Logical Methods in Computer Science
Volume 18, Issue 3, 2022, pp. 13:1–13:28
https://lmcs.episciences.org/

Submitted Dec. 20, 2018
Published Jul. 29, 2022

COMPARATOR AUTOMATA IN QUANTITATIVE VERIFICATION

SUGUMAN BANSAL [a], SWARAT CHAUDHURI [b], AND MOSHE Y. VARDI [c]

[a] University of Pennsylvania, Philadelphia, USA
e-mail address: suguman@seas.upenn.edu

[b] University of Texas, Austin, USA
e-mail address: swarat@cs.utexas.edu

[c] Rice University, Houston, USA
e-mail address: vardi@cs.rice.edu

Abstract.

The notion of comparison between system runs is fundamental in formal verification. This concept is implicitly present in the verification of qualitative systems, and is more pronounced in the verification of quantitative systems. In this work, we identify a novel mode of comparison in quantitative systems: the online comparison of the aggregate values of two sequences of quantitative weights. This notion is embodied by *comparator automata* (*comparators*, in short), a new class of automata that read two infinite sequences of weights synchronously and relate their aggregate values.

We show that aggregate functions that can be represented with Büchi automaton result in comparators that are finite-state and accept by the Büchi condition as well. Such *ω-regular comparators* further lead to generic algorithms for a number of well-studied problems, including the quantitative inclusion and winning strategies in quantitative graph games with incomplete information, as well as related non-decision problems, such as obtaining a finite representation of all counterexamples in the quantitative inclusion problem.

We study comparators for two aggregate functions: discounted-sum and limit-average. We prove that the discounted-sum comparator is ω-regular iff the discount-factor is an integer. Not every aggregate function, however, has an ω-regular comparator. Specifically, we show that the language of sequence-pairs for which limit-average aggregates exist is neither ω-regular nor ω-context-free. Given this result, we introduce the notion of *prefix-average* as a relaxation of limit-average aggregation, and show that it admits ω-context-free comparators i.e. comparator automata expressed by Büchi pushdown automata.

Key words and phrases: Comparator automata, aggregate functions, discounted-sum, limit-average, quantitative inclusion, PSPACE-complete, ω-regular, integer discount-factor.

An earlier version of this paper has appeared at FoSSaCS 2018 [BCV18b].

LOGICAL METHODS IN COMPUTER SCIENCE

DOI:10.46298/LMCS-18(3:13)2022

1. Introduction

Many classic questions in formal methods can be seen as involving *comparisons* between different system runs or inputs. Consider the problem of verifying if a system S satisfies a linear-time temporal property P. Traditionally, this problem is phrased language-theoretically: S and P are interpreted as sets of (infinite) words, and S is determined to satisfy P if $S \subseteq P$. The problem, however, can also be framed in terms of a *comparison* between words in S and P. Suppose a word w is assigned a weight of 1 if it belongs to the language of the system or property, and 0 otherwise. Then determining if $S \subseteq P$ amounts to checking whether the weight of every word in S is less than or equal to its weight in P [BK$^+$08].

The need for such a formulation is clearer in quantitative systems, in which every run of a word is associated with a sequence of (rational-valued) weights. The weight of a run is given by *aggregate function* $f : \mathbb{Q}^\omega \to \mathbb{R}$, which returns the real-valued *aggregate value* of the run's weight sequence. The weight of a word is given by the supremum or infimum of the weight of all its runs. Common examples of aggregate functions include discounted-sum and limit-average.

In a well-studied class of problems involving quantitative systems, the objective is to check if the aggregate value of words of a system exceed a constant threshold value [DAFH$^+$04, DAFS04, DDG$^+$10]. This is a natural generalization of emptiness problems in qualitative systems. Known solutions to the problem involve arithmetic reasoning via linear programming and graph algorithms such as negative-weight cycle detection, computation of maximum weight of cycles etc [And06, Kar78].

A more general notion of comparison relates aggregate values of two weight sequences. Such a notion arises in the *quantitative inclusion problem* for weighted automata [ABK11], where the goal is to determine whether the weight of words in one weighted automaton is less than that in another. Here it is necessary to compare the aggregate value along runs between the two automata. Approaches based on arithmetic reasoning do not, however, generalize to solving such problems. In fact, the known solution to discounted-sum inclusion with integer discount-factor combines linear programming with a *specialized* subset-construction-based determinization step, rendering an EXPTIME algorithm [And06, BH14]. Yet, this approach does not match the PSPACE lower bound for discounted-sum inclusion.

In this paper, we present an automata-theoretic formulation of this form of comparison between weighted sequences. Specifically, we introduce *comparator automata* (*comparators*, in short), a class of automata that read pairs of infinite weight sequences synchronously, and compare their aggregate values in an online manner. While comparisons between weight sequences happen implicitly in prior approaches to quantitative systems, comparator automata make these comparisons explicit. We show that this has many benefits, including generic algorithms for a large class of quantitative reasoning problems, as well as a direct solution to the problem of discounted-sum inclusion that also closes its complexity gap.

A *comparator for aggregate function f for relation R* is an automaton that accepts a pair (A, B) of sequences of bounded natural numbers iff $f(A)\ R\ f(B)$, where R is an inequality relation ($>$, $<$, $\geq$, $\leq$, $\neq$) or the equality relation $=$. A comparator could be finite-state or (pushdown) infinite-state. This paper studies such comparators.

A comparator is *ω-regular* if it is finite-state and accepts by the Büchi condition. We relate ω-regular comparators to *ω-regular aggregate functions* [CSV13], and show that ω-regular aggregate-functions entail ω-regular comparators. However, the other direction is still open: Does an ω-regular comparator for an aggregate function and a relation

imply that the aggregate function is also ω-regular? Furthermore, we show that ω-regular comparators lead to generic algorithms for a number of well-studied problems including the quantitative inclusion problem, and in solving quantitative games with incomplete information. Our algorithm yields PSPACE-completeness of quantitative inclusion when the ω-regular comparator is provided. The same algorithm extends to obtaining finite-state representations of counterexample words in inclusion.

Next, we show that the discounted-sum aggregation function admits an ω-regular comparator for all relations R iff the discount-factor $d > 1$ is an integer. We use this result to prove that discounted-sum aggregate function for discount-factor $d > 1$ is ω-regular iff d is an integer. Furthermore, we use properties of ω-regular comparators to conclude that the discounted-sum inclusion is PSPACE-complete, hence resolving the complexity gap (under a unary representation of numbers).

Finally, we investigate the limit-average comparator. Since limit-average is only defined for sequences in which the average of prefixes converge, limit-average comparison is not well-defined. We show that even a Büchi pushdown automaton cannot separate sequences for which limit-average exists from those for which it does not. Hence, we introduce the novel notion of *prefix-average comparison* as a relaxation of limit-average comparison. We show that the prefix-average comparator admits a comparator that is ω-context-free, i.e., given by a Büchi pushdown automaton, and we discuss the utility of this characterization.

This paper is organized as follows: Preliminaries are given in § 2. Comparator automata are formally defined in § 3. The connections between ω-regular aggregate functions and ω-regular comparators is discussed in Section 3.1. Generic algorithms for ω-regular comparators are discussed in § 3.2-3.3. § 4 discusses discounted-sum aggregate function and its comparators with non-integer rational discount-factors (§ 4.1) and integer discount-factors (§ 4.2). The construction and properties of prefix-average comparator are given in § 5. We conclude with future directions in § 6.

1.1. **Related work.** The notion of comparison has been widely studied in quantitative settings. Here we mention only a few of them. Such aggregate-function based notions appear in weighted automata [ABK11, DKV09], quantitative games including mean-payoff and energy games [DDG$^+$10], discounted-payoff games [AC13, And06], in systems regulating cost, memory consumption, power consumption, verification of quantitative temporal properties [DAFH$^+$04, DAFS04], and others. Common solution approaches include graph algorithms such as weight of cycles or presence of cycle [Kar78], linear-programming-based approaches, fixed-point-based approaches [CD10], and the like. The choice of approach for a problem typically depends on the underlying aggregate function. In contrast, in this work we present an automata-theoretic approach that unifies solution approaches to problems on different aggregate functions. We identify a class of aggregate functions, ones that have an ω-regular comparator, and present generic algorithms for some of these problems.

While work on finite-representations of counterexamples and witnesses in the qualitative setting is known [BK$^+$08], we are not aware of such work in the quantitative verification domain. This work can be interpreted as automata-theoretic arithmetic, which has been explored in regular real analysis [CSV13].

2. Preliminaries

Let Σ be a finite set of alphabet. The set of finite and infinite words over Σ is denoted by Σ^* and Σ^ω, respectively. An *aggregate function* $f : \mathbb{Q}^\omega \to \mathbb{R}$ takes the aggregate of an infinite-length weight sequence.

Definition 2.1 (Büchi automaton [TW$^+$02])**.** A (finite-state) *Büchi automaton* is a tuple $\mathcal{A} = (S, \Sigma, \delta, \mathit{Init}, \mathcal{F})$, where S is a finite set of *states*, Σ is a finite *input alphabet*, $\delta \subseteq (S \times \Sigma \times S)$ is the *transition relation*, $\mathit{Init} \subseteq S$ is the set of *initial states*, and $\mathcal{F} \subseteq S$ is the set of *accepting states*.

A Büchi automaton is *deterministic* if for all states s and inputs a, $|\{s' | (s, a, s') \in \delta \text{ for some } s'\}| \leq 1$ and $|\mathit{Init}| = 1$. Otherwise, it is *nondeterministic*. A Büchi automaton is *complete* if for all states s and inputs a, $|\{s' | (s, a, s') \in \delta \text{ for some } s'\}| \geq 1$. For a word $w = w_0 w_1 \cdots \in \Sigma^\omega$, a *run* ρ of w is a sequence of states $s_0 s_1 \ldots$ s.t. $s_0 \in \mathit{Init}$, and $\tau_i = (s_i, w_i, s_{i+1}) \in \delta$ for all i. Let $\mathit{inf}(\rho)$ denote the set of states that occur infinitely often in run ρ. A run ρ is an *accepting run* if $\mathit{inf}(\rho) \cap \mathcal{F} \neq \emptyset$. A word w is an *accepting word* if it has an accepting run. Büchi automata are closed under set-theoretic union, intersection, and complementation [TW$^+$02]. Languages accepted by these automata are called *ω-regular languages*.

Reals over ω-words [CSV13]**.** Given an integer base $\beta \geq 2$, its *digit set* is $\mathsf{Digit}(\beta) = \{0, \ldots, \beta - 1\}$. Let $x \in \mathbb{R}$, then there exist unique words $\mathsf{Int}(x, \beta) = z_0 z_1 \cdots \in \mathsf{Digit}(\beta)^* \cdot 0^\omega$ and $\mathsf{Frac}(x, \beta) = f_0 f_1 \cdots \in \mathsf{Digit}(\beta)^\omega \setminus \mathsf{Digit}(\beta)^* \cdot (\beta - 1)^\omega$ such that $|x| = \Sigma_{i=0}^{\infty} \beta^i \cdot z_i + \Sigma_{i=0}^{\infty} \frac{f_i}{\beta^i}$. Thus, z_i and f_i are respectively the i-th least significant digit in the base β representation of the integer part of x, and the i-th most significant digit in the base β representation of the fractional part of x. Then, a real-number $x \in \mathbb{R}$ in base β is represented by $\mathsf{rep}(x, \beta) = \mathsf{sign} \cdot (\mathsf{Int}(x, \beta), \mathsf{Frac}(x, \beta))$, where $\mathsf{sign} = +$ if $x \geq 0$, $\mathsf{sign} = -$ if $x < 0$, and $(\mathsf{Int}(x, \beta), \mathsf{Frac}(x, \beta))$ is the interleaved word of $\mathsf{Int}(x, \beta)$ and $\mathsf{Frac}(x, \beta)$. Clearly, $x = \mathsf{sign} \cdot |x| = \mathsf{sign} \cdot (\Sigma_{i=0}^{\infty} \beta^i \cdot z_i + \Sigma_{i=0}^{\infty} \frac{f_i}{\beta^i})$. For all integer $\beta \geq 2$, we denote the alphabet of representation of real-numbers in base β by $\mathsf{AlphaRep}(\beta) = \{+, -\} \cup \mathsf{Digit}(\beta) \times \mathsf{Digit}(\beta)$.

We adopt the definitions of function automata and regular functions [CSV13] w.r.t. aggregate functions as follows:

Definition 2.2 (Aggregate function automaton, ω-Regular aggregate function)**.** Let Σ be a finite set, and $\beta \geq 2$ be an integer-valued base. A Büchi automaton $\mathcal{A}$ over alphabet $\Sigma \times \mathsf{AlphaRep}(\beta)$ is an *aggregate function automaton of type* $\Sigma^\omega \to \mathbb{R}$ if for all $A \in \Sigma^\omega$, there exists exactly one $x \in \mathbb{R}$ such that $(A, \mathsf{rep}(x, \beta)) \in \mathcal{L}(\mathcal{A})$.

Equivalently, the language could be represented by a Parity automaton instead of a Büchi automaton.

Σ and $\mathsf{AlphaRep}(\beta)$ are the *input* and *output* alphabets, respectively. An aggregate function is an arbitrary function $f : \Sigma^\omega \to \mathbb{R}$. An aggregate function $f : \Sigma^\omega \to \mathbb{R}$ is said to be *ω-regular* under integer base $\beta \geq 2$ if there exists an aggregate function automaton $\mathcal{A}$ over alphabet $\Sigma \times \mathsf{AlphaRep}(\beta)$ such that for all sequences $A \in \Sigma^\omega$ and $x \in \mathbb{R}$, $f(A) = x$ iff $(A, \mathsf{rep}(x, \beta)) \in L(\mathcal{A})$.

Definition 2.3 (Weighted automaton [CDH10, Moh09])**.** A *weighted automaton* over infinite words is a tuple $\mathcal{A} = (\mathcal{M}, \gamma, f)$, where $\mathcal{M} = (S, \Sigma, \delta, \mathit{Init}, S)$ is a Büchi automaton with all states as accepting, $\gamma : \delta \to \mathbb{Q}$ is a *weight function*, and $f : \mathbb{Q} \to \mathbb{R}$ is the *aggregate function*.

Words and *runs* in weighted automata are defined as they are in Büchi automata. The *weight-sequence* of run $\rho = s_0 s_1 \dots$ of word $w = w_0 w_1 \dots$ is given by $wt_\rho = n_0 n_1 n_2 \dots$ where $n_i = \gamma(s_i, w_i, s_{i+1})$ for all i. The *weight of a run* ρ, denoted by $f(\rho)$, is given by $f(wt_\rho)$. Here the *weight of a word* $w \in \Sigma^\omega$ in weighted automata is defined as $wt_\mathcal{A}(w) = sup\{f(\rho) | \rho$ is a run of w in $\mathcal{A}\}$. In general, weight of a word can also be defined as the infimum of the weight of all its runs. Note, an automaton need not accept every word, even though all its states are accepting, since it need not be complete. By convention, if a word $w \notin \mathcal{L}(\mathcal{M})$ its weight $wt_\mathcal{A}(w) = -\infty$.

Definition 2.4 (Quantitative inclusion)**.** Let P and Q be weighted ω-automata with the *same* aggregate function f. The *strict quantitative inclusion problem*, denoted by $P \subset_f Q$, asks whether for all words $w \in \Sigma^\omega$, $wt_P(w) < wt_Q(w)$. The *non-strict quantitative inclusion problem*, denoted by $P \subseteq_f Q$, asks whether for all words $w \in \Sigma^\omega$, $wt_P(w) \leq wt_Q(w)$.

Quantitative inclusion, strict and non-strict, is PSPACE-complete for limsup and liminf [CDH10]. Non-strict quantitative inclusion is undecidable for limit-average [DDG$^+$10], while decidability of the strict variant is still open. For discounted-sum with integer discount-factor it is in EXPTIME [BH14, CDH10], and decidability is unknown for rational discount-factors.

Definition 2.5 (Incomplete-information quantitative games)**.** An *incomplete-information quantitative game* is a tuple $\mathcal{G} = (S, s_\mathcal{I}, O, \Sigma, \delta, \gamma, f)$, where S, O, Σ are sets of *states*, *observations*, and *actions*, respectively, $s_\mathcal{I} \in S$ is the *initial state*, $\delta \subseteq S \times \Sigma \times S$ is the *transition relation*, $\gamma : S \to \mathbb{N} \times \mathbb{N}$ is the *weight function*, and $f : \mathbb{N}^\omega \to \mathbb{R}$ is the *aggregate function*.

The transition relation δ is *complete*, i.e., for all states p and actions a, there exists a state q s.t. $(p, a, q) \in \delta$. A *play* ρ is a sequence $s_0 a_0 s_1 a_1 \dots$, where $\tau_i = (s_i, a_i, s_{i+1}) \in \delta$. The *observation of state* s, by abuse of notation, is denoted by $O(s) \in O$. The *observed play* o_ρ of ρ is the sequence $o_0 a_0 o_1 a a_1 \dots$, where $o_i = O(s_i)$. Player P_0 has incomplete information about the game $\mathcal{G}$; it only perceives the observation play o_ρ. Player P_1 receives full information and witnesses play ρ. Plays begin in the initial state $s_0 = s_\mathcal{I}$. For $i \geq 0$, Player P_0 selects action a_i. Next, player P_1 selects the state s_{i+1}, such that $(s_i, a_i, s_{i+1}) \in \delta$. The *weight of state* s is the pair of payoffs $\gamma(s) = (\gamma(s)_0, \gamma(s)_1)$. The *weight sequence* wt_i of player P_i along ρ is given by $\gamma(s_0)_i \gamma(s_1)_i \dots$, and its payoff from ρ is given by $f(wt_i)$ for aggregate function f, denoted by $f(\rho_i)$, for simplicity. A play on which a player receives a greater payoff than the other player is said to be a *winning play* for the player. A strategy for player P_0 is given by a function $\alpha : O^* \to \Sigma$ since it only sees observations. Player P_0 agrees with strategy α if for all i, $a_i = \alpha(o_0 \dots o_i)$. A strategy α is said to be a *winning strategy* for player P_0 if all plays agreeing with α are winning plays for P_0.

Definition 2.6 (Büchi pushdown automaton [CG77])**.** A *Büchi pushdown automaton (Büchi PDA)* is a tuple $\mathcal{A} = (S, \Sigma, \Gamma, \delta, Init, Z_0, \mathcal{F})$, where S, Σ, Γ, and $\mathcal{F}$ are finite sets of *states*, *input alphabet*, *pushdown alphabet* and *accepting states*, respectively. $\delta \subseteq (S \times \Gamma \times (\Sigma \cup \{\epsilon\}) \times S \times \Gamma)$ is the *transition relation*, $Init \subseteq S$ is a set of *initial states*, $Z_0 \in \Gamma$ is the *start symbol*.

A *run* ρ on a word $w = w_0 w_1 \dots \in \Sigma^\omega$ of a Büchi PDA $\mathcal{A}$ is a sequence of configurations $(s_0, \gamma_0), (s_1, \gamma_1) \dots$ satisfying (1) $s_0 \in Init$, $\gamma_0 = Z_0$, and (2) $(s_i, \gamma_i, w_i, s_{i+1}, \gamma_{i+1}) \in \delta$ for all i. Büchi PDA consists of a *stack*, elements of which are the tokens Γ, and initial element

Z_0. Transitions *push* or *pop* token(s) to/from the top of the stack. Let $inf(\rho)$ be the set of states that occur infinitely often in state sequence $s_0 s_1 \dots$ of run ρ. A run ρ is an *accepting run* in Büchi PDA if $inf(\rho) \cap \mathcal{F} \neq \emptyset$. A word w is an *accepting word* if it has an accepting run. Languages accepted by Büchi PDA are called *ω-context-free languages* (ω-CFL).

Notation and Terminology. For an infinite sequence $A = (a_0, a_1, \dots)$, $A[i]$ denotes its i-th element, and $A[n, m]$ denotes the finite word $A[n]A[n+1] \dots A[m]$. An infinite weight-sequence A is said to be *bounded* if there exists a value $b \in \mathbb{Q}$ such that $|A[i]| < b$ for all $i \geq 0$. Abusing notation, we write $w \in \mathcal{A}$ and $\rho \in \mathcal{A}$ if w and ρ are an accepting word and an accepting run of $\mathcal{A}$ respectively. The Symbol $\cdot$ is used to denote both multiplication of real numbers and concatenation of sequences. The meaning will be clear in context.

3. Comparator automata

Comparator automata (often abbreviated as *comparators*) are a class of automata that can read pairs of weight sequences synchronously and establish an equality or inequality relationship between these sequences. Formally, we define:

Definition 3.1 (Comparator automata)**.** Let Σ be a finite set of rational numbers, and $f : \mathbb{Q}^\omega \to \mathbb{R}$ denote an aggregate function. A *comparator automaton for aggregate function f* with inequality or equality relation $R \in \{\leq, <, \geq, >, =, \neq\}$ is an automaton over the alphabet $\Sigma \times \Sigma$ that accepts a pair (A, B) of (infinite) weight sequences iff $f(A)\ R\ f(B)$.

From now on, unless mentioned otherwise, we assume that all weight sequences are bounded, natural number sequences. The boundedness assumption is justified since the set of weights forming the alphabet of a comparator is bounded. For all aggregate functions considered in this paper, the result of comparison of weight sequences is preserved by a uniform linear transformation that converts rational-valued weights into natural numbers; justifying the natural number assumption.

When the comparator for an aggregate function and a relation is a Büchi automaton, we call it an *ω-regular comparator*. Likewise, when the comparator is a Büchi pushdown automaton, we call it an *ω-context-free comparator*.

Theorem 3.2. *Let Σ be a finite alphabet of natural numbers. Let $f : \mathbb{N}^\omega \to \mathbb{R}$ be an aggregate function. If the comparator automata for aggregate function f for any one inequality relation $R \in \{\leq, <, \geq, >\}$ is ω-regular, then the comparator for f is ω-regular for all relations in $\{\leq, <, \geq, >, =, \neq\}$.*

Proof sketch. Suppose the comparator automata $\mathcal{A}$ for relation $\leq$ is ω-regular for aggregate function f. Then the pair of weight sequences $(A, B) \in \mathcal{A}$ iff $f(A) \leq f(B)$ holds. Since $f(A) \leq f(B)$ iff $f(B) \geq f(A)$ the automaton obtained by altering transition $s \xrightarrow{(a,b)} t$ in $\mathcal{A}$ to transition $s \xrightarrow{(b,a)} t$ is the ω-regular comparator automata for relation $\geq$. The ω-regular comparator for $=$ can be obtained by taking the intersection of the comparator for $\leq$ and $\geq$ since Büchi automata are closed under intersection. Finally, since Büchi automata are also closed under complementation, we get that the comparator automata for the other three relations, namely $<$,$>$,$\neq$, is also ω-regular. □

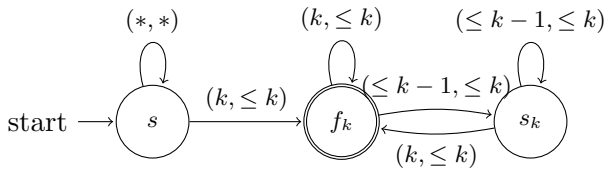

Figure 1: State f_k is an accepting state. Automaton $\mathcal{A}_k$ accepts (A, B) iff $\mathsf{LimSup}(\mathsf{A}) = k$, $\mathsf{LimSup}(\mathsf{B}) \leq k$. $*$ denotes $\{0, 1 \ldots \mu\}$, $\leq m$ denotes $\{0, 1 \ldots, m\}$

Later, we see that discounted-sum comparator is ω-regular (§ 4) and prefix-average comparator with $\geq$ (or $\leq$) is ω-context-free (§ 5).

Limsup comparator. We explain comparators through an example. The *limit supremum* (limsup, in short) of a bounded, integer sequence A, denoted by $\mathsf{LimSup}(\mathsf{A})$, is the largest integer that appears infinitely often in A. The *limsup comparator for relation* $\geq$ is a Büchi automaton that accepts the pair (A, B) of sequences iff $\mathsf{LimSup}(\mathsf{A}) \geq \mathsf{LimSup}(\mathsf{B})$.

The working of the limsup comparator for relation $\geq$ is based on non-deterministically guessing the limsup of sequences A and B, and then verifying that $\mathsf{LimSup}(\mathsf{A}) \geq \mathsf{LimSup}(\mathsf{B})$. Formal construction of the limsup comparator is given here. Suppose all sequences are natural number sequences, bounded by μ. The limsup comparator is the Büchi automaton $\mathcal{A}_{\succ LS} = (S, \Sigma, \delta, \mathit{Init}, \mathcal{F})$ where,

- $S = \{s\} \cup \{s_0, s_1 \ldots, s_\mu\} \cup \{f_0, f_1 \ldots, f_\mu\}$
- $\Sigma = \{(a, b) : 0 \leq a, b \leq \mu\}$ where a and b are integers.
- $\delta \subseteq S \times \Sigma \times S$ is defined as follows:
  (1) Transitions from start state s: $(s, (a, b), p)$ for all $(a, b) \in \Sigma$, and for all $p \in \{s\} \cup \{f_0, f_1, \ldots, f_\mu\}$.
  (2) Transitions between f_k and s_k for each k:
    i (f_k, α, f_k) for $\alpha \in \{k\} \times \{0, 1, \ldots k\}$.
    ii (f_k, α, s_k) for $\alpha \in \{0, 1, \ldots k-1\} \times \{0, 1, \ldots k\}$.
    iii (s_k, α, s_k) for $\alpha \in \{0, 1, \ldots k-1\} \times \{0, 1, \ldots k\}$.
    iv (s_k, α, f_k) for $\alpha \in \{k\} \times \{0, 1, \ldots k\}$.
- $\mathit{Init} = \{s\}$
- $\mathcal{F} = \{f_0, f_1 \ldots, f_\mu\}$

Fig. 1 illustrates the basic building block of the limsup comparator for relation $\geq$. For $k \in \{0 \ldots, \mu\}$, Fig 1 represents the segment of the limsup comparator consisting of the initial state s, accepting state f_k and state s_k. We denote this by Büchi automaton $\mathcal{A}_k$. We show that automaton $\mathcal{A}_k$ accepts pair (A, B) of number sequences iff $\mathsf{LimSup}(\mathsf{A}) = k$, and $\mathsf{LimSup}(\mathsf{B}) \leq k$, for integer k. (Lemma 3.3).

Lemma 3.3. *Let A and B be non-negative integer sequences bounded by μ. Büchi automaton $\mathcal{A}_k$ (Fig. 1) accepts (A, B) iff $\mathsf{LimSup}(\mathsf{A}) = k$, and $\mathsf{LimSup}(\mathsf{A}) \geq \mathsf{LimSup}(\mathsf{B})$.*

Proof. Let (A, B) have an accepting run in $\mathcal{A}_k$. We show that $\mathsf{LimSup}(\mathsf{A}) = k \geq \mathsf{LimSup}(\mathsf{B})$. The accepting run visits state f_k infinitely often. Note that all incoming transitions to accepting state f_k occur on alphabet $(k, \leq k)$ while all transitions between states f_k and s_k occur on alphabet $(\leq k-1, \leq k)$, where $\leq k$ denotes the set $\{0, 1, \ldots k\}$. So, the integer k must appear infinitely often in A and all elements occurring infinitely often in A and B

are less than or equal to k. Therefore, if (A, B) is accepted by $\mathcal{A}_k$ then $\mathsf{LimSup}(\mathsf{A}) = k$, and $\mathsf{LimSup}(\mathsf{B}) \leq k$, and $\mathsf{LimSup}(\mathsf{A}) \geq \mathsf{LimSup}(\mathsf{B})$.

Conversely, let $\mathsf{LimSup}(\mathsf{A}) = k > \mathsf{LimSup}(\mathsf{B})$. We prove that (A, B) is accepted by $\mathcal{A}_k$. For an integer sequence A when $\mathsf{LimSup}(\mathsf{A}) = k$ integers greater than k can occur only a finite number of times in A. Let l_A denote the index of the last occurrence of an integer greater than k in A. Similarly, since $\mathsf{LimSup}(\mathsf{B}) \leq k$, let l_B be index of the last occurrence of an integer greater than k in B. Therefore, for sequences A and B integers greater than k will not occur beyond index $l = max(l_A, l_B)$. Büchi automaton $\mathcal{A}_k$ (Fig. 1) non-deterministically determines l. On reading the l-th element of input word (A, B), the run of (A, B) exits the start state s and shifts to accepting state f_k. Note that all runs beginning at state f_k occur on alphabet (a, b) where $a, b \leq k$. Therefore, (A, B) can continue its infinite run even after transitioning to f_k. To ensure that this is an accepting run, the run must visit accepting state f_k infinitely often. But this must be the case, as transition on alphabet (k, k') for $k' \leq k$ must be taken infinitely often as k is the limsup of A and limsup of B is less than or equal to k. Transitions on this alphabet always return to the accepting state f_k. Hence, for all integer sequences A,B bounded by μ, if $\mathsf{LimSup}(\mathsf{A}) = k$, and $\mathsf{LimSup}(\mathsf{A}) \geq \mathsf{LimSup}(\mathsf{B})$, the automaton accepts (A, B). □

Theorem 3.4. *There exists an ω-regular comparator for the limsup aggregation function.*

Proof. The construction given above contains a Büchi automata $\mathcal{A}_k$ for all $k \in \{0, \ldots, \mu\}$. Therefore, from Lemma 3.3, we conclude that the construction corresponds to the limsup comparator with inequality $\geq$. Therefore, limsup comparator with relation $\geq$ is ω-regular.

From Lemma 3.2 we know that limsup comparator is ω-regular for all relations. □

Due to closure properties of Büchi automata, this implies that limsup comparator for all inequalities and equality relation is also ω-regular. The *limit infimum* (liminf, in short) of an integer sequence is the smallest integer that appears infinitely often in it; its comparator has a similar construction to the limsup comparator. One can further prove that the limsup and liminf aggregate functions are also ω-regular aggregate functions.

3.1. ω-Regular aggregate functions.

This section draws out the relationship between ω-regular aggregate functions and ω-regular comparators. We begin with the following Lemma in order to show that ω-regular aggregate functions entail ω-regular comparators for the aggregate function.

Lemma 3.5. *Let $\mu > 0$ be the upper-bound on weight sequences, and $\beta \geq 2$ be the integer base. Then there exists a Büchi automaton $\mathcal{A}_\beta$ such that for all $a, b \in \mathbb{R}$, $\mathcal{A}_\beta$ accepts $(\mathsf{rep}(a, \beta), \mathsf{rep}(b, \beta))$ iff $a > b$.*

Proof. Let $a, b \in \mathbb{R}$, and $\beta > 2$ be an integer base. Let $\mathsf{rep}(a, \beta) = \mathsf{sign}_a \cdot (\mathsf{Int}(a, \beta), \mathsf{Frac}(a, \beta))$ and $\mathsf{rep}(b, \beta) = \mathsf{sign}_b \cdot (\mathsf{Int}(b, \beta), \mathsf{Frac}(b, \beta))$. Then, the following statements can be proven using simple evaluation from definitions:

- When $\mathsf{sign}_a = +$ and $\mathsf{sign}_b = -$. Then $a > b$.
- When $\mathsf{sign}_a = \mathsf{sign}_b = +$
  - If $\mathsf{Int}(a, \beta) \neq \mathsf{Int}(b, \beta)$: Since $\mathsf{Int}(a, \beta)$ and $\mathsf{Int}(b, \beta)$ eventually only see digit 0 i.e. they are necessarily identical eventually, there exists an index i such that it is the last position where $\mathsf{Int}(a, \beta)$ and $\mathsf{Int}(b, \beta)$ differ. If $\mathsf{Int}(a, \beta)[i] > \mathsf{Int}(b, \beta)[i]$, then $a > b$. If $\mathsf{Int}(a, \beta)[i] < \mathsf{Int}(b, \beta)[i]$, then $a < b$.

  - If $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$ but $\mathsf{Frac}(a,\beta) \neq \mathsf{Frac}(b,\beta)$: Let i be the first index where $\mathsf{Frac}(a,\beta)$ and $\mathsf{Frac}(b,\beta)$ differ. If $\mathsf{Frac}(a,\beta)[i] > \mathsf{Frac}(b,\beta)[i]$ then $a > b$. If $\mathsf{Frac}(a,\beta)[i] < \mathsf{Frac}(b,\beta)[i]$ then $a < b$.
  - Finally, if $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$ and $\mathsf{Frac}(a,\beta) = \mathsf{Frac}(b,\beta)$: Then $a = b$.
- When $\mathsf{sign}_a = \mathsf{sign}_b = -$
  - If $\mathsf{Int}(a,\beta) \neq \mathsf{Int}(b,\beta)$: Since $\mathsf{Int}(a,\beta)$ and $\mathsf{Int}(b,\beta)$ eventually only see digit 0 i.e. they are necessarily identical eventually. Therefore, there exists an index i such that it is the last position where $\mathsf{Int}(a,\beta)$ and $\mathsf{Int}(b,\beta)$ differ. If $\mathsf{Int}(a,\beta)[i] > \mathsf{Int}(b,\beta)[i]$, then $a < b$. If $\mathsf{Int}(a,\beta)[i] < \mathsf{Int}(b,\beta)[i]$, then $a > b$.
  - If $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$ but $\mathsf{Frac}(a,\beta) \neq \mathsf{Frac}(b,\beta)$: Let i be the first index where $\mathsf{Frac}(a,\beta)$ and $\mathsf{Frac}(b,\beta)$ differ. If $\mathsf{Frac}(a,\beta)[i] > \mathsf{Frac}(b,\beta)[i]$ then $a < b$. If $\mathsf{Frac}(a,\beta)[i] < \mathsf{Frac}(b,\beta)[i]$ then $a > b$.
  - Finally, if $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$ and $\mathsf{Frac}(a,\beta) = \mathsf{Frac}(b,\beta)$: Then $a = b$.
- When $\mathsf{sign}_a = -$ and $\mathsf{sign}_b = +$. Then $a < b$.

Since the conditions given above are exhaustive and mutually exclusive, we conclude that for all $a, b \in \mathbb{R}$ and integer base $\beta \geq 2$, letting $\mathsf{rep}(a,\beta) = \mathsf{sign}_a \cdot (\mathsf{Int}(a,\beta), \mathsf{Frac}(a,\beta))$ and $\mathsf{rep}(b,\beta) = \mathsf{sign}_b \cdot (\mathsf{Int}(b,\beta), \mathsf{Frac}(b,\beta))$, $a > b$ iff one of the following conditions occurs:

(1) $\mathsf{sign}_a = +$ and $\mathsf{sign}_b = -$.
(2) $\mathsf{sign}_a = \mathsf{sign}_b = +$, $\mathsf{Int}(a,\beta) \neq \mathsf{Int}(b,\beta)$, and $\mathsf{Int}(a,\beta)[i] > \mathsf{Int}(b,\beta)[i]$ when i is the last index where $\mathsf{Int}(a,\beta)$ and $\mathsf{Int}(b,\beta)$ differ.
(3) $\mathsf{sign}_a = \mathsf{sign}_b = +$, $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$, $\mathsf{Frac}(a,\beta) \neq \mathsf{Frac}(b,\beta)$, and $\mathsf{Int}(a,\beta)[i] > \mathsf{Int}(b,\beta)[i]$ when i is the first index where $\mathsf{Frac}(a,\beta)$ and $\mathsf{Frac}(b,\beta)$ differ.
(4) $\mathsf{sign}_a = \mathsf{sign}_b = -$, $\mathsf{Int}(a,\beta) \neq \mathsf{Int}(b,\beta)$, and $\mathsf{Int}(a,\beta)[i] < \mathsf{Int}(b,\beta)[i]$ when i is the last index where $\mathsf{Int}(a,\beta)$ and $\mathsf{Int}(b,\beta)$ differ.
(5) $\mathsf{sign}_a = \mathsf{sign}_b = -$, $\mathsf{Int}(a,\beta) = \mathsf{Int}(b,\beta)$, $\mathsf{Frac}(a,\beta) \neq \mathsf{Frac}(b,\beta)$, and $\mathsf{Int}(a,\beta)[i] < \mathsf{Int}(b,\beta)[i]$ when i is the first index where $\mathsf{Frac}(a,\beta)$ and $\mathsf{Frac}(b,\beta)$ differ.

Note that each of these five conditions can be easily expressed by a Büchi automaton over alphabet $\mathsf{AlphaRep}(\beta)$ for an integer $\beta \geq 2$. For an integer $\beta \geq 2$, the union of all these Büchi automata will result in a Büchi automaton $\mathcal{A}_\beta$ such that for all $a, b \in \mathbb{R}$ and $A = \mathsf{rep}(a,\beta)$ and $B = \mathsf{rep}(b,\beta)$, $a > b$ iff interleaved word $(A,B) \in \mathcal{L}(\mathcal{A}_\beta)$. □

We finally show that ω-regular aggregate functions entail ω-regular comparators for the aggregate function.

Theorem 3.6. *Let $\mu > 0$ be the upper-bound on weight sequences, and $\beta \geq 2$ be the integer base. Let $f : \{0, 1, \ldots, \mu\}^\omega \to \mathbb{R}$ be an aggregate function. If aggregate function f is ω-regular under base β, then its comparator for all inequality and equality relations is also ω-regular.*

Proof. We show that if an aggregate function is ω-regular under base β, then its comparator for relation $>$ is ω-regular. By closure properties of ω-regular comparators, this implies that comparators of the aggregate function are ω-regular for all inequality and equality relations.

Let $f : \Sigma^\omega \to \mathbb{R}$ be an ω-regular aggregate function with aggregate function automata $\mathcal{A}_f$. We will construct an ω-regular comparator for f with relation $>$. From Lemma 3.5 we know that (X,Y) is present in the comparator iff $(X,M), (Y,N) \in \mathcal{A}_f$ for $M, N \in \mathsf{AlphaRep}(\beta)^\omega$ and $(M,N) \in \mathcal{A}_\beta$, for $\mathcal{A}_\beta$ as described above. Since $\mathcal{A}_f$ and $\mathcal{A}_\beta$ are both Büchi automata, the comparator for function f with relation $>$ is also a Büchi auotmaton. Therefore, the comparator for aggregate function f with relation $>$ is ω-regular. □

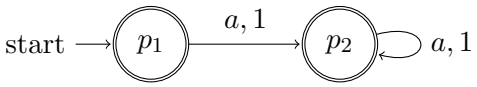

Figure 2: Weighted automaton P

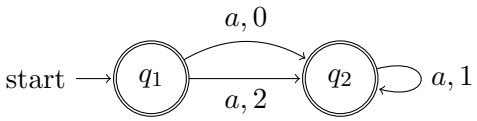

Figure 3: Weighted automaton Q

The converse direction of whether ω-regular comparator for an aggregate function f for all inequality or equality relations will entail ω-regular functions under an integer base $\beta \geq 0$ is trickier. For all aggregate functions considered in this paper, we see that whenever the comparator is ω-regular, the aggregate function is ω-regular as well. However, the proofs for this have been done on a case-by-cass basis, and we do not have an algorithmic procedure to derive a function (Büchi) automaton from its ω-regular comparator. We also do not have an example of an aggregate function for which the comparator is ω-regular but the function is not. Therefore, we arrive at the following conjecture:

Conjecture 3.7. Let $\mu > 0$ be the upper-bound on weight sequences, and $\beta \geq 2$ be the integer base. Let $f : \{0, 1, \ldots, \mu\}^\omega \to \mathbb{R}$ be an aggregate function. If the comparator for an aggregate function f is ω-regular for all inequality and equality relations, then its aggregate function is also ω-regular under base β.

3.2. **Quantitative inclusion.** The aggregate function or comparator of a quantitative inclusion problem refer to the aggregate function or comparator of the associated aggregate function. This section presents a generic algorithm (Algorithm 1) to solve quantitative inlcusion between ω-weighted automata P and Q with ω-comparators. This section focusses on the non-strict quantitative inclusion. InclusionReg (Algorithm 1) is an algorithm for quantitative inclusion between weighted ω-automata P and Q with ω-regular comparator $\mathcal{A}_f$ for relation $\geq$. InclusionReg takes P,Q and $\mathcal{A}_f$ as input, and returns True iff $P \subseteq_f Q$. The results for strict quantitative inclusion are similar. We use the following motivating example to explain steps of Algorithm 1.

Motivating example. Let weighted ω-automata P and Q be as illustrated in Fig. 2-3 with the limsup aggregate function. The word $w = a^\omega$ has one run $\rho_1^P = p_1 p_2^\omega$ with weight sequence $wt_1^P = 1^\omega$ in P and two runs $\rho_1^Q = q_1 q_2^\omega$ with weight sequence $wt_1^Q = 0, 1^\omega$ and run $\rho_2^Q = q_1 q_2^\omega$ with weight sequence $wt_2^Q = 2, 1^\omega$. Clearly, $wt_P(w) \leq wt_Q(w)$. Therefore $P \subseteq_f Q$. From Theorem 3.4 we know that the limsup comparator $\mathcal{A}_{\mathsf{LS}}^{\leq}$ for $\leq$ is ω-regular.

We use Algorithm 1 to show that $P \subseteq_f Q$ using its ω-regular comparator for $\leq$. Intuitively, the algorithm must be able to identify that for run ρ_1^P of w in P, there exists a run ρ_2^Q in Q s.t. (wt_1^P, wt_2^Q) is accepted by the limsup comparator for $\leq$.

Key ideas. A run ρ_P in P on word $w \in \Sigma^\omega$ is said to be *dominated* w.r.t $P \subseteq_f Q$ if there exists a run ρ_Q in Q on the same word w such that $wt_P(\rho_P) \leq wt_Q(\rho_Q)$. $P \subseteq_f Q$ holds if for every run ρ_P in P is dominated w.r.t. $P \subseteq_f Q$.

InclusionReg constructs Büchi automaton *Dom* that consists of exactly the dominated runs of P w.r.t $P \subseteq_f Q$. InclusionReg returns True iff *Dom* contains all runs of P. To obtain *Dom*, it constructs Büchi automaton *DomProof* that accepts word (ρ_P, ρ_Q) iff ρ_P and ρ_Q are runs of the same word in P and Q respectively, and $wt_P(\rho_P) \leq wt_Q(\rho_Q)$ i.e. if w_P and

Algorithm 1 InclusionReg$(P, Q, \mathcal{A}_f)$, Is $P \subseteq_f Q$?

1: **Input:** Weighted automata P, Q, and ω-regular comparator $\mathcal{A}_f$ (Inequality $\leq$)
2: **Output:** True if $P \subseteq_f Q$, False otherwise
3: $\hat{P} \leftarrow$ AugmentWtAndLabel(P)
4: $\hat{Q} \leftarrow$ AugmentWtAndLabel(Q)
5: $\hat{P} \times \hat{Q} \leftarrow$ MakeProduct$(\hat{P}, \hat{Q})$
6: *DomProof* $\leftarrow$ Intersect$(\hat{P} \times \hat{Q}, \mathcal{A}_{\geq})$
7: *Dom* $\leftarrow$ FirstProject(*DomProof*)
8: **return** $\hat{P} \equiv$ *Dom*

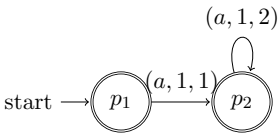

Figure 4: $\hat{P}$

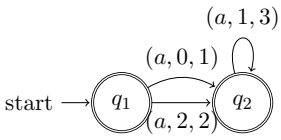

Figure 5: $\hat{Q}$

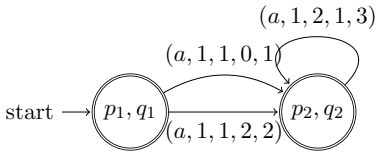

Figure 6: $\hat{P} \times \hat{Q}$

w_Q are weight sequence of ρ_P and ρ_Q, respectively, then (w_P, w_Q) is present in the ω-regular comparator $\mathcal{A}_f^{\leq}$ for aggregate function f with relation $\leq$. The projection of *DomProof* on runs of P results in *Dom*.

Algorithm details. For sake a simplicity, we assume that every word present in P is also present in Q i.e. $P \subseteq Q$ (qualitative inclusion). InclusionReg has three steps: (a). UniqueId (Lines 3-4): Enables unique identification of runs in P and Q through *labels*. (b). Compare (Lines 5-7): Compares weight of runs in P with weight of runs in Q, and constructs *Dom*. (c). DimEnsure (Line 8): Ensures if all runs of P are diminished.

(1) UniqueId: AugmentWtAndLabel transforms weighted ω-automaton $\mathcal{A}$ into Büchi automaton $\hat{\mathcal{A}}$ by converting transition $\tau = (s, a, t)$ with weight $\gamma(\tau)$ in $\mathcal{A}$ to transition $\hat{\tau} = (s, (a, \gamma(\tau), l), t)$ in $\hat{\mathcal{A}}$, where l is a unique label assigned to transition τ. The word $\hat{\rho} = (a_0, n_0, l_0)(a_1, n_1, l_1) \cdots \in \hat{A}$ iff there exists a run $\rho \in \mathcal{A}$ on word $a_0 a_1 \ldots$ with weight sequence $n_0 n_1 \ldots$. Labels ensure bijection between runs in $\mathcal{A}$ and words in $\hat{\mathcal{A}}$. Words of $\hat{A}$ have a single run in $\hat{A}$. Hence, transformation of weighted ω-automata P and Q to Büchi automata $\hat{P}$ and $\hat{Q}$ enables disambiguation between runs of P and Q (Line 3-4).

The corresponding $\hat{A}$ for weighted ω-automata P and Q from Figure 2- 3 are given in Figure 4- 5 respectively.

(2) Compare: The output of this step is the Büchi automaton *Dom* that contains the word $\hat{\rho} \in \hat{P}$ iff ρ is a dominated run in P w.r.t $P \subseteq_f Q$ (Lines 5-7).

MakeProduct$(\hat{P}, \hat{Q})$ constructs $\hat{P} \times \hat{Q}$ s.t. word $(\hat{\rho_P}, \hat{\rho_Q}) \in \hat{P} \times \hat{Q}$ iff ρ_P and ρ_Q are runs of the same word in P and Q respectively (Line 5). Concretely, for transition $\hat{\tau_{\mathcal{A}}} = (s_{\mathcal{A}}, (a, n_{\mathcal{A}}, l_{\mathcal{A}}), t_{\mathcal{A}})$ in automaton $\mathcal{A}$, where $\mathcal{A} \in \{\hat{P}, \hat{Q}\}$, transition $\hat{\tau_P} \times \hat{\tau_Q} = ((s_P, s_Q), (a, n_P, l_P, n_Q, l_Q), (t_P, t_Q))$ is in $\hat{P} \times \hat{Q}$, as shown in Figure 6.

Intersect intersects the weight components of $\hat{P} \times \hat{Q}$ with comparator $\mathcal{A}_f^{\leq}$ (Line 6). The resulting automaton *DomProof* accepts word $(\hat{\rho_P}, \hat{\rho_Q})$ iff $f(\rho_P) \leq f(\rho_Q)$, and ρ_P

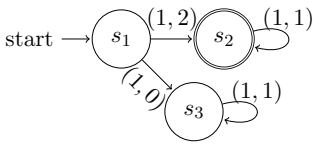

Figure 7: Snippet of limsup comparator $\mathcal{A}^{\leq}_{\mathsf{LS}}$ for relation $\leq$

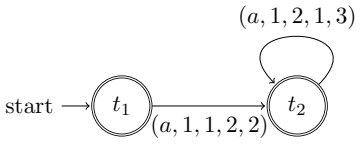

Figure 8: Snippet of Intersect. $t_1 = (p_1, q_1, s_1)$ and $t_2 = (p_2, q_2, s_2)$.

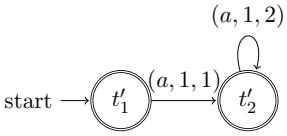

Figure 9: Snippet of *Dom*

and ρ_Q are runs on the same word in P and Q respectively. The result of Intersect between $\hat{P} \times \hat{Q}$ with the limsup comparator $\mathcal{A}^{\leq}_{\mathsf{LS}}$ for relation $\leq$ (Figure 7) is given in Figure 8.

The projection of *DomProof* on the alphabet of $\hat{P}$ returns *Dom* which contains the word $\hat{\rho_P}$ iff ρ_P is a dominated run in P w.r.t $P \subseteq_f Q$ (Line 7), as shown in Figure 9.

(3) DimEnsure: $P \subseteq_f Q$ iff $\hat{P} \equiv Dom$ (qualitative equivalence) since $\hat{P}$ consists of all runs of P and *Dom* consists of all dominated runs w.r.t $P \subseteq_f Q$ (Line 8).

Lemma 3.8. *Büchi automaton Dom consists of all dominated runs in P w.r.t $P \subseteq_f Q$.*

Proof. Let $\mathcal{A}^{\leq}_f$ be the comparator for ω-regular aggregate function f and relation $\leq$ s.t. $\mathcal{A}_f$ accepts (A, B) iff $f(A) \leq f(B)$. A run ρ over word w with weight sequence wt in P (or Q) is represented by the unique word $\hat{\rho} = (w, wt, l)$ in $\hat{P}$ (or $\hat{Q}$) where l is the unique label sequence associated with each run in P (or Q). Since every label on each transition is unique, $\hat{P}$ and $\hat{Q}$ are deterministic automata. Now, $\hat{P} \times \hat{Q}$ is constructed by ensuring that two transitions are combined in the product only if their alphabet is the same. Therefore if $(w, wt_1, l_1, wt_2, l_2) \in \hat{P} \times \hat{Q}$, then $\hat{\rho} = (w, wt_1, l_1) \in \hat{P}$, $\hat{\sigma} = (w, wt_2, l_2) \in \hat{Q}$. Hence, there exist runs ρ and σ with weight sequences wt_1 and wt_2 in P and Q, respectively. Next, $\hat{P} \times \hat{Q}$ is intersected over the weight sequences with ω-regular comparator $\mathcal{A}^{\leq}_f$ for aggregate function f and relation $\leq$. Therefore $(w, wt_1, l_1, wt_2, l_2) \in DomProof$ iff $f(wt_1) \leq f(wt_2)$. Therefore runs ρ in P and σ in Q are runs on the same word s.t. aggregate weight in P is less than or equal to that of σ in Q. Therefore *Dom* consists of $\hat{\rho}$ only if ρ is a dominated run in P w.r.t $P \subseteq_f Q$.

Every step of the algorithm has a two-way implication, hence it is also true that every dominated run in P w.r.t $P \subseteq_f Q$ is present in *Dom*. □

Lemma 3.9. *Given weighted ω-automata P and Q and their ω-regular comparator $\mathcal{A}^{\leq}_f$ for aggregate function f and relation $\leq$.* InclusionReg$(P, Q, \mathcal{A}_f)$ *returns* True *iff $P \subseteq_f Q$.*

Proof. $\hat{P}$ consists of all runs of P. *Dom* consists of all dominated run in P w.r.t $P \subseteq_f Q$. $P \subseteq_f Q$ iff every run of P is dominated w.r.t $P \subseteq_f Q$. Therefore $P \subseteq_f Q$ is given by whether $\hat{P} \equiv Dom$, where $\equiv$ denotes qualitative equivalence. □

Algorithm InclusionReg is adapted for *strict* quantitative inclusion $P \subset_f Q$ by repeating the same procedure with ω-regular comparator $\mathcal{A}^{<}_f$ for aggregate function f and relation $<$. Here, a run ρ_P in P on word $w \in \Sigma^\omega$ is said to be *dominated* w.r.t $P \subset_f Q$ if there exists a run ρ_Q in Q on the same word w such that $wt_P(\rho_P) < wt_Q(\rho_Q)$. Similarly for quantitative equivalence $P \equiv_f Q$.

We give the complexity analysis of quantitative-inclusion with ω-regular comparators.

Theorem 3.10. *Let P and Q be weighted ω-automata and $\mathcal{A}_f$ be an ω-regular comparator. Quantitative inclusion problem, quantitative strict-inclusion problem, and quantitative equivalence problem for ω-regular aggregate function f is* PSPACE*-complete.*

Proof. All operations in InclusionReg until Line 7 are polytime operations in the size of weighted ω-automata P, Q and comparator $\mathcal{A}_f$. Hence, *Dom* is polynomial in size of P, Q and $\mathcal{A}_f$. Line 8 solves a PSPACE-complete problem. Therefore, the quantitative inclusion for ω-regular aggregate function f is in PSPACE in size of the inputs P, Q, and $\mathcal{A}_f$.

The PSPACE-hardness of the quantitative inclusion is established via reduction from the *qualitative* inclusion problem, which is PSPACE-complete. The formal reduction is as follows: Let P and Q be Büchi automata (with all states as accepting states). Reduce P, Q to weighted automata $\overline{P}$, $\overline{Q}$ by assigning a weight of 1 to each transition. Since all runs in $\overline{P}$, $\overline{Q}$ have the same weight sequence, weight of all words in $\overline{P}$ and $\overline{Q}$ is the same for any function f. It is easy to see $P \subseteq Q$ (qualitative inclusion) iff $\overline{P} \subseteq_f \overline{Q}$ (quantitative inclusion). □

Theorem 3.10 extends to weighted ω-automata when weight of words is the *infimum* of weight of runs. The key idea for $P \subseteq_f Q$ here is to ensure that for every run ρ_Q in Q there exists a run on the same word in ρ_P in P s.t. $f(\rho_P) \leq f(\rho_Q)$.

Representation of counterexamples. When $P \not\subseteq_f Q$, there exists word(s) $w \in \Sigma^*$ s.t $wt_P(w) > wt_Q(w)$. Such a word w is said to be a *counterexample word*. Previously, finite-state representations of counterexamples have been useful in verification and synthesis in qualitative systems [BK$^+$08], and could be useful in quantitative settings as well. However, we are not aware of procedures for such representations in the quantitative settings. Here we show that a trivial extension of InclusionReg yields Büchi automata-representations for all counterexamples of the quantitative inclusion problem for ω-regular functions.

Theorem 3.11. *All counterexamples of the quantitative inclusion problem for an ω-regular aggregate function can be expressed by a Büchi automaton.*

Proof. For word w to be a counterexample, it must contain a run in P that is not dominated. Clearly, all non-dominated runs of P w.r.t to the quantitative inclusion are members of $\hat{P} \setminus Dom$. The counterexamples words can be obtained from $\hat{P} \setminus Dom$ by modifying its alphabet to the alphabet of P by dropping transition weights and their unique labels. □

3.3. **Incomplete-information quantitative games.** Given an incomplete-information quantitative game $\mathcal{G} = (S, s_{\mathcal{I}}, O, \Sigma, \delta, \gamma, f)$, our objective is to determine if player P_0 has a winning strategy $\alpha : O^* \to \Sigma$ for ω-regular aggregate function f. We assume we are given the ω-regular comparator $\mathcal{A}_f$ for function f. We provide an informal description of the algorithm to describe the intuition.

Note that a function $A^* \to B$ can be treated like a B-labeled A-tree, and vice-versa. Hence, we proceed by finding a Σ-labeled O-tree — the *winning strategy tree*. Every branch of a winning strategy-tree is an observed play o_ρ of $\mathcal{G}$ for which every actual play ρ is a winning play for P_0.

We first consider all *game trees* of $\mathcal{G}$ by interpreting $\mathcal{G}$ as a tree-automaton over Σ-labeled S-trees. Nodes $n \in S^*$ of the game-tree correspond to states in S and labeled by actions in Σ taken by player P_0. Thus, the *root node* ε corresponds to $s_{\mathcal{I}}$, and a node $s_{i_0}, \ldots, s_{i_k}$ corresponds to the state s_{i_k} reached via $s_{\mathcal{I}}, s_{i_0}, \ldots, s_{i_{k-1}}$. Consider now a node

x corresponding to state s and labeled by an action σ. Then x has children $xs_1, \ldots xs_n$, for every $s_i \in S$. If $s_i \in \delta(s, \sigma)$, then we call xs_i a *valid* child, otherwise we call it an *invalid* child. Branches that contain invalid children correspond to invalid plays.

A game-tree τ is a *winning tree* for player P_0 if every branch of τ is either a winning play for P_0 or an invalid play of $\mathcal{G}$. One can check, using an automata, if a play is invalid by the presence of invalid children. Furthermore, the winning condition for P_0 can be expressed by the ω-regular comparator $\mathcal{A}_f$ that accepts (A, B) iff $f(A) > f(B)$. To use the comparator $\mathcal{A}_f$, it is determinized to Parity automaton D_f. Thus, a product of game $\mathcal{G}$ with D_f is a deterministic Parity tree-automaton accepting precisely winning-trees for player P_0.

Winning trees for player P_0 are Σ-labeled S-trees. We need to convert them to Σ-labeled O-trees. Recall that every state has a unique observation. We can simulate these Σ-labeled S-trees on strategy trees using the technique of *thinning* states S to observations O [KV00]. The resulting alternating Parity tree automaton $\mathcal{M}$ will accept a Σ-labeled O-tree τ_o iff for all actual game-tree τ of τ_o, τ is a winning-tree for P_0 with respect to the strategy τ_o. The problem of existence of winning-strategy for P_0 is then reduced to non-emptiness checking of $\mathcal{M}$.

Using the above, we get the following result: Given an incomplete-information quantitative game $\mathcal{G}$ and ω-regular comparator $\mathcal{A}_f$ for the aggregate function f, the time complexity of determining whether P_0 has a winning strategy is exponential in $|\mathcal{G}| \cdot |D_f|$, where $|D_f| = |\mathcal{A}_f|^{O(|\mathcal{A}_f|)}$.

Observe that since D_f is obtained by determinization of A_f, we obtain that $|D_f| = |\mathcal{A}_f|^{O(|\mathcal{A}_f|)}$. The thinning operation is linear in size of $|\mathcal{G} \times D_f|$, therefore $|\mathcal{M}| = |\mathcal{G}| \cdot |D_f|$. Non-emptiness checking of alternating Parity tree automata is exponential. Therefore, our procedure is doubly exponential in size of the comparator and exponential in size of the game. The question of tighter bounds is open.

4. Discounted-sum comparator

The discounted-sum of an infinite sequence A with discount-factor $d > 1$, denoted by $DS(A, d)$, is defined as $\Sigma_{i=0}^{\infty} A[i]/d^i$, and the discounted-sum of a finite sequence A is $\Sigma_{i=0}^{|A|-1} A[i]/d^i$. The discounted-sum comparator (DS-comparator, in short) for discount-factor d and relation R, denoted by $\mathcal{A}_d^R$, accepts a pair (A, B) of (infinite length) weight sequences iff $DS(A, d)$ R $DS(B, d)$. We investigate properties of the DS-comparator, and show that the DS-comparator is ω-regular iff the discount-factor $d > 1$ is an integer. We also show that the discounted-sum aggregate function is ω-regular iff the discount-factor is an integer. Finally, we show the repercussions of the above results on quantitative inclusion with discounted-sum aggregate function (DS-inclusion, in short). Section 4.1 and Section 4.2 deal with the non-integer rational discount-factors and integer discount-factors, respectively.

4.1. Non-integer, rational discount-factor.

We prove that for non-integer discount factors, the discounted-sum comparator is not ω-regular. For a weighted ω-automaton $\mathcal{A}$ and a real number $r \in \mathbb{R}$, the *cut-point language* of $\mathcal{A}$ w.r.t. r is defined as $L^{\geq r} = \{w \in L(\mathcal{A}) | wt_{\mathcal{A}}(w) \geq r\}$ [CDH09]. When the discount factor is a rational value $1 < d < 2$, it is known that not all deterministic weighted ω-automaton with discounted-sum aggregate function (DS-automaton, in short) have an ω-regular cut-point language for an $r \in \mathbb{R}$ [CDH09]. In this section, we extended this result to all non-integer, rational

discount-factors $d > 1$. Finally, we use this to prove that discounted-sum is not an ω-regular aggregate function when its discount-factor is a non-integer rational number.

Ambiguous Words. Let $d > 2$ be a non-integer, rational discount-factor. We consider finite weight-sequences over the alphabet $\{0, 1, \ldots, \lceil d \rceil - 1\}$. We say a weight-sequence w is *ambiguous* if $1 - \frac{\lceil d \rceil - 1}{d^{|w|-1} \cdot (d-1)} \leq DS(w, d) < 1$. Intuitively, a weight-sequence is ambiguous if it could be extended to an infinite word with length less than 1 and greater than 1.

We will establish that there exists an infinite word such that its discounted-sum is equal to 1 but all of its finite prefixes are ambiguous. We prove by induction on length of prefixes. Let $w = 0$. Since $d > 2$, $\frac{\lceil d \rceil - 1}{d-1} > 1$. So, w is ambiguous. Now, we prove that if w is ambiguous, then at least one of $w \cdot 0, \ldots, w \cdot (\lceil d \rceil - 1)$ is ambiguous.

We prove this in cases: Suppose $DS(w, d) < 1 - \frac{\lceil d \rceil - 1}{d^{|w|}}$, then we show that $w \cdot (\lceil d \rceil - 1)$ is ambiguous. First, since w is ambiguous, we know that $DS(w, d) \geq 1 - \frac{\lceil d \rceil - 1}{d^{|w|-1} \cdot (d-1)}$. It is easy to show that $DS(w \cdot (\lceil d \rceil - 1), d) = DS(w, d) + \frac{\lceil d \rceil - 1}{d^{|w|}} \geq 1 - \frac{\lceil d \rceil - 1}{d^{|w|-1} \cdot (d-1)} + \frac{\lceil d \rceil - 1}{d^{|w|}} = 1 - \frac{\lceil d \rceil - 1}{d^{|w|} \cdot (d-1)}$. Next, since $DS(w, d) < 1 - \frac{\lceil d \rceil - 1}{d^{|w|}}$ we get that $DS(w \cdot (\lceil d \rceil - 1), d) = DS(w, d) + \frac{\lceil d \rceil - 1}{d^{|w|}} \leq 1$. Thus, $w \cdot (\lceil d \rceil - 1)$ is ambiguous.

In the next case, suppose $1 - \frac{\lceil d \rceil - 1}{d^{|w|}} \leq DS(w, d) < 1 - \frac{\lceil d \rceil - 2}{d^{|w|}}$. We will show that $w \cdot (\lceil d \rceil - 2)$ is ambiguous. First, as show earlier, it is easy to see that $DS(w \cdot (\lceil d \rceil - 2), d) < 1$ since $DS(w, d) < 1 - \frac{\lceil d \rceil - 2}{d^{|w|}}$ has been assumed. Next, since $DS(w, d) > 1 - \frac{\lceil d \rceil - 1}{d^{|w|}}$, we get that $DS(w \cdot (\lceil d \rceil - 2), d) \geq 1 - \frac{\lceil d \rceil - 1}{d^{|w|}} + \frac{\lceil d \rceil - 2}{d^{|w|}} = 1 - \frac{1}{d^{|w|}}$. Since $d > 2$, we know that $\frac{\lceil d \rceil - 1}{d-1} > 1$, so $DS(w \cdot (\lceil d \rceil - 2), d) \geq 1 - \frac{\lceil d \rceil - 1}{d^{|w|} \cdot (d-1)}$. Thus, $w \cdot (\lceil d \rceil - 2)$ is ambiguous.

Similarly, when $1 - \frac{i+1}{d^{|w|}} \leq DS(w \cdot i, d) < 1 - \frac{i}{d^{|w|}}$, we can show that $w \cdot i$ is ambiguous for $i \in \{1, \lceil d \rceil - 2\}$.

In the final case, $DS(w, d) \geq 1 - \frac{1}{d^{|w|}}$. Since $DS(w \cdot 0, d) = DS(w, d)$, clearly, $DS(w \cdot 0, d) < 1$ since w is ambiguous. Further, $DS(w, d) \geq 1 - \frac{1}{d^{|w|}}$, we get that $DS(w \cdot 0, d) \geq 1 - \frac{\lceil d \rceil - 1}{d^{|w|} \cdot (d-1)}$ as $\frac{\lceil d \rceil - 1}{d-1} > 1$. Thus, $w \cdot 0$ is ambiguous.

Theorem 4.1. *For all non-integer, rational discount-factor $d > 1$, there exists a deterministic discounted-sum automata $\mathcal{A}$ and rational value $r \in \mathbb{R}$ for which its cut-point language is not ω-regular.*

Proof. Since the proof for $1 < d < 2$ has been presented in [CDH09], we skip that case.

The proof presented here extends the earlier result on $1 < d < 2$ from [CDH09] to all non-integer, rational discount factors $d > 2$.

Let $d > 2$ be a non-integer, rational discount-factor. Define deterministic discounted-sum automata $\mathcal{A}$ over the alphabet $\{0, 1, \ldots, \lceil d \rceil - 1\}$ such that the weight of transitions on alphabet $n \in \{0, 1, \ldots, \lceil d \rceil - 1\}$ is n. Therefore, weight of word $w \in \mathcal{A}$ is $DS(w, d)$.

Consider its cut-point language $L^{\geq 1}$. Let us assume that the language $L^{\geq 1}$ is ω-regular and represented by Büchi automaton $\mathcal{B}$. For $n < m$, let the n- and m-length prefixes of $w^{\geq}$, denoted $w^{\geq}[0, n-1]$ and $w^{\geq}[0, m-1]$, respectively, be such that they reach the same states in $\mathcal{B}$. Then there exists an infinite length word w_s such that $DS(w^{\geq}[0, n-1] \cdot w_s, d) = DS(w^{\geq}[0, m-1] \cdot w_s, d) = 1$. Now, $DS(w^{\geq}[0, n-1] \cdot w_s, d) = DS(w^{\geq}[0, n-1], d) + \frac{1}{d^n} \cdot DS(w_s, d)$ and $DS(w^{\geq}[0, m-1] \cdot w_s, d) = DS(w^{\geq}[0, m-1], d) + \frac{1}{d^m} \cdot DS(w_s, d)$. Eliminating

$DS(w_s, d)$ from the equations and simplification, we get:

$$d^{m-1} \cdot (DS(w^{\geq}[0, m-1], d) - 1) + d^{n-1} \cdot (DS(w^{\geq}[0, n-1], d) - 1) = 0$$

The above is a polynomial over d with degree $m-1$ and integer co-efficients. Specifically, $d = \frac{p}{q} > 2$ such that integers $p, q > 1$, and p and q are mutually prime. Since $d = \frac{p}{q}$ is a root of the above equation, q must divide co-efficient of the highest degree term, in this case it is $m-1$. The co-efficient of the highest degree term in the polynomial above is $(w^{\geq}[0] - (\lceil d \rceil - 1))$. Recall from construction of the infinite-length word with ambiguous prefixes $w^{\geq}$ from above, $w^{\geq}[0] = 0$. So the co-efficient of the highest degree term is -1, which is not divisible by integer $q > 1$. Contradiction. □

Finally, we use Theorem 4.1 to prove the discounted-sum comparator is not ω-regular when the discount-factor $d > 1$ is non-integer, rational number.

Theorem 4.2. *DS-comparator for non-integer, rational discount-factors $d > 1$ for all inequalities and equality are not ω-regular.*

Proof. If the comparator for an aggregate function for any one inequality is not ω-regular, then the comparator for all inequalities and equality relation will also not be ω-regular. Therefore, it is sufficient to prove that the discounted-sum comparator with non-integer, rational value for relation $\geq$ is not ω-regular.

Let $d > 1$ be a non-integer, rational discount-fact. Let $\mathcal{A}$ be the discounted-sum automaton as described in proof of Lemma 4.1. Consider its cut-point language $L^{\geq 1}$. From Lemma 4.1 and [CDH09], we know that $L^{\geq 1}$ is not an ω-regular language.

Suppose there exists an ω-regular DS-comparator $\mathcal{A}_d^{\leq}$ for non-integer rational discount factor $d > 1$ for relation $\geq$. We define the Büchi automaton $\mathcal{P}$ s.t. $\mathcal{L}(\mathcal{P}) = \{(w, v) | w \in \mathcal{L}(\mathcal{A}), v = \lceil d \rceil - 1 \cdot 0^{\omega}\}$. Note that $DS(\lceil d \rceil - 1 \cdot 0^{\omega}, d) = \lceil d \rceil - 1$. Then the cut-point language $L^{\geq(\lceil d \rceil - 1)}$ of deterministic discounted-sum automata $\mathcal{A}$ can be constructed by taking the intersection of $\mathcal{P}$ with $\mathcal{A}_d^{\geq}$. Since all actions are closed under ω-regular operations, $L^{\geq 1}$ can be represented by a Büchi automaton. Contradiction to Theorem 4.1. □

Theorem 4.3. *Let $d > 1$ be a non-integer, rational discount-factor. The discounted-sum aggregate function with discount-factor d is not ω-regular.*

Proof. Immediate from Lemma 4.1 and Theorem 3.6. □

Since the DS-comparator for all non-integer, rational discount-factor $d > 1$ is not ω-regular, the ω-regular-based algorithm for quantitative inclusion described in Algorithm 1 does not apply to DS-inclusion. In fact, the decidability of DS-inclusion with non-integer, rational discount-factors is still open. Finally, we have shown is follow-up work that comparators for approximations of discounted-sum with non-integer discount factors $1 < d < 2$ can be made ω-regular [BKVW22].

4.2. **Integer discount-factor.** In this section, we provide an explicit construction of an ω-regular comparator for discounted-sum with integer discount-factors. We use this construction to prove that discounted-sum aggregate function with integer discount-factor is ω-regular. Finally, we use the ω-regular DS-comparator in Algorithm 1 to establish that PSPACE-completeness of DS-inclusion with integer discount-factors.

Discounted-sum comparator. Let integer $\mu > 0$ be the upper-bound on sequences. The core intuition is that bounded sequences can be converted to their value in base d via a finite-state transducer. Lexicographic comparison of the converted sequences renders the desired DS-comparator. Conversion of sequences to base d requires a certain amount of *look-ahead* by the transducer. Here we describe a method that directly incorporates the look-ahead with lexicographic comparison to obtain the DS-comparator for integer discount-factor $d > 1$. Here we construct the discounted-sum comparator for relation $<$.

We explain the construction in detail now. For weight sequence A and integer discount-factor $d > 1$, $DS(A, d)$ can be interpreted as a value in base d i.e. $DS(A, d) = A[0] + \frac{A[1]}{d} + \frac{A[2]}{d^2} + \cdots = (A[0].A[1]A[2]\dots)_d$ [CSV13]. Unlike comparison of numbers in base d, the lexicographically larger sequence may not be larger in value since (i) The elements of weight sequences may be larger in value than base d, and (ii) Every value has multiple infinite-sequence representations.

To overcome these challenges, we resort to arithmetic techniques in base d. Note that $DS(B, d) > DS(A, d)$ iff there exists a sequence C such that $DS(B, d) = DS(A, d) + DS(C, d)$, and $DS(C, d) > 0$. Therefore, to compare the discounted-sum of A and B, we obtain a sequence C. Arithmetic in base d also results in sequence X of carry elements. Then:

Lemma 4.4. *Let A, B, C, X be weight sequences, $d > 1$ be a positive integer such that the following equations hold:*

(1) $A[0] + C[0] + X[0] = B[0]$
(2) *For $i \geq 1$,* $A[i] + C[i] + X[i] = B[i] + d \cdot X[i-1]$

Then $DS(B, d) = DS(A, d) + DS(C, d)$.

Proof. $DS(A, d) + DS(C, d) = \Sigma_{i=0}^{\infty} A[i]\frac{1}{d^i} + \Sigma_{i=0}^{\infty} C[i]\frac{1}{d^i} = \Sigma_{i=0}^{\infty}(A[i] + C[i])\frac{1}{d^i} = (B[0] - X[0]) + \Sigma_{i=1}^{\infty}(B[i] + d \cdot X[i-1] - X[i])\frac{1}{d^i} = (B[0] - X[0]) + \Sigma_{i=1}^{\infty}(B[i] + d \cdot X[i-1] - X[i])\frac{1}{d^i} = \Sigma_{i=0}^{\infty} B[i] \cdot \frac{1}{d^i} - \Sigma_{i=0}^{\infty} X[i] + \Sigma_{i=0}^{\infty} X[i] = \Sigma_{i=0}^{\infty} B[i] \cdot \frac{1}{d^i} = DS(B, d)$ □

Hence to determine $DS(B, d) - DS(A, d)$, systematically guess sequences C and X using the equations, element-by-element beginning with the 0-th index and moving rightwards. There are two crucial observations here: (i) Computation of i-th element of C and X only depends on i-th and $(i-1)$-th elements of A and B. Therefore guessing $C[i]$ and $X[i]$ requires *finite memory* only. (ii) Intuitively, C refers to a representation of value $DS(B, d) - DS(A, d)$ in base d and X is the carry-sequence. If we can prove that X and C are also bounded-sequences and can be constructed from a finite-set of integers, we would be able to further proceed to construct a Büchi automaton for the desired comparator.

We proceed by providing an inductive construction of sequences C and X that satisfy the properties in Lemma 4.4 (Lemma 4.5), and show that these sequences are bounded when A and B are bounded. In particular, when A and B are bounded integer-sequences, then sequences C and X constructed here are also bounded-integer sequences. Therefore, they can be constructed from a finite-set of integers. Proofs for sequence C are in Lemma 4.6-Lemma 4.8, and proof for sequence X is in Lemma 4.9.

We begin with introducing some notation. Let $DS^-(B, A, d, i) = \Sigma_{j=0}^{i}(B[j] - A[j]) \cdot \frac{1}{d^j}$ for all index $i \geq 0$. Also, let $DS^-(B, A, d, \cdot) = \Sigma_{j=0}^{\infty}(B[j] - A[j]) \cdot \frac{1}{d^j} = DS(B, d) - DS(A, d)$. Define $maxC = \mu \cdot \frac{d}{d-1}$. We define the residual function $Res : \mathbb{N} \cup \{0\} \mapsto \mathbb{R}$ as follows:

$$Res(i) = \begin{cases} DS^-(B, A, d, \cdot) - \lfloor DS^-(B, A, d, \cdot) \rfloor & \text{if } i = 0 \\ Res(i-1) - \lfloor Res(i-1) \cdot d^i \rfloor \cdot \frac{1}{d^i} & \text{otherwise} \end{cases}$$

Then we define $C[i]$ as follows:

$$C[i] = \begin{cases} \lfloor DS^-(B, A, d, \cdot) \rfloor & \text{if } i = 0 \\ \lfloor Res(i-1) \cdot d^i \rfloor & \text{otherwise} \end{cases}$$

Intuitively, $C[i]$ is computed by *stripping off* the value of the i-th digit in a representation of $DS^-(B, A, d, \cdot)$ in base d. $C[i]$ denotes the numerical value of the i-th position of the difference between B and A. The residual function denotes the numerical value of the difference remaining after assigning the value of $C[i]$ until that i.

We define function $CSum(i) : \mathbb{N} \cup \{0\} \to \mathbb{Z}$ s.t. $CSum(i) = \Sigma_{j=0}^{i} C[j] \cdot \frac{1}{d^j}$. Then, we define $X[i]$ as follows:

$$X[i] = (DS^-(B, A, d, i) - CSum(i)) \cdot d^i$$

Therefore, we have defined sequences C and X as above. We now prove the desired properties one-by-one.

First, we establish that sequences C, X as defined here satisfy Equations 1-2 from Lemma 4.4. Therefore, ensuring that C is indeed the difference between sequences B and A, and X is their carry-sequence.

Lemma 4.5. *Let A and B be bounded integer sequences and C and X be defined as above. Then,*

(1) $B[0] = A[0] + C[0] + X[0]$
(2) *For $i \geq 1$,* $B[i] + d \cdot X[i-1] = A[i] + C[i] + X[i]$

Proof. We prove this by induction on i using definition of function X.

When $i = 0$, then $X[0] = DS^-(B, A, d, 0) - CSum(0) \implies X[0] = B[0] - A[0] - C[0] \implies B[0] = A[0] + C[0] + X[0]$.

When $i = 1$, then $X[1] = (DS^-(B, A, d, 1) - CSum(1)) \cdot d = (B[0] + B[1] \cdot \frac{1}{d}) - (A[0] + A[1] \cdot \frac{1}{d}) - (C[0] + C[1] \cdot \frac{1}{d}) \cdot d \implies X[1] = B[0] \cdot d + B[1] - (A[0] \cdot d + A[1]) - (C[0] \cdot d + C[1])$. From the above we obtain $X[1] = d \cdot X[0] + B[1] - A[1] - C[1] \implies B[1] + d \cdot X[0] = A[1] + C[1] + X[1]$.

Suppose the invariant holds true for all $i \leq n$, we show that it is true for $n+1$. $X[n+1] = (DS^-(B, A, d, n+1) - CSum(n+1)) \cdot d^{n+1} \implies X[n+1] = (DS^-(B, A, d, n) - CSum(n)) \cdot d^{n+1} + (B[n+1] - A[n+1] - C[n+1]) \implies X[n+1] = X[n] \cdot d + B[n+1] - A[n+1] - C[n+1] \implies B[n+1] + X[n] \cdot d = A[n+1] + C[n+1] + X[n+1]$. □

Next, we establish the sequence C is a bounded integer sequence, therefore it can be represented by a finite-set of integers. First of all, by definition of $C[i]$ it is clear that $C[i]$ is an integer for all $i \geq 0$. We are left with proving boundedness of C. Lemma 4.6-Lemma 4.8 establish boundedness of $C[i]$.

Lemma 4.6. *For all $i \geq 0$, $Res(i) = DS^-(B, A, d, \cdot) - CSum(i)$.*

Proof. Proof by simple induction on the definitions of functions *Res* and C.

(1) When $i = 0$, $Res(0) = DS^-(B, A, d, \cdot) - \lfloor DS^-(B, A, d, \cdot) \rfloor$. By definition of $C[0]$, $Res(0) = DS^-(B, A, d, \cdot) - C[0] \iff Res(0) = DS^-(B, A, d, \cdot) - CSum(0)$.
(2) Suppose the induction hypothesis is true for all $i < n$. We prove it is true when $i = n$. When $i = n$, $Res(n) = Res(n-1) - \lfloor Res(n-1) \cdot d^n \rfloor \cdot \frac{1}{d^n}$. By definition of $C[n]$ and I.H, we get $Res(n) = (DS^-(B, A, d, \cdot) - CSum(n-1)) - C[n] \cdot \frac{1}{d^n}$. Therefore $Res(n) = DS^-(B, A, d, \cdot) - CSum(n)$. □

Lemma 4.7. *If $DS^-(B, A, d, \cdot) \geq 0$, then for all $i \geq 0$, $0 \leq Res(i) < \frac{1}{d^i}$.*

Proof. Since $DS^-(B,A,d,\cdot) \geq 0$, $Res(0) = DS^-(B,A,d,\cdot) - \lfloor DS^-(B,A,d,\cdot)\rfloor \geq 0$ and $Res(0) = DS^-(B,A,d,\cdot) - \lfloor DS^-(B,A,d,\cdot)\rfloor < 1$. Specifically, $0 \leq Res(0) < 1$.

Suppose for all $i \leq k$, $0 \leq Res(i) < \frac{1}{d^i}$. We show this is true even for $k+1$.

Since $Res(k) \geq 0$, $Res(k)\cdot d^{k+1} \geq 0$. Let $Res(k)\cdot d^{k+1} = x + f$, for integral $x \geq 0$, and fractional $0 \leq f < 1$. Then, from definition of Res, we get $Res(k+1) = \frac{x+f}{d^{k+1}} - \frac{x}{d^{k+1}} \implies Res(k+1) < \frac{1}{d^{k+1}}$.

Also, $Res(k+1) \geq 0$ since $a - \lfloor a\rfloor \geq 0$ for all positive values of a (Lemma 4.6). □

Lemma 4.8. *Let $maxC = \mu \cdot \frac{d}{d-1}$. When $DS^-(B,A,d,\cdot) \geq 0$, for $i = 0$, $0 \leq C(0) \leq maxC$, and for $i \geq 1$, $0 \leq C(i) < d$.*

Proof. Since both A and B are non-negative bounded weight sequences, maximum value of $DS^-(B,A,d,\cdot)$ is when $B = \{\mu\}_i$ and $A = \{0\}_i$. In this case $DS^-(B,A,d,\cdot) = maxC$. Therefore, $0 \leq C[0] \leq maxC$.

From Lemma 4.7, we know that for all i, $0 \leq Res(i) < \frac{1}{d^i}$. Alternately, when $i \geq 1$, $0 \leq Res(i-1) < \frac{1}{d^{i-1}} \implies 0 \leq Res(i-1)\cdot d^i < \frac{1}{d^{i-1}}\cdot d^i \implies 0 \leq Res(i-1)\cdot d^i < d \implies 0 \leq \lfloor Res(i-1)\cdot d^i\rfloor < d \implies 0 \leq C[i] < d$. □

Therefore, we have established that sequence C is non-negative integer-valued and is bounded by $maxC = \mu \cdot \frac{d}{d-1}$.

Finally, we prove that sequence X is also a bounded-integer sequence, thereby proving that it is bounded, and can be represented with a finite-set of integers. Note that for all $i \geq 0$, by expanding out the definition of $X[i]$ we get that $X[i]$ is an integer for all $i \geq 0$. We are left with proving boundedness of X:

Lemma 4.9. *Let $maxX = 1 + \frac{\mu}{d-1}$. When $DS^-(B,A,d,\cdot) \geq 0$, then for all $i \geq 0$, $|X(i)| \leq maxX$.*

Proof. From definition of X, we know that $X(i) = (DS^-(B,A,d,i) - CSum(i))\cdot d^i \implies X(i)\cdot\frac{1}{d^i} = DS^-(B,A,d,i) - CSum(i)$. From Lemma 4.6 we get $X(i)\cdot\frac{1}{d^i} = DS^-(B,A,d,i) - (DS^-(B,A,d,\cdot) - Res(i)) \implies X(i)\cdot\frac{1}{d^i} = Res(i) - (DS^-(B,A,d,\cdot) - DS^-(B,A,d,i)) \implies X(i)\cdot\frac{1}{d^i} = Res(i) - (\Sigma_{j=i+1}^{\infty}(B[j]-A[j])\cdot\frac{1}{d^j}) \implies |X(i)\cdot\frac{1}{d^i}| \leq |Res(i)| + |(\Sigma_{j=i+1}^{\infty}(B[j]-A[j])\cdot\frac{1}{d^j})| \implies |X(i)\cdot\frac{1}{d^i}| \leq |Res(i)| + \frac{1}{d^{i+1}}\cdot|(\Sigma_{j=0}^{\infty}(B[j+i+1]-A[j+i+1])\cdot\frac{1}{d^j})| \implies |X(i)\cdot\frac{1}{d^i}| \leq |Res(i)| + \frac{1}{d^{i+1}}\cdot|maxC|$. From Lemma 4.7, this implies $|X(i)\cdot\frac{1}{d^i}| \leq \frac{1}{d^i} + \frac{1}{d^{i+1}}\cdot|maxC| \implies |X(i)| \leq 1 + \frac{1}{d}\cdot|maxC| \implies |X(i)| \leq 1 + \frac{\mu}{d-1} \implies |X(i)| \leq maxX$ □

We summarize our results from Lemma 4.5-Lemma 4.9 as follows:

Corollary 4.10. *Let $d > 1$ be an integer discount-factor. Let A and B be non-negative integer sequences bounded by μ, and $DS(A,d) < DS(B,d)$. Then there exists bounded integer-valued sequences X and C that satisfy the conditions in Lemma 4.4. Furthermore, C and X are bounded as follows:*

(1) $0 \leq C[0] \leq \mu \cdot \frac{d}{d-1}$ *and for all* $i \geq 1$, $0 \leq C[i] < d$,
(2) *For all* $i \geq 0$, $0 \leq |X[i]| \leq 1 + \frac{\mu}{d-1}$

Intuitively, we construct a Büchi automaton $\mathcal{A}_d^{\leq}$ with states of the form (x,c) where x and c range over all possible values of X and C, respectively, and a special initial state s. Transitions over alphabet (a,b) replicate the equations in Lemma 4.4. i.e. transitions from the start state $(s,(a,b),(x,c))$ satisfy $a + c + x = b$ to replicate Equation 1 (Lemma 4.4) at

the 0-th index, and all other transitions $((x_1, c_1), (a, b), (x_2, c_2))$ satisfy $a + c_2 + x_2 = b + d \cdot x_1$ to replicate Equation 2 (Lemma 4.4) at indexes $i > 0$. Full construction is as follows:

Construction. Let $\mu_C = \mu \cdot \frac{d}{d-1}$ and $\mu_X = 1 + \frac{\mu}{d-1}$. $\mathcal{A}_d^< = (S, \Sigma, \delta_d, Init, \mathcal{F})$

- $S = \{s\} \cup \mathcal{F} \cup S_\perp$ where
  $\mathcal{F} = \{(x, c) || x| \leq \mu_X, 0 \leq c \leq \mu_C\}$, and
  $S_\perp = \{(x, \perp) || x| \leq \mu_X\}$ where $\perp$ is a special character, and $c \in \mathbb{N}$, $x \in \mathbb{Z}$.
- State s is the initial state, and $\mathcal{F}$ are accepting states
- $\Sigma = \{(a, b) : 0 \leq a, b \leq \mu\}$ where a and b are integers.
- $\delta_d \subseteq S \times \Sigma \times S$ is defined as follows:
  (1) Transitions from start state s:
    i $(s, (a, b), (x, c))$ for all $(x, c) \in \mathcal{F}$ s.t. $a + x + c = b$ and $c \neq 0$
    ii $(s, (a, b), (x, \perp))$ for all $(x, \perp) \in S_\perp$ s.t. $a + x = b$
  (2) Transitions within $S_\perp$: $((x, \perp), (a, b), (x', \perp))$ for all $(x, \perp)$, $(x', \perp) \in S_\perp$, if $a + x' = b + d \cdot x$
  (3) Transitions within $\mathcal{F}$: $((x, c), (a, b), (x', c'))$ for all (x, c), $(x', c') \in \mathcal{F}$ where $c' < d$, if $a + x' + c' = b + d \cdot x$
  (4) Transition between $S_\perp$ and $\mathcal{F}$: $((x, \perp), (a, b), (x', c'))$ for all $(x, \perp) \in S_\perp$, $(x', c') \in \mathcal{F}$ where $0 < c' < d$, if $a + x' + c' = b + d \cdot x$

Theorem 4.11. *Let $d > 1$ be an integer discount-factor, and $\mu > 1$ be an integer upper-bound. Büchi automaton $\mathcal{A}_d^<$ accepts pair of bounded sequences (A, B) iff $DS(A, d) \leq DS(B, d)$. The Büchi automaton has $\mathcal{O}(\frac{\mu^2}{d})$-many states.*

Proof. Corollary 4.10 proves that if $DS(A, d) < DS(B, d)$ then sequences X and C satisfying the integer sequence criteria and bounded-criteria will exist. Let these sequences be $X = X[0]X[1]\ldots$ and $C = [0]C[1]\ldots$. Since $DS(C, d) > 0$, there exists an index $i \geq 0$ where $C[i] > 0$. Let the first position where $C[i] > 0$ be index j. By construction of $\mathcal{A}_d^<$, the state sequence given by $s, (X[0], \perp) \ldots, (X[j-1], \perp), (X[j], C[j]), (X[j+1], C[j+1]) \ldots$, where for all $i \geq j$, $C[i] \neq \perp$, forms a run of word (A, B) in the Büchi automaton. Furthermore, this run is accepting since state (x, c) where $c \neq \perp$ are accepting states. Therefore, (A, B) is an accepting word in $\mathcal{A}_d^<$.

To prove the other direction, suppose the pair of sequences (A, B) has an accepting run with state sequence s, $(x_0, \perp), \ldots (x_{j-1}, \perp), (x_j, c_j), (x_{j+1}, c_{j+1}) \ldots$, where for all $i \geq j$, $c_j \neq \perp$. Construct sequences X and C as follows: For all $i \geq 0$, $X[i] = x_i$. For all $i < j$, $C[i] = 0$ and for all $i \geq j$ $C[i] = c_i$. Then the transitions of $\mathcal{A}_d^<$ guarantees Equations 1- 2 from Lemma 4.4 to hold for sequences A,B and C,X. Therefore, it must be the case that $DS(B, d) = DS(A, d) + DS(C, d)$. Furthermore, since the first transition to accepting states (x, c) where $c \neq \perp$ is possible only if $c > 0$, $DS(C, d) > 0$. Therefore, $DS(A, d) < DS(B, d)$. Therefore, $\mathcal{A}_d^<$ accepts (A, B) if $DS(A, d) < DS(B, d)$. □

Corollary 4.12. *DS-comparator for integer discount-factors $d > 1$ for all inequalities and equality are ω-regular.*

Proof. Immediate from Theorem 4.11, and closure properties of Büchi automaton. □

Constructions of DS-comparator with integer discount-factor $d > 1$ for non-strict inequality $\leq$ and equality $=$ follow similarly and also have $\mathcal{O}(\frac{\mu^2}{d})$-many states.

Discounted-sum aggregate function. We use the ω-regular comparator for DS-aggregate function for integer discount-factor to prove that discounted-sum with integer discount-factors is an ω-regular aggregate function.

Theorem 4.13. *Let $d > 1$ be an integer discount-factor. The discounted-sum aggregate function with discount-factor d is ω-regular under base d.*

Proof. We define the discounted-sum aggregate function automaton (DS-function automaton, in short): For integer $\mu > 0$, let $\Sigma = \{0, 1, \dots \mu\}$ be the input alphabet of DS-function, and $d > 1$ be its integer base. Büchi automaton $\mathcal{A}_d^\mu$ over alphabet $\Sigma \times \mathsf{AlphaRep}(d)$ is a DS-function automaton of type $\Sigma^\omega \to \mathbb{R}$ if for all $A \in \Sigma^\omega$, $(A, \mathsf{rep}(DS(A, d)), d) \in \mathcal{A}_d^\mu$. Here we prove that such a $\mathcal{A}_d^\mu$ exists.

Let $\mu > 0$ be the integer upper-bound. Let $\mathcal{A}_d^=$ be the DS-comparator for integer discount-factor $d > 1$ for relation $=$. Intersect $\mathcal{A}_d^=$ with the Büchi automata consisting of all infinite words from alphabet $\{0, 1 \dots \mu\} \times \{0, \dots, d-1\}$. The resulting automaton $\mathcal{B}$ accepts (A, B) for $A \in \{0, \dots, \mu\}^\omega$ and $B \in \{0, \dots, d-1\}^\omega$ iff $DS(A, d) = DS(B, d)$. Next, we want to ensure that for all (A, B) accepted by $\mathcal{B}$, B is not of the form $\{0, \dots, d-1\}^* \cdot (d-1)^\omega$. I.e., either $DS(B, d)$ is an irrational number or $DS(B, d)$ is a rational number in which case we prevent the finite-representation which ends in an infinite series of $(d-1)$. This can be represented by the Büchi automata $\mathcal{B} \setminus \mathcal{C}$, where $\mathcal{C}$ accepts words of the form (A, B) s.t. $A \in \{0, \dots, \mu\}^\omega$ and $B \in \{0, \dots, d-1\}^* \cdot (d-1)^\omega$. A non-deterministic Büchi automaton for $\mathcal{C}$ can be constructed by hand. The automaton for $\mathcal{B} \setminus \mathcal{C}$ will be a Parity automaton.

Since all elements of B are bounded by $d-1$, $DS(B, d)$ can be represented as an ω-word as follows: Let $B = B[0], B[1] \dots$, then its ω-word representation in base d is given by $+ \cdot (\mathsf{Int}(DS(B, d), d), \mathsf{Frac}(DS(B, d), d))$ where $\mathsf{Int}(DS(B, d), d) = B[0] \cdot 0^\omega$ and $\mathsf{Frac}(DS(B, d), d) = B[1], B[2] \dots$. This transformation of integer sequence B into its ω-regular word form in base d can be achieved with a simple transducer $\mathcal{T}$.

Therefore, application of transducer $\mathcal{T}$ to Parity automaton $\mathcal{B} \setminus \mathcal{C}$ will result in a Parity automaton over the alphabet $\Sigma \times \mathsf{AlphaRep}(d)$ such that for all $A \in \Sigma^\omega$ the automaton accepts $(A, \mathsf{rep}(DS(A, d), d))$. This is exactly the DS-function automaton over input alphabet Σ and integer base $d > 1$. Therefore, the discounted-sum aggregate function with integer discount-factors in ω-regular. □

Recall, this proof works only for the discounted-sum aggregate function with integer discount-factor. In general, there is no known procedure to derive a function automaton from an ω-regular comparator (Conjecture 3.7).

DS-inclusion. For discounted-sum with integer discount-factor it is in EXPTIME [BH14, CDH10] which does not match with its existing PSPACE lower bound. In this section, we use the ω-regular DS-comparator for integer to close the gap, and establish PSPACE-completeness of DS-inclusion under a unary representation of numbers.

Corollary 4.14. *Let integer $\mu > 1$ be the maximum weight on transitions in DS-automata P and Q, and $d > 1$ be an integer discount-factor. Let μ and d be represented in unary form. Then DS-inclusion, DS-strict-inclusion, and DS-equivalence between P and Q are* PSPACE*-complete.*

Proof. Since size of DS-comparator is polynomial w.r.t. to upper bound μ, when represented in unary, (Theorem 4.11), DS-inclusion is PSPACE in size of input weighted ω-automata and μ (Theorem 3.10). □

Not only does this result improve upon the previously known upper bound of EXPTIME but it also closes the gap between upper and lower bounds for DS-inclusion. Note, however, if the numbers are represented in binary, then the comparator-based algorithm will incur prohibitively large overhead since the size of the comparator will be exponential in μ.

We observe the algorithmic benefits of comparator-based solutions. The earlier known EXPTIME upper bound in complexity is based on an exponential determinization construction (subset construction) combined with arithmetical reasoning [BH14, CDH10]. We observe that the determinization construction can be performed on-the-fly in PSPACE. To perform, however, the arithmetical reasoning on-the-fly in PSPACE would require essentially using the same bit-level ((x, c)-state) techniques that we have used to construct DS-comparator. This point is corroborated in empirical evaluations where comparator-based approach comprehensively outperforms the determinization-based approach [BCV18a]. The performance of comparator-based approach has further been improved using additional language-theoretic properties of DS comparators, namely their safety and co-safety properties [BV19].

5. Limit-average comparator

The limit-average of an infinite sequence M is the point of convergence of the average of prefixes of M. Let $\mathsf{Sum}(M[0, n-1])$ denote the sum of the n-length prefix of sequence M. The *limit-average infimum*, denoted by $\mathsf{LimInfAvg}(M)$, is defined as $\liminf_{n\to\infty} \frac{1}{n} \cdot \mathsf{Sum}(M[0, n-1])$. Similarly, the *limit-average supremum*, denoted by $\mathsf{LimSupAvg}(M)$, is defined as $\limsup_{n\to\infty} \frac{1}{n} \cdot \mathsf{Sum}(M[0, n-1])$. The limit-average of sequence M, denoted by $\mathsf{LimAvg}(M)$, is defined *only if* the limit-average infimum and limit-average supremum coincide, and then $\mathsf{LimAvg}(M) = \mathsf{LimInfAvg}(M)$ $(= \mathsf{LimSupAvg}(M))$. Note that while limit-average infimum and supremum exist for all bounded sequences, the limit-average may not. To work around this limitation of limit-average, most applications simply use limit-average infimum or limit-average supremum of sequences [BCD+11, CDH10, CHJ05, ZP96]. However, the usage of limit-average infimum or limit-average supremum in lieu of limit-average for purpose of comparison can be misleading. For example, consider sequence A s.t. $\mathsf{LimSupAvg}(A) = 2$ and $\mathsf{LimInfAvg}(A) = 0$, and sequence B s.t. $\mathsf{LimAvg}(B) = 1$. Clearly, limit-average of A does not exist. So while it is true that $\mathsf{LimInfAvg}(A) < \mathsf{LimInfAvg}(B)$, indicating that at infinitely many indices the average of prefixes of A is lower, this renders an incomplete picture since at infinitley many indices, the average of prefixes of B is greater as $\mathsf{LimSupAvg}(A) = 2$.

Such inaccuracies in limit-average comparison may occur when the limit-average of at least one sequence does not exist. However, it is not easy to distinguish sequences for which limit-average exists from those for which it doesn't.

We define *prefix-average comparison* as a relaxation of limit-average comparison. Prefix-average comparison coincides with limit-average comparison when limit-average exists for both sequences. Otherwise, it determines whether eventually the average of prefixes of one sequence are greater than those of the other. This comparison does not require the limit-average to exist to return intuitive results. Further, we show that the *prefix-average comparator* is ω-context-free.

5.1. Limit-average language and comparison.

Let $\Sigma = \{0, 1, \ldots, \mu\}$ be a finite alphabet with $\mu > 0$. The *limit-average language* $\mathcal{L}_{LA}$ contains the sequence (word) $A \in \Sigma^\omega$ iff

its limit-average exists. We begin with the intuition to why limit-average language is neither ω-regular nor ω-context free. The formal argument is available in Theorem 5.1.

Suppose $\mathcal{L}_{LA}$ were ω-regular, then $\mathcal{L}_{LA} = \bigcup_{i=0}^{n} U_i \cdot V_i^\omega$, where $U_i, V_i \subseteq \Sigma^*$ are regular languages over *finite* words. The limit-average of sequences is determined by its behavior in the limit, so limit-average of sequences in V_i^ω exists. Additionally, the average of all (finite) words in V_i must be the same. If this were not the case, then two words in V_i with unequal averages l_1 and l_2, can generate a word $w \in V_i^\omega$ s.t the average of its prefixes oscillates between l_1 and l_2. This cannot occur, since limit-average of w exists. Let the average of sequences in V_i be a_i, then limit-average of sequences in V_i^ω and $U_i \cdot V_i^\omega$ is also a_i. This is contradictory since there are sequences with limit-average different from a_i. Similarly, since every ω-CFL is represented by $\bigcup_{i=1}^{n} U_i \cdot V_i^\omega$ for CFLs U_i, V_i over finite words [CG77], a similar argument proves that $\mathcal{L}_{LA}$ is not ω-context-free.

Theorem 5.1. *$\mathcal{L}_{LA}$ is neither an ω-regular nor an ω-context-free language.*

Proof. We first prove that $\mathcal{L}_{LA}$ is not ω-regular.

Let us assume that the language $\mathcal{L}_{LA}$ is ω-regular. Then there exists a finite number n s.t. $\mathcal{L}_{LA} = \bigcup_{i=0}^{n} U_i \cdot V_i^\omega$, where U_i and $V_i \in \Sigma^*$ are regular languages over finite words.

For all $i \in \{0, 1, \ldots n\}$, the limit-average of any word in $U_i \cdot V_i^\omega$ is given by the suffix of the word in V_i^ω. Since $U_i \cdot V_i^\omega \subseteq \mathcal{L}_{LA}$, limit-average exists for all words in $U_i \cdot V_i^\omega$. Therefore, limit-average of all words in V_i^ω must exist. As discussed above, we conclude that the average of all words in V_i must be the same. Furthermore, we know that the limit-average of all words in V_i^ω must be the same, say $\mathsf{LimAvg}(w) = a_i$ for all $w \in V_i^\omega$.

Then the limit-average of all words in $\mathcal{L}_{LA}$ is one of $a_0, a_1 \ldots a_n$. Let $a = \frac{p}{q}$ s.t $p < q$, and $a \neq a_i$ for $i \in \{0, 1, \ldots, \mu\}$. Consider the word $w = (1^p 0^{q-p})^\omega$. It is easy to see that $\mathsf{LimAvg}(w) = a$. However, this word is not present in $\mathcal{L}_{LA}$ since the limit-average of all words in $\mathcal{L}_{LA}$ is equal to a_0 or a_1 $\ldots$ or a_n.

Therefore, our assumption that $\mathcal{L}_{LA}$ is an ω-regular language has been contradicted.

Next we prove that $\mathcal{L}_{LA}$ is not an ω-CFL.

Every ω-context-free language can be written in the form of $\bigcup_{i=0}^{n} U_i \cdot V_i^\omega$ where U_i and V_i are context-free languages over finite words. The rest of this proof is similar to the proof for non-ω-regularity of $\mathcal{L}_{LA}$. □

In the next section, we will define *prefix-average comparison* as a relaxation of limit-average comparison. To show how prefix-average comparison relates to limit-average comparison, we will require the following two lemmas: Quantifiers $\exists^\infty i$ and $\exists^f i$ denote the existence of *infinitely* many and *only finitely* many indices i, respectively.

Lemma 5.2. *Let A, B be sequences s.t their limit-average exists. If $\mathsf{LimAvg}(A) > \mathsf{LimAvg}(B)$ then $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$.*

Proof. Let the limit-average of sequences A, B be L_a, L_b respectively. Since the limit average of both A and B exists, for every $\epsilon > 0$, there exists N_ϵ s.t. for all $n > N_\epsilon$, $|\mathsf{Avg}(A[1, n]) - L_a| < \epsilon$ and $|\mathsf{Avg}(B[1, n]) - L_b| < \epsilon$.

Let $L_a - L_b = k > 0$. Let us take $\epsilon = \frac{k}{4}$. Then, for all $n > N_{\frac{k}{4}}$, we get that $\mathsf{Avg}(A[0, n-1]) - \mathsf{Avg}(B[0, n-1]) > \frac{k}{2}$, since $L_a - L_b = k$ and $|\mathsf{Avg}(A[1, n]) - L_a| < \frac{k}{4}$ and $|\mathsf{Avg}(B[1, n]) - L_b| < \frac{k}{4}$. Thus, for all $n > N_{\frac{k}{4}}$, we get that $\mathsf{Sum}(A[0, n-1]) > \mathsf{Sum}(B[0, n-1])$. In particular, we get that $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$. □

The implication does not hold the other way since for sequences A and B with equal limit-average it is possible that $\exists^{\infty} i, \mathsf{Sum}(A[0, n-1]) > \mathsf{Sum}(B[0, n-1])$ and $\exists^{\infty} i, \mathsf{Sum}(B[0, n-1]) > \mathsf{Sum}(A[0, n-1])$.

Lemma 5.3. *Let A and B be sequences s.t. their limit average exists. If $\exists^{f} i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$, then $\mathsf{LimAvg}(A) \geq \mathsf{LimAvg}(B)$.*

Proof. We prove by contradiction.

Suppose, $\mathsf{LimAvg}(A) < \mathsf{LimAvg}(B)$. Then, from Lemma 5.2, we know that

$$\exists^{f} i, \mathsf{Sum}(A[0, i-1]) \geq \mathsf{Sum}(B[0, i-1]).$$

But, $\exists^{f} i, \mathsf{Sum}(A[0, i-1]) \geq \mathsf{Sum}(B[0, i-1])$ and $\exists^{f} i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$ cannot hold together since the sequences A and B are of infinite length. Hence, contradiction. □

5.2. **Prefix-average comparison and comparator.** The previous section relates limit-average comparison with the sums of equal length prefixes of the sequences (Lemma 5.2-5.3). The comparison criterion is based on the number of times sum of prefix of one sequence is greater than the other, which does not rely on the existence of limit-average. Unfortunately, this comparison criterion is not necessary and sufficient for limit-average comparison due to the one-way implication of Lemma 5.2. Instead, we use this criteria to define *prefix-average comparison*. In this section, we define prefix-average comparison and explain how it relaxes limit-average comparison. Lastly, we construct the prefix-average comparator, and prove that it is not ω-regular but is ω-context-free.

Definition 5.4 (Prefix-average comparison)**.** Let A and B be number sequences. We say $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$ if $\exists^{f} i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$.

Note, the definition implies that $\exists^{\infty} i, \mathsf{Sum}(A[0, i-1]) > \mathsf{Sum}(B[0, i-1])$.

Intuitively, prefix-average comparison states that $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$ if eventually the sum of prefixes of A are always greater than those of B. We use $\succeq$ since the average of prefixes may be equal when the difference between the sum converges to 0. Definition 5.4 and Lemma 5.2-5.3 relate limit-average comparison and prefix-average comparison:

Corollary 5.5. *When limit-average of A and B exists, then*

- $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B) \implies \mathsf{LimAvg}(A) \geq \mathsf{LimAvg}(B)$.
- $\mathsf{LimAvg}(A) > \mathsf{LimAvg}(B) \implies \mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$.

Proof. The first item falls directly from Definition 5.4 and Lemma 5.3. The second item falls directly from Definition 5.4 and Lemma 5.2. □

Therefore, on sequences for which limit-average exists, the first bullet says that prefix-average comparison returns the same result as limit-average comparison. In addition, when limit-average may not exist, the prefix-average comparison can return intuitive results. For example, suppose limit-average of A and B do not exist, but $\mathsf{LimInfAvg}(A) > \mathsf{LimSupAvg}(B)$, then $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$. This way, prefix-average comparison relaxes limit-average comparison.

The rest of this section describes *prefix-average comparator*, denoted by $\mathcal{A}^{\succeq}_{\mathsf{PA}}$, an automaton that accepts the pair (A, B) of sequences iff $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$.

Lemma 5.6. ***(Pumping Lemma for ω-regular language*** [ADMW09]***)*** *Let L be an ω-regular language. There exists $p \in \mathbb{N}$ such that, for each $w = u_1 w_1 u_2 w_2 \cdots \in L$ such that $|w_i| \geq p$ for all i, there are sequences of finite words $(x_i)_{i\in\mathbb{N}}$, $(y_i)_{i\in\mathbb{N}}$, $(z_i)_{i\in\mathbb{N}}$ s.t., for all i, $w_i = x_i y_i z_i$, $|x_i y_i| \leq p$ and $|y_i| > 0$ and for every sequence of pumping factors $(j_i)_{i\in\mathbb{N}} \in \mathbb{N}$, the pumped word $u_1 x_1 y_1^{j_1} z_1 u_2 x_2 y_2^{j_2} z_2 \cdots \in L$.*

Theorem 5.7. *The prefix-average comparator is not ω-regular.*

Proof Sketch. We use Lemma 5.6 to prove that $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ is not ω-regular. Suppose $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ were ω-regular. For $p > 0 \in \mathbb{N}$, let $w = (A, B) = ((0,1)^p (1,0)^{2p})^\omega$. The segment $(0,1)^*$ can be pumped s.t the resulting word is no longer in $\mathcal{A}^{\succeq}_{\mathsf{PA}}$.

Concretely, $A = (0^p 1^{2p})^\omega$, $B = (1^p 0^{2p})^\omega$, $\mathsf{LimAvg}(A) = \frac{2}{3}$, $\mathsf{LimAvg}(B) = \frac{1}{3}$. So, $w = (A, B) \in \mathcal{A}^{\succeq}_{\mathsf{PA}}$. Select as factor w_i (from Lemma 5.6) the sequence $(0,1)^p$. Pump each y_i enough times so that the resulting word is $\hat{w} = (\hat{A}, \hat{B}) = ((0,1)^{m_i}(1,0)^{2p})^\omega$ where $m_i > 4p$. It is easy to show that $\hat{w} = (\hat{A}, \hat{B}) \notin \mathcal{A}^{\succeq}_{\mathsf{PA}}$. □

We discuss key ideas and sketch the construction of the prefix average comparator. The term *prefix-sum difference at i* indicates $\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1])$, i.e. the difference between sum of i-length prefix of A and B.

Key ideas. For sequences A and B to satisfy $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$, $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$. This implies there exists an index N s.t. for all indices $i > N$, $\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1]) > 0$. While reading a word, the prefix-sum difference is maintained by states and the stack of ω-PDA: states maintain whether it is negative or positive, while number of tokens in the stack equals its absolute value. The automaton non-deterministically guesses the aforementioned index N, beyond which the automaton ensures that prefix-sum difference remains positive.

Construction sketch. The push-down comparator $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ consists of three states: (i) State s_P and (ii) State s_N that indicate that the prefix-sum difference is greater than zero or not respectively, (iii) accepting state s_F. An execution of (A, B) begins in state s_N with an empty stack. On reading letter (a, b), the stack pops or pushes $|(a-b)|$ tokens from the stack depending on the current state of the execution. From state s_P, the stack pushes tokens if $(a - b) > 0$, and pops otherwise. The opposite occurs in state s_N. State transition between s_N and s_P occurs only if the stack action is to pop but the stack consists of $k < |a - b|$ tokens. In this case, stack is emptied, state transition is performed and $|a - b| - k$ tokens are pushed into the stack. For an execution of (A, B) to be an accepting run, the automaton non-deterministically transitions into state s_F. State s_F acts similar to state s_P except that execution is terminated if there aren't enough tokens to pop out of the stack. $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ accepts by accepting state.

To see why the construction is correct, it is sufficient to prove that at each index i, the number of tokens in the stack is equal to $|\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1])|$. Furthermore, in state s_N, $\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1]) \leq 0$, and in state s_P and s_F, $\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1]) > 0$. Next, the index at which the automaton transitions to the accepting state s_F coincides with index N. The execution is accepted if it has an infinite execution in state s_F, which allows transitions only if $\mathsf{Sum}(A[0, i-1]) - \mathsf{Sum}(B[0, i-1]) > 0$.

Construction. We provide a sketch of the construction of the Büchi push-down autoamaton $\mathcal{A}^{\succeq}_{\mathsf{PA}}$, and then prove that it corresponds to the prefix average comparator.

Let μ be the bound on sequences. Then $\Sigma = \{0, 1, \ldots, n\}$ is the alphabet of sequences. Let $\mathcal{A}^{\succeq}_{\mathsf{PA}} = (S, \Sigma \times \Sigma, \Gamma, \delta, s_0, Z_0)$ where:

- $S = \{s_N, s_P, s_F\}$ is the set of states of the automaton.
- $\Sigma \times \Sigma$ is the alphabet of the language.
- $\Gamma = \{Z_0, \alpha\}$ is the alphabet.
- $s_0 = s_N$ is the start state of the automata.
- Z_0 is the start symbol of the stack.
- s_F is the accepting state of the automaton. Automaton $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ accepts words by final state.
- Here we give a sketch of the behavior of the transition function δ.
  - When $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ is in configuration (s_P, τ) for $\tau \in \Gamma$, push a number of α-s into the stack. Next, pop b number of α-s. If after popping k α-s where $k < b$, the PDA's configuration becomes (s_P, Z_0), then first move to state (s_N, Z_0) and then resume with pushing $b - k$ α-s into the stack.
  - When $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ is in configuration (s_N, τ) for $\tau \in \Gamma$, push b number of α-s into the stack Next, pop a number of α-s. If after popping k α-s where $k < a$, the PDA's configuration becomes (s_N, Z_0), then first move to state (s_P, Z_0) and then resume with pushing $a - k$ α-s into the stack.
  - When $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ is in configuration (s_P, τ) for $\tau \neq Z_0$, first move to configuration (s_F, τ) and then push a number of α-s and pop b number of α-s. Note that there are no provisions for popping α if the stack hits Z_0 along this transition.
  - When $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ is in configuration (s_F, τ) for $\tau \neq Z_0$, push a α-s then pop b α-s. Note that there are no provisions for popping α if the stack hits Z_0 along this transition.

Lemma 5.8. *Pushdown automaton $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ accepts a pair of sequences (A, B) iff $\mathsf{PrefixAvg}(A) \succeq \mathsf{PrefixAvg}(B)$.*

Proof sketch. To prove this statement, it is sufficient to demonstrate that $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ accepts a pair of sequences (A, B) iff there are only finitely many indexes where $\mathsf{Sum}(B[1, i]) > \mathsf{Sum}(A[1, i])$. On $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ this corresponds to the condition that there being only finitely many times when the PDA is in state N during the run of (A, B). This is ensured by the pushdown automaton since the word can be accepted only in state F and there is no outgoing edge from F. Therefore, every word that is accepted by $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ satisfies the condition $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$.

Conversely, for every word (A, B) that satisfies $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$ there is a point, call it index k, such that for all indexes $m > k$, $\mathsf{Sum}(B[1, m]) \not\geq \mathsf{Sum}(A[1, m])$. If a run of (A, B) switches to F at this m, then it will be accepted by the automaton. Since $\mathcal{A}^{\succeq}_{\mathsf{PA}}$ allows for non-deterministic move to (F, τ) from (P, τ), the run of (A, B) will always be able to move to F after index m. Hence, every (A, B) satisfying $\exists^f i, \mathsf{Sum}(B[0, i-1]) \geq \mathsf{Sum}(A[0, i-1])$ will be accepted by $\mathcal{A}^{\succeq}_{\mathsf{PA}}$. □

Theorem 5.9. *The prefix-average comparator is an ω-CFL.*

While ω-CFL can be easily expressed, they do not possess closure properties, and several problems on ω-CFL are easily undecidable. Hence, the application of ω-context-free comparator will require further investigation. For example, it is unclear whether ω-context free comparators can solve quantitative inclusion since complementation of ω-CFL is undecidable. Problems like membership in ω-CFG are decidable. We will have to

investigate applications where reductions using ω-context-free comparators require decidable operations such as membership.

6. Concluding remarks

In this paper, we identified a novel mode for comparison in quantitative systems: the online comparison of aggregate values of sequences of quantitative weights. This notion is embodied by comparators automata that read two infinite sequences of weights synchronously and relate their aggregate values. We showed that all ω-regular aggregate functions have ω-regular comparators. However, the converse direction is still open: Are functions with ω-regular comparators also ω-regular? We showed that ω-regular comparators not only yield generic algorithms for problems including quantitative inclusion and winning strategies in incomplete-information quantitative games, they also result in algorithmic advances [BCV21]. We establish that when the weights are represented in unary, the discounted-sum inclusion problem is PSAPCE-complete for integer discount-factor, hence closing a complexity gap. We showed that discounted-sum aggregate function and their comparators are ω-regular iff the discount-factor $d > 1$ is an integer. We showed that prefix-average comparator are ω-context-free.

We believe comparators, especially ω-regular comparators, can be of significant utility in verification and synthesis of quantitative systems [Ban20], as demonstrated by the existence of finite-representation of counterexamples of the quantitative inclusion problem. Another potential application is computing equilibria in quantitative games. Applications of the prefix-average comparator, in general ω-context-free comparators, is open to further investigation. Another direction to pursue is to study aggregate functions in more detail, and attempt to solve the conjecture relating ω-regular aggregate functions and ω-regular comparators.

Acknowledgment

We thank the anonymous reviewers for their comments. We thank K. Chatterjee, L. Doyen, T. Henzinger, G. A. Perez and J. F. Raskin for corrections to earlier drafts, and their contributions to this paper. We thank P. Ganty and R. Majumdar for preliminary discussions on the limit-average comparator. This work was partially supported by NSF Grant No. 1704883, "Formal Analysis and Synthesis of Multiagent Systems with Incentives".

References

[ABK11] Shaull Almagor, Udi Boker, and Orna Kupferman. Whatś decidable about weighted automata? In *Proc. of ATVA*, pages 482–491. Springer, 2011.

[AC13] Garrett Andersen and Vincent Conitzer. Fast equilibrium computation for infinitely repeated games. In *Proc. of AAAI*, pages 53–59, 2013.

[ADMW09] Rajeev Alur, Aldric Degorre, Oded Maler, and Gera Weiss. On omega-languages defined by mean-payoff conditions. In *Proc. of FOSSACS*, pages 333–347. Springer, 2009.

[And06] Daniel Andersson. An improved algorithm for discounted payoff games. In *ESSLLI Student Session*, pages 91–98, 2006.

[Ban20] Suguman Bansal. Automata-based quantitative verification. *PhD Thesis*, 2020.

[BCD+11] Lubos Brim, Jakub Chaloupka, Laurent Doyen, Raffaella Gentilini, and Jean-François Raskin. Faster algorithms for mean-payoff games. *Formal methods in system design*, 38(2):97–118, 2011.

[BCV18a] Suguman Bansal, Swarat Chaudhuri, and Moshe Y Vardi. Automata vs linear-programming discounted-sum inclusion. In *Proc. of CAV*, 2018.

[BCV18b] Suguman Bansal, Swarat Chaudhuri, and Moshe Y. Vardi. Comparator automata in quantitative verification. In *Proc. of FoSSaCS*, 2018.

[BCV21] Suguman Bansal, Krishnendu Chatterjee, and Moshe Y. Vardi. On satisficing in quantitative games. In *Proc. of TACAS*, 2021.

[BH14] Udi Boker and Thomas A. Henzinger. Exact and approximate determinization of discounted-sum automata. *LMCS*, 10(1), 2014.

[BK+08] Christel Baier, Joost-Pieter Katoen, et al. *Principles of model checking*. MIT press Cambridge, 2008.

[BKVW22] Suguman Bansal, Lydia Kavraki, Moshe Y Vardi, and Andrew Wells. Synthesis from satisficing and temporal goals. 2022.

[BV19] Suguman Bansal and Moshe Y Vardi. Safety and co-safety comparator automata for discounted-sum inclusion. In *Proc. of CAV*, 2019.

[CD10] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. In *Proc. of ICALP*, pages 599–610. Springer, 2010.

[CDH09] Krishnendu Chatterjee, Laurent Doyen, and Thomas A Henzinger. Expressiveness and closure properties for quantitative languages. In *Proc. of LICS*, pages 199–208. IEEE, 2009.

[CDH10] Krishnendu Chatterjee, Laurent Doyen, and Thomas A Henzinger. Quantitative languages. *Transactions on Computational Logic*, 11(4):23, 2010.

[CG77] Rina S Cohen and Arie Y Gold. Theory of ω-languages: Characterizations of ω-context-free languages. *Journal of Computer and System Sciences*, 15(2):169–184, 1977.

[CHJ05] Krishnendu Chatterjee, Thomas A Henzinger, and Marcin Jurdzinski. Mean-payoff parity games. In *Proc. of LICS*, pages 178–187. IEEE, 2005.

[CSV13] Swarat Chaudhuri, Sriram Sankaranarayanan, and Moshe Y. Vardi. Regular real analysis. In *Proc. of LICS*, pages 509–518, 2013.

[DAFH+04] Luca De Alfaro, Marco Faella, Thomas A Henzinger, Rupak Majumdar, and Mariëlle Stoelinga. Model checking discounted temporal properties. In *Proc. of TACAS*, pages 77–92. Springer, 2004.

[DAFS04] Luca De Alfaro, Marco Faella, and Mariëlle Stoelinga. Linear and branching metrics for quantitative transition systems. In *Proc. of ICALP*, pages 97–109. Springer, 2004.

[DDG+10] Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In *International Workshop on Computer Science Logic*, pages 260–274. Springer, 2010.

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler. *Handbook of weighted automata*. Springer, 2009.

[Kar78] Richard M Karp. A characterization of the minimum cycle mean in a digraph. *Discrete mathematics*, 23(3):309–311, 1978.

[KV00] Orna Kupferman and Moshe Y Vardi. Synthesis with incomplete informatio. In *Advances in temporal logic*, pages 109–127. Springer, 2000.

[Moh09] Mehryar Mohri. Weighted automata algorithms. In *Handbook of weighted automata*, pages 213–254. Springer, 2009.

[TW+02] Wolfgang Thomas, Thomas Wilke, et al. *Automata, logics, and infinite games: A guide to current research*, volume 2500. Springer Science & Business Media, 2002.

[ZP96] Uri Zwick and Mike Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1):343–359, 1996.