Safety and Co-safety Comparator Automata for Discounted-Sum Inclusion

Suguman Bansal(✉) and Moshe Y. Vardi

Rice University, Houston, TX 77005, USA
sugumanb@gmail.com

Abstract. *Discounted-sum inclusion* (DS-inclusion, in short) formalizes the goal of comparing quantitative dimensions of systems such as cost, resource consumption, and the like, when the mode of aggregation for the quantitative dimension is discounted-sum aggregation. *Discounted-sum comparator automata*, or *DS-comparators* in short, are Büchi automata that read two infinite sequences of weights synchronously and relate their discounted-sum. Recent empirical investigations have shown that while DS-comparators enable competitive algorithms for DS-inclusion, they still suffer from the scalability bottleneck of Büchi operations.

Motivated by the connections between discounted-sum and Büchi automata, this paper undertakes an investigation of language-theoretic properties of DS-comparators in order to mitigate the challenges of Büchi DS-comparators to achieve improved scalability of DS-inclusion. Our investigation uncovers that DS-comparators possess safety and co-safety language-theoretic properties. As a result, they enable reductions based on subset construction-based methods as opposed to higher complexity Büchi complementation, yielding tighter worst-case complexity and improved empirical scalability for DS-inclusion.

1 Introduction

The analysis of quantitative dimensions of computing systems such as cost, resource consumption, and distance metrics [6,10,28] has been studied thoroughly to design efficient computing systems. Cost-aware program-synthesis [14,16] and low-cost program-repair [25] have found compelling applications in robotics [24,29], education [22], and the like. *Quantitative verification* facilitates efficient system design by automatically determining if a system implementation is more efficient than a specification model. Investigations in quantitative verification have demonstrated their high computational complexity and practically intractable [17,23]. This work addresses practical intractability of quantitative verification.

At the core of quantitative verification lies the problem of *quantitative inclusion* which formalizes the goal of determining which of two given systems is more efficient [17,23,31]. In quantitative inclusion, quantitative systems are abstracted as weighted automata [7,21,32]. A run in a weighted automaton is associated with a sequence of weights. The quantitative dimension of these runs is determined by the weight of runs, which is computed by taking an aggregate of the

I. Dillig and S. Tasiran (Eds.): CAV 2019, LNCS 11561, pp. 60–78, 2019.
https://doi.org/10.1007/978-3-030-25540-4_4

run's weight sequence. Quantitative inclusion can be thought of as the quantitative generalization of (qualitative) language inclusion.

A commonly appearing mode of aggregation is that of *Discounted-sum (DS) aggregation* which captures the intuition that weights incurred in the near future are more significant than those incurred later on [19]. The convergence of DS aggregation for all bounded infinite weight-sequences makes it a preferred mode of aggregation across domains: Reinforcement learning [37], planning under uncertainty [34], and game-theory [33]. This work examines the problem of *Discounted-sum inclusion* or *DS-inclusion* that is quantitative inclusion when *discounted sum* is the mode of aggregation.

In theory, DS-inclusion is PSPACE-complete [12]. Recent algorithmic approaches have tapped into language-theoretic properties of discounted-sum aggregate function [12,18] to design practical algorithms for DS-inclusion [11,12]. These algorithms use *DS-comparator automata* (*DS-comparator*, in short) as their main technique, and are *purely* automata-theoretic. While these algorithms outperform other existing approaches for DS-inclusion in runtime [15,17], even these do not scale well on weighted-automata with more than few hundreds of states [11]. This work contributes novel techniques and algorithms for DS-inclusion to address the scalability challenge of DS-inclusion

An in-depth examination of the DS-comparator based algorithm exposes their scalability bottleneck. DS-comparator is a Büchi automaton that relates the discounted-sum aggregate of two (bounded) weight-sequences A and B by determining the membership of the interleaved pair of sequences (A, B) in the language of the comparator. As a result, DS-comparators reduce DS-inclusion to language inclusion between (non-deterministic) Büchi automaton. In spite of the fact that many techniques have been proposed to solve Büchi language inclusion efficiently in practice [4,20], none of them can avoid at least an exponential blow-up of $2^{\mathcal{O}(n \log n)}$, for an n-sized input, caused by a direct or indirect involvement of Büchi complementation [36,40].

This work meets the scalability challenge of DS-inclusion by delving deeper into language-theoretic properties of discounted-sum aggregate functions [18] in order to obtain algorithms for DS-inclusion that render both tighter theoretical complexity and improved scalability. Specifically, we prove that DS-comparators are expressed as *safety automata* or *co-safety automata* [26] (Sect. 3.1), and have compact deterministic constructions (Sect. 3.2). Safety and co-safety automata have the property that their complementation is performed by simpler and lower $2^{\mathcal{O}(n)}$-complexity subset-construction methods [27]. As a result, they facilitate a procedure for DS-inclusion that uses subset-construction based intermediate steps instead of Büchi complementation, yielding an improvement in theoretical complexity from $2^{\mathcal{O}(n \cdot \log n)}$ to $2^{\mathcal{O}(n)}$. Our subset-construction based procedure has yet another advantage over Büchi complementation as they support efficient on-the-fly implementations, yielding practical scalability as well (Sect. 4).

An empirical evaluation of our prototype tool QuIPFly for the proposed procedure against the prior DS-comparator algorithm and other existing approaches for DS-inclusion shows that QuIPFly outperforms them by orders of magnitude both in runtime and the number of benchmarks solved (Sect. 4).

2 Preliminaries and Related Work

A weight-sequence, finite or infinite, is *bounded* if the absolute value of all of its elements are bounded by a fixed number.

Büchi Automaton: A *Büchi automaton* is a tuple $\mathcal{A} = (S, \Sigma, \delta, s_{\mathcal{I}}, \mathcal{F})$, 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*, state $s_{\mathcal{I}} \in S$ is the *initial state*, and $\mathcal{F} \subseteq S$ is the set of *accepting states* [39]. A Büchi automaton is *deterministic* if for all states s and inputs a, $|\{s'|(s,a,s') \in \delta \text{ for some } s'\}| \leq 1$. 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 $w = w_0 w_1 \dots \in \Sigma^\omega$, a *run* ρ of w is a sequence of states $s_0 s_1 \dots$ s.t. $s_0 = s_{\mathcal{I}}$, 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. The language of Büchi automaton $\mathcal{A}$, denoted by $\mathcal{L}(\mathcal{A})$ is the set of all words accepted by $\mathcal{A}$. By abuse of 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 $\mathcal{A}$. Büchi automata are closed under set-theoretic union, intersection, and complementation [39].

Safety and Co-safety Properties: Let $\mathcal{L} \subseteq \Sigma^\omega$ be a language over alphabet Σ. A finite word $w \in \Sigma^*$ is a *bad prefix* for $\mathcal{L}$ if for all infinite words $y \in \Sigma^\omega$, $x \cdot y \notin \mathcal{L}$. A language $\mathcal{L}$ is a *safety language* if every word $w \notin \mathcal{L}$ has a bad prefix for $\mathcal{L}$. A language $\mathcal{L}$ is a *co-safety language* if its complement language is a safety language [5]. When a safety or co-safety language is an ω-regular language, the Büchi automaton representing it is called a safety or co-safety automaton, respectively [26]. Wlog, safety and co-safety automaton contain a *sink state* from which every outgoing transitions loops back to the sink state and there is a transition on every alphabet symbol. All states except the sink state are accepting in a safety automaton, while only the sink state is accepting in a co-safety automaton. Unlike Büchi complementation, complementation of safety and co-safety automaton is conducted by simpler subset construction with a lower $2^{\mathcal{O}(n)}$ blow-up. The complementation of safety automaton is a co-safety automaton, and vice-versa. Safety automata are closed under intersection, and co-safety automata are closed under union.

Comparator Automaton: For a finite-set of integers Σ, an aggregate function $f : \mathbb{Z}^\omega \rightarrow \mathbb{R}$, and equality or inequality relation $\mathsf{R} \in \{<, >, \leq, \geq, =, \neq\}$, the *comparison language for* f *with relation* R is a language of infinite words over the alphabet $\Sigma \times \Sigma$ that accepts a pair (A, B) iff $f(A)$ R $f(B)$ holds. A *comparator automaton (comparator, in short) for aggregate function* f *and relation* R is an automaton that accepts the comparison language for f with R [12]. A comparator is said to be *regular* if its automaton is a Büchi automaton.

Weighted Automaton: A *weighted automaton* over infinite words is a tuple $\mathcal{A} = (\mathcal{M}, \gamma, f)$, where $\mathcal{M} = (S, \Sigma, \delta, s_{\mathcal{I}}, S)$ is a complete Büchi automaton with all states as accepting, $\gamma : \delta \rightarrow \mathbb{N}$ is a *weight function*, and $f : \mathbb{N}^\omega \rightarrow \mathbb{R}$ is the *aggregate function* [17,31]. *Words* and *runs* in weighted automata are defined as 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 $\mathcal{A}\}$.

Quantitative Inclusion: Let P and Q be weighted automata with the *same* aggregate function. The *strict quantitative inclusion problem*, denoted by $P \subset Q$, asks whether for all words $w \in \Sigma^\omega$, $wt_P(w) < wt_Q(w)$. The *non-strict quantitative inclusion problem*, denoted by $P \subseteq Q$, asks whether for all words $w \in \Sigma^\omega$, $wt_P(w) \leq wt_Q(w)$. *Comparison language or comparator of a quantitative inclusion* problem refer to the comparison language or comparator of the associated aggregate function.

Discounted-sum Inclusion: Let $A = A_0, A_1, \dots$ be a weight sequence, $d > 1$ be a rational number. The *discounted-sum* (DS in short) of A with *integer* discount-factor $d > 1$ is $DS(A, d) = \Sigma_{i=0}^{\infty} \frac{A_i}{d^i}$. DS-comparison language and DS-comparator with discount-factor $d > 1$ are the comparison language and comparator obtained for the discounted-sum aggregate function with discount-factor $d > 1$, respectively. Strict or non-strict discounted-sum inclusion is strict or non-strict quantitative inclusion with the discounted-sum aggregate function, respectively. For brevity, we abbreviate discounted-sum inclusion to DS-inclusion.

Related Work. The decidability of DS-inclusion is an open problem when the discount-factor $d > 1$ is arbitrary. Recent work has established that DS-inclusion is PSPACE-complete when the discount-factor is an integer [12]. This work investigates algorithmic approaches to DS-inclusion with integer discount-factors.

Two contrasting solution approaches have been identified for DS-inclusion. The first approach is *hybrid* [17]. It separates out the language-theoretic aspects of weighted-automata from the numerical aspects, and solves each separately [15,17]. More specifically, the hybrid approach solves the language-theoretic aspects by DS-determinization [15] and the numerical aspect is performed by linear programming [8,9] sequentially. To the best of our knowledge, this procedure cannot be performed in parallel. As a result, this approach must always incur the exponential cost of DS-determinization.

The second approach is *purely*-automata theoretic [12]. This approach uses regular DS-comparator to reduce DS-inclusion to language inclusion between non-deterministic Büchi automata [11,12]. While the purely automata-theoretic approach scales better than the hybrid approach in runtime [11], its scalability suffers from fundamental algorithmic limitations of Büchi language inclusion. A key ingredient of Büchi language-inclusion is Büchi complementation [36]. Büchi complementation is $2^{\mathcal{O}(n \log n)}$ in the worst-case, and is practically intractable [40]. These limitations also feature in the theoretical complexity and practical performance of DS-inclusion. The complexity of DS-inclusion between weighted automata P and Q with regular DS-comparator C for integer discount-factor $d > 1$ is $|P| \cdot 2^{\mathcal{O}(|P||Q||C| \cdot \log(|P||Q||C|))}$.

This work improves the worst-case complexity and practical performance of the purely automata theoretic approach for DS-inclusion by a closer investigation of language-theoretic properties of DS-comparators. In particular, we identify that DS-comparator for integer discount-factor form a safety or co-safety automata (depending on the relation R). We show that complementation advantage of safety/co-safety automata not only improves the theoretical complexity of DS-inclusion with integer discount-factor but also facilitate on-the-fly implementations that significantly improve practical performance.

3 DS-inclusion with Integer Discount-Factor

This section covers the core technical contributions of this paper. We uncover novel language-theoretic properties of DS-comparison languages and utilize them to obtain tighter theoretical upper-bound for DS-inclusion with integer discount-factor. Unless mentioned otherwise, the discount-factor is an integer.

In Sect. 3.1 we prove that DS-comparison languages are either safety or co-safety for all rational discount-factors. Since DS-comparison languages are ω-regular for integer discount-factors [12], we obtain that DS-comparators for integer discount-factors form safety or co-safety automata. Next, Sect. 3.2 makes use of newly obtained safety/co-safety properties of DS-comparator to present the first deterministic constructions for DS-comparators. These deterministic construction are compact in the sense that they match their non-deterministic counterparts in number of states [11]. Section 3.3 evaluates the complexity of quantitative inclusion with regular safety/co-safety comparators, and observes that its complexity is lower than the complexity for quantitative inclusion with regular comparators. Finally, since DS-comparators are regular safety/co-safety, our analysis shows that the complexity of DS-inclusion is improved as a consequence of the complexity observed for quantitative-inclusion with regular safety/co-safety comparators.

We begin with formal definitions of safety/co-safety comparison languages and safety/co-safety comparators:

Definition 1 (Safety and co-safety comparison languages). *Let Σ be a finite set of integers, $f : \mathbb{Z}^\omega \rightarrow \mathbb{R}$ be an aggregate function, and $\mathsf{R} \in \{\leq, <, \geq, >, =, \neq\}$ be a relation. A comparison language L over $\Sigma \times \Sigma$ for aggregate function f and relation R is said to be a safety comparison language (or a co-safety comparison language) if L is a safety language (or a co-safety language).*

Definition 2 (Safety and co-safety comparators). *Let Σ be a finite set of integers, $f : \mathbb{Z}^\omega \rightarrow \mathbb{R}$ be an aggregate function, and $\mathsf{R} \in \{\leq, <, \geq, >, =, \neq\}$ be a relation. A comparator for aggregate function f and relation R is a* safety comparator *(or* co-safety comparator*) is the comparison language for f and R is a safety language (or co-safety language).*

A safety comparator is *regular* if its language is ω-regular (equivalently, if its automaton is a safety automaton). Likewise, a co-safety comparator is *regular* if its language is ω-regular (equivalently, automaton is a co-safety automaton).

By complementation duality of safety and co-safety languages, comparison language for an aggregate function f for non-strict inequality $\leq$ is safety iff the comparison language for f for strict inequality $<$ is co-safety. Since safety languages and safety automata are closed under intersection, safety comparison languages and regular safety comparator for non-strict inequality renders the same for equality. Similarly, since co-safety languages and co-safety automata are closed under union, co-safety comparison languages and regular co-safety comparators for non-strict inequality render the same for the inequality relation. Therefore, it suffices to examine the comparison language for one relation only.

It is worth noting that for weight-sequences A and B and all relations R, we have that $DS(A, d) \ \mathsf{R} \ DS(B, d)$ iff $DS(A - B, d) \ \mathsf{R} \ 0$, where $(A - B)_i = A_i - B_i$ for all $i \geq 0$. Prior work [11] shows that we can define *DS-comparison language* with upper bound μ, discount-factor $d > 1$, and relation R to accept infinite and bounded weight-sequence C over $\{-\mu, \dots, \mu\}$ iff $DS(C, d) \ \mathsf{R} \ 0$ holds. Similarly, DS-comparator with the same parameters μ, $d > 1$, accepts the DS-comparison language with parameters μ, d and R. We adopt these definitions for DS-comparison languages and DS-comparators

Throughout this section, the concatenation of finite sequence x with finite or infinite sequence y is denoted by $x \cdot y$ in the following.

3.1 DS-comparison Languages and Their Safety/Co-safety Properties

The central result of this section is that DS-comparison languages are safety or co-safety languages for all (integer and non-integer) discount-factors (Theorem 1). In particular, since DS-comparison languages are ω-regular for integer discount-factors [12], this implies that DS-comparators for integer discount-factors form safety or co-safety automata (Corollary 1).

The argument for safety/co-safety of DS-comparison languages depends on the property that the discounted-sum aggregate of all bounded weight-sequences exists for all discount-factors $d > 1$ [35].

Theorem 1. *Let $\mu > 1$ be the upper bound. For rational discount-factor $d > 1$*

1. *DS-comparison languages are safety languages for relations* $\mathsf{R} \in \{\leq, \geq, =\}$
2. *DS-comparison language are co-safety languages for relations* $\mathsf{R} \in \{<, >, \neq\}$.

Proof (Proof sketch). Due to duality of safety/co-safety languages, it suffices to show that DS-comparison language with $\leq$ is a safety language.

Let DS-comparison language with upper bound μ, rational discount-factor $d > 1$ and relation $\leq$ be denoted by $\mathcal{L}_{\leq}^{\mu,d}$. Suppose that $\mathcal{L}_{\leq}^{\mu,d}$ is not a safety language. Let W be a weight-sequence in the complement of $\mathcal{L}_{\leq}^{\mu,d}$ such that W does not have a bad prefix. Then the following hold: (a). $DS(W, d) > 0$ (b).

For all $i \geq 0$, the i-length prefix $W[i]$ of W can be extended to an infinite and bounded weight-sequence $W[i] \cdot Y^i$ such that $DS(W[i] \cdot Y^i, d) \leq 0$.

Note that $DS(W, d) = DS(W[i], d) + \frac{1}{d^i} \cdot DS(W[i \dots], d)$ where $W[i \dots] = W_i W_{i+1} \dots$ and $DS(W[i], d)$ is the discounted-sum of the finite sequence $W[i]$ i.e. $DS(W[i], d) = \Sigma_{j=0}^{j=i-1} \frac{W[j]}{d^j}$. Similarly, $DS(W[i] \cdot Y^i, d) = DS(W[i], d) + \frac{1}{d^i} \cdot DS(Y^i, d)$. The contribution of tail sequences $W[i \dots]$ and Y^i to the discounted-sum of W and $W[i] \cdot Y^i$, respectively, diminishes exponentially as the value of i increases. In addition, since W and $W[i] \cdot Y^i$ share a common i-length prefix $W[i]$, their discounted-sum values must converge to each other. The discounted sum of W is fixed and greater than 0, due to convergence there must be a $k \geq 0$ such that $DS(W[k] \cdot Y^k, d) > 0$. Contradiction to (b).

Therefore, DS-comparison language with $\leq$ is a safety language. □

Semantically this result implies that for a bounded-weight sequence C and rational discount-factor $d > 1$, if $DS(C, d) > 0$ then C must have a finite prefix C_{pre} such that the discounted-sum of the finite prefix is so large that no infinite extension by bounded weight-sequence Y can reduce the discounted-sum of $C_{\mathsf{pre}} \cdot Y$ with the same discount-factor d to zero or below.

Prior work shows that DS-comparison languages are expressed by Büchi automata iff the discount-factor is an integer [13]. Therefore:

Corollary 1. *Let $\mu > 1$ be the upper bound. For* integer *discount-factor $d > 1$*

1. *DS-comparators are regular safety for relations* $\mathsf{R} \in \{\leq, \geq, =\}$
2. *DS-comparators are regular co-safety for relations* $\mathsf{R} \in \{<, >, \neq\}$.

Lastly, it is worth mentioning that for the same reason [13] DS-comparators for non-integer rational discount-factors do not form safety or co-safety automata.

3.2 Deterministic DS-comparator for Integer Discount-Factor

This section issues deterministic safety/co-safety constructions for DS-comparators with integer discount-factors. This is different from prior works since they supply non-deterministic Büchi constructions only [11,12]. An outcome of DS-comparators being regular safety/co-safety (Corollary 1) is a proof that DS-comparators permit deterministic Büchi constructions, since non-deterministic and deterministic safety automata (and co-safety automata) have equal expressiveness [26]. Therefore, one way to obtain deterministic Büchi construction for DS-comparators is to determinize the non-deterministic constructions using standard procedures [26,36]. However, this will result in exponentially larger deterministic constructions. To this end, this section offers direct deterministic safety/co-safety automata constructions for DS-comparator that not only avoid an exponential blow-up but also match their non-deterministic counterparts in number of states (Theorem 3).

Key ideas. Due to duality and closure properties of safety/co-safety automata, we only present the construction of deterministic safety automata for DS-comparator with upper bound μ, integer discount-factor $d > 1$ and relation $\leq$, denoted by $\mathcal{A}^{\mu,d}_{\leq}$. We proceed by obtaining a *deterministic finite automaton*, (DFA), denoted by $\mathsf{bad}(\mu, d, \leq)$, for the language of bad-prefixes of $\mathcal{A}^{\mu,d}_{\leq}$ (Theorem 2). Trivial modifications to $\mathsf{bad}(\mu, d, \leq)$ will furnish the coveted deterministic safety automata for $\mathcal{A}^{\mu,d}_{\leq}$ (Theorem 3).

Construction. We begin with some definitions. Let W be a *finite* weight-sequence. By abuse of notation, the discounted-sum of finite-sequence W with discount-factor d is defined as $DS(W, d) = DS(W \cdot 0^{\omega}, d)$. The *recoverable-gap* of a finite weight-sequences W with discount factor d, denoted $\mathsf{gap}(W, d)$, is its normalized discounted-sum: If $W = \varepsilon$ (the empty sequence), $\mathsf{gap}(\varepsilon, d) = 0$, and $\mathsf{gap}(W, d) = d^{|W|-1} \cdot DS(W, d)$ otherwise [15]. Observe that the recoverable-gap has an inductive definition i.e. $\mathsf{gap}(\varepsilon, d) = 0$, where ε is the empty weight-sequence, and $\mathsf{gap}(W \cdot v, d) = d \cdot \mathsf{gap}(W, d) + v$, where $v \in \{-\mu, \ldots, \mu\}$.

This observation influences a sketch for $\mathsf{bad}(\mu, d, \leq)$. Suppose all possible values for recoverable-gap of weight sequences forms the set of states. Then, the transition relation of the DFA can mimic the inductive definition of recoverable gap i.e. there is a transition from state s to t on alphabet $v \in \{-\mu, \ldots, \mu\}$ iff $t = d \cdot s + v$, where s and v are recoverable-gap values of weight-sequences. There is one caveat here: There are infinitely many possibilities for the values of recoverable gap. We need to limit the recoverable gap values to finitely many values of interest. The core aspect of this construction is to identify these values.

First, we obtain a lower bound on recoverable gap for bad-prefixes of $\mathcal{A}^{\mu,d}_{\leq}$:

Lemma 1. *Let μ and $d > 1$ be the bound and discount-factor, resp. Let $\mathsf{T} = \frac{\mu}{d-1}$ be the threshold value. Let W be a non-empty, bounded, finite weight-sequence. Weight sequence W is a bad-prefix of $\mathcal{A}^{\mu,d}_{\leq}$ iff $\mathsf{gap}(W, d) > \mathsf{T}$.*

Proof. Let a finite weight-sequence W be a bad-prefix of $\mathcal{A}^{\mu,d}_{\leq}$. Then, $DS(W \cdot Y, d) > 0$ for all infinite and bounded weight-sequences Y. Since $DS(W \cdot Y, d) = DS(W, d) + \frac{1}{d^{|W|}} \cdot DS(Y, d)$, we get $\inf(DS(W, d) + \frac{1}{d^{|W|}} \cdot DS(Y, d)) > 0 \implies DS(W, d) + +\frac{1}{d^{|W|}} \cdot \inf(DS(Y, d)) > 0$ as W is a fixed sequence. Hence $DS(W, d) + \frac{-\mathsf{T}}{d^{|W|-1}} > 0 \implies \mathsf{gap}(W, d) - T > 0$. Conversely, for all infinite, bounded, weight-sequence Y, $DS(W \cdot Y, d) \cdot d^{|W|-1} = \mathsf{gap}(W, d) + \frac{1}{d} \cdot DS(Y, d)$. Since $\mathsf{gap}(W, d) > T$, $\inf(DS(Y, d)) = -\mathsf{T} \cdot d$, we get $DS(W \cdot Y, d) > 0$. □

Since all finite and bounded extensions of bad-prefixes are also bad-prefixes, Lemma 1 implies that if the recoverable-gap of a finite sequence is strinctly lower that threshold T, then recoverable gap of all of its extensions also exceed T. Since recoverable gap exceeding threshold T is the precise condition for bad-prefixes, all states with recoverable gap exceeding T can be merged into a single state. Note, this state forms an accepting sink in $\mathsf{bad}(\mu, d, \leq)$.

Next, we attempt to merge very low recoverable gap value into a single state. For this purpose, we define *very-good prefixes* for $\mathcal{A}_{\leq}^{\mu,d}$: A finite and bounded weight-sequence W is a *very good* prefix for language of $\mathcal{A}_{\leq}^{\mu,d}$ if for all infinite, bounded extensions of W by Y, $DS(W\cdot Y, d) \leq 0$. A proof similar to Lemma 1 proves an upper bound for the recoverable gap of very-good prefixes of $\mathcal{A}_{\leq}^{\mu,d}$:

Lemma 2. *Let μ and $d>1$ be the bound and discount-factor, resp. Let $\mathsf{T} = \frac{\mu}{d-1}$ be the threshold value. Let W be a non-empty, bounded, finite weight-sequence. Weight-sequence W is a very-good prefix of $\mathcal{A}_{\leq}^{\mu,d}$ iff $\mathsf{gap}(W,d) \leq -\mathsf{T}$.*

Clearly, finite extensions of very-good prefixes are also very-good prefixes. Further, $\mathsf{bad}(\mu,d,\leq)$ must not accept very-good prefixes. Thus, by reasoning as earlier we get that all recoverable gap values that are less than or equal to $-\mathsf{T}$ can be merged into one non-accepting sink state in $\mathsf{bad}(\mu,d,\leq)$.

Finally, for an integer discount-factor the recoverable gap is an integer. Let $\lfloor x \rfloor$ denote the floor of $x\in\mathbb{R}$ e.g. $\lfloor 2.3\rfloor = 2$, $\lfloor -2\rfloor = -2$, $\lfloor -2.3\rfloor = -3$. Then,

Corollary 2. *Let μ be the bound and $d>1$ an integer discount-factor. Let $\mathsf{T} = \frac{\mu}{d-1}$ be the threshold. Let W be a non-empty, bounded, finite weight-sequence.*

- *W is a bad prefix of $\mathcal{A}_{\leq}^{\mu,d}$ iff $\mathsf{gap}(W,d) > \lfloor \mathsf{T}\rfloor$*
- *W is a very-good prefix of $\mathcal{A}_{\leq}^{\mu,d}$ iff $\mathsf{gap}(W,d) \leq \lfloor -\mathsf{T}\rfloor$*

So, the recoverable gap value is either one of $\{\lfloor -\mathsf{T}\rfloor + 1, \dots, \lfloor \mathsf{T}\rfloor\}$, or less than or equal to $\lfloor -\mathsf{T}\rfloor$, or greater than $\lfloor \mathsf{T}\rfloor$. This curbs the state-space to $\mathcal{O}(\mu)$-many values of interest, as $\mathsf{T} = \frac{\mu}{d-1} < \frac{\mu\cdot d}{d-1}$ and $1 < \frac{d}{d-1} \leq 2$. Lastly, since $\mathsf{gap}(\varepsilon, d) = 0$, state 0 must be the initial state.

Construction of $\mathsf{bad}(\mu,d,\leq)$. Let μ be the upper bound, and $d>1$ be the integer discount-factor. Let $\mathsf{T} = \frac{\mu}{d-1}$ be the threshold value. The finite-state automata $\mathsf{bad}(\mu,d,\leq) = (S, s_I, \Sigma, \delta, \mathcal{F})$ is defined as follows:

- States $S = \{\lfloor -\mathsf{T}\rfloor + 1, \dots, \lfloor \mathsf{T}\rfloor\} \cup \{\mathsf{bad}, \mathsf{veryGood}\}$
- Initial state $s_I = 0$, Accepting states $\mathcal{F} = \{\mathsf{bad}\}$
- Alphabet $\Sigma = \{-\mu, -\mu+1, \dots, \mu-1, \mu\}$
- Transition function $\delta \subseteq S\times\Sigma \rightarrow S$ where $(s,a,t)\in\delta$ then:
  1. If $s \in \{\mathsf{bad}, \mathsf{veryGood}\}$, then $t = s$ for all $a\in\Sigma$
  2. If $s \in \{\lfloor -\mathsf{T}\rfloor + 1, \dots, \lfloor \mathsf{T}\rfloor\}$, and $a\in\Sigma$
     (a) If $\lfloor -\mathsf{T}\rfloor < d\cdot s + a \leq \lfloor \mathsf{T}\rfloor$, then $t = d\cdot s + a$
     (b) If $d\cdot s + a > \lfloor \mathsf{T}\rfloor$, then $t = \mathsf{bad}$
     (c) If $d\cdot s + a \leq \lfloor -\mathsf{T}\rfloor$, then $t = \mathsf{veryGood}$

Theorem 2. *Let μ be the upper bound, $d>1$ be the integer discount-factor.* $\mathsf{bad}(\mu,d,\leq)$ *accepts finite, bounded, weight-sequence iff it is a bad-prefix of $\mathcal{A}_{\leq}^{\mu,d}$.*

Proof (Proof sketch). First note that the transition relation is deterministic and complete. Therefore, every word has a unique run in $\mathsf{bad}(\mu,d,\leq)$. Let last be the last state in the run of finite, bounded, weight-sequence W in the DFA. Use induction on the length of W to prove the following:

- $\mathsf{last} \in \{\lfloor -\mathsf{T} \rfloor + 1, \ldots, \lfloor \mathsf{T} \rfloor\}$ iff $\mathsf{gap}(W, d) = \mathsf{last}$
- $\mathsf{last} = \mathsf{bad}$ iff $\mathsf{gap}(W, d) > \lfloor \mathsf{T} \rfloor$
- $\mathsf{last} = \mathsf{veryGood}$ iff $\mathsf{gap}(W, d) \leq \lfloor -\mathsf{T} \rfloor$

Therefore, a finite, bounded weight-sequence is accepted iff its recoverable gap is greater than $\lfloor \mathsf{T} \rfloor$. In other words, iff it is a bad-prefix of $\mathcal{A}^{\mu,d}_{\leq}$. □

$\mathcal{A}^{\mu,d}_{\leq}$ is obtained from $\mathsf{bad}(\mu, d, \leq)$ by applying co-Büchi acceptance condition.

Theorem 3. *Let μ be the upper bound, and $d > 1$ be the integer discount-factor. DS-comparator for all inequalities and equality are either deterministic safety or deterministic co-safety automata with $\mathcal{O}(\mu)$ states.*

As a matter of fact, the most compact non-deterministic DS-comparator constructions with parameters μ, d and R also contain $\mathcal{O}(\mu)$ states [11].

3.3 Quantitative Inclusion with Safety/Co-safety Comparators

This section investigates quantitative language inclusion with regular safety/co-safety comparators. Unlike quantitative inclusion with regular comparators, quantitative inclusion with regular safety/co-safety comparators is able to circumvent Büchi complementation with intermediate subset-construction steps. As a result, complexity of quantitative inclusion with regular safety/co-safety comparator is lower than the same with regular comparators [12] (Theorem 4). Finally, since DS-comparators are regular safety/co-safety comparators, the algorithm for quantitative inclusion with regular safety/co-safety comparators applies to DS-inclusion yielding a lower complexity algorithm for DS-inclusion (Corollary 5).

Key Ideas A run of word w in a weighted-automaton is *maximal* if its weight is the supremum weight of all runs of w in the weighted-automaton. A run ρ_P of w in P is a *counterexample* for $P \subseteq Q$ (or $P \subset Q$) iff there exists a maximal run sup_Q of w in Q such that $wt(\rho_P) > wt(sup_Q)$ (or $wt(\rho_P) \geq wt(sup_Q)$). Consequently, $P \subseteq Q$ (or $P \subset Q$) iff there are no counterexample runs in P. Therefore, the roadmap to solve quantitative inclusion for regular safety/co-safety comparators is as follows:

1. Use regular safety/co-safety comparators to construct the *maximal automaton* of Q i.e. an automaton that accepts all maximal runs of Q (Corollary 3).
2. Use the regular safety/co-safety comparator and the maximal automaton to construct a *counterexample automaton* that accepts all counterexample runs of the inclusion problem $P \subseteq Q$ (or $P \subset Q$) (Lemma 5).

3. Solve quantitative inclusion for safety/co-safety comparator by checking for emptiness of the counterexample (Theorem 4).
   Finally, since DS-comparators are regular safety/co-safety automaton (Corollary 1), apply Theorem 4 to obtain an algorithm for DS-inclusion that uses regular safety/co-safety comparators (Corollary 5).

Let W be a weighted automaton. Then the *annotated automaton* of W, denoted by $\hat{W}$, is the Büchi automaton obtained by transforming transition $s \xrightarrow{a} t$ with weight v in W to transition $s \xrightarrow{a,v} t$ in $\hat{W}$. Observe that $\hat{W}$ is a safety automaton since all its states are accepting. A run on word w with weight sequence wt in W corresponds to an *annotated word* (w, wt) in $\hat{W}$, and vice-versa.

Maximal Automaton. This section covers the construction of the *maximal automaton* from a weighted automaton. Let W and $\hat{W}$ be a weighted automaton and its annotated automaton, respectively. We call an annotated word (w, wt_1) in $\hat{W}$ *maximal* if for all other words of the form (w, wt_2) in $\hat{W}$, $wt(wt_1) \geq wt(wt_2)$. Clearly, (w, wt_1) is a maximal word in $\hat{W}$ iff word w has a run with weight sequence wt_1 in W that is maximal. We define *maximal automaton* of weighted automaton W, denoted $\mathsf{Maximal}(W)$, to be the automaton that accepts all maximal words of its annotated automata $\hat{W}$.

We show that when the comparator is regular safety/co-safety, the construction of the maximal automata incurs a $2^{\mathcal{O}(n)}$ blow-up. This section exposes the construction for maximal automaton when comparator for non-strict inequality is regular safety. The other case when the comparator for strict inequality is regular co-safety has been deferred to the appendix.

Lemma 3. *Let W be a weighted automaton with regular safety comparator for non-strict inequality. Then the language of* $\mathsf{Maximal}(W)$ *is a safety language.*

Proof (Proof sketch). An annotated word (w, wt_1) is not maximal in $\hat{W}$ for one of the following two reasons: Either (w, wt_1) is not a word in $\hat{W}$, or there exists another word (w, wt_2) in $\hat{W}$ s.t. $wt(wt_1) < wt(wt_2)$ (equivalently (wt_1, wt_2) is not in the comparator non-strict inequality). Both $\hat{W}$ and comparator for non-strict inequality are safety languages, so the language of maximal words must also be a safety language. □

We now proceed to construct the safety automata for $\mathsf{Maximal}(W)$

Intuition. The intuition behind the construction of maximal automaton follows directly from the definition of maximal words. Let $\hat{W}$ be the annotated automaton for weighted automaton W. Let $\hat{\Sigma}$ denote the alphabet of $\hat{W}$. Then an annotated word $(w, wt_1) \in \hat{\Sigma}^\omega$ is a word in $\mathsf{Maximal}(W)$ if (a) $(w, wt_1) \in \hat{W}$, and (b) For all words $(w, wt_2) \in \hat{W}$, $wt(wt_1) \geq wt(wt_2)$.

The challenge here is to construct an automaton for condition (b). Intuitively, this automaton simulates the following action: As the automaton reads word (w, wt_1), it must spawn all words of the form (w, wt_2) in $\hat{W}$, while also ensuring that $wt(wt_1) \geq wt(wt_2)$ holds for every word (w, wt_2) in $\hat{W}$. Since $\hat{W}$ is a safety automaton, for a word $(w, wt_1) \in \hat{\Sigma}^\omega$, all words of the form $(w, wt_2) \in \hat{W}$ can be traced by subset-construction. Similarly since the comparator C for non-strict inequality ($\geq$) is a safety automaton, all words of the form $(wt_1, wt_2) \in C$ can be traced by subset-construction as well. The construction needs to carefully align the word (w, wt_1) with the all possible $(w, wt_2) \in \hat{W}$ and $(wt_1, wt_2) \in C$.

Construction of $\mathsf{Maximal}(W)$. Let W be a weighted automaton, with annotated automaton $\hat{W}$ and C denote its regular safety comparator for non-strict inequality. Let S_W denote the set of states of W (and $\hat{W}$) and S_C denote the set of states of C. We define $\mathsf{Maximal}(W) = (S, s_I, \hat{\Sigma}, \delta, \mathcal{F})$ as follows:

- Set of states S consists of tuples of the form (s, X), where $s \in S_W$, and $X = \{(t, c) | t \in S_W, c \in S_C\}$
- $\hat{\Sigma}$ is the alphabet of $\hat{W}$
- Initial state $s_I = (s_w, \{(s_w, s_c)\})$, where s_w and s_c are initial states in $\hat{W}$ and C, respectively.
- Let states $(s, X), (s, X') \in S$ such that $X = \{(t_1, c_1), \ldots, (t_n, c_n)\}$ and $X' = \{(t'_1, c'_1), \ldots, (t'_m, c'_m)\}$. Then $(s, X) \xrightarrow{(a,v)} (s', X') \in \delta$ iff
  1. $s \xrightarrow{(a,v)} s'$ is a transition in $\hat{W}$, and
  2. $(t'_j, c'_j) \in X'$ if there exists $(t_i, c_i) \in X$, and a weight v' such that $t_i \xrightarrow{a,v'} t'_j$ and $c_i \xrightarrow{v,v'} c'_j$ are transitions in $\hat{W}$ and C, respectively.
- $(s, \{(t_1, c_1), \ldots, (t_n, c_n)\}) \in \mathcal{F}$ iff s and all t_i are accepting in $\hat{W}$, and all c_i is accepting in C.

Lemma 4. *Let W be a weighted automaton with regular safety comparator C for non-strict inequality. Then the size of* $\mathsf{Maximal}(W)$ *is* $|W| \cdot 2^{\mathcal{O}(|W| \cdot |C|)}$.

Proof (Proof sketch). A state $(s, \{(t_1, c_1), \ldots, (t_n, c_n)\})$ is non-accepting in the automata if one of s,t_i or c_j is non-accepting in underlying automata $\hat{W}$ and the comparator. Since $\hat{W}$ and the comparator automata are safety, all outgoing transitions from a non-accepting state go to non-accepting state in the underlying automata. Therefore, all outgoing transitions from a non-accepting state in $\mathsf{Maximal}(W)$ go to non-accepting state in $\mathsf{Maximal}(W)$. Therefore, $\mathsf{Maximal}(W)$ is a safety automaton. To see correctness of the transition relation, one must prove that transitions of type (1.) satisfy condition (a), while transitions of type (2.) satisfy condition (b). $\mathsf{Maximal}(W)$ forms the conjunction of (a) and (b), hence accepts the language of maximal words of W.

A similar construction proves that the maximal automata of weighted automata W with regular safety comparator C for strict inequality contains $|W| \cdot 2^{\mathcal{O}(|W| \cdot |C|)}$ states. In this case, however, the maximal automaton may not be a safety automaton. Therefore, Lemma 4 generalizes to:

Corollary 3. *Let W be a weighted automaton with regular safety/co-safety comparator C. Then* $\mathsf{Maximal}(W)$ *is a Büchi automaton of size* $|W| \cdot 2^{\mathcal{O}(|W| \cdot |C|)}$.

Counterexample Automaton. This section covers the construction of the counterexample automaton. Given weighted-automata P and Q, an annotated word (w, wt_P) in annotated automata $\hat{P}$ is a *counterexample word* of $P \subseteq Q$ (or $P \subset Q$) if there exists (w, wt_Q) in $\mathsf{Maximal}(Q)$ s.t. $wt(wt_P) > wt(wt_Q)$ (or $wt(wt_P) \geq wt(wt_Q)$). Clearly, annotated word (w, wt_P) is a counterexample word iff there exists a counterexample run of w with weight-sequence wt_P in P.

For this section, we abbreviate strict and non-strict to strct and nstrct, respectively. For $\mathsf{inc} \in \{\mathsf{strct}, \mathsf{nstrct}\}$, the *counterexample automaton* for inc-quantitative inclusion, denoted by $\mathsf{Counterexample}(\mathsf{inc})$, is the automaton that contains all counterexample words of the problem instance. We construct the counterexample automaton as follows:

Lemma 5. *Let P, Q be weighted-automata with regular safety/co-safety comparators. For $\mathsf{inc} \in \{\mathsf{strct}, \mathsf{nstrct}\}$, $\mathsf{Counterexample}(\mathsf{inc})$ is a Büchi automaton.*

Proof. We construct Büchi automaton $\mathsf{Counterexample}(\mathsf{inc})$ for $\mathsf{inc} \in \{\mathsf{strct}, \mathsf{nstrct}\}$ that contains the counterexample words of inc-quantitative inclusion. Since the comparator are regular safety/co-safety, $\mathsf{Maximal}(Q)$ is a Büchi automaton (Corollary 3). Construct the product $\hat{P} \times \mathsf{Maximal}(Q)$ such that transition $(p_1, q_1) \xrightarrow{a, v_1, v_2} (p_1, q_2)$ is in the product iff $p_1 \xrightarrow{a, v_1} p_1$ and $q_1 \xrightarrow{a, v_2} q_2$ are transitions in $\hat{P}$ and $\mathsf{Maximal}(Q)$, respectively. A state (p, q) is accepting if both p and q are accepting in $\hat{P}$ and $\mathsf{Maximal}(Q)$. One can show that the product accepts (w, wt_P, wt_Q) iff (w, wt_P) and (w, wt_Q) are words in $\hat{P}$ and $\mathsf{Maximal}(Q)$, respectively.

If $\mathsf{inc} = \mathsf{strct}$, intersect $\hat{P} \times \mathsf{Maximal}(Q)$ with comparator for $\geq$. If $\mathsf{inc} = \mathsf{nstrct}$, intersect $\hat{P} \times \mathsf{Maximal}(Q)$ with comparator for $>$. Since the comparator is a safety or co-safety automaton, the intersection is taken without the cyclic counter. Therefore, $(s_1, t_1) \xrightarrow{a, v_1, v_2} (s_2, t_2)$ is a transition in the intersection iff $s_1 \xrightarrow{a, v_1, v_2} s_2$ and $t_1 \xrightarrow{v_1, v_2} t_2$ are transitions in the product and the appropriate comparator, respectively. State (s, t) is accepting if both s and t are accepting. The intersection will accept (w, wt_P, wt_Q) iff (w, wt_P) is a counterexample of inc-quantitative inclusion. $\mathsf{Counterexample}(\mathsf{inc})$ is obtained by projecting out the intersection as follows: Transition $m \xrightarrow{a, v_1, v_2} n$ is transformed to $m \xrightarrow{a, v_1} n$. □

Quantitative Inclusion and DS-inclusion. In this section, we give the final algorithm for quantitative inclusion with regular safety/co-safety comparators. Since DS-comparators are regular safety/co-safety comparators, this gives us an algorithm for DS-inclusion with improved complexity than previous results.

Theorem 4. *Let P, Q be weighted-automata with regular safety/co-safety comparators. Let $C_{\leq}$ and $C_{<}$ be the comparators for $\leq$ and $<$, respectively. Then*

- *Strict quantitative inclusion $P \subset Q$ is reduced to emptiness checking of a Büchi automaton of size $|P||C_{\leq}||Q| \cdot 2^{\mathcal{O}(|Q| \cdot |C_{<}|)}$.*
- *Non-strict quantitative inclusion $P \subseteq Q$ is reduced to emptiness checking of a Büchi automaton of size $|P||C_{<}||Q| \cdot 2^{\mathcal{O}(|Q| \cdot |C_{<}|)}$.*

Proof. Strict and non-strict are abbreviated to strct and nstrct, respectively. For $\mathsf{inc} \in \{\mathsf{strct}, \mathsf{nstrct}\}$, inc-quantitative inclusion holds iff $\mathsf{Counterexample}(\mathsf{inc})$ is empty. Size of $\mathsf{Counterexample}(\mathsf{inc})$ is the product of size of P, $\mathsf{Maximal}(Q)$ (Corollary 3), and the appropriate comparator as described in Lemma 5. □

In contrast, quantitative inclusion with regular comparators reduces to emptiness of a Büchi automaton with $|P| \cdot 2^{\mathcal{O}(|P||Q||C| \cdot \log(|P||Q||C|))}$ states [12]. The $2^{\mathcal{O}(n \log n)}$ blow-up is unavoidable due to Büchi complementation. Hence, quantitative inclusion with regular safety/co-safety has lower worst-case complexity.

Lastly, we use the results of developed in previous sections to solve DS-inclusion. Since DS-comparators are regular safety/co-safety (Corollary 1), an immediate consequence of Theorem 4 is an improvement in the worst-case complexity of DS-inclusion in comparison to prior results with regular DS-comparators. Furthermore, since the regular safety/co-safety DS-comparators are of the same size for all inequalities (Theorem 3), we get:

Corollary 4. *Let P, Q be weighted-automata, and C be a regular safety/co-safety DS-comparator with integer discount-factor $d > 1$.* Strict DS-inclusion *reduces to emptiness checking of a* safety automaton *of size* $|P||C||Q| \cdot 2^{\mathcal{O}(|Q| \cdot |C|)}$.

Proof (Proof sketch). When comparator for non-strict inequality is safety-automaton, as it is for DS-comparator, the maximal automaton is a safety automaton (Lemma 3). One can then show that the counterexample automata is also a safety automaton.

A similar argument proves *non-strict DS-inclusion* reduces to emptiness of a *weak-Büchi automaton* [27] of size $|P||C||Q| \cdot 2^{\mathcal{O}(|Q| \cdot |C|)}$ (see Appendix).

Corollary 5 ([DS-inclusion with safety/co-safety comparator)**.** *Let P, Q be weighted-automata, and C be a regular (co)-safety DS-comparator with integer discount-factor $d > 1$.The complexity of DS-inclusion is* $|P||C||Q| \cdot 2^{\mathcal{O}(|Q| \cdot |C|)}$.

4 Implementation and Experimental Evaluation

The goal of the empirical analysis is to examine performance of DS-inclusion with integer discount-factor with safety/co-safety comparators against existing tools to investigate the practical merit of our algorithm. We compare against (a) Regular-comparator based tool QuIP, and (b) DS-determinization and linear-programming tool DetLP.

QuIP is written in C++, and invokes state-of-the-art Büchi language inclusion-solver RABIT [2]. We enable the `-fast` flag in RABIT, and tune its Java-threads with Xss, Xms, Xmx set to 1GB, 1GB and 8GB, respectively. DetLP is also written in C++, and uses linear programming solver GLPSOL provided by GLPK (GNU Linear Prog. Kit) [1]. We compare these tools along two axes: runtime and number of benchmarks solved.

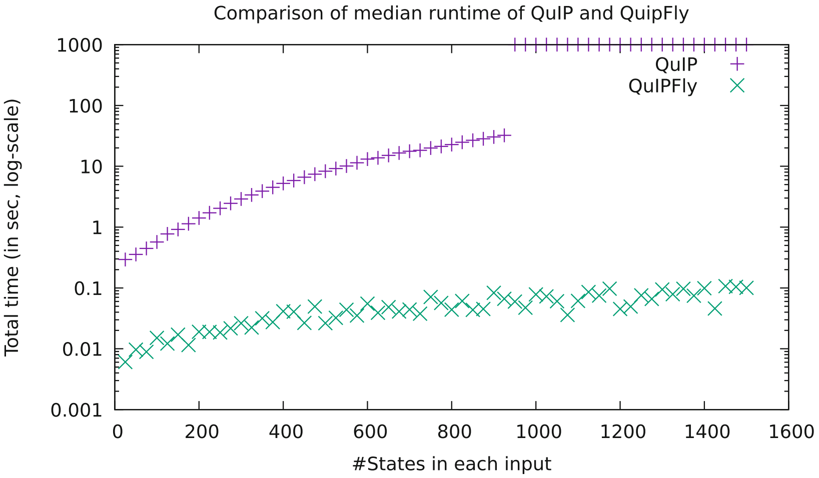

Fig. 1. $s_P = s_Q$ on x-axis, $wt = 4$, $\delta = 3$, $d = 3$, $P \subset Q$

Implementation Details. The algorithm for strict-DS-inclusion with integer discount factor $d > 1$ proposed in Corollary 4 and non-strict DS-inclusion checks for emptiness of the counterexample automata. A naive algorithm will construct the counterexample automata fully, and then check if they are empty by ensuring the absence of an *accepting lasso*.

We implement a more efficient algorithm. In our implementation, we make use of the fact that the constructions for DS-inclusion use subset-construction intermediate steps. This facilitates an *on-the-fly procedure* since successor states of state in the counterexample automata can be determined directly from input weighted automata and the comparator automata. The algorithm terminates as soon as an accepting lasso is detected. When an accepting lasso is absent, the algorithm traverses all states and edges of the counterexample automata.

We implement the optimized on-the-fly algorithm in a prototype QuIPFly. QuIPFly is written in Python 2.7.12. QuIPFly employs basic implementation-level optimizations to avoid excessive re-computation.

Design and Setup for Experiments. Due to lack of standardized benchmarks for weighted automata, we follow a standard approach to performance evaluation of automata-theoretic tools [3,30,38] by experimenting with *randomly generated* benchmarks, using random benchmark generation procedure described in [11].

The parameters for each experiment are number of states s_P and s_Q of weighted automata, transition density δ, maximum weight wt, integer discount-factor d, and $\mathsf{inc} \in \{\mathsf{strct}, \mathsf{nstrct}\}$. In each experiment, weighted automata P and Q are randomly generated, and runtime of inc-DS-inclusion for all three tools is reported with a timeout of 900 s. We run the experiment for each parameter tuple 50 times. All experiments are run on a single node of a high-performance cluster consisting of two quad-core Intel-Xeon processor running at 2.83 GHz,

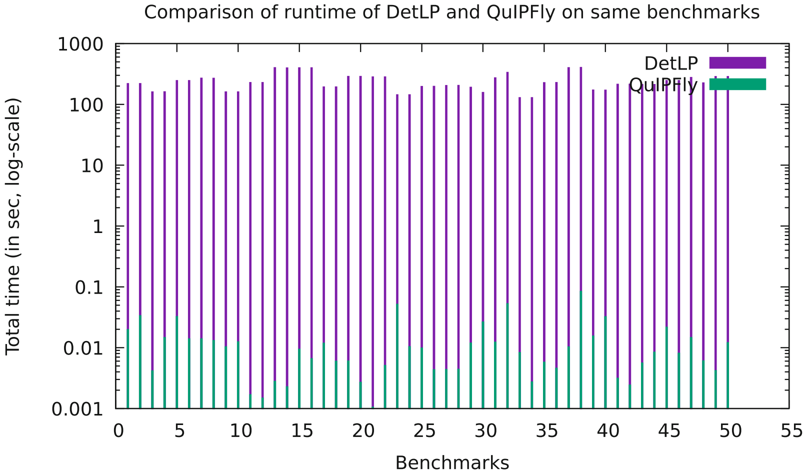

Fig. 2. $s_P = s_Q = 75$, $wt = 4$, $\delta = 3$, $d = 3$, $P \subset Q$

with 8 GB of memory per node. We experiment with $s_P = s_Q$ ranging from 0–1500 in increments of 25, $\delta \in \{3, 3.5, 4\}$, $d = 3$, and $wt \in \{d^1 + 1, d^3 - 1, d^4 - 1\}$.

Observations and Inferences.[1] For clarity of exposition, we present the observations for only one parameter-tuple. Trends and observations for other parameters were similar.

QuIPFly Outperforms. QuIP by at least an order of magnitude in runtime. Figure 1 plots the median runtime of all 50 experiments for the given parameter-values for QuIP and QuIPFly. More importantly, QuIPFly solves all of our benchmarks within a fraction of the timeout, whereas QuIP struggled to solve at least 50% of the benchmarks with larger inputs (beyond $s_P = s_Q = 1000$). Primary cause of failure is memory overflow inside RABIT. We conclude that regular safety/co-safety comparators outperform their regular counterpart, giving credit to the simpler subset-constructions vs. Büchi complementation.

QuIPFly Outperforms. DetLP comprehensively in runtime and in number of benchmarks solved. We were unable to plot DetLP in Fig. 1 since it solved fewer than 50% benchmarks even with small input instances. Figure 2 compares the runtime of both tools on the same set of 50 benchmarks for a representative parameter-tuple on which all 50 benchmarks were solved. The plot shows that QuIPFly beats DetLP by 2–4 orders of magnitude on all benchmarks.

Overall Verdict. Overall, QuIPFly outperforms QuIP and DetLP by a significant margin along both axes, runtime and number of benchmarks solved. This analysis gives unanimous evidence in favor of our safety/co-safety approach to solving DS-inclusion.

[1] Figures are best viewed online and in color.

5 Concluding Remarks

The goal of this paper was to build scalable algorithms for DS-inclusion. To this end, this paper furthers the understanding of language-theoretic properties of discounted-sum aggregate function by demonstrating that DS-comparison languages form safety and co-safety languages, and utilizes these properties to obtain a decision procedure for DS-inclusion that offers both tighter theoretical complexity and improved scalability. All in all, the key insights of this work are:

1. Pure automata-theoretic techniques of DS-comparator are better for DS-inclusion;
2. In-depth language-theoretic analysis improve both theoretical complexity and practical scalability of DS-inclusion;
3. DS-comparators are compact deterministic safety or co-safety automata.

To the best of our knowledge, this is the first work that applies language-theoretic properties such as safety/co-safety in the context of quantitative reasoning.

More broadly, this paper demonstrates that the close integration of language-theoretic and quantitative properties can render novel algorithms for quantitative reasoning that can benefit from advances in qualitative reasoning.

Acknowledgements. We thank anonymous reviewers for their comments. We thank D. Fried, L. M. Tabajara, and A. Verma for their valuable inputs on initial drafts of the paper. This work was partially supported by NSF Grant No. CCF-1704883.

References

1. GLPK. https://www.gnu.org/software/glpk/
2. Rabit-Reduce. http://www.languageinclusion.org/
3. Abdulla, P.A., et al.: Simulation subsumption in ramsey-based büchi automata universality and inclusion testing. In: Proceedings of CAV, pp. 132–147. Springer (2010)
4. Abdulla, P.A., et al.. Advanced ramsey-based büchi automata inclusion testing. In: Proceedings of CONCUR, vol. 11, pp. 187–202. Springer (2011)
5. Alpern, B., Schneider, F.B.: Recognizing safety and liveness. Distrib. Comput. **2**(3), 117–126 (1987)
6. Alur, R., Mamouras, K.: An introduction to the streamqre language. Dependable Softw. Syst. Eng. **50**, 1 (2017)
7. Aminof, B., Kupferman, O., Lampert, R.: Reasoning about online algorithms with weighted automata. Trans. Algorithms **6**(2), 28 (2010)
8. Andersen, G., Conitzer, V.: Fast equilibrium computation for infinitely repeated games. In: Proceedings of AAAI, pp. 53–59 (2013)
9. Andersson, D.: An improved algorithm for discounted payoff games. In: ESSLLI Student Session, pp. 91–98 (2006)
10. Baier, C.: Probabilistic model checking. In: Dependable Software Systems Engineering, pp. 1–23 (2016)
11. Bansal, S., Chaudhuri, S., Vardi, M.Y.: Automata vs linear-programming discounted-sum inclusion. In: Proceedings of International Conference on Computer-Aided Verification (CAV) (2018)

12. Bansal, S., Chaudhuri, S., Vardi, M.Y. : Comparator automata in quantitative verification. In: Proceedings of International Conference on Foundations of Software Science and Computation Structures (FoSSaCS) (2018)
13. Bansal, S., Chaudhuri, S., Vardi, M.Y.: Comparator automata in quantitative verification (full version). CoRR, abs/1812.06569 (2018)
14. Bloem, R., Chatterjee, K., Henzinger, T.A., Jobstmann, B.: Better quality in synthesis through quantitative objectives. In: Bouajjani, A., Maler, O. (eds.) CAV 2009. LNCS, vol. 5643, pp. 140–156. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02658-4_14
15. Boker, U., Henzinger, T.A.: Exact and approximate determinization of discounted-sum automata. LMCS **10**(1), 1–13 (2014)
16. Chakrabarti, A., Chatterjee, K., Henzinger, T.A., Kupferman, O., Majumdar, R.: Verifying quantitative properties using bound functions. In: Borrione, D., Paul, W. (eds.) CHARME 2005. LNCS, vol. 3725, pp. 50–64. Springer, Heidelberg (2005). https://doi.org/10.1007/11560548_7
17. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. Trans. Computat. Logic **11**(4), 23 (2010)
18. Chaudhuri, S., Sankaranarayanan, S., Vardi, M.Y.: Regular real analysis. In: Proceedings of LICS, pp. 509–518 (2013)
19. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1022–1037. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45061-0_79
20. Doyen, L., Raskin, J.-F.: Antichain algorithms for finite automata. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 2–22. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-12002-2_2
21. Droste, M., Kuich, W., Vogler, H.: Handbook of Weighted Automata. Springer, Berlin (2009)
22. D'Antoni, L., Samanta, R., Singh, R.: Qlose: program repair with quantitative objectives. In: Proceedings of CAV, pp. 383–401. Springer (2016)
23. Filiot, E., Gentilini, R., Raskin, J.-F.: Quantitative languages defined by functional automata. In: Koutny, M., Ulidowski, I. (eds.) CONCUR 2012. LNCS, vol. 7454, pp. 132–146. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32940-1_11
24. He, K., Lahijanian, M., Kavraki, L.E., Vardi, M.Y.: Reactive synthesis for finite tasks under resource constraints. In: 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5326–5332. IEEE (2017)
25. Hu, Q., DAntoni, L.: Syntax-guided synthesis with quantitative syntactic objectives. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 386–403. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_21
26. Kupferman, O., Vardi, M.Y.: Model checking of safety properties. In: Halbwachs, N., Peled, D. (eds.) CAV 1999. LNCS, vol. 1633, pp. 172–183. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48683-6_17
27. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. Trans. Computat. Logic **2**(3), 408–429 (2001)
28. Kwiatkowska, M.: Quantitative verification: models, techniques and tools. In: Proceedings 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), pp. 449–458. ACM Press, September 2007

29. Lahijanian, M., Almagor, S., Fried, D., Kavraki, L.E., Vardi, M.Y.: This time the robot settles for a cost: a quantitative approach to temporal logic planning with partial satisfaction. In: AAAI, pp. 3664–3671 (2015)
30. Mayr, R., Clemente, L.: Advanced automata minimization. ACM SIGPLAN Not. **48**(1), 63–74 (2013)
31. Mohri, M.: Weighted automata algorithms. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2009). https://doi.org/10.1007/978-3-642-01492-5_6
32. Mohri, M., Pereira, F., Riley, M.: Weighted finite-state transducers in speech recognition. Comput. Speech Lang. **16**(1), 69–88 (2002)
33. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT press, Cambridge (1994)
34. Puterman, M.L.: Markov decision processes. Handbooks Oper. Res. Manag. Sci. **2**, 331–434 (1990)
35. Rudin, W.: Principles of Mathematical Analysis, vol. 3. McGraw-Hill, New York (1964)
36. Safra, S.: On the complexity of ω-automata. In: Proceedings of FOCS, pp. 319–327. IEEE (1988)
37. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning, vol. 135. MIT press, Cambridge (1998)
38. Tabakov, D., Vardi, M.Y.: Experimental evaluation of classical automata constructions. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS (LNAI), vol. 3835, pp. 396–411. Springer, Heidelberg (2005). https://doi.org/10.1007/11591191_28
39. Thomas, W., Wilke, T., et al.: Automata, Logics, and Infinite Games: A Guide to Current Research, vol. 2500. Springer Science & Business Media, Berlin (2002)
40. Vardi, M.Y.: The büchi complementation saga. In: Annual Symposium on Theoretical Aspects of Computer Science, pp. 12–22. Springer (2007)