Quantitative Languages Defined by Functional Automata

Emmanuel Filiot, Raffaella Gentilini, Jean-François Raskin

Quantitative Languages Defined by Functional Automata

Emmanuel Filiot[1], Raffaella Gentilini[2], and Jean-François Raskin[1]

1 **Université Libre de Bruxelles**
2 **Università degli Studi di Perugia**

Abstract

In this paper, we study several decision problems for functional weighted automata. To associate values with runs, we consider four different measure functions: the sum, the mean, the discounted sum of weights along edges and the ratio between rewards and costs. On the positive side, we show that the existential and universal threshold problems, the language inclusion problem and the equivalence problem are all decidable for the class of functional weighted automata and the four measure functions that we consider. On the negative side, we also study the quantitative extension of the realizability problem and show that it is undecidable for sum, mean and ratio. Finally, we show how to decide if the quantitative language defined by a functional weighted discounted sum automaton can be defined with a deterministic automata (it was already known for sum and mean).

1998 ACM Subject Classification Algorithms, Theory, Verification

Keywords and phrases Weighted automata, quantitative languages, functionality, synthesis, computer-aided verification

1 Introduction

Recently, there have been several efforts made to lift the foundations of computer aided verification and synthesis from the basic *Boolean* case to the richer *quantitative* case, e.g. [10, 8, 1]. This paper belongs to this line of research and contributes to the study of quantitative languages over finite words.

Our paper proposes a systematic study of the algorithmic properties of several classes of *functional* weighted automata (defining quantitative languages). A functional weighted automaton is a *nondeterministic* weighted automaton such that any two accepting runs ρ_1, ρ_2 on a word w associate with this word a unique value $\mathsf{V}(\rho_1) = \mathsf{V}(\rho_2)$. As we show in this paper, several important verification problems are decidable for nondeterministic functional weighted automata while they are undecidable (or unknown to be decidable) for the full class of nondeterministic weighted automata. As functional weighted automata are a natural generalization of *unambiguous* weighted automata, and as unambiguity captures most of the nondeterminism that is useful in practice, our results are both theoretically and practically important. Also, the notion of functionality leads to useful insight into the relation between deterministic and nondeterministic weighted automata and into algorithmic idea for testing equivalence for example.

In this paper, we study automata in which an integer weight, or a pair of integer weights, is associated with each of their transitions. From those weights, an (accepting) run ρ on a word w associates a sequence of weights with the word, and this sequence is mapped to a rational value by a *measure function*. We consider four different measure functions[1]: (i) Sum computes the sum of the weights along the sequence, (ii) Avg returns the mean value of the weights, (iii) Dsum_λ computes

[1] We do not consider the measure functions Min and Max that map a sequence to the minimal and the maximal value that appear in the sequence as the nondeterministic automata that use those measure functions can be made deterministic and all the decision problems for them have known and simple solutions.

the discounted sum of the weights for a given discount factor $\lambda \in \mathbb{Q} \cap]0,1[$, and (iv) Ratio is applied to a sequence of pairs of weights, and it returns the ratio between the sum of weights appearing as the first component (rewards) and the sum of the weights appearing as the second component (costs). The value associated with a word w accepted by A is denoted by $L_A(w)$. While Sum and, to some extent, Avg are known because they can be seen as operations over a semiring of values [20], the case of Dsum_λ and Ratio are less studied. Those two measures are motivated by applications in computer aided verification and synthesis, see for example [12, 7].

Contributions Functionality is a semantical property. We show that it can be decided for the four classes of measure functions that we consider (either in polynomial time or polynomial space). Then we solve the following decision problems, along the line of [10]. First, we consider *threshold* problems. The *existential* (*universal*, respectively) *threshold* problem asks, given a weighted automaton A and a threshold $\nu \in \mathbb{Q}$, if there exists a word (if for all words, respectively) w accepted by A: $L_A(w) \geq \nu$. Those problems can be seen as generalizations of the emptiness and universality problems for finite state automata. Second, we consider the *quantitative language inclusion problem* that asks, given two weighted automata A and B, if all words accepted by A are also accepted by B, and for all accepted words w of A, we have $L_A(w) \leq L_B(w)$. We show that all those problems are decidable for the four classes of measure functions that we consider in this paper when the automaton is functional. We show that the inclusion problem is PSPACEC for Sum, Avg and Dsum_λ. For Ratio, we show decidability of the problem using a recent algorithm to solve quadratic diophantine equations [14], this is a new deep result in mathematics and the complexity of the algorithm is not yet known. Note that those decidability results are in sharp contrast with the corresponding results for the full class of nondeterministic weighted automata: for that class, only the existential threshold problem is known to be decidable, the language inclusion problem is undecidable for Sum, Avg, and Ratio while the problem is open for Dsum_λ. We also show that the equivalence problem can be decided in polynomial space for Ratio via an easy reduction to functionality.

Then, we consider a quantitative variant of the *realizability* problem introduced by Church, which is part of the foundations of game theory played on graphs [23] and synthesis of reactive systems [21]. It can be formalized as a game in which two players alternates in choosing letters in their respective alphabet. By doing so, they form a word which is obtained by concatenating the successive choices of the players. The realizability problem asks, given a weighted automaton A, alphabet $\Sigma = \Sigma_1 \times \Sigma_2$, if there exists a strategy for choosing the letters in Σ_1 in the word forming game such that no matter how the adversary chooses his letters in Σ_2, the word w that is obtained belongs to the language of A and $A(w) \geq 0$. We show that this problem is undecidable for Sum, Avg, and Ratio even when considering unambiguous automata (the case Dsum_λ is left open). However, we show that the realizability problem is decidable for the deterministic versions of the automata studied in this paper. This motivates the study of the *determinizability* problem.

The determinizability problem asks, given a functional weighted automaton A, if the quantitative language defined by A is also definable by a *deterministic* automaton. This problem has been solved for Sum, Avg in [16]. It is known that Dsum_λ-automata are not determinizable in general [10]. We give here a decidable *necessary* and *sufficient* condition for the determinizability of functional Dsum_λ automata, and we show how to construct a deterministic automaton from the functional one when this is possible.

Related Works Motivated by computer-aided verification issues, our work follows the same line as [10]. However [10] is mainly concerned with weighted automata on infinite words, either nondeterministic, for which some important problems are undecidable (e.g. inclusion of Avg-automata), or deterministic ones, which are strictly less expressive than functional automata. The Ratio measure is not considered either. Their domains of quantitative languages are assumed to be total (as all states are accepting and their transition relation is total) while we can define partial quantitave languages thanks to an acceptance condition.

Weighted automata over semirings have been extensively studied (see [20] for a survey), and

more generally rational series [4]. For instance, the functionality problem for weighted automata over the tropical semiring, i.e. Sum-automata, is known to be in PTime [16]. Moreover, it is known that determinizability of functional Sum-automata is decidable in PTime [16], as well as for the strictly more expressive class of polynomially ambiguous Sum-automata [15], for which the termination of Mohri's determinization algorithm [20] is decidable. However, the Dsum_λ and Ratio-automata are not automata over any semiring, and therefore results on automata over semirings cannot be directly applied to those measures. The technics we use for deciding functionality and determinization are inspired by technics from word transducers [22, 6, 3, 11, 24].

The functionality problem has been studied for finite state (word) transducers. It was proved to be decidable in [22], and later in [6]. Based on a notion of delay between runs, an efficient PTime procedure for testing functionality has been given in [3]. The functionality problem for Sum,Avg and Dsum_λ-automata is also based on a notion of delay. Based on the *twinning property* [11] and the notion of delay, efficient procedures for deciding determinizability can be devised [3, 24]. This also inspired our determinization procedure for functional Dsum_λ-automata. In [9], Boker et. al. show that Dsum_λ-automata on infinite words with a trivial accepting condition (all states are accepting), but not necessarily functional, are determinizable for any discount factor of the form $1/n$ for some $n \in \mathbb{N}_{\geq 2}$. Their proof is based on a notion of *recoverable gap*, similar to that of delays. In our paper, we provide a sufficient and necessary condition to check whether a functional Dsum_λ-automaton (over finite words) is determinizable. Finally in [13], the relation between discounted weighted automata over a semiring and weighted logics is studied.

To the best of our knowledge, our results on Dsum_λ and Ratio-automata, as well as on the realizability problem, are new. Our main and most technical results are functionality of Dsum_λ and Ratio-automata, inclusion problems, determinizability of functional Dsum_λ-automata, undecidability of the realizability of unambiguous Sum-automata, and solvability of the deterministic versions of the realizability problem. The latter reduce to games on graphs that are to the best of our knowledge new, namely finite Sum, Avg, Dsum_λ, Ratio-games on weighted graphs with a combination of a reachability objective and a quantitative objective.

Omitted proofs can be found in the Appendix section.

2 Quantitative Languages and Functionality

Let Σ be a finite alphabet. We denote by Σ^+ the set of non-empty finite words over Σ. A *quantitative language* L over Σ is a mapping $L : \Sigma^+ \to \mathbb{Q} \cup \{\bot\}$[2]. For all $w \in \Sigma^+$, $L(w)$ is called the *value* of w. $L(w) = \bot$ means that the value of w is undefined. For all $x \in \mathbb{Q}$, we let $\max(x, \bot) = x$, $\max(\bot, x) = x$ and $\max(\bot, \bot) = \bot$.

Let $n \geq 0$. Given a finite sequence $v = v_0 \dots v_n$ of integers (resp. a finite sequence $v' = (r_0, c_0) \dots (r_n, c_n)$ of pairs of natural numbers, $c_i > 0$ for all i) and $\lambda \in \mathbb{Q}$ such that $0 < \lambda < 1$, we define the following functions:

$$\mathsf{Sum}(v) = \sum_{i=0}^{n} v_i \quad \mathsf{Avg}(v) = \frac{\mathsf{Sum}(v)}{n} \quad \mathsf{Dsum}_\lambda(v) = \sum_{i=0}^{n} \lambda^i v_i \quad \mathsf{Ratio}(v') = \frac{\sum_{i=0}^{n} r_i}{\sum_{i=0}^{n} c_i}$$

For empty sequences ϵ, we also set $\mathsf{Sum}(\epsilon) = \mathsf{Avg}(\epsilon) = \mathsf{Dsum}_\lambda(\epsilon) = \mathsf{Ratio}(\epsilon) = 0$.

Weighted Automata Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. A *weighted V-automaton* over Σ is a tuple $A = (Q, q_I, F, \delta, \gamma)$ where Q is a finite set of states, F is a set of final states, $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation, and $\gamma : \delta \to \mathbb{Z}$ (resp. $\gamma : \delta \to \mathbb{N} \times (\mathbb{N} - 0)$ if $V = \mathsf{Ratio}$) is a *weight function*.

[2] As in [10], we do not consider the empty word as our weighted automata do not have initial and final weight functions. This eases our presentation but all our results carry over to the more general setting with initial and final weight function [20].

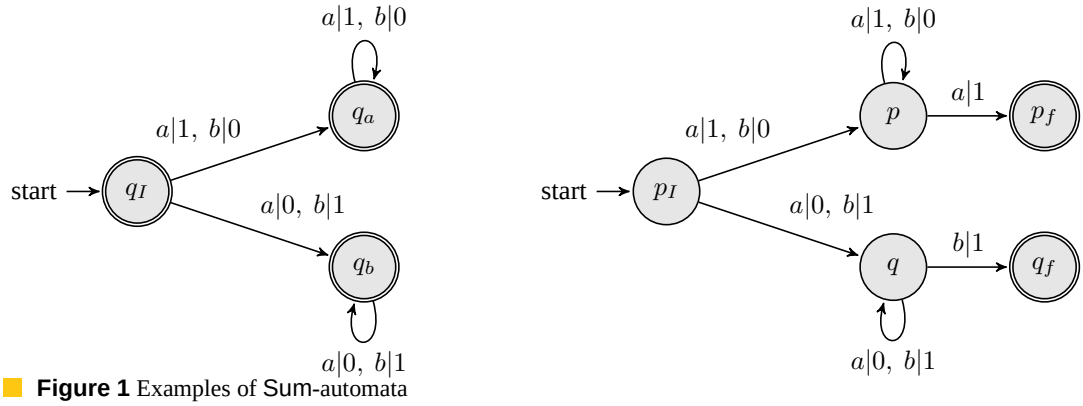

Figure 1 Examples of Sum-automata

The size of A is defined by $|A| = |Q| + |\delta| + \sum_{t\in\delta} log_2(\gamma(t))$. Note that (Q, q_I, F, δ) is a classical finite state automaton. We say that A is *deterministic* (resp. unambiguous) if (Q, q_I, F, δ) is.

A run ρ of A over a word $w = \sigma_1 \dots \sigma_n \in \Sigma^*$ is a sequence $\rho = q_0\sigma_1 q_1\sigma_2 \dots \sigma_n q_n$ such that $q_0 = q_I$ and for all $i \in \{0, \dots, n-1\}$, $(q_i, \sigma_{i+1}, q_{i+1}) \in \delta$. It is *accepting* if $q_n \in F$. We write $\rho : q_0 \xrightarrow{w} q_n$ to denote that ρ is a run on w starting at q_0 and ending in q_n. The domain of A, denoted by $\mathrm{dom}(A)$, is defined as the set of words $w \in \Sigma^+$ on which there exists some accepting run of A.

The function V is naturally extended to runs as follows:

$$V(\rho) = \begin{cases} V(\gamma(q_0, \sigma_1, q_1) \dots \gamma(q_{n-1}, \sigma_n, q_n)) & \text{if } \rho \text{ is accepting} \\ \bot & \text{otherwise} \end{cases}$$

The *relation induced by* A is defined by $R_A^V = \{(w, V(\rho)) \mid w \in \Sigma^+, \rho \text{ is a accepting run of } A \text{ on } w\}$. It is *functional* if for all words $w \in \Sigma^+$, we have $|\{v \mid (w, v) \in R_A^V,\ v \neq \bot\}| \leq 1$. In that case we say that A is functional. The *quantitative language* $L_A : \Sigma^+ \to \mathbb{Q} \cup \{\bot\}$ defined by A is defined by $L_A : w \mapsto \max\{v \mid (w, v) \in R_A^V\}$.

▶ **Example 1.** Fig. 1 illustrates two Sum-automata over the alphabet $\{a, b\}$. The first automaton (on the left) defines the quantitative language $w \in \Sigma^+ \mapsto \max(\#_a(w), \#_b(w))$, where $\#_\alpha(w)$ denotes the number of occurences of the letter α in w. Its induced relation is $\{(w, \#_a(w)) \mid w \in \Sigma^+\} \cup \{(w, \#_b(w)) \mid w \in \Sigma^+\}$. The second automaton (on the right) defines the quantitative language that maps any word of length at least 2 to the number of occurences of its last letter.

We say that a state q is *co-accessible* (resp. *accessible*) by some word $w \in \Sigma^*$ if there exists some run $\rho : q \xrightarrow{w} q_f$ for some $q_f \in F$ (resp. some run $\rho : q_I \xrightarrow{w} q$). If such a word exists, we say that q is co-accessible (resp. accessible). A pair of states (q, q') is co-accessible if there exists a word w such that q and q' are co-accessible by w. In the sequel, we use the term V-automata to denote either Sum, Dsum_λ, Avg or Ratio-automata.

Functional Weighted Automata The Sum-automaton on the left of Fig. 1 is not functional (e.g. the word abb maps to the values 1 and 2), while the one of the right is functional (and even unambiguous).

Concerning the expressiveness of functional automata, we can show that deterministic automata are strictly less expressive than functional automata which are again strictly less expressive than non-deterministic automata.

▶ **Lemma 2.** *Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. The following hold:*
Deterministic < Functional *There exists a functional V-automaton that cannot be defined by any deterministic V-automaton;*
Functional < Non-deterministic *There exists a non-deterministic V-automaton that cannot be defined by any functional V-automaton.*

Proof. Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. The automata of Fig. 1 can be seen as V-automata (with a constant cost 1 if $V = \mathsf{Ratio}$). The right V-automaton cannot be expressed by any deterministic V-automaton because the value of a word depends on its last letter. The left V-automaton cannot be expressed by any functional V-automaton. ◀

As proved in Appendix, functional V-automata are equally expressive as unambiguous V-automata (i.e. at most one accepting run per input word). However we inherit the succinctness property of non-deterministic finite state automata wrt unambiguous finite state automata, as a direct consequence functional V-automata are exponentially more succinct than unambiguous V-automata. Moreover, considering unambiguous V-automata does not simplify the proofs of our results neither lower the computational complexity of the decision problems. Finally, testing functionality often relies on a notion of delay that gives strong insights that are useful for determinization procedures, and will allow us to test equivalence of functional (and even unambiguous) Ratio-automata with a better complexity than using our results on inclusion.

3 Functionality

In this section, we show that it is decidable whether a V-automaton $A = (Q, q_I, F, \delta, \gamma)$ is functional for all $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$.

3.1 Functionality of Sum and Avg-Automata

It is clear that a Sum-automaton A is functional iff the Avg-automaton A is functional. Indeed, let $w \in \mathrm{dom}(A)$ and $v_1, v_2 \in \mathbb{Z}$. We have $(w, v_1), (w, v_2) \in R_A^{\mathsf{Sum}}$ iff $(w, \frac{v_1}{|w|}), (w, \frac{v_2}{|w|}) \in R_A^{\mathsf{Avg}}$. The result follows as $v_1 \neq v_2$ iff $\frac{v_1}{|w|} \neq \frac{v_2}{|w|}$. So we can rephrase the following result of [16]:

▶ **Theorem 3** ([16]). *Functionality is decidable in* PTime *for* Sum *and* Avg*-automata.*

The algorithm of [16] for checking functionality of Sum-automata is based on the notion of *delay* between two runs. This notion has been first introduced for deciding functionality of finite state (word) transducers [3]. Let $w \in \Sigma^+$ and ρ, ρ' be two runs of a Sum-automaton A on w. The delay between ρ and ρ' is defined as $\mathsf{delay}(\rho, \rho') = \mathsf{Sum}(\rho) - \mathsf{Sum}(\rho')$. For all pairs (p, q), we define $\mathsf{delay}(p, q)$ as the set of delays $\mathsf{delay}(p, q) = \{\mathsf{delay}(\rho, \rho') \mid \exists w \in \Sigma^* \cdot \rho : q_I \xrightarrow{w} p,\ \rho' : q_I \xrightarrow{w} q\}$.

It is proved in [16] that a Sum-automaton A is functional iff for all co-accessible pairs of states (p, q), $|\mathsf{delay}(p, q)| \leq 1$. Intuitively, if A is functional, then any delay $\mathsf{delay}(p, q)$ associated with a pair (p, q) co-accessible with the (same) word w has to be recovered when reading w. If there are at least two different delays associated with (p, q), one of them cannot be recovered when reading the same word w, therefore A is not functional. The algorithm then consists first in computing all co-accessible pairs of states, and then all the triples (p, q, k) in a forward manner, where k represents some delay of (p, q). If two triples (p, q, k) and (p, q, k') with $k \neq k'$ are reached, then A is not functional. Termination is obtained by a small witness property for non-functionality, which ensures that the triples need to be visited at most twice. A similar algorithm with another notion of delay is used for deciding functionality of Dsum_λ-automata.

3.2 Functionality of Dsum_λ-automata

▶ **Definition 4** (Dsum_λ Delay). Let $p, q \in Q$ and $d \in \mathbb{Q}$. The rational d is a *delay* for (p, q) if A admits two runs $\rho : q_I \xrightarrow{w} p$, $\rho' : q_I \xrightarrow{w} q$ on $w \in \Sigma^*$ such that

$$\mathsf{delay}(\rho, \rho') =_{def} \frac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho')}{\lambda^{|w|}} = d$$

As for Sum-automata, at most one delay can be associated with co-accessible pairs of states of functional Dsum_λ automata.

▶ **Lemma 5** (One Delay). *Let $A = (Q, q_I, F, \delta, \gamma)$ be a functional Dsum_λ-automaton. For all pairs of states (p, q): If (p, q) is co-accessible, then (p, q) admits at most one delay.*

We now define an algorithm that checks whether a Dsum_λ-automaton is functional. In a first step, it computes all co-accessible pairs of states. Then it explores the set of accessible pairs of states in a forward manner and computes the delays associated with those pairs. If two different delays are associated with the same pair, or if a pair of final states with a non-zero delay is reached, it stops and returns that the automaton is not functional, otherwise it goes on until all co-accessible (and accessible) pairs have been visited and concludes that the automaton is functional.

Algorithm 1: Functionality test for Dsum_λ-automata. (DSumFunTest)

Data: Dsum_λ-automaton $A = (Q, q_I, F, \delta, \gamma)$.
Result: Boolean certifying whether A is functional.
begin
1 CoAcc $\leftarrow$ all co-accessible pairs of states;
2 visited $\leftarrow \emptyset$; delay$(q_I, q_I) \leftarrow 0$; PUSH$(S, ((q_I, q_I), 0))$;
3 **while** $S \neq \emptyset$ **do**
4 $((p, q), d) \leftarrow$ POP(S);
5 **if** $(p, q) \in F^2 \wedge d \neq 0$ **then returns** No;
6 **if** $(p, q) \in$ visited **then**
if delay$(p, q) \neq d$ **then returns** No
else
7 visited $\leftarrow$ visited $\cup \{(p, q)\}$;
8 delay$(p, q) \leftarrow d$;
9 **foreach** $(p', q') \in$ CoAcc *s.t.* $\exists a \in \Sigma \cdot (p, a, p') \in \delta \wedge (q, a, q') \in \delta$ **do**
PUSH$(S, ((p', q'), \gamma(p, a, p') - \gamma(q, a, q') + d))$;
10 **returns** Yes

▶ **Lemma 6.** *Let $A = (Q, q_I, F, \delta, \gamma)$ be a Dsum_λ-automaton. If A is not functional, there exists a word $w = \sigma_0 \dots \sigma_n$ and two accepting runs $\rho = q_0\sigma_0 \dots q_n, \rho' = q'_0\sigma_0 \dots q'_n$ on it such that $\mathsf{Dsum}_\lambda(\rho) \neq \mathsf{Dsum}_\lambda(\rho')$ and for all positions $i < j$ in w, either (i) $(p_i, q_i) \neq (p_j, q_j)$ or (ii) $\mathsf{delay}(\rho_i, \rho'_i) \neq \mathsf{delay}(\rho_j, \rho'_j)$, where ρ_i and ρ'_i (resp. ρ_j and ρ'_j) denote the prefixes of the runs ρ and ρ' until position i (resp. position j).*

We can now prove the correctness of Algorithm DSumFunTest.

▶ **Theorem 7.** *Given a Dsum_λ-automaton A, Algorithm* DSumFunTest *applied to A returns* YES *iff A is functional and terminates within $O(|A|^2)$ steps.*

Sketch, full proof in Appendix. If DSumFunTest(A) returns NO, it is either because a pair of accepting states with non-null delay has been reached, which gives a counter-example to functionality, or it finds a pair of states with two different delays, so A is not functional by Lemma 5.

Conversely, if A is non-functional, by Lemma 6, there exists a word w with two accepting runs having different values such that either no pair of states is repeated twice, in which case the algorithm can find a pair of final states with a non-null delay, or there is a pair of states that repeat twice (take the first that repeat) and has necessarily two different delays, in which case the algorithm will return NO at line 6, if not before. ◀

3.3 Functionality of Ratio-automata

Unlike Sum,Avg or Dsum_λ-automata, it is still open whether there exists a good notion of delay for Ratio-automata that would allow us to design an efficient algorithm to test functionality. However deciding functionality can be done by using a short witness property of non-functionality.

▶ **Lemma 8** (Pumping). *Let A be a* Ratio*-automaton with n states. A is not functional iff there exist a word w such that $|w| < 4n^2$ and two accepting runs ρ, ρ' on w such that* $\mathsf{Ratio}(\rho) \neq \mathsf{Ratio}(\rho')$.

Proof. We prove the existence of a short witness for non-functionality. The other direction is obvious. Let w be a word such that $|w| \geq 4n^2$ and there exists two accepting runs ρ_1, ρ_2 on w such that $\mathsf{Ratio}(\rho) \neq \mathsf{Ratio}(\rho')$. Since $|w| \geq 4n^2$, there exist states $p, q \in Q, p_f, q_f \in F$ and words w_0, w_1, w_2, w_3, w_4 such that $w = w_0w_1w_2w_3w_4$ and ρ, ρ' can be decomposed as follows:

$$\begin{array}{lccccccccccc} \rho: & q_I & \xrightarrow{w_0|(r_0,c_0)} & p & \xrightarrow{w_1|(r_1,c_1)} & p & \xrightarrow{w_2|(r_2,c_2)} & p & \xrightarrow{w_3|(r_3,c_3)} & p & \xrightarrow{w_4|(r_4,c_4)} & p_f \\ \rho': & q_I & \xrightarrow{w_0|(r'_0,c'_0)} & q & \xrightarrow{w_1|(r'_1,c'_1)} & q & \xrightarrow{w_2|(r'_2,c'_2)} & q & \xrightarrow{w_3|(r'_3,c'_3)} & q & \xrightarrow{w_4|(r'_4,c'_4)} & q_f \end{array}$$

where r_i, c_i denotes the sum of the rewards and the costs respectively on the subruns of ρ on w_i, and similarly for r'_i, c'_i.

By hypothesis we know that $(\sum_{i=0}^4 r_i) \cdot (\sum_{i=0}^4 c'_i) \neq (\sum_{i=0}^4 c_i) \cdot (\sum_{i=0}^4 r'_i)$. For all subsets $X \subseteq \{1,2,3\}$, we denote by w_X the word $w_0w_{i_1} \dots w_{i_k}w_4$ if $X = \{i_1 < \dots < i_k\}$. For instance, $w_{\{1,2,3\}} = w$, $w_{\{1\}} = w_0w_1w_4$ and $w_{\{\}} = w_0w_4$. Similarly, we denote by ρ_X, ρ'_X the corresponding runs on w_X. We will show that there exists $X \subsetneq \{1,2,3\}$ such that $\mathsf{Ratio}(\rho_X) \neq \mathsf{Ratio}(\rho'_X)$. Suppose that for all $X \subsetneq \{1,2,3\}$, we have $\mathsf{Ratio}(\rho_X) = \mathsf{Ratio}(\rho'_X)$. We now show that it implies that $\mathsf{Ratio}(\rho) = \mathsf{Ratio}(\rho')$, which contradicts the hypothesis. For all $X \subseteq \{1,2,3\}$, we let:

$$L_X = (\sum_{i \in X \cup \{0,4\}} r_i) \cdot (\sum_{i \in X \cup \{0,4\}} c'_i) \qquad R_X = (\sum_{i \in X \cup \{0,4\}} c_i) \cdot (\sum_{i \in X \cup \{0,4\}} r'_i)$$

By hypothesis, $L_{\{1,2,3\}} \neq R_{\{1,2,3\}}$ and for all $X \subsetneq \{1,2,3\}$, $L_X = R_X$. We now prove the following equalities:

$$\begin{array}{ccccccccccccc} L_{\{\}} & + & L_{\{1,2\}} & + & L_{\{1,3\}} & + & L_{\{2,3\}} & - & L_{\{1\}} & - & L_{\{2\}} & - & L_{\{3\}} & = & L_{\{1,2,3\}} \\ R_{\{\}} & + & R_{\{1,2\}} & + & R_{\{1,3\}} & + & R_{\{2,3\}} & - & R_{\{1\}} & - & R_{\{2\}} & - & R_{\{3\}} & = & R_{\{1,2,3\}} \end{array}$$

We only prove the equality with the L values as it is symmetric for the R values. For all $i, j \in \{0,4\}$, the subterm $r_ic'_j$ occurs once in all expressions L_X, and $1+1+1+1-1-1-1-1=1$. For all $i, j \in \{1,2,3\}$ such that $i \neq j$, the subterm $r_ic'_j$ appears once in $L_{\{i,j\}}$ and once in $L_{\{1,2,3\}}$. For all $i \in \{1,2,3\}$, the subterm $r_ic'_i$ appears once in all L_X such that $i \in X$, and there are exactly two such L_X that are added to the left of the equation, one that is substracted to the left, and one added to the right. For instance, the subterm $r_1c'_1$ appears in $L_{\{1,2,3\}}, L_{\{1,2\}}, L_{\{1,3\}}$ and $L_{\{1\}}$. It can be checked similarly that on the left of the equation, the coefficients for all other subterms are 1.

Therefore, since by hypothesis we have $L_X = R_X$ for all $X \subsetneq \{1,2,3\}$, we get $L_{\{1,2,3\}} = R_{\{1,2,3\}}$, which is a contradiction. Thus there exists $X \subsetneq \{1,2,3\}$ such that $L_X \neq R_X$. In other words, there exists $X \subsetneq \{1,2,3\}$ such that $\mathsf{Ratio}(\rho_X) \neq \mathsf{Ratio}(\rho'_X)$. This shows that when a witness of non-functionality has length at least $4n^2$, we can find a strictly smaller witness of functionality. This achieves to prove the lemma. ◀

As a consequence, we can design a non-deterministic PSpace procedure that will check non-functionality by guessing runs of length at most $4n^2$, where n is the number of states:

▶ **Theorem 9.** *Functionality is decidable in* PSpace *for* Ratio*-automata, and in* NLogSpace *if the weights are encoded in unary.*

▶ Remark. The pumping lemma states that if some Ratio-automaton with n states is not functional, there exists a witness of non-functionality whose length is bounded by $4n^2$, where n is the number of states. Such a property also holds for Dsum$_\lambda$-automata (and is well-known for Sum and Avg-automata), but with the smaller bound $3n^2$. Those bounds are used to state the existence of two runs on the same word such that the same pair of states is repeated 3 or 4 times along the two runs. Then it is proved that one can remove some part in between two repetitions and get a smaller word with two different output values. However for Ratio-automata, three repetitions are not enough to be able to shorten non-functionality witnesses. For instance, consider the following two runs on the alphabet $\{a, b, c, d\}$ and states $\{q_I, p, q, p_f, q_f\}$ where p_f, q_f are final (those two runs can easily be realized by some Ratio-automaton):

$$\begin{array}{lccccccccc} \rho: & q_I & \xrightarrow{a|(2,2)} & p & \xrightarrow{b|(2,1)} & p & \xrightarrow{c|(2,2)} & p & \xrightarrow{d|(1,1)} & p_f \\ \rho': & q_I & \xrightarrow{a|(1,2)} & q & \xrightarrow{b|(2,1)} & q & \xrightarrow{c|(1,1)} & q & \xrightarrow{d|(2,1)} & q_f \end{array}$$

It is easy to verify that the word $abcd$ has two outputs given by ρ and ρ' while the words ad, abd and acd has one output. For instance, the two runs $q_I \xrightarrow{a|(2,2)} p \xrightarrow{d|(1,1)} p_f$ and $q_I \xrightarrow{a|(1,2)} q \xrightarrow{d|(2,1)} q_f$ on ad have both value 1.

4 Decision Problems

In this section we investigate several decision problems for functional V-automata as defined in [10], $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. Given two V-automata A, B over Σ (and with the same discount factor when $V = \mathsf{Dsum}_\lambda$) and a threshold $\nu \in \mathbb{Q}$, we define the following decision problems:

Inclusion	$L_A \leq L_B$	holds if for all $w \in \Sigma^+$, $L_A(w) \leq L_B(w)$
Equivalence	$L_A = L_B$	holds if for all $w \in \Sigma^+$, $L_A(w) = L_B(w)$
$\sim \nu$-Emptiness	$L_A^{\sim\nu} \neq \varnothing$	holds if there exists $w \in \Sigma^+$ such that $L_A(w) \sim \nu$, $\sim \in \{>, \geq\}$
$\sim \nu$-Universality	$\nu \sim L_A$	holds if for all $w \in \mathrm{dom}(A)$, $L_A(w) \sim \nu$, where $\sim \in \{>, \geq\}$.

It is known that inclusion is undecidable for non-deterministic Sum-automata [18], and therefore is also undecidable for Avg and Ratio-automata. To the best of our knowledge, it is open whether it is decidable for Dsum$_\lambda$-automata.

▶ **Theorem 10.** *Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$ and let A, B be two V-automata such that B is functional. The inclusion problem $L_A \leq L_B$ is decidable. If $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda\}$ then it is* PSpace-c *and if additionnaly B is deterministic, it is in* PTime.

Proof. Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda\}$. In a first step, we test the inclusion of the domains $\mathrm{dom}(A) \subseteq \mathrm{dom}(B)$ (it is PSpace-c and in PTime if B is deterministic). Then we construct the product $A \times B$ as follows: $(p, q) \xrightarrow{a|n_A - n_B} (p', q') \in \delta_{A\times B}$ iff $p \xrightarrow{a|n_A} p' \in \delta_A$ and $q \xrightarrow{a|n_B} q' \in \delta_B$. Then $L_A \not\leq L_B$ iff there exists a path in $A \times B$ from a pair of initial states to a pair of accepting states with strictly positive sum if $V \in \{\mathsf{Sum}, \mathsf{Avg}\}$, and with strictly positive discounted sum if $V = \mathsf{Dsum}_\lambda$. This can be checked in PTime for all those three measures, with shortest path algorithms for Sum and Avg, and as a consequence of a result of [2] about single player discounted games, for Dsum$_\lambda$.

Let $V = \mathsf{Ratio}$. As for the other measures we first check inclusion of the domains. Then let $\delta_A = \{x_1, \ldots, x_n\}$ and $\delta_B = \{y_1, \ldots, y_m\}$. Let $r_A = (r_1, \ldots, r_n)$ be the rewards associated with the transitions $x_1, \ldots, x_n$ respectively. Similarly, let $c_A = (c_1, \ldots, c_n)$ be the costs associated with $x_1, \ldots, x_n$. The vectors r_B and c_B are defined similarly.

We define the product $A \times B$ of A and B similarly as before, except that the values of transitions are quadruples (r_1, c_1, r_2, c_2) of rewards and costs of A and B respectively. Let $\delta_{A\times B}$ denotes the transitions of $A \times B$. It is clear by construction of $A \times B$ that any transition $t \in \delta_{A\times B}$ can be associated with a unique pair of transitions in $\delta_A \times \delta_B$, denoted by $(\alpha_A(t), \alpha_B(t))$.

Given a vector $a = (a_1, \ldots, a_n) \in \mathbb{N}^n$ and a vector $b = (b_1, \ldots, b_m) \in \mathbb{N}^m$, we say that the pair (a, b) fits $A \times B$ if there exists $w \in \Sigma^+$ and an accepting run ρ of $A \times B$ on w such that for all transitions $x_i \in \delta_A$ (resp. $y_j \in \delta_B$), ρ visits the transitions $t \in \delta_{A\times B}$ such that $\alpha_A(t) = x_i$ (resp. $\alpha_B(t) = y_j$) exactly a_i times (resp. b_j times). In other words, if n_t denotes the number of times a transition $t \in \delta_{A\times B}$ is visited by ρ, we require that for all transitions $x_i \in \delta_A$ and all transitions $y_j \in \delta_B$, $a_i = \sum\{n_t \mid t \in \delta_{A\times B}, \alpha_A(t) = x_i\}$ and $b_j = \sum\{n_t \mid t \in \delta_{A\times B}, \alpha_B(t) = y_j\}$. We denote by $F(A \times B)$ the set of pairs (a, b) fitting $A \times B$. By using Parikh's theorem, it is easy to show that $F(A \times B)$ is a semi-linear set which can be effectively represented as the solutions of a system of linear equations over natural numbers. We finally define the set Γ as follows:

$$\Gamma = \{(a, b) \mid (a, b) \in F(A \times B), a.r_A \cdot (b.c_B)^T > a.c_A \cdot (b.r_B)^T\}$$

where $.$ denotes the pairwise multiplication, $\cdot$ the matrix multiplication, and $.^T$ the transposite. It is easy to check that $\Gamma \neq \varnothing$ iff $L_A \not\leq L_B$. The set Γ can be defined as the solutions over natural numbers of a system of equations in linear and quadratic forms (i.e. in which products of two variables are permitted). It is decidable whether such a system has a solution [25, 14]. ◀

There is no known complexity bound for solving quadratic equations, so the proof above does not give us a complexity bound for the inclusion problem of functional Ratio-automata. However, thanks to the functionality test, which is in PSpace for Ratio-automata, we can test equivalence of two functional Ratio-automata A_1 and A_2 in PSpace: first check in PSpace that $\mathrm{dom}(A_1) = \mathrm{dom}(A_2)$ and check that the union of A_1 and A_2 is functional. This algorithm can also be used for the other measures:

▶ **Theorem 11.** *Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. Equivalence of functional V-automata is* PSpace-c.

▶ **Theorem 12.** *Let $\nu \in \mathbb{Q}$. The $> \nu$-emptiness (resp. $\geq \nu$-emptiness) problem is in* PTime *for* Sum-, Avg-, Ratio-, *and* Dsum_λ*-automata (resp.* Sum-, Avg-, *and* Ratio*-automata).*

It is open how to decide $\geq \nu$ for Dsum_λ-automata. Dually:

▶ **Theorem 13.** *Let $\nu \in \mathbb{Q}$. The $\geq \nu$-universality (resp. $> \nu$-universality) problem is* PSpace-c *for* Sum-, Avg-, Ratio-, *and* Dsum_λ*-automata (resp.* Sum-, Avg-, *and* Ratio*-automata).*

5 Realizability

In this section, we consider the problem of *quantitative language realizability*. The realizability problem is better understood as a game between two players: the 'Player input' (the environment, also called Player I) and the 'Player output' (the controller, also called Player O). Player I (resp. Player O) controls the letters of a finite alphabet Σ_I (resp. Σ_O). We assume that $\Sigma_O \cap \Sigma_I = \varnothing$ and that Σ_O contains a special symbol $\#$ whose role is to stop the game. We let $\Sigma = \Sigma_O \cup \Sigma_I$.

Formally, the realizability game is a turn-based game played on an arena defined by a weighted automaton $A = (Q = Q_O \uplus Q_I, q_0, F, \delta = \delta_I \cup \delta_O, \gamma)$, whose set of states is partitioned into two sets, $\delta_O \subseteq Q_O \times \Sigma_O \times Q_I$, $\delta_I \subseteq Q_I \times \Sigma_I \times Q_O$, and such that $\mathrm{dom}(A) \subseteq (\Sigma \backslash \{\#\})^*\#$. Player O starts by giving an initial letter $o_0 \in \Sigma_O$, Player I responds providing a letter $i_0 \in \Sigma_I$, then Player O gives o_1 and Player I responds i_1, and so on. Player O has also the power to stop the game at any turn with the distinguishing symbol $\#$. In this case, the game results in a finite word $(o_0 i_0)(o_1 i_1) \ldots (o_j i_j)\# \in \Sigma^*$, otherwise the outcome of the game is an infinite word $(o_0 i_0)(o_1 i_1) \cdots \in \Sigma^\omega$.

The players play according to strategies. A strategy for Player O (resp. Player I) is a mapping $\lambda_O : (\Sigma_O \Sigma_I)^* \to \Sigma_O$ (resp. $\lambda_I : \Sigma_O (\Sigma_I \Sigma_O)^* \to \Sigma_I$). The outcome of the strategies λ_O, λ_I is the word $w = o_0 i_0 o_1 i_1 \ldots$ denoted by $\mathsf{outcome}(\lambda_O, \lambda_I)$ such that for all $0 \leq j \leq |w|$ (where

$|w| = +\infty$ if w is infinite), $o_j = \lambda_O(o_0 i_0 \dots i_{j-1})$ and $i_j = \lambda(o_0 i_0 \dots o_j)$, and such that if $\# = o_j$ for some j, then $w = o_0 i_0 \dots o_j$. We denote by Λ_O (resp. Λ_I) the set of strategies for Player O (resp. Player I).

A strategy $\lambda_O \in \Lambda_O$ is winning for Player O if for all strategies $\lambda_I \in \Lambda_I$, $\mathsf{outcome}(\lambda_O, \lambda_I)$ is finite and $L_A(\mathsf{outcome}(\lambda_O, \lambda_I)) > 0$. The *quantitative language realizability problem* for the weighted automaton A asks whether Player O has a winning strategy and in that case, we say that A is *realizable*.

Our first result on realizability is negative: we show that it is undecidable for weighted functional Sum-, Avg-automata, and Ratio-automata. In particular, we show that the halting problem for deterministic 2-counter Minsky machines [19] can be reduced to the quantitative language realizability problem for (functional) Sum-automata (resp. Avg-automata).

▶ **Theorem 14.** *Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Ratio}\}$. The realizability problem for functional weighted V-automata is undecidable.*

The proof of Theorem 14 (in Appendix) relies on the use of a nondeterministic weighted automaton. Indeed, as stated in the next theorem, the quantitative language realizability problem is decidable for the four measures when the automaton is deterministic, in $\mathsf{NP} \cap \mathsf{coNP}$ (see Appendix), though memoryfull strategies are necessary for winning those games.

▶ **Theorem 15.** *The quantitative language realizability problem for deterministic weighted V-automata, $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$, is in $\mathsf{NP} \cap \mathsf{coNP}$.*

6 Determinization

A V-automaton $A = (Q, q_I, F, \delta, \gamma)$ is *determinizable* if it is effectively equivalent to a deterministic V-automaton[3]. V-automata are not determinizable in general. For example, consider the right automaton on Fig. 1. Seen as a Sum, Avg or Dsum_λ-automaton for any λ, it cannot be determinized, because there are infinitely many delays associated with the pair of states (p, q). Those delays can for instance be obtained by the family of words of the form a^n.

We show that it can be decided whether a functional V-automaton is determinizable for $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda\}$. However, it is still open for Ratio-automata, for which we do not have an adequate notion of delay.

To ease notations, for all V-automaton A over an alphabet Σ, we assume that there exists a special ending symbol $\# \in \Sigma$ such that any word $w \in \mathrm{dom}(A)$ is of the form $w'\#$ with $w' \in (\Sigma - \#)^*$.

Determinizability is already known to be decidable in PTime for functional Sum-automata [16][4]. Determinizable functional Sum-automata are characterized by the so called *twinning property*, that has been introduced for finite word transducers [11]. Two states p, q are *twinned* if both p and q are co-accessible and for all words $w_1, w_2 \in \Sigma^*$, for all $n_1, n_2, m_1, m_2 \in \mathbb{Z}$, if $q_I \xrightarrow{w_1|n_1} p \xrightarrow{w_2|n_2} p$ and $q_I \xrightarrow{w_1|m_1} q \xrightarrow{w_2|m_2} q$, then $n_2 = m_2$. In other words, the delays between the two runs cannot increase on the loop. If all pairs of states are twinned, then it is proved that the number of different accumulated delays on parallel runs is finite. The determinization for Sum-automata extends the classical determinization procedure of finite automata with delays. States are (partial) functions from states to delays. Clearly, a Sum-automaton A is determinizable iff the Avg-automaton A is determinizable. We can even use exactly the same determinization procedure as for Sum-automata.

[3] With the existence of an ending symbol, the notion of determinizability corresponds to the notion of subsequentializability [11].

[4] See [17, 15] for determinizability results on more general classes of Sum-automata.

▶ **Theorem 16** ([16]). *It is decidable in* PTime *whether a functional* Sum *or* Avg*-automaton is determinizable.*

We now explain the determinization procedure for Dsum_λ-automata.

▶ **Definition 17.** We say that two states p, q are *twinned* if both p and q are co-accessible and for all words $w_1, w_2 \in \Sigma^*$, for all runs $\rho_1 : q_I \xrightarrow{w_1} p$, $\rho_2 : p \xrightarrow{w_2} p$, $\rho_1' : q_I \xrightarrow{w_1} q$, $\rho_2' : q \xrightarrow{w_2} q$, we have $\mathsf{delay}(\rho_1, \rho_1') = \mathsf{delay}(\rho_1\rho_2, \rho_1'\rho_2')$.

A Dsum_λ-automaton A satisfies the *twinning property* if all pairs of states are twinned. We show that the twinning property is a decidable characterization of determinizable functional Dsum_λ-automata. First, we prove that it is decidable in PSpace:

▶ **Lemma 18.** *Is it decidable in* PSpace *whether a* Dsum_λ*-automaton satisfies the twinning property.*

We denote by $\mathcal{D}$ the set of possible delays between two runs of A, i.e. $\mathcal{D}$ is the set of delays $\mathsf{delay}(\rho, \rho')$ for all runs ρ, ρ' on the same input word, such that the last states of ρ and ρ' are both co-accessible.

▶ **Lemma 19.** *If the twinning property holds, then $\mathcal{D}$ is finite of size at most $|\Sigma|^{|Q|^2}$.*

Proof. As delays must be identical on parallel loops, any delay can be obtained with some pair of runs of length $|Q|^2$ at most (on longer pairs of runs, there must exist a parallel loop with identical delays that can be removed without affecting the value of the global delay of both runs, see Lemma 24 of the Appendix). ◀

Determinization Assume that the twinning property holds. We define a determinization procedure that constructs from a functional Dsum_λ-automaton $A = (Q, q_I, F, \delta, \gamma)$ a deterministic Dsum_λ-automaton $A_d = (Q_d, f_d, F_d, \delta_d, \gamma_d)$. Wlog we assume that all states are co-accessible (otherwise we can remove non co-accessible states in linear time). We define $Q' = \mathcal{D}^Q$ (which is finite by Lemma 19), the set of partial functions from states Q to delays.We let $f_I' : q_I \mapsto 0$ and F' is defined as $\{f \in Q' \mid \mathrm{dom}(f) \cap F \neq \varnothing\}$. Then, given partial functions $f, f' \in Q'$ and a symbol $a \in \Sigma$, we let:

$$\begin{array}{lcl} \gamma'(f, a, f') & = & \min\{\dfrac{f(q)}{\lambda} + \gamma(q, a, q') \mid q \in \mathrm{dom}(f) \wedge (q, a, q') \in \delta\} \\ (f, a, f') \in \delta' & \text{iff} & \text{for all } q' \in \mathrm{dom}(f') \text{ there exists } q \in \mathrm{dom}(f) \text{ such that } (q, a, q') \in \delta \text{ and} \\ & & f'(q') = \dfrac{f(q)}{\lambda} + \gamma(q, a, q') - \gamma'(f, a, f') \end{array}$$

Let $Q_d \subseteq Q'$ be the accessible states of $A' := (Q', f_I', F', \delta', \gamma')$. We define $A_d = (Q_d, f_d, \delta_d, \gamma_d)$ as the restriction of A' to the accessible states.

▶ **Lemma 20.** *If the twinning property holds, A_d and A are equivalent, A_d is deterministic and has $O(|\Sigma|^{|Q|^3})$ states.*

The proof is based on the following lemma:

▶ **Lemma 21.** *Let $f \in Q_d$ be state of A_d accessible by a run ρ_d on some word $w \in \Sigma^*$. Then $dom(f)$ is the set of states q such that there exists a run on w reaching q. Moreover, if $q \in dom(f)$ and ρ is a run on w reaching q, then $f(q) = \max\{\mathsf{delay}(\rho, \rho') \mid \rho' \text{ is a run of } A \text{ on } w\} = \dfrac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho_d)}{\lambda^{|w|}}$ and $\mathsf{Dsum}_\lambda(\rho_d) = \min\{\mathsf{Dsum}_\lambda(\xi) \mid \xi \text{ is a run of } A \text{ on } w\}$.*

If the twinning property does not hold, we show that $\mathcal{D}$ is infinite and that A cannot be determinized. Therefore we get the following theorem:

▶ **Theorem 22.** *A functional* Dsum_λ*-automaton is determinizable iff it satisfies the twinning property. Therefore determinizability is decidable in* PSpace *for functional* Dsum_λ*-automata.*

References

1 Benjamin Aminof, Orna Kupferman, and Robby Lampert. Rigorous approximated determinization of weighted automata. In *LICS*, pages 345–354, 2011.
2 D Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.
3 Beal, Carton, Prieur, and Sakarovitch. Squaring transducers: An efficient procedure for deciding functionality and sequentiality. *TCS: Theoretical Computer Science*, 292, 2003.
4 Jean Berstel and Christophe Reutenauer. *Rational Series and Their Languages*. Number 12 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1988.
5 Henrik Björklund and Sergei Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Appl. Math.*, 155:210–229, 2007.
6 Meera Blattner and Tom Head. Single-valued a-transducers. *JCSS*, 15(3):310–327, 1977.
7 Roderick Bloem, Karin Greimel, Thomas A. Henzinger, and Barbara Jobstmann. Synthesizing robust systems. In *FMCAD*, pages 85–92. IEEE, 2009.
8 Udi Boker, Krishnendu Chatterjee, Thomas A. Henzinger, and Orna Kupferman. Temporal specifications with accumulative values. In *LICS*, pages 43–52, 2011.
9 Udi Boker and Thomas A. Henzinger. Determinizing discounted-sum automata. In *CSL*, pages 82–96, 2011.
10 Krishnendu Chatterjee, Laurent Doyen, and Thomas A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log*, 11(4), 2010.
11 Christian Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977.
12 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.
13 Manfred Droste and George Rahonis. Weighted automata and weighted logics with discounting. In *CIAA*, pages 73–84, 2007.
14 Fritz Grunewald and Dan Segal. On the integer solutions of quadratic equations. *Journal für die reine und angewandte Mathematik*, 569:13–45, 2004.
15 Daniel Kirsten. A burnside approach to the termination of mohri's algorithm for polynomially ambiguous min-plus-automata. *ITA*, 42(3):553–581, 2008.
16 Daniel Kirsten and Ina Mäurer. On the determinization of weighted automata. *Journal of Automata, Languages and Combinatorics*, 10(2/3):287–312, 2005.
17 Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theor. Comput. Sci.*, 327(3):349–373, 2004.
18 Daniel Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Int. Journal of Algebra and Computation*, 4(3):405–425, 1994.
19 Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, 1967.
20 Mehryar Mohri. Weighted automata algorithms. *Handbook of Weighted Automata*, pages 213–254, 2009.
21 A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *ACM Symposium on Principles of Programming Languages (POPL)*. ACM, 1989.
22 Marcel Paul Schützenberger. Sur les relations rationnelles. In *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 209–213, 1975.
23 Wolfgang Thomas. Church's problem and a tour through automata theory. In *Pillars of Computer Science*, pages 635–655, 2008.
24 Andreas Weber and Reinhard Klemm. Economy of description for single-valued transducers. *Inf. Comput.*, 118(2):327–340, 1995.
25 Karianto Wong, Aloys Krieg, and Wolfgang Thomas. On intersection problems for polynomially generated sets. In *ICALP*, volume 4052 of *LNCS*, pages 516–527. Springer, 2006.

A Quantitative Languages and Functionality

A.1 Functionality and Unambiguity

▶ **Lemma 23.** *Let $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda, \mathsf{Ratio}\}$. For all functional V-automaton with n states we can construct an equivalent unambiguous V-automaton with $O(n.2^n)$ states.*

Proof. Our proof is independent on the measure. Let $A = (Q, q_I, F, \delta, \gamma)$ be a functional V-automaton. We order the transitions of δ by a total order denoted by $<_\delta$. We construct an equivalent unambiguous automaton $A' = (Q', q'_I, F', \delta', \gamma')$, where:

- $Q' = Q \times 2^Q$;
- $q'_i = (q_I, \varnothing)$;
- $F' = F \times \{P \subseteq Q \mid F \cap P = \varnothing\}$;
- $\gamma' \ : \ ((p, P), a, (p', P')) \mapsto \gamma(p, a, p')$;

Before defining δ' formally, let us explain intuitively the semantics of the states in Q'. The automaton A' will guess a run of A on first state component (called the *current run*). A pair (p, P) represents the state p of current run in the original automaton A while P represents the states reached by all the runs that are greater than the current run (for the order $<_\delta$ lexigraphically extended to runs). At the end of the word, the run is accepting iff p is accepting and there is no accepting state in P. In other words, a run of A' on a word w is accepting iff the run it defines on the first component is the smallest accepting run of A on w.

When a new letter $a \in \Sigma$ is read, A' guesses a transition from p to some state p', and goes to the state $(p', S_P \cup S_{p,a,p'})$, where S_P are the successor states of of P by δ on the input a, and $S_{p,a,p'}$ are all the states reached from p by a transition on a bigger than (p, q, p').

Formally, $((p, P), a, (p', P')) \in \delta'$ iff

- $(p, a, p') \in \delta$;
- $P' = \{q' \mid \exists q \in P,\ (q, a, q') \in \delta\} \cup \{p'' \mid (p, a, p'') \in \delta \wedge (p, a, p') <_\delta (p, a, p'')\}$.

It is clear by construction that A and A' defines the same domain. As A is functional, they also define the same function, because the value of a word is equal to the value of any run on it, and in particular to the value of the smallest run. ◀

B Functionality

B.1 Proof of Lemma 5

Proof. Consider a co-accessible pair of states (p, q). Assume that (p, q) admits two delays d_1, d_2. We show that if A is functional, then $d_1 = d_2$. Let $\rho_1 : q_0 \overset{w_1}{\rightsquigarrow} p$, $\rho'_1 : q_0 \overset{w_1}{\rightsquigarrow} q$ (resp. $\rho_2 : q_0 \overset{w_2}{\rightsquigarrow} p$, $\rho'_2 : q_0 \overset{w_2}{\rightsquigarrow} q$) be two runs witnessing the delay d_1 (resp. d_2), i.e.:

$$\frac{\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1)}{\lambda^{|w_1|}} = d_1, \frac{\mathsf{Dsum}_\lambda(\rho_2) - \mathsf{Dsum}_\lambda(\rho'_2)}{\lambda^{|w_2|}} = d_2$$

Since (p, q) is co-accessible, there exists a word u and two runs $\rho_3 : p \overset{u}{\rightsquigarrow} f \in F$, $\rho'_3 : q \overset{u}{\rightsquigarrow} f' \in F$. Moreover, the hypothesis of functionality on A implies:

$$\mathsf{Dsum}_\lambda(\rho_1\rho_3) - \mathsf{Dsum}_\lambda(\rho'_1\rho'_3) = 0 \tag{1}$$

$$\mathsf{Dsum}_\lambda(\rho_2\rho_3) - \mathsf{Dsum}_\lambda(\rho'_2\rho'_3) = 0 \tag{2}$$

Let $v_{1,0} \dots v_{1,m}$ (resp. $v'_{1,0} \dots v'_{1,m}$, $v_{3,0} \dots v_{3,l}$, $v'_{3,0} \dots v'_{3,l}$) the sequence of weights occuring along ρ_1 (resp. ρ'_1, ρ_3, ρ'_3), where $m = |w_1| - 1$ and $l = |u| - 1$.

Then, Equation 1 implies:

$$\sum_{i=0}^{m} v_{1,i}\lambda^i + \lambda^{m+1}\sum_{i=0}^{l} v_{3,i}\lambda^i = \sum_{i=0}^{m} v'_{1,i}\lambda^i + \lambda^{m+1}\sum_{i=0}^{l} v'_{3,i}\lambda^i \tag{3}$$

Let $v_{2,0} \dots v_{2,n}$ (resp. $v'_{2,0} \dots v'_{2,n}$) be the sequence of weights occuring along ρ_2 (resp. ρ'_2), where $n = |w_2| - 1$.

Then, Equation 2 implies:

$$\sum_{i=0}^{n} v_{2,i}\lambda^i + \lambda^{n+1}\sum_{i=0}^{l} v_{3,i}\lambda^i = \sum_{i=0}^{n} v'_{2,i}\lambda^i + \lambda^{n+1}\sum_{i=0}^{l} v'_{3,i}\lambda^i \tag{4}$$

Equations 3 and 4 yield:

$$(\sum_{i=0}^{m} v_{1,i}\lambda^i - \sum_{i=0}^{m} v'_{1,i}\lambda^i) = \lambda^{m+1}(\sum_{i=0}^{l} v'_{3,i}\lambda^i - \sum_{i=0}^{l} v_{3,i}\lambda^i) \tag{5}$$

$$(\sum_{i=0}^{n} v_{2,i}\lambda^i - \sum_{i=0}^{n} v'_{2,i}\lambda^i) = \lambda^{n+1}(\sum_{i=0}^{l} v'_{3,i}\lambda^i - \sum_{i=0}^{l} v_{3,i}\lambda^i) \tag{6}$$

Dividing both the members of Equation 5 by λ^{m+1} and both the members of Equation 6 by λ^{n+1}, and finally subtracting the obtained results, we get our thesis i.e.

$$\frac{\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1)}{\lambda^{|w_1|}} = \frac{\mathsf{Dsum}_\lambda(\rho_2) - \mathsf{Dsum}_\lambda(\rho'_2)}{\lambda^{|w_2|}}$$

◀

B.2 Proof of Lemma 6

We first prove the following key result:

▶ **Lemma 24.** *Let $A = (Q, q_I, F, \delta, \gamma)$ be a Dsum_λ-automaton. Let $w_1, w_2, w_3 \in \Sigma^*$ such that there exist $p, p', q, q' \in Q$ and the following runs:*

$$\begin{array}{lll} \rho_1 : q_I \xrightarrow{w_1} p & \rho_2 : p \xrightarrow{w_2} p & \rho_3 : p \xrightarrow{w_3} q \\ \rho'_1 : q_I \xrightarrow{w_1} p' & \rho_2 : p' \xrightarrow{w_2} p' & \rho_3 : p' \xrightarrow{w_3} q' \end{array}$$

and such that $\mathsf{delay}(\rho_1, \rho'_1) = \mathsf{delay}(\rho_1\rho_2, \rho'_1\rho'_2)$. *Then* $\mathsf{delay}(\rho_1\rho_2\rho_3, \rho'_1\rho'_2\rho'_3) = \mathsf{delay}(\rho_1\rho_3, \rho'_1\rho'_3)$.

Proof. By hypothesis, we have the following equality:

$$\frac{\mathsf{Dsum}_\lambda(\rho_1) + \lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho_2) - \mathsf{Dsum}_\lambda(\rho'_1) - \lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho'_2)}{\lambda^{|w_1|+|w_2|}} = \frac{\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_2)}{\lambda^{|w_1|}} \tag{7}$$

which implies:

$$\mathsf{Dsum}_\lambda(\rho_1)+\lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho_2)-\mathsf{Dsum}_\lambda(\rho'_1)-\lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho'_2) = \lambda^{|w_2|}(\mathsf{Dsum}_\lambda(\rho_1)-\mathsf{Dsum}_\lambda(\rho'_2))$$

(8)

By definition of the delays, we have:
$\mathsf{delay}(\rho_1\rho_2\rho_3, \rho'_1\rho'_2\rho'_3) =$

$$\frac{\mathsf{Dsum}_\lambda(\rho_1) + \lambda^{|w_1}\mathsf{Dsum}_\lambda(\rho_2) + \lambda^{|w_1w_2|}\mathsf{Dsum}_\lambda(\rho_3) - \mathsf{Dsum}_\lambda(\rho'_1) - \lambda^{|w_1}\mathsf{Dsum}_\lambda(\rho'_2) - \lambda^{|w_1w_2|}\mathsf{Dsum}_\lambda(\rho'_3)}{\lambda^{|w_1w_2w_3|}} \tag{9}$$

Thanks to Equation 8, it can be simplified into:

$$\mathsf{delay}(\rho_1\rho_2\rho_3, \rho'_1\rho'_2\rho'_3) = \frac{\lambda^{|w_2|}(\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1)) + \lambda^{|w_1w_2|}\mathsf{Dsum}_\lambda(\rho_3) - \lambda^{|w_1w_2|}\mathsf{Dsum}_\lambda(\rho'_3)}{\lambda^{|w_1w_2w_3|}} \tag{10}$$

We can now simplify this expression by $\lambda^{|w_2|}$ and we get:

$$\mathsf{delay}(\rho_1\rho_2\rho_3, \rho'_1\rho'_2\rho'_3) = \frac{\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1) + \lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho_3) - \lambda^{|w_1|}\mathsf{Dsum}_\lambda(\rho'_3)}{\lambda^{|w_1w_3|}} \tag{11}$$

which exactly means: $\mathsf{delay}(\rho_1\rho_2\rho_3, \rho'_1\rho'_2\rho'_3) = \mathsf{delay}(\rho_1\rho_3, \rho'_1\rho'_3)$.

◀

Lemma 6. Let $w \in \mathrm{dom}(A)$ such that $|R_A(w)| > 1$. Clearly, there exist two runs ρ, ρ' on w such that $\mathsf{Dsum}_\lambda(\rho) \neq \mathsf{Dsum}_\lambda(\rho')$. Moreover if ρ and ρ' can be decomposed so that the premises of Lemma 24 are satisfied, then we can find a strictly shorter word with two runs on it which have different delays. We can repeat this operation until the goals of Lemma 6 are satisfied. ◀

B.3 Proof of Theorem 7

Proof. We start to prove that A is not functional iff DSumFunTest(A) returns NO.

($\Leftarrow$) The following invariant holds overall the execution of the algorithm: If the stack S contains the pair $((p,q),d)$, then (p,q) is co-accessible and A admits two runs $\rho : q_I \xrightarrow{w} p, \rho' : q_I \xrightarrow{w} q$ such that $\frac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho)}{\lambda^{|w|}} = d$. This can be proved by a simple inductive argument on the number of iterations of the while loop at Line 3. Suppose that DSumFunTest(A) returns NO. There are two cases to consider. If the pair $((p,q) \in F^2, d \neq 0)$ is popped from the stack, then it witnesses the existence of two accepting runs $\rho : q_I \xrightarrow{w} p, \rho' : q_I \xrightarrow{w} q$ in A for which $\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho') \neq 0$. Thus, A is not functional. In the second case, the pair $((p,q),d)$ popped from the stack witnesses that A admits two delays for the co-accessible pair of states (p,q). By Lemma 5, this implies that A is not functional.

($\Rightarrow$) Let A be a non functional Dsum_λ-automaton. By Lemma 6, there exists a word w such that A admits two runs $\rho : q_I \xrightarrow{w} q_f \in F, \rho' : q_I \xrightarrow{w} q'_f \in F$ on w such that $\mathsf{Dsum}_\lambda(\rho) \neq \mathsf{Dsum}_\lambda(\rho')$. Suppose that for all positions $i < j$, $(p_i, q_i) \neq (p_j, q_j)$. In that case DSumFunTest(A) will output No at Line 5 (if not before). Otherwise let i_2 be the least position on the two runs ρ, ρ' such that there exists $i_1 < i_2$ such that ρ (resp. ρ') reach the state p (resp. q) at positions i_1 and i_2. By condition (ii) of Lemma 6, the delays are different at i_1 and i_2 and therefore DSumFunTest(A) will determine that A is not functional at Line 6, after processing ρ, ρ' upon position i_2 (if not before).

This algorithm terminates as any pair of states is visited at most twice. It is well-known that co-accessible states of a finite automaton can be computed in linear time. We can apply this procedure on the product of A with itself to compute the co-accessible pairs of states in quadratic time. ◀

B.4 Proof of Theorem 9

Proof. We give an non-deterministic CoPSpace (and hence PSpace) algorithm to test non-functionality. By the pumping lemma, if a ratio-automaton with n states is not functional, there exists a word of length at most $4n^2$ with two different values. We use the following non-deterministic iterative algorithm: non-deterministically choose two transitions in parallel on the same input letter and count the current length (up to $4n^2$), and compute the current respective rewards and costs of the two current runs. Non-deterministically choose to stop before the length exceeds $4n^2$ and check that the reached states are accepting and that the respective ratio of the two chosen runs are different. We therefore need to store the two current states, reward sum and cost sum, which are bounded by $4n^2M_r$ and $4n^2M_c$, where M_r and M_c are the maximal reward and cost (in absolute value). Those sums are represented in pspace, and in nlogspace if the weights are unary encoded. ◀

C Decision Problems

C.1 Proof of Theorem 12

Proof. For Dsum_λ automata, we show that $L_A^{>\nu} \neq \emptyset$ iff Player 0 has a strategy to ensure a play from v_0 with discounted sum greater than ν in the one player (infinite) Dsum_λ game $\Gamma = (V, E, w, \langle V_0, V_1\rangle)$, where:

- $V = \{p \mid p \in Q \wedge \exists w \in \Sigma^*(p \overset{w}{\rightsquigarrow} f \in F)\}$
- $V_0 = V, V_1 = \emptyset$
- $E = (V \times (\Sigma \cup \zeta) \times V) \cap (\{(p, a, p') \mid (p, a, p') \in \delta\} \cup \{(p, \zeta, p) \mid p \in F\})$, where $\zeta \notin \Sigma$ is a fresh symbol
- For each $e = (p, a, p') \in E$: If $(p, a, p') \in \delta$, then $w(e) = \gamma(p, a, p')$, else $w(e) = 0$.

Once proved the above equivalence, our complexity bound follows easily, since checking wether $L_A^{>\nu} \neq \emptyset$ reduces to solving a 1 player Dsum_λ game (that is in PTime [2]).

($\Rightarrow$) If $L_A^{>\nu} \neq \emptyset$, then A admits an accepting run $r : q_0^1 = r_0 \overset{w}{\rightsquigarrow} r_n \in F$ such that $\mathsf{Dsum}_\lambda(\gamma(r) > \nu$. By construction, Γ admits an (infinite) path p with a positive discounted sum, i.e. Player 0 has a (memoryless) strategy to win the one-player discounted sum game Γ.

($\Leftarrow$) S uppose that Player 0 has a strategy to win the one-player discounted sum game Γ. Let p be an infinite path on Γ consistent with a winning strategy for player 0. Then $\mathsf{Dsum}_\lambda(r) > 0$. Let W be the maximum absolute weight in Γ. For each prefix r_i of length i of r we have:

$$\mathsf{Dsum}_\lambda(r_i) + \frac{W}{1-\lambda} \geq \mathsf{Dsum}_\lambda(r) \Rightarrow$$
$$\mathsf{Dsum}_\lambda(r_i) \geq \mathsf{Dsum}_\lambda(r) - \frac{W\lambda^i}{1-\lambda} \tag{12}$$

Since $\mathsf{Dsum}_\lambda(r) > \nu$, there exists i^* such that $\mathsf{Dsum}_\lambda(r) - \frac{W\lambda^{i^*}}{1-\lambda} > \nu$ that implies $\mathsf{Dsum}_\lambda(r_{i^*}) > \nu$. By construction, each path in Γ can be extended to reach a node in F. Let $r'_i = r'_0 \dots r'_m \in F$ be such a continuation of r^{i^*}. By Equation 12, our choice of i^* guarantees that $\mathsf{Dsum}_\lambda(r'_i) > \nu$. Since A is functional, r' witnesses the existence of a word w such that $L_A(w) > \nu$.

For Sum automata, let A be a Sum-automaton. $L_A^{\sim\nu} \neq \emptyset$ iff A admits a path to a final state whose sum of the weights is $\sim \nu$. This can be easily checked in PTime, using e.g. a shortest path algorithm (once the edges have been reversed).

For Avg-automata, let A be an Avg-automaton. We can assume $\nu = 0$ since the $\sim \nu$-emptiness problem for Avg-automata reduces to the ~ 0-emptiness problem for Avg-automata, by simply reweighting the input automaton [5]. $L_A^{\sim 0} \neq \emptyset$ iff A admits a path to a final state whose sum of the weights is ~ 0, that can be easily checked in PTime.

Finally, let A be a Ratio-automaton, let $\nu = \frac{m}{n}$. We consider the Sum automaton A', where each edge of A having reward r and cost c is replaced by an edge of weight $rn - cm$. It can be easily proved that $L_A^{\sim\nu} \neq \emptyset$ iff $L_{A'}^{\sim\nu} \neq \emptyset$.

◀

C.2 Proof of Theorem 13

Proof. Let A be a V-automaton, $V \in \{\mathsf{Sum}, \mathsf{Avg}, \mathsf{Dsum}_\lambda\}$ and consider the $\geq \nu$-universality (resp. $> \nu$-universality) problem for V-automata. We check wether A admits an accepting run with $V(\gamma(r)) < \nu$. This can be done in PTime for $V \in \{\mathsf{Sum}-, \mathsf{Avg}-, \mathsf{Ratio}, \mathsf{Dsum}_\lambda-\}$ (resp. $V \in \{\mathsf{Sum}-, \mathsf{Avg}-, \mathsf{Ratio}\}$), with a procedure similar to the one applied in the proof of Theorem 12. ◀

D Realizability

D.1 Proof of Theorem 14

Proof. A 2-*counter machine* M consists of a finite set of control states S, an initial state $s_I \in S$, a final state $s_F \in Q$, a set C of counters ($|C| = 2$) and a finite set δ_M of instructions manipulating two integer-valued counters. Instructions are of the form

$s:$ $c := c + 1$ **goto** s'

$s:$ **if** $c = 0$ **then goto** s' **else** $c := c - 1$ **goto** s''.

Formally, instructions are tuples (s, α, c, s') where $s, s' \in S$ are source and target states respectively, the action $\alpha \in \{inc, dec, 0?\}$ applies to the counter $c \in C$. We assume that M is deterministic: for every state $s \in S$, either there is exactly one instruction $(s, \alpha, \cdot, \cdot) \in \delta_M$ and $\alpha = inc$, or there are two instructions $(s, dec, c, \cdot), (s, 0?, c, \cdot) \in \delta_M$.

A *configuration* of M is a pair (s, v) where $s \in S$ and $v : C \to \mathbb{N}$ is a valuation of the counters. An *accepting run* of M is a finite sequence $\pi = (s_0, v_0)\delta_0(s_1, v_1)\delta_1 \ldots \delta_{n-1}(s_n, v_n)$ where $\delta_i = (s_i, \alpha_i, c_i, s_{i+1}) \in \delta_M$ are instructions and (s_i, v_i) are configurations of M such that $s_0 = s_I$, $v_0(c) = 0$ for all $c \in C$, $s_n = s_F$, and for all $0 \leq i < n$, we have $v_{i+1}(c) = v_i(c)$ for $c \neq c_i$, and (a) if $\alpha = inc$, then $v_{i+1}(c_i) = v_i(c_i) + 1$ (b) if $\alpha = dec$, then $v_i(c_i) \neq 0$ and $v_{i+1}(c_i) = v_i(c_i) - 1$, and (c) if $\alpha = 0?$, then $v_{i+1}(c_i) = v_i(c_i) = 0$. The corresponding *run trace* of π is the sequence of instructions $\bar{\pi} = \delta_0\delta_1 \ldots \delta_{n-1}$. The *halting problem* is to decide, given a 2-counter machine M, whether M has an accepting run. This problem is undecidable [19].

Given a 2-counters (deterministic) machine M, we construct a functional weighted functional Sum-automaton $A = (Q, q_0, \delta, \gamma)$ (resp. Avg-automata), where $Q = Q_O \cup Q_I$, $\Sigma = \Sigma_O \cup \Sigma_I$ and $\delta \subseteq Q \times \Sigma \times Q$ such that M halts if and only if $L(A)$ is realizable. In particular, $\Sigma_O = \delta_M$ and a strategy $\pi \in \Lambda_O$ for Player O is winning if and only if for each $\lambda_I \in \Lambda_I$, the projection of $\mathsf{outcome}(\pi, \gamma_2)$ onto Σ_O is an accepting run of M. The alphabet Σ_I for Player I is the set of letters $\Sigma_I = \{go\} \cup (\bigcup_{i=1,2}\{\textit{cheatCi+}, \textit{cheatCi-}\}) \cup (\bigcup_{0 \leq j < |S|}\{\textit{cheatR:sj}\})$. The role of Player I is that of observing the play of Player O and detecting whether he faithfully simulates M, or he cheats. In details, if Player O cheats by declaring the i-th counter equal to 0 when it is not (positive cheat), then Player I can use the action *cheatCi+*, $i \in \{1, 2\}$, to force all the runs but one (with weight ≤ 0) to die. Similarly, if Player O cheats by decrementing a counter with value zero (negative cheat) or on the structural properties of a run of M, then Player I can win by playing the corresponding observing action : *cheatCi-*, for negative cheats on counter $i \in \{1, 2\}$, or *cheatR:sj* for a cheat on the run through M detected at state sj.

The automaton A consists of a nondeterministic initial choice between different gadgets, described below. Each gadget checks one of the properties of the sequence of actions provided by

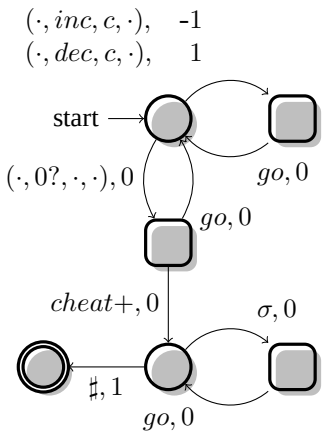

Figure 2 Gadget to check positive cheats

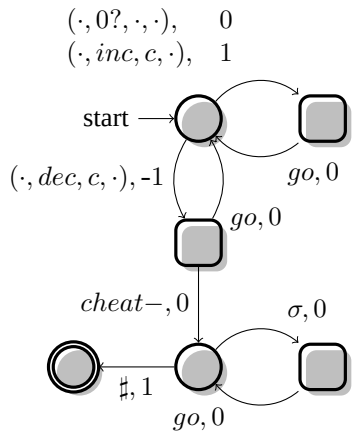

Figure 3 Gadget to check negative cheats

Player O, and verify whether Player O simulates faithfully M or he eventually cheats. Due to the initial nondeterministic choice, each final state (in one of the gadget) is accessible throughout the evolution of the play and Player O has to ensure that all the properties checked in the gadgets are fulfilled. Otherwise, Player I will have the ability to kill all the runs but one, and to ensure that the only surviving run (in the appropriate gadget) reaches the final state with weight ≤ 0.

In particular, Figure 2 represents the gadget to check a positive cheat on counter i, $i \in \{1, 2\}$. Player I observes the inverted value of the counter i throughout the path on M simulated by Player O. Whenever Player O declares that counter i is equal to 0, Player I can use the action *cheatCi+* to kill all the runs in A but the one within the observing gadget. The evolution of such a run up to *cheatCi+* will have a negative value (corresponding to the inverted value of the observed counter) if Player O was cheating. Hence, as soon as Player O playes $\sharp$ it will end in a final state with weight ≤ 0. Symmetrically, the gadget for checking negative cheats (represented in Figure 3) uses the weights on the edges to store the value of the observed counter. If Player O cheats decrementing counter i when its value is 0, Player I can use the action *cheatCi-* to kill all the runs but the one (with negative value) in the gadget observing negative cheats.

Finally, Player I can use the gadgets in Figures 5–6 to detect any structural cheat committed by Player O. If Player O initially provides an action different from $(s_0, \_, \_, \_)$, Player I can punish him by playing action *cheatR:s0*. Similarly, if Player O provides two actions that do not induce a (sub)-path in M, Player I can punish him within the gadget in Figure 6.

The automaton A will contain a gadget to observe positive/negative cheats for each counter $i \in \{1, 2\}$, a gadget to observe a structural cheat for each state $s \in S$ that can be traversed by a path in M, and a neutral gadget (represented in Figure 4), where Player I simply observes the run provided by Player O and let such a run to reach a final state as soon as Player O provides an action

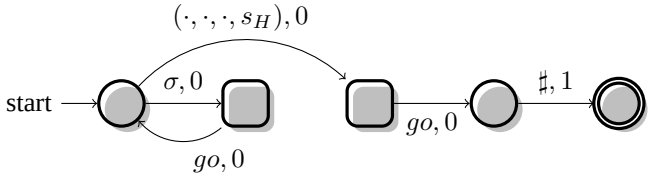

Figure 4 Neutral gadget.

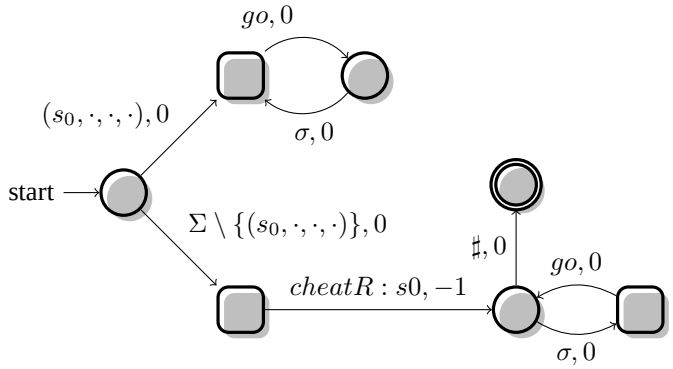

Figure 5 Gadget to check that Player 1 plays $(s_0, \cdot, \cdot, \cdot)$ at the beginning.

simulating a step toward the halting state of M. The proof is concluded by showing that M halts iff Player O has a strategy to win the realizability game on A. Namely, we show that Player O wins the realizability game iff he provides a word π which corresponds to an accepting run of M (and then stop the game).

($\Rightarrow$) Suppose that M halts. Let π be the run of M leading to the halting state, and consider $\lambda_O(\pi) \in \Lambda_O$, where $\lambda_O(\pi)$ denotes the strategy for Player O induced by π, in which Player$O$ provides the word π and then stop the game. Let $\lambda_I \in \Lambda_I$. There are two cases to consider.

1. In the first case, λ_I does not provide any action in:

$$(\bigcup_{i=1,2} \{\textit{cheatCi+}, \textit{cheatCi-}\}) \cup (\bigcup_{0 \leq j < |S|} \{\textit{cheatR:sj}\})$$

   Then, the only run to a final state in A is the one within the neutral gadget, having weight 1.
2. In the second case, γ_I contains an action in $(\bigcup_{i=1,2}\{\textit{cheatCi+}, \textit{cheatCi-}\})$. Let α be the first action in $(\bigcup_{i=1,2}\{\textit{cheatCi+}, \textit{cheatCi-}\})$ on λ_I. There is only one gadget allowing a run containing α. Since π is faithfully simulating M, such a run leads to a final state in the corresponding gadget with value > 0.

Note that λ_I can not contain an action $\alpha \in (\bigcup_{0 \leq j < |S|}\{\textit{cheatR:sj}\})$. In fact, Player I can never play *cheatR:sj*, since Player O does not commit any structural cheat on the run π. Hence, we conclude that $\forall \lambda_I \in \Lambda_I(L_A(\mathsf{outcome}(\lambda_O(\pi), \lambda_I) > 0)$.

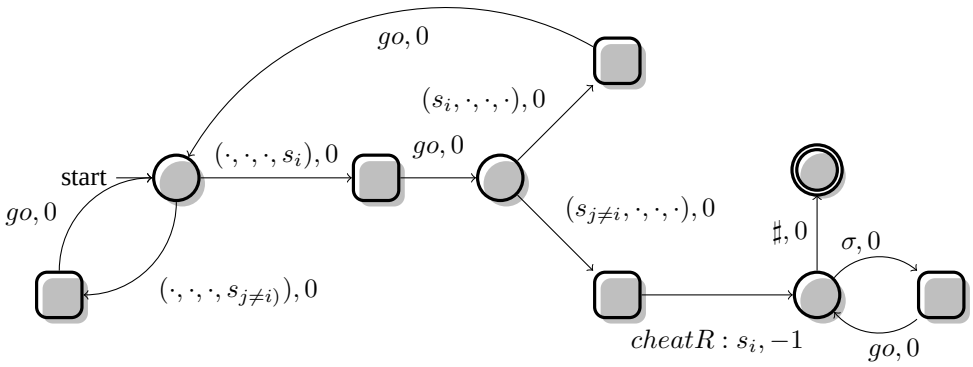

Figure 6 Gadget to check cheats along the run.

($\Leftarrow$) Suppose that the strategy $\lambda_O \in \Lambda_O$ is such that $\forall \lambda_I \in \Lambda_I(L_A(\mathsf{outcome}(\lambda_O, \lambda_I) \geq 0)$. By construction of A, λ_O allows Player 1 to survive in the gadgets for detecting positive, negative or structural cheats if and only if the projection of the outcome onto Σ_O is a faithful simulation of a run in M. If Player I can not use an action in $(\bigcup_{i=1,2}\{\mathit{cheatCi+}, \mathit{cheatCi-}\}) \cup (\bigcup_{0\leq j<|S|}\{\mathit{cheatR:sj}\})$ to win (using the gadget targeted to check the corresponding cheat), the only remaining strategy for Player I is playing indefinitely $\neg cheat$. In that case, Player O wins only if he eventually provides an action simulating a step leading to an halting state in M (and then stop the game). Thus, our hypothesis entail that λ_O consists in providing a run for M that leads to a final state, witnessing that M halts.

◀

D.2 Proof of Theorem 15

Proof. We first consider the case of deterministic Sum-automata. Let $A = (Q = Q_O \uplus Q_I, q_I, F, \delta = \delta_I \cup \delta_O, \gamma)$ be a deterministic Sum-automaton. Without lost of generality, we assume that A contains only one accepting state denoted by f which is absorbing. Then we consider A as a finite state game arena and compute the set of states $S \subseteq Q$ from which player 1 can force a visit to the accepting state f. Note that from any state s in S, player 1 has a strategy to force a visit to f within n steps, where $n = |Q|$. Note also that by determinacy, the complement of this set is the set of states of A from which player 2 has a strategy to prevent a visit to f. Clearly, player 1 has to avoid the states in $Q \setminus S$ at all cost and so they can be removed from A. Let A' be A where we have kept only the states in S.

Now, we construct from A' a finite tree as follows. We unfold A' and stop a branch at a node when:

- it is labeled with f and the sum of the weights on the branch up to the node is equal to $c > 0$,
- it is labeled by a state q that already appears on the branch from the root to the node. We call the node where q already appears the *ancestor* of the leaf.

Let us note L the set of leafs of this finite tree. We then partition the leafs of this tree into L_1, the set of leafs that are good for player 1 and L_2, the set of leafs that are good for player 2. L_1 contains:

- (C_1) the leafs that are annotated with f and for which the sum of weights is strictly positive and
- (C_2) the leafs labeled with a repeating state and for which the sum of weights from the root to the leaf is strictly larger than the sum of weights from the root to the ancestor.

$L_2 = Q \setminus L_1$ are the leafs that are good for player 2. Now, consider the game played on this finite tree where player 1 wants to reach L_1 and player 2 wants to reach L_2. The winner in this game can be determined by backward induction. We claim (and prove) below that player 1 win in this finite game tree iff he wins the original game.

Assume that player 1 wins the finite game tree. We show how to construct a winning strategy in the original game. The strategy is built as follows. In the original game, player 1 plays as in the final tree up to the point where he reaches a leaf (in L_1). If the leaf is of sort defined in C_1 above then we know that player 1 has won the original game. Otherwise, we know that the sum now is strictly greater than the sum up to the ancestor of the leaf that we have reached. Then player 1 continues to play as prescribed by its winning strategy in the tree from the ancestor. Continuing like that, each time that the game arrives at a leaf, the sum of weights has strictly increased from the last visit to that leaf. As a consequence, after a finite amount of time, the sum will be strictly larger than $n \cdot |-W|$ where $-W$ is the smallest negative weight in A'. From that point, player 1 can use his strategy that forces the state f and reach it with a sum that is strictly positive (this is because he can force f within n steps).

Now assume that player 2 wins the finite game tree. We show how to construct a winning strategy in the original game. The strategy simply follows the strategy of player 2 in the finite tree by applying the strategy from the ancestor when reaching a leaf. As only leaf in L_2 are reached when

playing that way, we know that the sum on successive visits to repeating states is non-increasing. As a consequence, as player 1 cannot force a visit to a node labeled with f and a strictly positive sum in the finite game tree, we know that this will not append in the original game neither when player 2 plays its strategy.

This proof clearly establishes that the problem belongs to NP ∩ coNP as we can guess for the winning player one edge per states and verifies in polynomial time that this leads to a winning strategy in the original game. Nevertheless, note that player 1 needs memory to win in the original game as he has to verify that he has reached a sufficiently high sum before applying the strategy that forces the visit to f.

As for the previous results Avg games can be reduced easily to Sum games, and as for the questions about thresholds, Ratio games can be reduced to Avg games.

We now turn to the case of Dsum_λ. The solution for Dsum_λ is obtained by first removing from A all states from which player 1 cannot force a visit to f. As above, we note the game where those states have been removed by A'. Then, we consider A' as an (infinite) discounted sum game where player 1 tries to maximize the value of the discounted sum while player 2 tries to minimize this value. Let v denotes the value of the initial state q_I in that game. We claim that player 1 wins the initial game iff the value v in q_I is strictly positive. Indeed, if player 1 has a winning strategy in the original game, i.e. a strategy to force the game into f with strictly positive discounted sum, then by playing this strategy in the discounted sum game, the infinite discounted sum will be equal to the discounted sum up to f as from there only the self loop on f is crossed and its weight is equal to 0. Now assume that player 1 has a strategy that force a value $v > 0$ in the discounted sum game. Then by playing that strategy for i steps in the original game with i large enough to make sure that $\lambda^i W + \cdots + \lambda^{i+n} W$ is small enough, he will be able to switch to its strategy that forces f after at most n steps and ensure to reach f with a strictly positive discounted sum. As infinite discounted sum games are in NP ∩ coNP [2] and since our reduction is polynomial, we also get that finite reachability discounted sum games are in NP ∩ coNP. ◀

E Determinization

E.1 Proof of Lemma 18

We prove the following short witness property for the twinning property:

▶ **Lemma 25.** *Let A be a Dsum_λ-automaton. If A does not satisfy the twinning property, there exist two words $w_1, w_2 \in \Sigma^*$ such that $|w_1| \leq 2|Q|^2$ and $|w_2| \leq 2|Q|^2$, two states $p, q \in Q$ such that p and q are both co-accessible, and runs $\rho_1 : q_I \xrightarrow{w_1} p$, $\rho_2 : p \xrightarrow{w_2} p$, $\rho_1' : q_I \xrightarrow{w_1} q$, $\rho_2' : q \xrightarrow{w_2} q$, such that $\mathsf{delay}(\rho_1, \rho_1') \neq \mathsf{delay}(\rho_1\rho_2, \rho_1'\rho_2')$.*

Proof. Suppose that $|w_2| > 2|Q|^2$ (the case $|w_1| > 2|Q|^2$ is proved exactly the same way) and that $w_1 w_2$ witnesses that the twinning property does not hold by the decomposition into runs $\rho_1, \rho_2, \rho_1', \rho_2'$ as in the premisses of the lemma. We will show that we can shorten the runs ρ_1, ρ_1' and still get a witness that the twinning property does not hold.

Since $|w_2| > 2|Q|^2$, there is a pair of states (p', q') that repeat three times along the two parallel runs ρ_2 and ρ_2', i.e. w_2 can be decomposed as $w_1' w_2' w_3' w_4'$ and ρ_2 and ρ_2' can be decomposed as $r_1 r_2 r_3 r_4$ and $r_1' r_2' r_3' r_4'$ respectively, where:

$$r_1 : p \xrightarrow{w_1'} p' \quad r_2 : p' \xrightarrow{w_2'} p' \quad r_3 : p' \xrightarrow{w_3'} p' \quad r_4 : p' \xrightarrow{w_4'} p$$
$$r_1' : q \xrightarrow{w_1'} q' \quad r_2' : q' \xrightarrow{w_2'} q' \quad r_3' : q' \xrightarrow{w_3'} q' \quad r_4' : p' \xrightarrow{w_4'} q$$

Note that r_1, r_1' and r_4, r_4' may be empty (in this case $p = p'$ and $q = q'$), but r_2, r_3, r_2', r_3' are assumed to be non-empty.

Now, there are two cases: $\mathsf{delay}(\rho_1 r_1, \rho'_1 r'_1) \neq \mathsf{delay}(\rho_1 r_1 r_2, \rho'_1 r'_1 r'_2)$ and in that case the word $w_1 w'_1 w'_2$ is a witness that the twinning property does not hold, and $|w_1 w'_1 w'_2| < |w_1 w_2|$. In the second case, we have $\mathsf{delay}(\rho_1 r_1, \rho'_1 r'_1) = \mathsf{delay}(\rho_1 r_1 r_2, \rho'_1 r'_1 r'_2)$, but in that case, we can apply Lemma 24 and we get $\mathsf{delay}(\rho_1 r_1 r_3 r_4, \rho'_1 r'_1 r'_3 r'_4) = \mathsf{delay}(\rho_1 \rho_2, \rho'_1 \rho'_2)$. Therefore $\mathsf{delay}(\rho_1, \rho'_1) \neq \mathsf{delay}(\rho_1 r_1 r_3 r_4, \rho'_1 r'_1 r'_3 r'_3)$ and $w_1 w'_1 w'_3 w'_4$ is a shorter witness that the twinning property does not hold.

We can iterate this reasoning until we find a witness whose size satisfy the premisses of the lemma. ◀

We are now ready to prove Lemma 18:

Proof of Lemma 18. We define a non-deterministic PSpace algorithm to check whether a Dsum_λ-automaton does not satisfy the twinning property. The idea is to guess two runs on the same input word of size at most $4|Q|^2$ and two positions in those runs, and check the pair of states at the two positions are equal and that the respective delays are different. This algorithm uses a polynomial space (denomitors of the form $\lambda^{|w|}$ are stored in polynomial space) and thanks to Lemma 25, is correct. ◀

E.2 Proofs of Lemma 21 and Lemma 20

We prove a stronger version of Lemma 21:

► **Lemma 26.** *Let f be an accessible state of A_d by a run ρ_d on some word $w \in \Sigma^*$.*
Then the following hold:

1. *dom(f) is the set of states q such that there exists a run on w reaching q;*
2. $\mathsf{Dsum}_\lambda(\rho_d T) = \min\{\mathsf{Dsum}_\lambda(\xi) \mid \xi \text{ is a run of } A \text{ on } w\}$.
3. *If $q \in$ dom(f) and ρ is a run on w reaching q, then*

$$f(q) = \max\{\mathsf{delay}(\rho, \rho') \mid \rho' \text{ is a run of } A \text{ on } w\} = \frac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho_d T)}{\lambda^{|w|}}$$

4. *For all $f' : Q \to \mathbb{Q}$ and all $a \in \Sigma$ such that dom$(f') = \{q' \mid \exists q \in$ dom$(f),\ (q, a, q') \in \delta\}$ and for all $q' \in$ dom(f'):*

$$f'(q') = \frac{f(q)}{\lambda} + \gamma(q, a, q') - \gamma_d(f, a, f') \text{ for some } q \in \text{dom}(f) \text{ such that } (q, a, q') \in \delta$$

we have $f' \in Q_d$ and $(f, a, f') \in \delta_d$.

Proof. The five statements are proved by induction on $|w|$. It is clear when $|w| = 0$.

Suppose that $|w| > 0$ and $w = w'a$ for some $a \in \Sigma$. Let $\rho_d : f_I \xrightarrow{w'} f$ be a run of A_d on w' and let f' such that $(f, a, f') \in \delta_d$, and $t_d = (f, a, f')$.

- The first statement is obvious by induction hypothesis and by definition of δ_d.

- The second statement is proved as follows:

$$
\begin{aligned}
& \mathsf{Dsum}_\lambda(\rho_d T) \\
= \;& \mathsf{Dsum}_\lambda(\rho_d) + \lambda^{|w|}\gamma_d(T) \\
= \;& \mathsf{Dsum}_\lambda(\rho_d) + \lambda^{|w|}\min\{\frac{f(q)}{\lambda} + \gamma(q,a,q') \mid q \in \mathrm{dom}(f) \wedge (q,a,q') \in \delta\} \\
= \;& \mathsf{Dsum}_\lambda(\rho_d) + \lambda^{|w|}\min\{\frac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho_d)}{\lambda^{|w|}} + \gamma(q,a,q') \mid q \in \mathrm{dom}(f) \wedge (q,a,q') \in \delta\}
\end{aligned}
$$

by induction hypothesis and for some run $\rho : q_I \xrightarrow{w'} q$. This is independ on the choice of ρ as any run ρ' reaching q on w' satisfies $\mathsf{Dsum}_\lambda(\rho') = \mathsf{Dsum}_\lambda(\rho)$, as A is functional and q is co-accessible.

$$
\begin{aligned}
= \;& \mathsf{Dsum}_\lambda(\rho_d) + \min\{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho_d) + \lambda^{|w|}\gamma(q,a,q') \mid q \in \mathrm{dom}(f) \wedge (q,a,q') \in \delta\} \\
= \;& \min\{\mathsf{Dsum}_\lambda(\rho) + \lambda^{|w|}\gamma(q,a,q') \mid q \in \mathrm{dom}(f) \wedge (q,a,q') \in \delta\} \\
= \;& \min\{\mathsf{Dsum}_\lambda(\rho(q,a,q')) \mid q \in \mathrm{dom}(f) \wedge (q,a,q') \in \delta\} \\
= \;& \min\{\mathsf{Dsum}_\lambda(\xi) \mid \xi \text{ is a run on } w\}
\end{aligned}
$$

(as it is independ on the choice of ρ)

- The third statement is proved as follows: Let $q' \in \mathrm{dom}(f')$.

$$
\begin{aligned}
f'(q') &= \frac{f(q)}{\lambda} + \gamma(q,a,q') - \gamma_d(f,a,f') \\
&\quad \text{(for some } q \in \mathrm{dom}(f) \text{ such that } (q,a,q') \in \delta\text{)} \\
&= \frac{\mathsf{Dsum}_\lambda(\rho) - \mathsf{Dsum}_\lambda(\rho_d)}{\lambda^{|w|}} + \gamma(q,a,q') - \gamma_d(f,a,f') \\
&\quad \text{(by induction hypothesis and for some } \rho : q_I \xrightarrow{w'} q\text{)} \\
&= \frac{\mathsf{Dsum}_\lambda(\rho) + \lambda^{|w|}\gamma(q,a,q') - \mathsf{Dsum}_\lambda(\rho_d) - \lambda^{|w|}\gamma_d(f,a,f')}{\lambda^{|w|}} \\
&= \frac{\mathsf{Dsum}_\lambda(\rho(q,a,q')) - \mathsf{Dsum}_\lambda(\rho_d T)}{\lambda^{|w|}}
\end{aligned}
$$

This value does not depend on the choice q. Indeed, any run ρ' that reaches q' on w satisfies $\mathsf{Dsum}_\lambda(\rho') = \mathsf{Dsum}_\lambda(\rho(q,a,q'))$, as q' is co-accessible and A is functional.

Then, we prove the second part of the third statement:

$$
\begin{aligned}
f'(q') &= \frac{\mathsf{Dsum}_\lambda(\rho(q,a,q')) - \mathsf{Dsum}_\lambda(\rho_d T)}{\lambda^{|w|}} \\
&= \frac{\mathsf{Dsum}_\lambda(\rho(q,a,q')) - \min\{\mathsf{Dsum}_\lambda(\xi) \mid \xi \text{ is a run on } w\}}{\lambda^{|w|}} \\
&= \max\frac{\mathsf{Dsum}_\lambda(\rho(q,a,q')) - \mathsf{Dsum}_\lambda(\xi)}{\lambda^{|w|}} \mid \xi \text{ is a run on } w\}\lambda^{|w|} \\
&= \max\{\mathsf{delay}(\rho(q,a,q'),\xi) \mid \xi \text{ is a run on } w\}
\end{aligned}
$$

which achieves to prove the lemma, as again, this value does not depend on the choice of the run $\rho(q,a,q')$.

- We prove the fifth statement. Let f'' be a function as defined in the fifth statement. We have seen that the value $f'(q')$ does not depend on the choice of q. We can therefore use exactly the same proof as f' to prove that for all $q'' \in \mathrm{dom}(f'')$:

$$f''(q'') = \max\{\mathsf{delay}(\rho,\rho') \mid \rho,\rho' \text{ are runs of } A \text{ on } wa \text{ s.t. } \rho \text{ reaches } q''\}$$

By definition of Q_d, we get $f'' \in Q_d$ and by definition of δ_d, $(f',a,f'') \in \delta_d$. ◄

Proof of Lemma 20. First note that A_d is complete. Indeed, for all $f \in Q_d$ and all $a \in \Sigma$, there exists $f' \in Q_d$ such that $(f,a,f') \in \delta_d$. It suffices to define f' as follows: for all $q' \in Q$:

$$f'(q') = \frac{f(q)}{\lambda} + \gamma(q,a,q') - \gamma_d(f,a,f') \text{ for some } q \in \mathrm{dom}(f) \text{ such that } (q,a,q') \in \delta$$

By Lemma 26 (statement 5), we get $(f, a, f') \in \delta_d$.

We show that A_d is deterministic. Suppose that there exists $f, f', f'' \in Q_d$ and $a \in \Sigma$ such that $(f, a, f') \in \delta_d$ and $(f, a, f'') \in \delta_d$. Clearly, by definition of δ_d, $\mathrm{dom}(f') = \mathrm{dom}(f'')$. Since f' and f'' are accessible by definition of A_d, we can apply Lemma 20 and we clearly get that $f'(q) = f''(q)$ for all $q \in \mathrm{dom}(f') = \mathrm{dom}(f'')$. Therefore A_d is deterministic.

Let us prove that $L_{A_d} = L_A$. Let $w \in \Sigma^+$. We show that $w \in \mathrm{dom}(A_d)$ iff $w \in \mathrm{dom}(A)$. If $w \in \mathrm{dom}(A_d)$, then there exists an accepting run $\rho_d : f_I \xrightarrow{w} f$ of A_d such that $f \in F_d$. Therefore there exists $q \in \mathrm{dom}(f)$ such that $q \in F$. By Lemma 26, there exists a run of A on w reaching $q \in F$, so that $w \in \mathrm{dom}(A)$.

Conversely, if $w \in \mathrm{dom}(A)$, then there exists an accepting run $\rho : q_I \xrightarrow{w} q$ with $q \in F$. Since A_d is complete, there exists an accepting run of A_d on w reaching some $f \in Q_d$. Again by Lemma 26, we get $q \in \mathrm{dom}(f)$ and therefore, since $q \in F$, we have $f \in F_d$. Hence $w \in \mathrm{dom}(A_d)$.

Let $w \in \mathrm{dom}(A)$, we show that $L_{A_d}(w) = L_A(w)$. Since $\mathrm{dom}(A) = \mathrm{dom}(A_d)$, $w \in \mathrm{dom}(A_d)$ and therefore there exists an accepting run of A_d on w that we denote by $\rho_d : f_I \xrightarrow{w} f \in F_d$. By Lemma 26, $\mathsf{Dsum}_\lambda(\rho_d) = \min\{\mathsf{Dsum}_\lambda(\xi) \mid \xi \text{ is a run of } A \text{ on } w\}$. Since $w \in \mathrm{dom}(A)$ and $\mathrm{dom}(A) \subseteq (\Sigma - \#)^*\#$, w has necessarily the form $w'\#$ and since all states of A are assumed to be co-accessible, all the runs of A on w are necessarily accepting. Therefore $\mathsf{Dsum}_\lambda(\rho_d) = \mathsf{Dsum}_\lambda(\xi)$ for some accepting run ξ of A on w (the choice of ξ is not important as A is functional). In other words, $L_{A_d}(w) = L_A(w)$. ◀

E.3 Proof of Theorem 22

Proof. The forth direction has been already proved (Lemma 20) . We prove the back direction (i.e. the twinning property is a necessary condition). Suppose that the twinning property does not hold. There exist states p, q such that p and q are co-accessible and there exists words $w_1, w_2 \in \Sigma^*$, and runs $\rho_1 : q_I \xrightarrow{w_1} p$, $\rho_2 : p \xrightarrow{w_2} p$, $\rho'_1 : q_I \xrightarrow{w_1} q$, $\rho'_2 : q \xrightarrow{w_2} q$, such that $\mathsf{delay}(\rho_1, \rho'_1) \neq \mathsf{delay}(\rho_1\rho_2, \rho'_1\rho'_2)$.

We first show that there are infinitely many different delays. For all $i \geq 0$, let $\Delta(i) = \mathsf{delay}(\rho_1(\rho_2)^i, \rho'_1(\rho'_2)^i)$. We show that for all $j \geq i \geq 0$, we have:

$$\lambda^{i|w_2|}(\Delta(j) - \Delta(i)) = \Delta(j-i) - \Delta(0) \tag{13}$$

Let us first develop the expression $\Delta(j) - \Delta(i)$:

$$\begin{aligned}
& \Delta(j) - \Delta(i) \\
= \; & \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^j) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^j)}{\lambda^{|w_1|+j|w_2|}} - \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^i)}{\lambda^{|w_1|+i|w_2|}} \\
= \; & \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^j) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^j) - \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i) + \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^i)}{\lambda^{|w_1|+j|w_2|}}
\end{aligned}$$

We can rewrite $\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^j) - \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i)$ into $\mathsf{Dsum}_\lambda(\rho_1) - \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho_1) + \lambda^{|w_1|}\mathsf{Dsum}_\lambda((\rho_2)^{j-i})$, i.e. $\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{j-i}) - \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho_1)$. A similar rewriting can be obtained for $\mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^j) - \lambda^{(j-i)|w_2|}\mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^i)$. Therefore we get:

$$\begin{aligned}
& \Delta(j) - \Delta(i) \\
= \; & \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{j-i}) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{j-i})}{\lambda^{|w_1|+j|w_2|}} - \frac{\lambda^{(j-i)|w_2|}(\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1))}{\lambda^{|w_1|+j|w_2|}} \\
= \; & \frac{1}{\lambda^{i|w_2|}}\left(\frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{j-i}) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{j-i})}{\lambda^{|w_1|+(j-i)|w_2|}} - \frac{\lambda^{(j-i)|w_2|}(\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1))}{\lambda^{|w_1|+(j-i)|w_2|}}\right) \\
= \; & \frac{1}{\lambda^{i|w_2|}}\left(\frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{j-i}) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{j-i})}{\lambda^{|w_1|+(j-i)|w_2|}} - \frac{\mathsf{Dsum}_\lambda(\rho_1) - \mathsf{Dsum}_\lambda(\rho'_1)}{\lambda^{|w_1|}}\right) \\
= \; & \frac{1}{\lambda^{i|w_2|}}(\Delta(j-i) - \Delta(0))
\end{aligned}$$

By Equation 13, for all $i \geq 0$, we have $\lambda^{i|w_2|}(\Delta(i+1) - \Delta(i)) = \Delta(1) - \Delta(0)$. Since by hypothesis, $\Delta(1) \neq \Delta(0)$, the sequence $(\Delta(i))_{i\geq 0}$ is either strictly increasing or strictly decreasing. Hence we get:

$$\forall i, j \geq 0, \quad (i \neq j) \implies \Delta(i) \neq \Delta(j) \tag{14}$$

We use Equation 14 to show that A cannot be determinized. We suppose that there exists a deterministic automaton $A_d = (Q_d, f_I, F_d, \delta_d, \gamma_d)$ equivalent to A and get a contradiction. We consider a word of the form $w_1(w_2)^i$, for i taken large enough to satisfy the existence of a run of A_d of the following form:

$$f_I \xrightarrow{w_1(w_2)^{k_1}} f \xrightarrow{(w_2)^{k_2}} f \xrightarrow{(w_2)^{i-k_2-k_1}} f'$$

for some k_1, k_2 such that $k_2 > 0$, and some $f, f' \in Q_d$.

Moreover, since p and q are both co-accessible, there exist two words w_3, w_3' and two runs of A_d of the form:

$$\rho_d \ : \ f_I \xrightarrow{w_1(w_2)^{k_1}} f \xrightarrow{(w_2)^{k_2}} f \xrightarrow{(w_2)^{i-k_2-k_1}w_3} g$$
$$\rho_d' \ : \ f_I \xrightarrow{w_1(w_2)^{k_1}} f \xrightarrow{(w_2)^{k_2}} f \xrightarrow{(w_2)^{i-k_2-k_1}w_3'} g'$$

for some accepting states $g, g' \in F_d$. Let $\rho_d = \rho_{d,1}\rho_{d_2}\rho_{d,3}$ and $\rho_d' = \rho_{d,1}\rho_{d_2}\rho_{d,3}'$ for some subruns $\rho_{d,1}, \rho_{d_2}$ that correspond to $w_1(w_2)^{k_1}$ and $(w_2)^{k_2}$ respectively, and some subruns $\rho_{d,3}$ and $\rho_{d,3}'$ that correspond to $(w_2)^{i-k_1-k_2}w_3$ and $(w_2)^{i-k_1-k_2}w_3'$ respectively. By equation 14, we know that $\Delta(k_1) \neq \Delta(k_1+k_2)$. We show that this leads to a contradiction. Let also $\rho_3 : p \xrightarrow{w_3} p_f$ and $\rho_3' : q \xrightarrow{w_3'} q_f$ be two runs in A, for some $p_f, q_f \in F$. Then we have:

$$\mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,2}\rho_{d,3}) = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i\rho_3) \tag{15}$$

$$\mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,2}\rho_{d,3}') = \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^i\rho_3') \tag{16}$$

$$\mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,3}) = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{i-k_2}\rho_3) \tag{17}$$

$$\mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,3}') = \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^{i-k_2}\rho_3') \tag{18}$$

From which we get:

$$\mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,2}\rho_{d,3})) - \mathsf{Dsum}_\lambda(\rho_{d,1}\rho_{d,2}\rho_{d,3}')) = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i\rho_3) - \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^i\rho_3') \tag{19}$$

which is equivalent to:

$$\lambda^{|w_1|+(k_1+k_2)|w_2|}(\mathsf{Dsum}_\lambda(\rho_{d,3}) - \mathsf{Dsum}_\lambda(\rho_{d,3}')) = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i\rho_3) - \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^i\rho_3') \tag{20}$$

Similarly:

$$\lambda^{|w_1|+k_1|w_2|}(\mathsf{Dsum}_\lambda(\rho_{d,3}) - \mathsf{Dsum}_\lambda(\rho_{d,3}')) = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{i-k_2}\rho_3) - \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^{i-k_2}\rho_3') \tag{21}$$

Dividing Equation 20 by $\lambda^{k_2|w_2|}$ and combining it with Equation 21 we get:

$$\frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i\rho_3) - \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^i\rho_3')}{\lambda^{k_2|w_2|}} = \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{i-k_2}\rho_3) - \mathsf{Dsum}_\lambda(\rho_1'(\rho_2')^{i-k_2}\rho_3') \tag{22}$$

Let us rewrite the left hand side of Equation 22:

$$\begin{aligned}
& \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^i\rho_3) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^i\rho'_3)}{\lambda^{k_2|w_2|}} \\
= \quad & \frac{\mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{k_1+k_2}) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{k_1+k_2})}{\lambda^{k_2|w_2|}} + \\
& + \frac{\lambda^{|w_1|+(k_1+k_2)|w_2|}(\mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3))}{\lambda^{k_2|w_2|}} \\
= \quad & \lambda^{|w_1|+k_1|w_2|}(\Delta(k_1+k_2) + \mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3))
\end{aligned}$$

Similarly, we rewrite the right hand side of Equation 22:

$$\begin{aligned}
& \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{i-k_2}\rho_3) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{i-k_2}\rho'_3) \\
= \quad & \mathsf{Dsum}_\lambda(\rho_1(\rho_2)^{k_1}) - \mathsf{Dsum}_\lambda(\rho'_1(\rho'_2)^{k_1}) + \\
& + \lambda^{|w_1|+k_1|w_2|}(\mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3)) \\
= \quad & \lambda^{|w_1|+k_1|w_2|}(\Delta(k_1) + \mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3))
\end{aligned}$$

Therefore Equation 22 rewrites into:

$$\begin{gathered}
\lambda^{|w_1|+k_1|w_2|}(\Delta(k_1+k_2) + \mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3)) \\
= \\
\lambda^{|w_1|+k_1|w_2|}(\Delta(k_1) + \mathsf{Dsum}_\lambda((\rho_2)^{i-k_2-k_1}\rho_3) - \mathsf{Dsum}_\lambda((\rho'_2)^{i-k_2-k_1}\rho'_3))
\end{gathered}$$

And therefore $\Delta(k_1) = \Delta(k_1+k_2)$, which is a contradiction. ◀