A = \lambda_D$, similarly to Lemma 3, the algorithm finds the infimum value of $\mathcal{C} \equiv \mathcal{A} - \mathcal{D}$ using $\hat{\mathcal{C}}$, and determines if an actual finite word attains this value using the process described above.

Otherwise, the algorithm converts \mathcal{A} and \mathcal{D} to $\hat{\mathcal{A}}$ and $\hat{\mathcal{D}}$, and proceeds as in Lemma 3 over $\hat{\mathcal{A}}$ and $\hat{\mathcal{D}}$. According to the above observations, we have that $\inf \{ \mathcal{A}(u) - \mathcal{D}(u) \mid u \in \Sigma^+ \} = \min \{ \hat{\mathcal{A}}(w) - \hat{\mathcal{D}}(w) \mid w \in \Sigma^\omega \}$, and that $\inf \{ \mathcal{A}(u) - \mathcal{D}(u) \}$ is attainable iff $\min \{ \hat{\mathcal{A}}(w) - \hat{\mathcal{D}}(w) \}$ is attainable by some word that has a τ transition. Hence, whenever computing LOWRUN or HIGHRUN, we also perform the process described above, to determine whether this value is attainable by a run that has a τ transition. We determine that $\inf \{ \mathcal{A}(u) - \mathcal{D}(u) \}$ is attainable iff exists a leaf of the computation tree that leads to it, for which the relevant values LOWRUN and HIGHRUN are attainable. \square

Complexity analysis We show below that the algorithm of Lemmas 3 and 5 only needs a polynomial space, with respect to the size of the input automata, implying a PSPACE algorithm for the corresponding decision problems. We define the size of an NDA \mathcal{N} , denoted by $|\mathcal{N}|$, as the maximum between the number of its transitions, the maximal binary representation of any weight in it, and the maximal unary representation of the discount factor. (Binary representation of the discount factors might cause our algorithm to use an exponential space, in case that the two factors are very close to each other.) The input NDAs may have rational weights, yet it will be more convenient to consider equivalent NDAs with integral weights that are obtained by multiplying all the weights by their common denominator [6]. (Observe that it causes the values of all words to be multiplied by this same ratio, and it keeps the same input size, up to a polynomial change.)

Before proceeding to the complexity analysis, we provide an auxiliary lemma.

Lemma 6. *For every integers $p > q \in \mathbb{N} \setminus \{0\}$, a $\frac{p}{q}$ -NDA \mathcal{A} with integral weights, and a lasso run $r = t_0, t_1, \dots, t_{x-1}, (t_x, t_{x+1}, \dots, t_{x+y-1})^\omega$ of \mathcal{A} , there exists an integer b , such that $\mathcal{A}(r) = \frac{b}{p^x(p^y - q^y)}$.*

Proof. Let $\lambda = \frac{p}{q}$ be \mathcal{A} 's discount factor, and γ its weight function. Consider a lasso run $r = t_0, t_1, \dots, t_{x-1}, (t_x, t_{x+1}, \dots, t_{x+y-1})^\omega$ of \mathcal{A} . Let $v_f = \gamma(t_0) + \frac{1}{\lambda}\gamma(t_1) + \dots + \frac{1}{\lambda^{x-1}}\gamma(t_{x-1})$ be its prefix value, and $v_\ell = \gamma(t_x) + \frac{1}{\lambda}\gamma(t_{x+1}) + \dots + \frac{1}{\lambda^{y-1}}\gamma(t_{x+y-1})$ its loop value.

Since all the weights are integers, we have that $v_f = \frac{a_f}{p^x}$ and $v_\ell = \frac{a_\ell}{p^y}$ for some integers a_f and a_ℓ . Recall that for a loop ℓ of length y and accumulated value v_ℓ in a λ -NDA, the accumulated value of its infinite repetition is $\sum_{i=0}^{\infty} \frac{v_\ell}{(\lambda^y)^i} = v_\ell \frac{\lambda^y}{\lambda^y - 1}$. Hence the value of r is

$$\begin{aligned} \gamma(r) &= v_f + \frac{1}{\lambda^x} \cdot v_\ell \frac{\lambda^y}{\lambda^y - 1} = \frac{a_f}{p^x} + \frac{a_\ell}{p^y} \cdot \frac{1}{\lambda^{x-y}(\lambda^y - 1)} = \frac{a_f}{p^x} + \frac{a_\ell \cdot q^{x-y}}{p^{y+x-y}(\frac{p^y - q^y}{q^y})} \\ &= \frac{a_f(p^y - q^y) + a_\ell \cdot q^x}{p^x(p^y - q^y)} \end{aligned}$$

\square

Proceeding to the complexity analysis, let the input size be $S = |\mathcal{A}| + |\mathcal{D}|$, the reduced forms of $\lambda_{\mathcal{A}}$ and $\lambda_{\mathcal{D}}$ be $\frac{p}{q}$ and $\frac{p_{\mathcal{D}}}{q_{\mathcal{D}}}$ respectively, the number of states in \mathcal{A} be n , and the maximal difference between transition weights in \mathcal{D} be M . Observe that $n \leq S, p \leq S, M \leq 2 \cdot 2^S, \frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{D}}-1} \leq \frac{p_{\mathcal{D}}}{p_{\mathcal{D}}-q_{\mathcal{D}}} \leq p_{\mathcal{D}} \leq S$, and for $\lambda_{\mathcal{D}} > \lambda_{\mathcal{A}} > 1$, we also have $\frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{A}}} = \frac{p \cdot q_{\mathcal{D}}}{q \cdot p_{\mathcal{D}}} \geq 1 + \frac{1}{S^2}$.

Observe that \mathcal{A} has a best infinite run (and \mathcal{D} has a worst infinite run), in a lasso form as in Lemma 6, with $x, y \in [1..n]$. Indeed, following preferred transitions, a run must complete a lasso, and then may forever repeat its choices of preferred transitions. Hence, $m_{\mathcal{A}}$, being the difference between two lasso runs, is in the form of

$$\begin{aligned} m_{\mathcal{A}} &= \frac{b_1}{p^{x_1}(p^{y_1} - q^{y_1})} - \frac{b_2}{p^{x_2}(p^{y_2} - q^{y_2})} = \frac{b_3}{p^n(p^{y_1} - q^{y_1})(p^{y_2} - q^{y_2})} > \frac{b_3}{p^n p^{y_1} p^{y_2}} \\ &\geq \frac{1}{p^{3n}} \geq \frac{1}{S^{3S}} \stackrel{\text{for } S \geq 1}{>} \frac{1}{(2^S)^{3S}} = \frac{1}{2^{3S^2}} \end{aligned}$$

for some $x_1, x_2, y_1, y_2 \leq n$ and some integers b_1, b_2, b_3 . (Similarly, we can show that $m_{\mathcal{D}} > \frac{1}{2^{3S^2}}$.) We have $\text{MAXDIFF}(\mathcal{D}) \leq M \cdot \frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{D}}-1}$, hence

$$\frac{\text{MAXDIFF}(\mathcal{D})}{m_{\mathcal{A}}} \leq \frac{M \cdot \frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{D}}-1}}{m_{\mathcal{A}}} \leq \frac{2^{1+S} \cdot S \stackrel{\text{(for } S \geq 1)}{<} 2^{3S}}{m_{\mathcal{A}}} < 2^{3S+3S^2}$$

Recall that we unfold the computation tree until level k , which is the minimal integer such that $(\frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{A}}})^k > \frac{\text{MAXDIFF}(\mathcal{D})}{m_{\mathcal{A}}}$. Observe that for $S \geq 1$ we have $(\frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{A}}})^{S^2} \geq (1 + \frac{1}{S^2})^{S^2} \geq 2$, hence for $k' = S^2 \cdot (3S + 3S^2)$, we have

$$\left(\frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{A}}}\right)^{k'} = \left(\left(\frac{\lambda_{\mathcal{D}}}{\lambda_{\mathcal{A}}}\right)^{S^2}\right)^{3S+3S^2} \geq 2^{3S+3S^2} > \frac{\text{MAXDIFF}(\mathcal{D})}{m_{\mathcal{A}}}$$

meaning that k is polynomial in S . Similar analysis shows that k is polynomial in S also for $\lambda_{\mathcal{D}} < \lambda_{\mathcal{A}}$.

Considering decision problems that use our algorithm, due to the equivalence of NPSPACE and PSPACE, the algorithm can nondeterministically guess an optimal prefix word u of size k , letter by letter, as well as a run ψ of \mathcal{A} on u , transition by transition, and then compute the value of $\mathcal{A}(\psi) + \frac{\text{LOWRUN}(\mathcal{A}^{\delta_{\mathcal{A}}(\psi)})}{\lambda_{\mathcal{A}}^k} - \mathcal{D}(u) - \frac{\text{HIGHRUN}(\mathcal{C}^{\delta_{\mathcal{A}}(\psi), \delta_{\mathcal{D}}(u)})}{\lambda_{\mathcal{D}}^k}$.

Observe that along the run of the algorithm, we need to save the following information, which can be done in polynomial space:

- The automaton $\mathcal{C} \equiv \mathcal{B} \times \mathcal{D}$ (or $\mathcal{A} \times \mathcal{B}$), which requires polynomial space.
- $\lambda_{\mathcal{A}}^k$ (for $\mathcal{A}(\psi)$) and $\lambda_{\mathcal{D}}^k$ (for $\mathcal{D}(u)$). Since we save them in binary representation, we have $\log_2(\lambda^k) \leq k \log_2(S)$, requiring polynomial space.

We thus get the following complexity result.

Theorem 4. *For input discount factors $\lambda_{\mathcal{A}}, \lambda_{\mathcal{D}} \in \mathbb{Q} \cap (1, \infty)$, $\lambda_{\mathcal{A}}$ -NDA \mathcal{A} and $\lambda_{\mathcal{D}}$ -DDA \mathcal{D} on finite or infinite words, it is decidable in PSPACE whether $\mathcal{A}(w) \geq \mathcal{D}(w)$ and whether $\mathcal{A}(w) > \mathcal{D}(w)$ for all words w .*

Proof. We use Lemma 3 in the case of infinite words and Lemma 5 in the case of finite words, checking whether $\min \{ \mathcal{A}(w) - \mathcal{D}(w) \} < 0$ and whether $\min \{ \mathcal{A}(w) - \mathcal{D}(w) \} \leq 0$. In the case of finite words, we also use the information of whether there is an actual word that gets the desired value. \square

Since integral NDAs can always be determinized [7], we get as a corollary that there is an algorithm to decide equivalence and strict and non-strict containment of integral NDAs with different (or the same) discount factors. Note, however, that it might not be in PSPACE, since determinization exponentially increases the number of states, resulting in k that is exponential in S , and storing in binary representation values in the order of λ^k might require exponential space.

Corollary 1. *There are algorithms to decide for input integral discount factors $\lambda_A, \lambda_B \in \mathbb{N}$, λ_A -NDA \mathcal{A} and λ_B -NDA \mathcal{B} on finite or infinite words whether or not $\mathcal{A}(w) > \mathcal{B}(w)$, $\mathcal{A}(w) \geq \mathcal{B}(w)$, or $\mathcal{A}(w) = \mathcal{B}(w)$ for all words w .*

5 Conclusions

The new decidability result, providing an algorithm for comparing discounted-sum automata with different integral discount factors, may allow to extend the usage of discounted-sum automata in formal verification, while the undecidability result strengthen the justification of restricting discounted-sum automata with multiple integral discount factors to tidy NMDAs. The new algorithm also extends the possible, more limited, usage of discounted-sum automata with rational discount factors, while further research should be put into this direction.

References

1. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: proceedings of ICALP. vol. 2719, pp. 1022–1037 (2003). https://doi.org/10.1007/3-540-45061-0_79
2. Almagor, S., Boker, U., Kupferman, O.: What’s decidable about weighted automata? *Information and Computatio* **282** (2022). <https://doi.org/10.1016/j.ic.2020.104651>
3. Almagor, S., Kupferman, O., Ringert, J.O., Velner, Y.: Quantitative assume guarantee synthesis. In: proceedings of CAV. pp. 353–374. Springer (2017). https://doi.org/10.1007/978-3-319-63390-9_19
4. Andersson, D.: An improved algorithm for discounted payoff games. In: proceedings of ESSLLI Student Session. pp. 91–98 (2006)
5. Bansal, S., Chaudhuri, S., Vardi, M.Y.: Comparator automata in quantitative verification. In: proceedings of FoSSaCS. LNCS, vol. 10803, pp. 420–437 (2018). https://doi.org/10.1007/978-3-319-89366-2_23
6. Boker, U., Hefetz, G.: Discounted-sum automata with multiple discount factors. In: proceedings of CSL. LIPIcs, vol. 183, pp. 12:1–12:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2021). <https://doi.org/10.4230/LIPIcs.CSL.2021.12>

7. Boker, U., Henzinger, T.A.: Exact and approximate determinization of discounted-sum automata. *Log. Methods Comput. Sci.* **10**(1) (2014). [https://doi.org/10.2168/LMCS-10\(1:10\)2014](https://doi.org/10.2168/LMCS-10(1:10)2014)
8. Boker, U., Henzinger, T.A., Otop, J.: The target discounted-sum problem. In: proceedings of LICS. pp. 750–761 (2015). <https://doi.org/10.1109/LICS.2015.74>
9. Boker, U., Lehtinen, K.: History determinism vs. good for gameness in quantitative automata. In: proceedings of FSTTCS. pp. 38:1–38:20 (2021). <https://doi.org/10.4230/LIPIcs.FSTTCS.2021.38>
10. Brenguier, R., Clemente, L., Hunter, P., Pérez, G.A., Randour, M., Raskin, J.F., Sankur, O., Sassolas, M.: Non-zero sum games for reactive synthesis. In: *Language and Automata Theory and Applications*. pp. 3–23. Springer (2016)
11. Chatterjee, K., Doyen, L., Henzinger, T.A.: Alternating weighted automata. In: proceedings of FCT. LNCS, vol. 5699, pp. 3–13 (2009). https://doi.org/10.1007/978-3-642-03409-1_2
12. Chatterjee, K., Doyen, L., Henzinger, T.A.: Probabilistic weighted automata. In: proceedings of CONCUR. LNCS, vol. 5710, pp. 244–258 (2009). https://doi.org/10.1007/978-3-642-04081-8_17
13. Chatterjee, K., Doyen, L., Henzinger, T.A.: Expressiveness and closure properties for quantitative languages. *Log. Methods Comput. Sci.* **6**(3) (2010), <http://arxiv.org/abs/1007.4018>
14. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. *ACM Trans. Comput. Log.* **11**(4), 23:1–23:38 (2010). <https://doi.org/10.1145/1805950.1805953>
15. Chatterjee, K., Forejt, V., Wojtczak, D.: Multi-objective discounted reward verification in graphs and MDPs. In: proceedings of LPAR. LNCS, vol. 8312, pp. 228–242 (2013). https://doi.org/10.1007/978-3-642-45221-5_17
16. Clarke, E.M., Draghicescu, I.A., Kurshan, R.P.: A unified approach for showing language containment and equivalence between various types of ω -automata. *Information Processing Letters* **46**, 301–308 (1993)
17. Degorre, A., Doyen, L., Gentilini, R., Raskin, J., Toruńczyk, S.: Energy and mean-payoff games with imperfect information. In: proceedings of CSL. LNCS, vol. 6247, pp. 260–274 (2010). https://doi.org/10.1007/978-3-642-15205-4_22
18. Droste, M., Kuske, D.: Skew and infinitary formal power series. *Theor. Comput. Sci.* **366**(3), 199–227 (2006). <https://doi.org/10.1016/j.tcs.2006.08.024>
19. Filiot, E., Gentilini, R., Raskin, J.: Finite-valued weighted automata. In: proceedings of FSTTCS. LIPIcs, vol. 29, pp. 133–145 (2014). <https://doi.org/10.4230/LIPIcs.FSTTCS.2014.133>
20. Filiot, E., Gentilini, R., Raskin, J.: Quantitative languages defined by functional automata. *Log. Methods Comput. Sci.* **11**(3) (2015). [https://doi.org/10.2168/LMCS-11\(3:14\)2015](https://doi.org/10.2168/LMCS-11(3:14)2015)
21. Filiot, E., Löding, C., Winter, S.: Synthesis from weighted specifications with partial domains over finite words. In: proceedings of FSTTCS. pp. 46:1–46:16 (2020). <https://doi.org/10.4230/LIPIcs.FSTTCS.2020.46>
22. Gimbert, H., Zielonka, W.: Limits of multi-discounted markov decision processes. In: proceedings of LICS. pp. 89–98 (2007). <https://doi.org/10.1109/LICS.2007.28>
23. Glendinning, P., Sidorov, N.: Unique representations of real numbers in non-integer bases. *Mathematical Research Letters* **8**(4), 535–543 (2001)
24. Hare, K.: Beta-expansions of pisot and salem numbers. In: *Waterloo Workshop in Computer Algebra* (2006)
25. Hojati, R., Touati, H., Kurshan, R., Brayton, R.: Efficient ω -regular language containment. In: proceedings of CAV. LNCS, vol. 663. springer (1992)

26. Hunter, P., Pérez, G.A., Raskin, J.: Reactive synthesis without regret. *Acta Informatica* **54**(1), 3–39 (2017). <https://doi.org/10.1007/s00236-016-0268-z>
27. Kupferman, O., Vardi, M., Wolper, P.: An automata-theoretic approach to branching-time model checking. *Journal of the ACM* **47**(2), 312–360 (2000)
28. Madani, O., Thorup, M., Zwick, U.: Discounted deterministic markov decision processes and discounted all-pairs shortest paths. *ACM Trans. Algorithms* **6**(2), 33:1–33:25 (2010). <https://doi.org/10.1145/1721837.1721849>
29. Mahler, K.: An unsolved problem on the powers of $\frac{3}{2}$. *The journal of the Australian mathematical society* **8**(2), 313–321 (1968)
30. Minsky, M.L.: *Computation: Finite and Infinite Machines*. Prentice-Hall Series in Automatic Computation, Prentice-Hall (1967)
31. Sutton, R.S., G.Barto, A.: *Introduction to Reinforcement Learning*. MIT Press (1998), <http://dl.acm.org/doi/book/10.5555/551283>
32. Tasiran, S., Hojati, R., Brayton, R.: Language containment using non-deterministic omega-automata. In: proceedings of CHARME. LNCS, vol. 987, pp. 261–277. springer (1995)
33. Vardi, M.Y.: Verification of concurrent programs: The automata-theoretic framework. In: proceedings of LICS. pp. 167–176 (1987)
34. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) *Logics for Concurrency: Structure versus Automata*. LNCS, vol. 1043, pp. 238–266 (1996)
35. Wang, Y., Ye, Q., Liu, T.: Beyond exponentially discounted sum: Automatic learning of return function. *CoRR* (2019), <http://arxiv.org/abs/1905.11591>
36. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. *Theor. Comput. Sci.* **158**, 343–359 (1996). [https://doi.org/10.1016/0304-3975\(95\)00188-3](https://doi.org/10.1016/0304-3975(95)00188-3)