Journal of Computer and System Sciences 106 (2019) 1–22

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Decidable weighted expressions with Presburger combinators

Emmanuel Filiot[1], Nicolas Mazzocchi[*,2], Jean-François Raskin[3]

Université libre de Bruxelles, Belgium

A R T I C L E   I N F O

Article history:
Received 4 January 2018
Received in revised form 5 April 2019
Accepted 21 May 2019
Available online 14 June 2019

Keywords:
Weighted automata
Presburger arithmetic
Quantitative languages
Weighted expressions

A B S T R A C T

In this paper, we investigate the expressive power and the algorithmic properties of weighted expressions, which define functions from finite words to integers. First, we consider a slight extension of an expression formalism, introduced by Chatterjee et al. in the context of infinite words, in order to combine values given by unambiguous *sum*-automata, using Presburger arithmetic. We show that important decision problems such as emptiness, universality, inclusion and equivalence are PSpace-C for these expressions. We then investigate the extension of these expressions with Kleene star. This allows to iterate an expression over smaller fragments of the input word, and to combine the results by taking their iterated sum. The decision problems turn out to be undecidable, but we introduce a decidable and still expressive class of synchronised expressions.

1. Introduction

Quantitative languages (QL) Quantitative languages of finite words generalise Boolean languages, which can be seen as (partial) functions from finite words to Boolean values, to (partial) functions from finite words to an arbitrary set of values S. Quantitative languages have been studied for long in the context of weighted automata [10], and they have recently received a particular attention from the verification community, for their application in modelling system *quality* [6], lifting classical Boolean verification problems to a quantitative setting.

In this paper, we consider the case of integer weights ($S = \mathbb{Z}$) and the classical decision problems of emptiness, universality and inclusion. The *inclusion* problem asks whether two given QL $f, g : \Sigma^* \to \mathbb{Z}$ satisfy $f \leq g$, i.e. whether $\mathrm{dom}(f) \subseteq \mathrm{dom}(g)$ (where dom() denotes the domain) and in that case whether $f(u) \leq g(u)$ for all $u \in \mathrm{dom}(f)$. The *equivalence* problem asks whether $f \leq g$ and $g \leq f$. The *universality* problem asks, given some threshold value $\nu \in \mathbb{Z}$ and some QL f, whether $\nu \leq f(u)$ for all $u \in \mathrm{dom}(f)$. The *emptiness* problem asks whether $\nu \leq f(u)$ for some $u \in \mathrm{dom}(f)$. We say that a formalism for QL is decidable if all these problems are decidable.

A popular formalism to define QL over $\mathbb{Z}$ is that of weighted automata over the tropical semiring $(\mathbb{Z}, \max, +)$ [10]. We only consider the tropical semiring in this paper and therefore just call these automata weighted automata (WA). WA are finite automata whose transitions are extended with values in $\mathbb{Z}$, the $+$ operator is used to compute the value of a run (as

* Corresponding author.
E-mail address: nicolas.mazzocchi@ulb.ac.be (N. Mazzocchi).

[1] E. Filiot is a research associate of F.R.S.-FNRS. This work has been supported by the following projects: the ARC Project Transform (Federation Wallonie-Brussels), the FNRS CDR project Flare CDR-J.0131.16.

[2] N. Mazzocchi is a PhD student funded by a FRIA fellowship from the F.R.S.-FNRS.

[3] J.-F. Raskin is supported by an ERC Starting Grant (279499: inVEST), by the ARC project — Non-Zero Sum Game Graphs: Applications to Reactive Synthesis and Beyond — funded by the Fédération Wallonie-Bruxelles, and by a Professeur Francqui de Recherche grant awarded by the Francqui Fondation.

https://doi.org/10.1016/j.jcss.2019.05.005
0022-0000/© 2019 Elsevier Inc. All rights reserved.

the sum of all values occurring on its transitions) and, for any word $u \in \Sigma^*$, max is used to aggregate all the values of the accepting runs on u, then defining *the* value of u. However, WA have undecidable inclusion, equivalence and universality problems [19], even if they are linearly ambiguous [8].

Decidable formalisms for quantitative languages and objectives The most expressive known class of WA enjoying decidability is that of finitely ambiguous WA [13], for which there is a uniform bound on the degree of non-determinism (there exists a constant c such that any word admits at most c accepting runs). This class is expressively equivalent to the class of finite-valued WA [13] (there exists a constant c such that for any word, the number of different values of its accepting runs is at most[4] c). Moreover, $(\max, +)$-automata are not closed under simple operations such as min and minus [17]. In particular, basic functions such as[5] $u \mapsto \min(\#_a(u), \#_b(u))$ and (as a consequence) $u \mapsto |f(u) - g(u)|$ are not definable by WA, even if f, g are [17].

To cope with the expressiveness and undecidability issues, a class of weighted expressions was introduced in [5] in the context of ω-words (where the mean-payoff is taken instead of the sum). Casted to finite words, the idea is to use (input) deterministic WA[6] as atoms, and to combine them using the operations max, min, $+$, and $-$. The decision problems defined before were shown to be PSPACE-C [22] over ω-words. One limitation of this formalism, casted to finite words, is that it is not expressive enough to capture finitely ambiguous WA as we show, yielding two incomparable classes of QL. In this paper, our objective is to push the expressiveness of weighted expressions as far as possible while retaining decidability, and to capture both finitely ambiguous WA and the expressions of [5], for finite words. Nevertheless we conjecture in Section 7 that our formalisms are incomparable to linearly ambiguous WA. We introduce two formalisms of strictly increasing expressive power, namely monolithic expressions and expressions with iterated sum, which extend monolithic expressions with some kind of unambiguous Kleene star operation.

Monolithic expressions with Presburger combinators We define in Section 3 the monolithic expressions, inspired from [5]. Compared to [5], we notice that taking *unambiguous* WA instead of deterministic WA does not change the complexity of the decision problems but strictly extend the expressive power of the expressions. To combine those atoms, monolithic expressions are allowed to use what we call *Presburger combinators* instead of just max, min, $+$, $-$, and this can be done for free (complexity-wise). Presburger combinators are any function definable in Existential Presburger arithmetic. For instance, the absolute value function $|x|$ is definable by the formula $\phi_{abs}(x, r) = x \le 0 \wedge x = -r \vee x > 0 \wedge x = r$, and $\max(x, y)$ is defined by the formula $\phi_{max}(x, y, r) = (x \le r \wedge y \le r) \wedge (x = r \vee y = r)$.

Let us precisely state our results. Since any finitely ambiguous WA A can be decomposed into a finite union of unambiguous WA[13], our formalism captures finitely ambiguous WA (by using the max operator).

We show that all the decision problems are PSPACE-C, matching the complexity of [22]. It is important to mention that this complexity result cannot be directly obtained from [22] which is on ω-words with mean-payoff automata as atoms (hence the value of an infinite word is prefix-independent). Instead, we rely on techniques that were used in the context of Parikh automata [18], in particular to show that the intersection emptiness of n Parikh automata is PSPACE-C [12]. The conference version of this contribution [15] was using reversal-bounded counter machines [16]. Using some result of [12] allowed us to simplify the proof, compared to that version.

Expressions with iterated sum The previous expressions are monolithic in the sense that first, some values are computed by weighted automata applied on the whole input word, and then these values are combined using Presburger combinators. It is not possible to iterate expressions on factors of the input word, and to aggregate all the values computed on these factors, for instance by a sum operation. The basic operator for iteration is that of Kleene star (extended to quantitative languages), which we call more explicitly *iterated-sum*. It has already been defined in [10], and its unambiguous version considered in [2] to obtain an expression formalism equivalent to unambiguous WA.

We investigate in Section 4 the extension of monolithic expressions with unambiguous iterated-sum, which we just call iterated-sum in the paper. The idea is, given an expression E which applies on a domain D, the expression $E^{\circledast}$ is defined only on words w that can be uniquely decomposed (hence the name unambiguous) into factors $w_1 w_2 \dots w_n = w$ such that $w_i \in D$, and the value of w is then $\sum_{i=1}^{n} E(w_i)$.

Unfortunately, we show that such an extension yields undecidability (if 2 or more iterated sum operations occur in parallel in the expression). The undecidability is caused by the fact that sub-expressions $E^{\circledast}$ may decompose the input word in different ways. We therefore define the class of so called *synchronised* expressions with iterated-sum, which forbids this behaviour. We show that while being expressive (for instance, they can define QL beyond finitely ambiguous WA), decidability is recovered.

The proof goes via a new weighted automata model (Section 5), called *weighted chop automata*, that "chop" the input word into smaller factors, recursively apply smaller chop automata on the factors to compute their values, which are then aggregated by taking their sum. In their synchronised version, we show decidability for chop automata.

[4] Note that it does not imply that there are at most c accepting runs.
[5] Where $\#_\sigma(u)$ is the number of occurrences of σ in u.
[6] Also known as sequential WA in the literature.

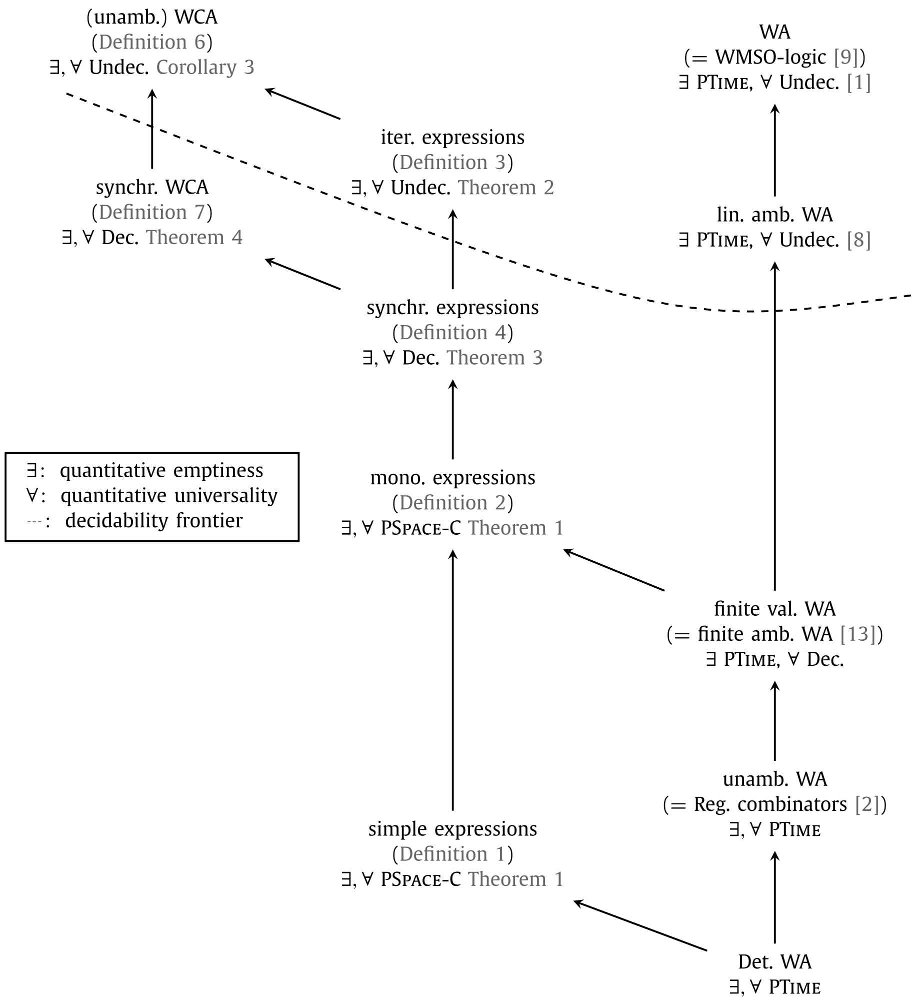


Fig. 1. Expressiveness and decidability properties of weighted expressions and automata (the arrows denote strict inclusion between the classes of quantitative languages defined by the formalisms)

Figure 1 gives a comparison between the weighted expression formalisms and the weighted chop automata (WCA) introduced in this paper, as well as classes of weighted automata (WA). The correspondence between weighted MSO (WMSO) and weighted automata can be found in [9] and a simple proof for the undecidability of universality for weighted automata is given in [1]. We remind the reader that weighted automata are over the tropical semiring in this paper.

2. Quantitative languages

Words, languages and quantitative languages Let Σ be a finite alphabet and Σ^* be the set of finite words over Σ, with ε the empty word. For $w \in \Sigma^*$, $|w|$ denotes its length, in particular $|\varepsilon| = 0$. Given two words $u, v \in \Sigma^*$, the distance between u and v is defined as $d(u, v) = |u| + |v| - 2|\sqcap(u, v)|$, where $\sqcap(u, v)$ denotes the longest common prefix of u and v. A *quantitative language*[7] (QL) is a partial function $f\colon \Sigma^* \to \mathbb{Z}$, whose domain is denoted by $\mathrm{dom}(f)$. E.g., consider the quantitative language mapping any word $w \in \Sigma^*$ to the number of occurrences $\#_\sigma(w)$ of some symbol $\sigma \in \Sigma$ in w. A QL f is *Lipschitz-continuous* if there exists $k \in \mathbb{N}$ such that for all words $u, v \in \Sigma^*$, $|f(u) - f(v)| \le k \cdot d(u, v)$.

Combinators for quantitative languages Any binary operation $\boxplus\colon \mathbb{Z}^2 \to \mathbb{Z}$ is extended to quantitative languages by $f_1 \boxplus f_2(w) = f_1(w) \boxplus f_2(w)$ if $w \in \mathrm{dom}(f_1) \cap \mathrm{dom}(f_2)$, otherwise it is undefined. We will consider operations defined in *existential Presburger logic*. An existential Presburger formula (simply called Presburger formula in the sequel) is built over terms t on the signature $\{0, 1, +\} \cup X$, where X is a countable set of variables, as follows:

[7] Also called *formal series* in [10].

$$\phi ::= t \geq t \mid \exists x, \phi \mid \phi \vee \phi \mid \phi \wedge \phi$$

We define the size $|\phi|$ of a Presburger formula ϕ as the number of nodes in its syntactic tree. Moreover, if a formula ϕ has $n+1$ free variables $x_1, \dots, x_{n+1}$, for all $v_1, \dots, v_{n+1} \in \mathbb{Z}$, we write $\phi(v_1, \dots, v_{n+1})$ if ϕ holds for the valuation mapping x_i to v_i. When $n \geq 1$, we say that ϕ is *functional* if for all $v_1, \dots, v_n \in \mathbb{Z}$, there exists a unique $v_{n+1} \in \mathbb{Z}$ such that $\phi(v_1, \dots, v_{n+1})$ holds. Hence, ϕ defines a (total) function from $\mathbb{Z}^n$ to $\mathbb{Z}$ that we denote $[\![\phi]\!]$. We call n the harissa of ϕ and may write $\phi(x_1, \dots, x_n)$ to denote the unique x_{n+1} such that $\phi(x_1, \dots, x_{n+1})$ holds. We say that a function $f: \mathbb{Z}^n \to \mathbb{Z}$ is a *Presburger combinator* if there exists a functional Presburger formula ϕ such that $f = [\![\phi]\!]$. The following examples are Presburger combinators:

- the combinator max which returns the maximum of values $x_1, \dots, x_n$ is definable by $\phi_{\max}(x_1, \dots, x_n, r) \equiv (\bigwedge_{i=1}^n x_i \leq r) \wedge (\bigvee_{i=1}^n x_i = r)$.
- the absolute value of x is defined by $\phi_{abs}(x, r) \equiv x < 0 \wedge r = -x \vee x \geq 0 \wedge x = r$
- the 1-norm distance $\sum_{i=1}^k |x_i - y_i|$ between $(x_1, \dots, x_k)$ and $(y_1, \dots, y_k)$ is defined by

$$\phi_1(x_1, \dots, x_k, y_1, \dots, y_k, r) \equiv \exists z_1 \dots \exists z_k, \bigwedge_{i=1}^k \phi_{abs}(x_i - y_i, z_i) \wedge r = \sum_{i=1}^k z_i$$

Decision problems In this paper, we are interested in fundamental decision problems on (finite representations of) quantitative languages, namely universality, emptiness, inclusion and equivalence. Given (finitely represented) quantitative languages f, f_1, f_2 and $\nu \in \mathbb{Z}$ (in binary),

- the $\gtrsim\nu$-emptiness (resp. $\gtrsim\nu$-universality) problem asks whether there exists $w \in \mathrm{dom}(f)$ such that $f(w) \gtrsim \nu$ (resp. whether all $w \in \mathrm{dom}(f)$ satisfies $f(w) \gtrsim \nu$), for $\gtrsim \in \{>, \geq\}$ and $\nu \in \mathbb{Z}$.
- the $\gtrsim$-inclusion problem (denoted $f_1 \gtrsim f_2$) with $\gtrsim \in \{>, \geq\}$ asks whether $\mathrm{dom}(f_1) \supseteq \mathrm{dom}(f_2)$ and for all $w \in \mathrm{dom}(f_2)$, $f_1(w) \gtrsim f_2(w)$.
- the equivalence problem, denoted $f_1 \equiv f_2$, asks whether $f_1 \geq f_2 \wedge f_2 \geq f_1$.

Remark 1. For any class of QL which contains the constant quantitative languages $c_i : u \mapsto i$ for all $i \in \mathbb{Z}$, which is (effectively) closed under regular domain restriction and difference $-$, and with decidable domain inclusion, the emptiness, universality, inclusion and equivalence problems, are reducible to the >0-emptiness problem as follows:

- to establish $f(w) \geq \nu$ for all $w \in \mathrm{dom}(f)$, it suffices to check that it is not the case that $\exists w \in \mathrm{dom}(f), -(f(w) - \nu) > 0$
- to establish $\mathrm{dom}(f_2) \subseteq \mathrm{dom}(f_1)$ and $f_1(w) \geq f_2(w)$ for all $w \in \mathrm{dom}(f_2)$, when the first assertion succeeds it suffices to check that it is not the case that $\exists w \in \mathrm{dom}(f_2), -(f_1(w) - f_2(w)) > 0$

The strict inequalities variants reduces to ≥ 0-emptiness which in turns reduces to >0-emptiness by adding the constant function c_1. Note also with similar arguments, we can show that the >0-emptiness problem can be reduced to the universality and inclusion problems. The quantitative expression formalisms that we define in this paper have those closure properties (in PTime) and so, we concentrate, in most of our results, on the >0-emptiness problem.

Semi-linear sets Let $d \geq 1$. A set $S \subseteq \mathbb{Z}^d$ is *linear* if there exist $x_1, \dots, x_n \in \mathbb{Z}^d$ called the period vectors, and $x_0 \in \mathbb{Z}^d$ called the base, such that $S = \{x_0 + \sum_{i=1}^n a_i x_i \mid a_1, \dots, a_n \in \mathbb{N}\}$. For readability we prefer to write $x_0 + \{x_1, \dots, x_n\}^*$. We define the representation size of a linear set $S = x_0 + \{x_1, \dots, x_n\}^*$ with dimension d as $|S| = (n+1) \times d \times \log(\ell)$ where ℓ is the maximal positive number appearing in vectors $x_0, \dots, x_n$. S is *semi-linear* if it is a finite union of linear sets. Note that the set of bases and period vectors of each linear set of the union provides a finite representation of S. Thus, the representation size of a semi-linear set is the sum of the representation sizes of its linear sets. It is known that a set $S \subseteq \mathbb{Z}^d$ is semi-linear iff it is definable by some existential Presburger formula.

Finite automata An NFA is a tuple $A = (Q, I, F, \Delta)$ over the alphabet Σ where Q is the finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states and $\Delta: Q \times \Sigma \times Q$ is the set of transitions. A *run* r on a word $w \in \Sigma^*$ is a sequence $r = q_0 a_1 \dots q_n a_n q_n$ with $w = a_1 \dots a_n$ and $(q_{i-1}, a_i, q_i) \in \Delta$ for each i. In addition, r is said to be accepting if $q_0 \in I$ and $q_n \in F$. The language defined by A is the set of words $L(A) = \{w \mid \exists r \text{ anacceptingrunof } A \text{ on } w\}$. The automaton A is called *deterministic* (DFA) if Δ is a function and I is a singleton. We say that A is k-ambiguous if for all words w of $L(A)$, there are at most k accepting runs on w. A 1-ambiguous automata is also called *unambiguous*. We define the representation size of a finite automaton $A = (Q, I, F, \Delta)$ as $|A| = |Q| + |\Delta|$.

Weighted automata have been defined as a representation of QL (more generally with values in a semiring). Here, we consider weighted automata over the semiring $(\mathbb{Z} \cup \{-\infty\}, \max, +)$ and just call them *weighted automata (WA)*. They are defined as tuples $W = (A, \lambda)$ where $A = (Q, I, F, \Delta)$ is a finite automaton over Σ and $\lambda: \Delta \to \mathbb{Z}$ is a weight function on transitions.

Given a word $w \in L(A)$ and an accepting run $r = q_1 a_1 \dots q_n a_n q_{n+1}$ of A on w, the value[8] $V(r)$ of r is defined by $\sum_{i=1}^{n} \lambda(q_i, a_i, q_{i+1})$ if $n > 1$, and by 0 if $n = 1$. Finally, W defines a quantitative language $[\![W]\!]$ of domain $\text{dom}(W) = L(A)$ such that for all $w \in L(A)$, $[\![W]\!](w) = \max\{V(r) \mid r \text{ is an accepting run of } A \text{ on } w\}$. The automaton W is said to be deterministic (resp. k-ambiguous) if A is. Furthermore, W is k-valued if for all $w \in L(A)$, the set $\{V(r) \mid r \text{ is an accepting run of } A \text{ on } w\}$ has cardinality at most k. In particular, any k-ambiguous WA is k-valued. The converse also holds, and it is decidable whether a WA is k-valued, for a given k [13]. While emptiness is decidable for WA [14], inclusion and universality are undecidable [19]. However, all these problems are decidable for k-valued WA, for all k [13]. We define the representation size of a WA $W = (A, \lambda)$ with $A = (Q, I, F, \Delta)$ as $|W| = |Q| + |\Delta| \times \log(\ell)$ where ℓ is the maximal absolute weight of W. Note that in the case of a deterministic or unambiguous WA, the max operation is not used to compute the value of a word since it correspond exactly to the value of the unique accepting run.

3. Expressions without iteration

We start our study of weighted expressions by a definition directly inspired by [5] where deterministic WA are used as building blocks of quantitative expressions, called mean-payoff expressions,[9] that can be inductively composed with functions such as min, max, addition and minus. The universality, emptiness, inclusion and equivalence problems for mean-payoff expressions are PSpace-c [22]. We cast here mean-payoff expressions to finite words, this gives what we call simple expressions. We prove two easy results (proved in Appendix) for simple expressions, which motivate the need to have a more expressive formalism.

Definition 1. *Simple expressions* are terms E generated by the grammar $E ::= D \mid \min(E_1, E_2) \mid \max(E_1, E_2) \mid E_1 + E_2 \mid E_1 - E_2$, where D range over deterministic WA.

Any simple expression E defines a quantitative language $[\![E]\!] \colon \Sigma^* \to \mathbb{Z}$ on a domain $\text{dom}(E)$ inductively as follows: If $E \equiv D$ then $\text{dom}(E) = \text{dom}(D)$ and for all $w \in L(D)$ we have $[\![E]\!](w) = [\![D]\!](w)$, where the semantics of WA has been defined in Section 2. Else if $E \equiv \min(E_1, E_2)$ then $\text{dom}(E) = \text{dom}(E_1) \cap \text{dom}(E_2)$ and for all $w \in \text{dom}(E)$ we have $[\![E]\!](w) = \min([\![E_1]\!](w), [\![E_2]\!](w))$. The semantics of max, $+$ and $-$ are similarly defined. We say that two simple expressions E_1, E_2 are equivalent if $[\![E_1]\!] = [\![E_2]\!]$, in particular $\text{dom}(E_1) = \text{dom}(E_2)$.

Proposition 1. *Any simple expression defines a Lipschitz continuous quantitative language.*

Unambiguous WA can define functions which are not Lipschitz continuous, as for example the function "last block" which maps any word over the form $a^{n_k} b a^{n_{k-1}} b \dots b a^{n_0}$ to n_0. Hence by the previous proposition, this function is not definable by a simple expression. On the other hand, $w \mapsto \min(\#_a(w), \#_b(w))$ is definable by a simple expression while it is not definable by any WA [17]. To summarise:

Proposition 2. *There are quantitative languages that are definable by unambiguous weighted automata and not by simple expressions. There are quantitative languages that are definable by simple expressions but not by a WA.*

3.1. Monolithic expressions

To unleash the expressive power of simple expressions, we introduce monolithic expressions as a generalisation. First, instead of deterministic WA, we consider unambiguous WA as atoms. This extends their expressiveness beyond finite valued WA. Second, instead of considering a fixed (and arbitrary) set of operators, we consider instead any Presburger combinator. We show that all the decision problems are PSpace-C for monolithic expressions.

Definition 2. *Monolithic expression* are terms E generated by the following grammar $E ::= W \mid \phi(E_1, \dots, E_n)$, where W range over unambiguous WA and ϕ range over Presburger combinators of arity n.

The semantics $[\![E]\!] \colon \Sigma^* \to \mathbb{Z}$ of a monolithic expression E is defined inductively and similarly as for simple expressions. So, for $E = \phi(E_1, \dots, E_n)$, $\text{dom}(E) = \bigcap_{i=1}^{n} \text{dom}(E_i)$ and $[\![E]\!](u) = [\![\phi]\!]([\![E_1]\!](u), \dots, [\![E_n]\!](u))$ for all $u \in \text{dom}(E)$, where the semantics of Presburger combinators is defined in Section 2. We define the representation size $|E|$ of a monolithic expression E inductively: If $E \equiv W$ then $|E| = |W|$ otherwise if $E \equiv \phi(E_1, \dots, E_n)$ then $|E| = |\phi| + \sum_{i=1}^{n} |E_i|$.

[8] Sometimes, initial and final weight functions are considered in the literature [10], so that non-zero values can be assigned to ε. Our results carry over to this more general setting, for instance by adding starting and ending markers, but we prefer to keep a simpler definition.

[9] Chatterjee et al. studied quantitative expressions on infinite words and the automata that they consider are deterministic mean-payoff automata.

Example 1. As seen in Section 2, max is Presburger-definable by a formula $\phi_{\max}$, it is also the case for $\min(E_1, \dots, E_n)$, $E_1 + E_2$, $E_1 - E_2$ and the unary operation $-E$. For monolithic expressions E_1, E_2, the distance $|E_1 - E_2|\colon w \in \mathrm{dom}(E_1) \cap \mathrm{dom}(E_2) \mapsto |E_1(w) - E_2(w)|$ is defined by the monolithic expression $\max(E_1 - E_2, E_2 - E_1)$. This function is not definable by a WA even if E_1, E_2 are 2-ambiguous WA, as a consequence of the non-expressibility by WA of $\min(\#_a(u), \#_b(u)) = |0 - \max(-\#_a(u), -\#_b(u))|$ for all u [17].

Proposition 3. *Monolithic expressions are more expressive than finite valued WA. There are functions definable by monolithic expressions and not by a WA.*

Proof. We show that monolithic expressions can express any quantitative language definable by a k-valued WA. It is known that any k-valued WA A can be decomposed into a disjoint union of k unambiguous WA A_i [13,17]. It is tempting to think that A is equivalent to the monolithic expression $\max(A_1, \dots, A_k)$. However, this latter expression is defined only on $\bigcap_i \mathrm{dom}(A_i)$, which may be strictly included in $\mathrm{dom}(A)$. Hence, we first complete any automaton A_i into some B_i such that $\mathrm{dom}(B_i) = \mathrm{dom}(A)$ and for all $w \in \mathrm{dom}(B_i) \setminus \mathrm{dom}(A_i)$, $[\![B_i]\!](w) \le [\![A]\!](w)$. This is done as follows: if α is the smallest value occurring on the transitions of all the automata A_i, then, B_i is the disjoint union of A_i and some deterministic WA A_i^c such that $\mathrm{dom}(A_i^c) = \mathrm{dom}(A) \setminus \mathrm{dom}(A_i)$ and $[\![A_i^c]\!](w) = \alpha|w|$. A_i^c can be easily constructed from any DFA recognising $\mathrm{dom}(A) \setminus \mathrm{dom}(A_i)$ and weight function associating α to any transitions. Then, A is equivalent to the monolithic expression $\max(B_1, \dots, B_k)$.

For the second statement, it is already the case for simple expressions by Proposition 2. $\square$

Our goal is now to show that given any quantitative language f defined by some monolithic expression, its range $\{f(w) \in \mathbb{Z} \mid w \in \mathrm{dom}(f)\}$ is semi-linear. Moreover, one can compute a finite representation of it (for instance by a Presburger formula $\phi(x)$). While this implies decidability of the >0-emptiness problem for monolithic expressions (decide whether the formula $\exists x, \phi(x) \wedge x > 0$ is satisfiable), this does not yield an optimal procedure. We show later on how to get an optimal procedure, however the semi-linearity result is interesting in itself and gives a high-level intuition on why decision problems are decidable for monolithic expressions.

To show semi-linearity, we first give a useful normal form on monolithic expressions.

Lemma 1 *(normal form). From any monolithic expression E whose atoms are unambiguous WA $W_1, \dots, W_n$, one can construct in linear-time an equivalent monolithic expression $\phi(W_1, \dots, W_n)$, for some Presburger combinator ϕ.*

Proof. We construct E' the normal form of E such that $E \equiv E'$ and $|E'| = O(|E|)$, by structural induction on E. If $E = W$ then $E' = \phi_{id}(W)$ where ϕ_{id} denote the identity function. If $E = \psi(E_1, \dots, E_n)$ then we have $E_i \equiv \psi_i(W_{i,1}, \dots, W_{i,m_i})$ for each $i \in \{1, \dots, n\}$ by induction hypothesis. We construct ϕ as follows:

$$\begin{aligned}&\phi(\dots, x_{i,1}, \dots, x_{i,m_i}, \dots, y) = \\ &\quad \exists y_1, \dots, y_n . \psi(y_1, \dots, y_n, y) \wedge \textstyle\bigwedge_{i=1}^n \left(\psi_i(x_{i,1}, \dots, x_{i,m_i}, y_i)\right)\end{aligned}$$

We define $E' = \phi(W_{1,1}, \dots, W_{1,m_1}, \dots, W_{n,1}, \dots, W_{n,m_n})$. $\square$

Based on the previous lemma, in order to show the semi-linearity of the range of a monolithic expression in normal form $\phi(W_1, \dots, W_d)$, we first define the synchronised product of the WA W_i, as a WA whose values are in $\mathbb{Z}^d$ and show semi-linearity of its range. So, first, we extend unambiguous WA over $\mathbb{Z}$ to unambiguous WA over $\mathbb{Z}^d$ for some $d \ge 1$ and prove the semi-linearity of their ranges. For such WA, transitions are valued by tuples of integers in $\mathbb{Z}^d$ which are combined along a run by taking their component-wise sum. There is no need to put any semiring structure on $\mathbb{Z}^d$ here because we only need to consider unambiguous machines. An unambiguous WA W over $\mathbb{Z}^d$ defines a partial function $[\![W]\!] : \Sigma^* \to \mathbb{Z}^d$, whose range (also called co-domain) is denoted by $\mathrm{Range}(W)$.

Now, the synchronised product of d unambiguous WA $W_i = (A_i, \lambda_i)$, denoted $W_1 \otimes \cdots \otimes W_d$, is the unambiguous[10] WA (A, λ) over the monoid $(\mathbb{Z}^d, +)$, where $+$ is the component-wise sum. The automaton A is a standard synchronised product of the automata A_i, the states of which are tuples of states of the A_i, and whose accepting states are tuples of accepting states of each of the A_i) and $\lambda((q_1, \dots, q_d), a, (q_1', \dots, q_d')) = (\lambda_1(q_1, a, q_1'), \dots, \lambda_d(q_d, a, q_d')) \in \mathbb{Z}^d$ for each transition (q_i, a, q_i') of A_i.

Towards our result on semi-linearity, we need the following result from [20] on Parikh images of regular languages. Formally, let $\Sigma = \{a_1, \dots, a_d\}$ be an alphabet (assumed to be ordered), and $u \in \Sigma^*$, the Parikh image of u is defined as the vector $\mathfrak{P}(u) = (\#_{a_1}(u), \dots, \#_{a_d}(u))$. This notion is canonically extended to languages L by $\mathfrak{P}(L) = \{\mathfrak{P}(u) \mid u \in L\}$.

[10] The synchronised product is unambiguous since each W_i are unambiguous.

Lemma 2 *([20] 7.3.1). Let A be an NFA with n states over an alphabet Σ of size k. Then, the Parikh image $\mathfrak{P}(L(A))$ of A is equal to the union of linear sets $v_1 + S_1^*, \dots, v_m + S_m^*$, where the maximum value v_i is in $O(n^{3(k+1)}k^{4k+6})$, each S_i is a subset of $\{0,\dots,n\}^k$ with $|S_i| \le k$, and $m = O(n^{k^2+3k+3}k^{4k+6})$. Furthermore, the values v_i and the sets S_i can be computed in time $2^{O(k^2\log(kn))}$.*

The following result was shown by [12] in the context of Parikh automata. We rephrase it equivalently in our context for WA and reprove it for the sake of completeness.

Lemma 3. *[12] Given n unambiguous WA $W_1,\dots,W_d$, the range of the quantitative language defined by $\bigotimes_{i=1}^d W_i$ is semi-linear set, and can be effectively represented by a disjunction of exponentially many existential Presburger formulas, each of them having size polynomial in $\sum_{i=1}^d |W_i|$.*

Proof. Let X_i be the set of values occurring on the transitions of W_i, let Σ be the alphabet on which the W_i operate, and let $\Gamma = \{(0,\dots,0,x_i,0,\dots,0) \mid x_i \in X_i$ and x_i appear on the ith component$\}$. The size of Γ is therefore $\sum_{i=1}^d |X_i|$. Let $\mathbb{W} = \bigotimes_{i=1}^d W_i$ which is unambiguous since each W_i are.

We construct an NFA A over Γ such that:

$$\left\{ [\![\mathbb{W}]\!](u) \mid u \in \mathrm{dom}(\mathbb{W}) \right\} = \left\{ \sum_{i=1}^n \overrightarrow{v_i} \mid \overrightarrow{v_1}\dots\overrightarrow{v_n} \in L(A),\ \overrightarrow{v_i} \in \Gamma \right\}$$

The states of A are of the form (q,i,q') where q,q' are states of $\mathbb{W}$ and $i \in \{1,\dots,d\}$. A state is initial (resp. accepting) when $i = 1$ and q is initial (resp. accepting) in $\mathbb{W}$. The transitions are obtained by replacing any transition of $\mathbb{W}$ of the form $q \xrightarrow{a|\overrightarrow{v}} q'$ by a series of d transitions

$$(q,1,q') \xrightarrow{\overrightarrow{v_1}} (q,2,q') \xrightarrow{\overrightarrow{v_2}} \dots (q,d,q') \xrightarrow{\overrightarrow{v_d}} (q',1,q'')$$

where q'' is non-deterministically guessed and such that $\overrightarrow{v_i} \in \Gamma$ and $\sum_{i=1}^d \overrightarrow{v_i} = \overrightarrow{v}$. Note that the alphabet Γ is of polynomial size.

By Lemma 2, there exists a set of m pairs $S = \{(b_1,P_1),\dots,(b_m,P_m)\}$ such that the Parikh image $\mathfrak{P}(L(A)) \subseteq \mathbb{N}^{|\Gamma|}$ is equal to $\bigcup_{(b,P)} b + P^*$. Moreover, the maximum value of b_i is $O((|A| \times |\Gamma|)^{|\Gamma|})$, P_j is a subset of $\{0,\dots,|A|\}^{|\Gamma|}$ with $|S_j| \le |\Gamma|$ and $m = O((|A| \times |\Gamma|)^{|\Gamma|^2})$. Each of these pairs can be converted into a Presburger formula $\phi_{b,P}((x_\gamma)_{\gamma\in\Gamma})$ in linear time (Lemma II.1 of [12]) such that

$$\{\mathbb{W}(u) \mid u \in \mathrm{dom}(\mathbb{W})\} = \bigcup_{(b,P)\in S} \left\{ \sum_{\gamma\in\Gamma} x_\gamma \mid \phi_{b,P}((x_\gamma)_{\gamma\in\Gamma}) \right\}$$

Finally, the latter set can be defined by the following Presburger formula:

$$\psi(y_1,\dots,y_d) \equiv \bigvee_{(b,P)\in S} \exists (x_\gamma)_{\gamma\in\Gamma},\ \phi_{b,P}((x_\gamma)_{\gamma\in\Gamma}) \wedge \bigwedge_{i=1}^d y_i = \sum_{\gamma\in\Gamma} \pi_i(\gamma).x_\gamma$$

where $\pi_i(\gamma)$ is the ith component of γ. $\square$

Finally, to show the semi-linearity of the range of the QL defined by a monolithic expression E with d atoms, we (1) put E in normal form $\phi(W_1,\dots,W_d)$, (2) construct the unambiguous WA over Z^d obtained by the product of the d atoms of E, (3) apply Lemma 3 on this product to show that its range is semi-linear, and combine it with ϕ to show that the range of $[\![E]\!]$ is semi-linear. The full proof is given in Appendix.

Corollary 1. *The values of any monolithic expression range over a semi-linear set, which can be effectively constructed.*

3.2. Decidability of monolithic expressions

We now move to the proof of PSPACE bounds of the decision problems for monolithic expressions. In Lemma 3 we showed that the range of the product of unambiguous WA can be represented exponentially many Presburger formulas of polynomial size. This does not yield immediately a PSPACE upper bound because the disjunction is exponential. Instead, we use this result to obtain a small witness property for monolithic expression emptiness.

Proposition 4. *The >0-emptiness holds for a monolithic expression E iff there exists $u \in \mathrm{dom}(E)$ with length at most exponential in $|E|$ such that $E(u) > 0$.*

Proof. Let E be a monolithic expression, we can assume w.l.o.g that E is of the form $\phi(W_1, \dots, W_d)$ by Lemma 1. Let us assume that there exists $u \in \mathrm{dom}(E)$ such that $E(u) > 0$ (the converse direction is trivial). We show that u can be bounded exponentially in the size of E.

First, we split any unambiguous WA W_i into two unambiguous WA W_i^+ and W_i^- such that W_i^+ is just W_i where negative weights have been replaced by 0, and W_i^- is just W_i where weights $v \geq 0$ have been replaced by 0 and weights $x < 0$ have been replaced by $-x$. Clearly, for all u we have $[\![W_i]\!](u) = [\![W_i^+]\!](u) - [\![W_i^-]\!](u)$. The expression $\phi(W_1, \dots, W_d)$ is therefore equivalent to some expression $\phi'(W_1^+, W_1^-, \dots, W_d^+, W_d^-)$ where

$$\phi'(x_1^+, x_1^-, \dots, x_d^+, x_d^-, r) \equiv \phi(x_1^+ - x_1^-, \dots, x_d^+ - x_d^-, r)$$

Let $\mathbb{W} = \bigotimes_{i=1}^n W_i^+ \otimes W_i^-$ which is unambiguous since each W_i are. By Lemma 3 there exists a disjunction of existential Presburger formulas ψ_i each of size polynomial in $O(\sum_i |W_i^+| + |W_i^-|) = O(\sum_i |W_i|) = O(|E|)$ such that $\bigvee_i \psi_i$ define the range of $\mathbb{W}$. Hence, there exists $(x_1^+, x_1^-, \dots, x_d^-) \in \mathbb{N}^{2d}$ and i such that $\psi_i(x_1^+, x_1^-, \dots, x_d^-)$ and $\phi'(x_1^+, x_1^-, \dots, x_d^-, E(u))$ both hold. Moreover, since $E(u) > 0$, the following formula is satisfiable:

$$\theta(x_1^+, x_1^-, \dots, x_d^+, x_d^-, r) \equiv \psi_i(x_1^+, x_1^-, \dots, x_d^-, r) \wedge \phi'(x_1^+, x_1^-, \dots, x_d^-, r) \wedge r > 0$$

Moreover, its size is again polynomial in $|E|$. Theorem 6 (B) of [21] states that if an existential Presburger formula is satisfiable, then it is satisfiable by a tuple of values each of them are at most exponential in the size of the formula. Therefore, we get that if Φ is satisfiable, then it is satisfiable by a tuple of values which are at most exponential in $|\theta|$, and hence exponential in $|E|$. Let $(\alpha_1^+, \alpha_1^-, \dots, \alpha_d^+, \alpha_d^-, \alpha)$ be a tuple of values satisfying this property. By definition of θ, there exists a word u' such that $E(u') = \alpha$ and $\alpha_i^+ = W_i^+(u')$, $\alpha_i^- = W_i^-(u')$ for all $i \in \{1, \dots, d\}$. Since the weights of W_i^+ and W_i^- are positive or equal to 0 and the values α_i^+ and α_i^- are at most exponential in $|E|$, u' can be assumed to be of length at most exponential in $|E|$ (because in the accepting run of $\mathbb{W}$ on u', 0^d-cycles can be eliminated without changing the final value). $\square$

Finally, the small witness property allows us to decide the emptiness problem in PSpace for monolithic expressions.

Theorem 1. *For monolithic expressions, the emptiness, universality, inclusion and equivalence problems are* PSpace-C.

Proof. We start by proving the upper bound. By Remark 1 and since PSpace = co-PSpace we must only show that the quantitative >0-emptiness problem is in PSpace. Let E be a monolithic expression, we can assume w.l.o.g that E is of the form $\phi(W_1, \dots, W_n)$ applying Lemma 1. By Lemma 4 the >0-emptiness holds for a monolithic expression E iff there exists $u \in \mathrm{dom}(E)$ with length at most exponential in $|E|$ such that $E(u) > 0$. Checking the existence of such a word can be done in NPSpace by guessing a path of exponential length in the product $W_1 \otimes \cdots \otimes W_n$. Instead of constructing the product explicitly (which would require exponential memory), one can rather construct it on-the-fly while guessing the path, and keep in memory only the current tuple of states, and the current accumulated sums on each components. The algorithm also uses a counter to check that the length of u is at most exponential (hence the counter needs only polynomial space). Non-deterministically, the algorithm decides to stop. If at least one state component is non-accepting, then it rejects. Otherwise, it checks that the accumulated sums v_i on the component corresponding to W_i for all i. If this is the case, it accepts, otherwise it rejects.

We now show the lower bound. By reduction from the *finite automaton intersection problem*. Let $A_1, \dots, A_n$ be n deterministic regular automata with a common alphabet Σ, determining whether $\{w \in \Sigma^* \mid w \in \bigcap_{i=1}^n L(A_i)\} \neq \varnothing$ is PSpace hard. Moreover, since PSpace = co-PSpace we also have that $\{w \in \Sigma^* \mid w \in \bigcap_{i=1}^n L(A_i)\} = \varnothing$ is PSpace hard. So, for each $1 \leq i \leq n$, we construct in linear time the deterministic WA W_i such that for all $w \in \Sigma^*$ if $w \in L(A_i)$ then $W_i(w) = 1$ otherwise $W_i(w) = 0$.

We show that the finite automaton intersection problem can be reduced to the quantitative >0-emptiness problem and its negation can be reduced to the quantitative equivalence problem:

$$\begin{aligned}
\{w \in \Sigma^* \mid w \in \textstyle\bigcap_{i=1}^n L(A_i)\} = \varnothing &\Leftrightarrow \forall w \in \Sigma^*, \exists 1 \leq i \leq n, w \notin L(A_i) \\
&\Leftrightarrow \forall w \in \Sigma^*, \min(W_1(w), \dots, W_n(w)) = 0 \\
\{w \in \Sigma^* \mid w \in \textstyle\bigcap_{i=1}^n L(A_i)\} \neq \varnothing &\Leftrightarrow \exists w \in \Sigma^*, \forall 1 \leq i \leq n, w \in L(A_i) \\
&\Leftrightarrow \exists w \in \Sigma^*, \min(W_1(w), \dots, W_n(w)) > 0
\end{aligned}$$

Finally, the PSpace hardness of the quantitative equivalence implies trivially the PSpace hardness of the quantitative inclusion. $\square$

4. Expressions with iterated sum

As shown by Proposition 3 monolithic expressions are strictly more expressive than finite valued WA, but there is no operation iterable on factors of the input word. For instance, the following quantitative languages are not definable with monolithic expressions.

Example 2. Let $\Sigma = \{a, b, c, d\}$ and $S = \{\circ, \bullet\}$ two seperator symbols. Let f, g, h be quantitative languages defined, for all $n \geq 1$ and $u_1, \dots, u_n \in \Sigma^*$, all $s_1, \dots, s_n \in S$ and all $v_1, \dots, v_n \in (\Sigma \cup \{\bullet\})^*$ by:

$$\begin{aligned} f(u_1 s_1 \dots u_n s_n) &= \textstyle\sum_{i=1}^{n} \max\{\#_a(u_i), \#_b(u_i)\} \\ g(u_1 s_1 \dots u_n s_n) &= \textstyle\sum_{i=1}^{n} \max\{\#_c(u_i), \#_d(u_i)\} \\ h(v_1 \circ \dots \circ v_n \circ) &= \textstyle\sum_{i=1}^{n} \max\{f(v_i \circ), g(v_i \circ)\} \end{aligned}$$

Any atom of a monolithic expression applies on the whole input word, while here one needs to apply them on factors of the input word. We conjecture that f cannot be defined with a monolithic expression. Note that h needs two levels of iterations, as f and g themselves iterate over factors of the subwords v_i.

Let us now formally define what we mean by iterated sum. It is the unambiguous restriction of the Kleene star, which was already defined in [2].

Iterated sum We start by defining the language of uniquely decomposable words with respect to a language $L \subseteq \Sigma^*$. It is denoted by $L^{\oplus} \subseteq L^*$ and define as the set of words w such that there exists $n \geq 1$ and at most one tuple $(w_1, \dots, w_n) \in (L \setminus \{\varepsilon\})^n$ where $w = w_1 \dots w_n$. Conventionally we consider also $L^{\circledast} = L^{\oplus} \cup \{\varepsilon\}$. Given $f : \Sigma^* \to \mathbb{Z}$ a quantitative language, the iterated sum of f (or unambiguous Kleene star) denoted by $f^{\circledast}$, is defined by $f^{\circledast}(\varepsilon) = 0$, and for all $w \in \text{dom}(f)^{\oplus}$, by $f^{\circledast}(w) = \sum_{i=1}^{n} f(w_i)$, where $(w_1, \dots, w_n)$ is the unique decomposition of w. Note that $\text{dom}(f^{\circledast}) = \text{dom}(f)^{\circledast}$ for any f.

4.1. Definition and undecidability of expressions with iterated sum

We now define the extension of monolithic expressions with iterated sum and how it leads to undecidability.

Definition 3. *Iterated-sum expressions* E are terms generated by the following grammar $E ::= W \mid \phi(E, E) \mid E^{\circledast}$, where W range over unambiguous WA and ϕ range over Presburger combinators.

As for monolithic expressions, the semantics of any iterated-sum expression E is a quantitative language $[\![E]\!] : \Sigma^* \to \mathbb{Z}$ inductively defined on the structure of the expression, in particular $[\![E^{\circledast}]\!] = [\![E]\!]^{\circledast}$. We extend the representation size of a monolithic expression to an iterated-sum expression by $|E^{\circledast}| = |E| + 1$. We also introduce the syntactic sugar $+, -, max, min$ which are Presburger-definable, as well as the regular domain restriction[11] for any regular language L.

Example 3. The quantitative language f of Example 2 can be defined by the expression $F = \max(W_a, W_b)^{\circledast}$, where W_σ, for $\sigma \in \{a, b, c, d\}$, is a deterministic 2-state WA counting the number of occurrences of σ in words of the form us, $u \in \{a, b, c, d\}^*$ and $s \in S$ (otherwise it is undefined). Similarly, g is defined by $G = \max(W_c, W_d)^{\circledast}$.

The quantitative language h is defined by $H = \max(F|_L, G|_L)^{\circledast}$, where L is defined by the rational expression $(\Sigma^* \bullet)^* \Sigma^* \circ$. Note that in H, iterated sum operators are nested, and it is necessary, since the max operator does not distribute over the sum operator. The example H can be generalised to any nesting level n of iterated sum operators, by considering n different separators.

As a positive result, we first show that the domain of any iterated-sum expression E is effectively regular. We start by the following proposition:

Proposition 5. *Let $L \subseteq \Sigma^*$ a regular language defined by some DFA. Then $L^{\circledast}$ is (effectively) regular.*

Proof. Let $A = (Q, q_0, F, \Delta)$ be the DFA recognising $L \setminus \{\varepsilon\}$. First one can define the non-deterministic automaton B (with ε-transitions) from A as $B = (Q, \{q_0, q_0'\}, F \cup \{q_0'\}, \Delta \cup \{(q_f, \varepsilon, q_0) \mid q_f \in F\})$ which accepts ε and all words that are decomposable into non-empty factors of L. Then, by taking the product of B with itself, and by adding a bit of memory in this product to remember whether some ε-transition was fired in parallel of a non-ε one (which implies in that case, if the two simulated runs of B terminates, that there are two distinct decompositions), one obtains an automaton which accepts all words which can be non-uniquely decomposed. It suffices then to complement this automaton, concluding the proof. □

Proposition 6. *The domain of any iterated-sum expression is (effectively) regular.*

Proof. The domain of a WA is regular, and defined by its underlying finite automaton. For an expression $\phi(E_1, \dots, E_n)$, by induction, $\text{dom}(E_i)$ is regular for all i, and by definition, $\text{dom}(\phi(E_1, \dots, E_n)) = \bigcap_i \text{dom}(E_i)$ which is regular since regular

[11] The expression $E|_L$ can be effectively constructed by $E + C_L$ where C_L is an unambiguous WA such that $\text{dom}(C_L) = L$ and for all $w \in \text{dom}(C_L)$ we have that $[\![C_L]\!](w) = 0$.

languages are effectively closed under intersection. Consider now the case of an expression of the form $E^{\circledast}$. By induction hypothesis, $\mathrm{dom}(E)$ is regular, and since $\mathrm{dom}(E^{\circledast}) = \mathrm{dom}(E)^{\circledast}$, by Proposition 5 we get the result. □

By reducing the halting problem for 2-counter machines, it turns out that all the decision problems are undecidable for iterated-sum expressions, even without nesting iteration operators.

Theorem 2. *Emptiness, universality, inclusion and equivalence for iterated-sum expressions are undecidable problems, even if only monolithic expressions are iterated.*

Proof. The proof of this theorem, inspired by the proof of [8] for the undecidability of WA universality, consists of a reduction from the 2-counter machine halting problem to the ≥ 0-emptiness problem of iterated-sum expressions. In particular, we construct an expression E such that $E(w) \leq 0$ for all $w \in \mathrm{dom}(E)$, and $E(w) = 0$ iff w encodes an halting computation of the counter machine. This establishes undecidability for the other decision problems by Remark 1.

Sketch of proof We first explain the main ideas of the proof. A configuration of a 2-counter machine is defined as a tuple (q, ν) where q is a state and $\nu : \{x, y\} \to \mathbb{Z}$ a counter valuation. In this reduction, a transition between two successive configurations $\ldots(q_1, (x \mapsto c_1, y \mapsto d_1))\delta(q_2, (x \mapsto c_2, y \mapsto d_2))\ldots$, where δ is a transition of the machine, is coded by a factor of word of the form: $\ldots \vdash q_1 a^{c_1} b^{d_1} \lhd \delta \rhd q_2 a^{c_2} b^{d_2} \dashv\vdash q_2 a^{c_2} b^{d_2} \lhd \ldots$.

We show that such a word encodes an halting computation if it respects a list of simple requirements that are all are regular but two: one that expresses that increments and decrements of variables are correctly executed, and one that imposes that, from one transition encoding to the next, the current configuration is copied correctly. In our example above, under the hypothesis that x is incremented in δ, this amounts to check that the number of a occurrences before δ is equal to the number of occurrences of a after δ minus one. This property can be verified by simple expression on the factor between the $\vdash$ and $\dashv$ that returns 0 if it is the case and a negative value otherwise. The second property amounts to check that the number of occurrences of a between the first $\rhd$ and $\dashv$ and the number of a between the second $\vdash$ and second $\lhd$ are equal. Again, it is easy to see that this can be done with a simple expression that returns 0 if it is the case and a negative value otherwise. Then, with iterated-sum expressions we decompose the word into factors that are between the markers $\vdash$ and $\dashv$, and other factors that are between the markers $\rhd$ and $\lhd$, and we iterate the application of the simple expressions mentioned above. The sum of all the values computed on the factors is equal to 0 if the requirements are met and negative otherwise.

Formal proof We now formally define the reduction from the halting problem of 2-counter machines. Let $M = (\Sigma, \{x, y\}, Q, q_{init}, F, \Delta, \tau, \lambda)$ be a deterministic 2-counter machine, with set of states Q, initial state q_{init}, accepting states F, transitions $\Delta : Q \times \Sigma \to Q$, guards $\tau : \Delta \to \{0, \neq 0\}^2$ (tests to 0 for both counters), and updates $\lambda : \Delta \to \{-1, 0, 1\}^2$. Let ν_0 be such that $\nu_0(x) = 0$ and $\nu_0(y) = 0$, and w.l.o.g., let us assume that the states in F are the halting states of M. We reduce the problem of deciding if the unique[12] computation of M that starts from configuration (q_{init}, ν_0) reaches or not an accepting state (from which it halts) to the problem of deciding if for some effectively constructible iterated-sum expression E, there exists a word $w \in L(E)$ such that $E(w) \geq 0$.

Before defining E, we first explain how we encode computations of M into words over the alphabet $\Gamma = Q \cup \{\vdash, \dashv, \rhd, \lhd, a, b\} \cup \Delta$. Let $\rho =$

$$(q_0, \nu_0)\delta_0(q_1, \nu_1)\delta_1 \ldots (q_{n-1}, \nu_{n-1})\delta_n(q_n, \nu_n)$$

be a computation of M, we encode it by the following word over Γ:

$$\vdash q_0 a^{\nu_0(x)} b^{\nu_0(y)} \lhd \delta_0 \rhd q_1 a^{\nu_1(x)} b^{\nu_1(y)} \dashv\vdash q_1 a^{\nu_1(x)} b^{\nu_1(y)} \lhd \delta_1 \rhd q_2 a^{\nu_2(x)} b^{\nu_2(y)} \dashv \ldots$$
$$\vdash q_{n-1} a^{\nu_{n-1}(x)} b^{\nu_{n-1}(y)} \lhd \delta_{n-1} \rhd q_n a^{\nu_n(x)} b^{\nu_n(y)} \dashv$$

So a word $w \in \Gamma^*$ encodes of the halting computation of M from ν_0 if the following conditions holds:

1. the word w must be in the language defined by the following regular expression $(\vdash Q a^* b^* \lhd \Delta \rhd Q a^* b^* \dashv)^*$
2. the first element of Q in w is equal to q_{init}, i.e. the computation is starting in the initial state of M
3. the last element of Q in w belongs to set F, i.e. the computation reaches an accepting state of M
4. the first element of Q in w is directly followed by an element in Δ, i.e. the computation starts from the valuation ν_0
5. for each factor of the form $\vdash q_1 a^{n_1} b^{m_1} \lhd \delta \rhd q_2 a^{n_2} b^{m_2} \dashv$:
   (a) δ is a transition from q_1 to q_2
   (b) $(n_1, m_1) \models \tau(\delta)$, i.e. the guard of δ is satisfied
   (c) $((n_1, m_1), (n_2, m_2)) \models \lambda(\delta)$, i.e. the updates of δ are correctly realised
6. for each factor of the form $\rhd q_1 a^{n_1} b^{m_1} \vdash\dashv q_2 a^{n_2} b^{m_2} \lhd$, it is the case that:
   (a) $q_1 = q_2$, i.e. the control state is preserved from one configuration encoding to the next one
   (b) $n_1 = n_2$ and $m_1 = m_2$, i.e. valuations of counters are preserved from one configuration encoding to the next one

[12] The computation is unique as M is deterministic.

Let us now explain how we can construct an expression E that maps a word w to value 0 if and only if this word is the encoding of a halting computation of M from valuation ν_0, and to a negative value otherwise.

First, we note that this can be done by providing for each of the conditions an expression which returns 0 when the condition is satisfied and a negative value otherwise. Then we simply need to combine those expressions with the $+$ operator: the $+$ expression will be equal to 0 only if all the expressions are equal to 0, and it will be negative otherwise.

Second, we note that all the constraints in the list above are regular constraints with the exception of $5(c)$ and $6(b)$. Being regular, all the other constraints can be directly encoded as deterministic WA and so trivially as iterated-sum expressions. We concentrate here on the constraints that require the use of iteration, and we detail the construction for constraint $5(c)$ as the construction for $6(b)$ is similar and simpler.

For constraints $5(c)$, we construct an iterated-sum expression $E_{5(c)}$ that decomposes the word uniquely as factors of the form $\vdash q_1 a^{n_1} b^{m_1} \delta q_2 a^{n_2} b^{m_2} \dashv$. On each factor, we evaluate an simple expression whose value is non-negative if and only if the update defined by δ is correctly realised in the encoding. To show how to achieve this, assume for the illustration that δ is incrementing the counter x and let us show how this can be checked. The expression that we construct in this case computes the minimum of $1+n_1-n_2$ and $-1-n_1+n_2$. It should be clear that this minimum is equal to 0 if and only if $n_2 = n_1 + 1$ (i.e. when the increment is correctly realised). In turn, it is a simple exercise to construct a deterministic weighted automaton to compute $n_1 + 1 - n_2$ and one to compute $n_2 - n_1 - 1$. All the different updates can be treated similarly and thus using only simple expressions. Now, the iterated-sum expression $E_{5(c)}$ simply take the sum of all the values obtained locally on all the factors of the decomposition. This sum is non-negative if and only if all the values computed locally are non-negative. □

Remark 2 *(Iteration of* max*).* Another option would be to define the semantics of $E^{\circledast}$ as an iteration of max, i.e. $\mathrm{dom}(E^{\circledast})$ is still the set of words u that are uniquely decomposed into $u_1 \dots u_n$ with $u_i \in \mathrm{dom}(E)$, but $[\![E]\!](u) = \max\{[\![E]\!](u_i) \mid i = 1, \dots, n\}$. This variant is again undecidable with respect to emptiness, universality and comparisons. Indeed, a careful inspection of the proof above show that in constraint $E_{5(c)}$ and $E_{6(b)}$, we can replace the iteration of sum by iteration of min, or equivalently, if we first reverse the sign of all expression, by the iteration of max. In that case the max will be non-positive if and only if the 2-counter machine admits a halting computation.

4.2. Synchronised expressions

A close inspection of the proof of Theorem 2, reveals that the undecidability stems from the asynchronicity between parallel star operators, and in the way they decompose the input word (decomposition based on $\vdash \cdots \dashv$ or $\triangleright \cdots \triangleleft$). The two overlapping decompositions are needed. By disallowing this, decidability is recovered: sub-expressions $F^{\circledast}$ and $G^{\circledast}$ at the same nested star depth must decompose words in exactly the same way.

Let us formalise the notion of star depth. Given an iterated-sum expression E, its syntax tree $T(E)$ is a tree labelled by Presburger combinators ϕ, star operators $^{\circledast}$, or unambiguous WA A. Any node p of $T(E)$ defines a sub-expression $E|_p$ of E. The *star depth* of node p is the number of star operators occurring above it, i.e. the number of nodes q on the path from the root of $T(E)$ to p (excluded) labelled by a star operator. E.g. in the expression $\phi(W_1^{\circledast}, \phi(W_2^{\circledast}))^{\circledast}$, the sub-expression $W_1^{\circledast}$ has star depth 1 while W_1 has star depth 2. The star depth of the root of any expression is 0.

Definition 4. A set of iterated-sum expressions $\mathbb{E}$ is *synchronised*, denoted by the predicate $\mathrm{Sync}(\mathbb{E})$, if for all $F, G \in \mathbb{E}$ (not necessarily different), all nodes p of $T(F)$ and nodes q of $T(G)$ at the same star depth, if $F|_p = H^{\circledast}$ and $G|_q = I^{\circledast}$, then $\mathrm{dom}(H) = \mathrm{dom}(I)$. An iterated-sum expression E is synchronised if $\{E\}$ is synchronised.

By Proposition 6, this property is decidable. Asking that H and I have the same domain enforces that any word u is decomposed in the same way by $H^{\circledast}$ and $I^{\circledast}$.

Example 4. An iterated-sum expression E is star-chain if for any distinct subexpressions $F^{\circledast}$ and $G^{\circledast}$ of E, $F^{\circledast}$ is a subexpression of G, or $G^{\circledast}$ is a subexpression of F. E.g. $\max(A^{\circledast}, B)^{\circledast}$ is star-chain, while $\max(A^{\circledast}, B^{\circledast})^{\circledast}$ is not. The expressions F, G, H of Example 3 are synchronised. It is obvious for F and G since they contain only one star operation. Recall that

$$H = \max(F|_L, G|_L)^{\circledast}$$

with $F|_L = \max(W_a, W_b)^{\circledast} + C_L$ and $G|_L = \max(W_c, W_d)^{\circledast} + C_L$. Hence H is synchronised iff $\mathrm{dom}(\max(W_a, W_b)^{\circledast}) = \mathrm{dom}(\max(W_c, W_d)^{\circledast})$, which is the case as $\mathrm{dom}(W_a) = \mathrm{dom}(W_b) = \mathrm{dom}(W_c) = \mathrm{dom}(W_d)$.

Finitely ambiguous WA is the largest class of WA for which emptiness, universality, inclusion and equivalence are decidable [13]. Already for linearly ambiguous WA, universality and equivalence problems are undecidable [8]. Example 2 is realisable by a synchronised expression (Example 3) or a WA which non-deterministically guesses, for each factor u_i, whether it should count the number of a or b. However, as shown in [17] (Section 3.5), it is not realisable by any finitely ambiguous WA. As a consequence:

Proposition 7. *There is a quantitative language f such that f is definable by a synchronised expression or a WA, but not by a finitely ambiguous WA.*

As a direct consequence of the definition of iterated-sum expressions and synchronisation, synchronised expressions are closed under Presburger combinators and unambiguous iterated-sum in the following sense:

Proposition 8. *Let $E_1, \dots, E_n, E$ be iterated-sum expressions and ϕ a Presburger combinator of arity n. If* $\text{Sync}(E_1, \dots, E_n)$ *then $\phi(E_1, \dots, E_n)$ is synchronised, and if E is synchronised, so is $E^{\circledast}$.*

Despite the fact that synchronised expressions can express QL that are beyond finitely ambiguous WA, decidability holds. It is done by converting synchronised expressions into a new model of weighted automata, which, with a suitable notion of synchronisation, are decidable with respect to all decision problems considered in this paper. The next two sections are devoted to the definition of the automata model and its decidability in its synchronised restriction.

Theorem 3. *The emptiness and universality problems are decidable for synchronised expressions. The inclusion and equivalence problems for iterated-sum expressions E_1, E_2 such that* $\text{Sync}\{E_1, E_2\}$ *holds are decidable.*

5. Weighted chop automata

In this section, we introduce a new weighted automata model, called *weighted chop automata* (WCA), into which we can transform iterated-sum expressions. As a consequence, the undecidability of iterated-sum expressions (Theorem 2) carries over to WCA. We therefore introduce the class of synchronised WCA, into which synchronised expressions can be compiled and decidability is recovered, proving Theorem 3. The intuitive behaviour of a WCA is as follows. A generalised NFA (whose transitions are not reading single letters but words in some regular language) "chop" the input word into factors, on which expressions of the form $\phi(C_1, \dots, C_n)$, where C_i are smaller WCA, are applied to obtain intermediate values, which are then summed to obtain the value of the whole input word. In the following, we first introduce generalised NFA, then WCA and finally their synchronised restriction.

Definition 5 *(GNFA).* A *generalised NFA* is a tuple $A = (Q, I, F, \Delta)$ where Q is a set of states, I its initial states and F its final states, and Δ maps any pair $(p, q) \in Q^2$ to a regular language $\Delta(p, q) \subseteq \Sigma^*$ (finitely represented by some (classical) NFA).

A run of A over a word $u \in \Sigma^*$ is a sequence $r = q_0 u_1 \dots q_{n-1} u_n q_n$ such that $u = u_1 \dots u_n$ and $u_i \in \Delta(q_{i-1}, q_i)$ for each i. It is accepting if $q_0 \in I$ and $q_n \in F$. We say that A is unambiguous if for all $u \in \Sigma^*$, there is at most one accepting run of A on u (and hence its decomposition $u_1 \dots u_n$ is unique). We define the representation size of a generalised finite automaton $A = (Q, I, F, \Delta)$ as $\sum_{p,q \in Q} n_{p,q} + |Q| + |\Delta|$ where $n_{p,q}$ is the number of states of the NFA recognising $\Delta(p, q)$.

Proposition 9. *We can decide whether a GNFA is unambiguous in* PTIME.

Proof. Let $A = (Q, I, F, \Delta)$ be a GNFA whose languages $\Delta(p, q)$ are given by NFA $A_{p,q} = (Q_{p,q}, I_{p,q}, F_{p,q}, \Delta_{p,q})$. We construct in polynomial time a finite transducer $T = (P, I, \{p_f\}, \Delta')$ which, given a word $u \in L(A)$, outputs any run of A on u (seen as a word over the alphabet $Q \cup \Sigma$). Clearly, T defines a function iff A is unambiguous. We refer the reader for instance to [4] for a definition of finite transducers, but let us recall that it is an automaton extended with outputs. In particular, the transition relation Δ' has type $\Delta' \subseteq P \times \Sigma^* \times \Gamma^* \times P$, where Γ is the output alphabet. Transitions are denoted by $p \xrightarrow{u|v} q$ where u is the input word and v the output word. In general, a transducer defines a binary relation from input to output words, but functionality is decidable in PTIME for finite transducers [4].

To construct T, the idea is as follows. We take $\Gamma = Q \cup \Sigma$ has output alphabet. Then, $P = \{p_f\} \cup Q \uplus \biguplus_{p,q \in Q} Q_{p,q}$ where p_f is a new state. The transition function contains the following rules:

- $q \xrightarrow{\varepsilon|q} s_0$ for all $q \in Q$ and $s_0 \in \bigcup_{q' \in Q} I_{q,q'}$: when entering a new subautomaton $A_{q,q'}$, the transducer starts by writing the state q on the output.
- $s \xrightarrow{\sigma|\sigma} s'$ for all $(s, \sigma, s') \in \bigcup_{p,q \in Q} \Delta_{p,q}$: inside a subautomaton $A_{p,q}$, only symbols from Σ are written on the output.
- $s \xrightarrow{\sigma|\sigma} q$ if there are $p, q \in Q$ and $s' \in F_{p,q}$ such that $(s, \sigma, s') \in \Delta_{p,q}$: when reaching an accepting state in a subautomaton, the transducer make exit the subautomaton.
- $q \xrightarrow{\varepsilon|q} p_f$ for all $q \in F$: the transducer write the last seen accepting state (when the end of the word is reached) on the output and goes to p_f, which is accepting and is a deadlock (no outgoing transitions). $\square$

5.1. Definition and closure properties

We formally define WCA, then investigate their closure properties.

Definition 6 *(WCA).* The class of *weighted chop automata* is inductively defined as follows.

A 0-weighted chop automaton is an unambiguous WA.

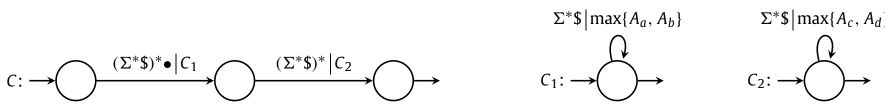
Let $m > 0$. An *m-weighted chop automaton* (m-WCA) is a tuple $C = (A, \lambda)$ where A is an unambiguous GNFA and λ is a function mapping any pair $(p, q) \in Q^2$ to some term $\phi(C_1, \ldots, C_k)$ where for all i, C_i is an m'-WCA, with $m' < m$ and ϕ is a Presburger combinator of arity k. Moreover, it is required that at least one C_i is an $(m-1)$-WCA. A WCA C is an m-WCA for some m. By definition of m-WCA, m is unique and is called the chop level of C.

A WCA C defines a quantitative language $[\![C]\!]$ of domain $\mathrm{dom}(C)$ inductively defined as follows:

- If C is a 0-WCA, then its semantics is that of unambiguous WA defined in Section 2.
- Otherwise $C = (A, \lambda)$, and the set $\mathrm{dom}(C)$ is the set of words $u = u_1 \ldots u_n$ on which there exists one accepting run $r = q_0 u_1 \ldots q_{n-1} u_n q_n$ of A such that for all $1 \le i \le n$, if $\lambda(q_{i-1}, q_i)$ is of the form $\phi(C_1, \ldots, C_k)$, then $u_i \in \bigcap_{j=1}^{k} \mathrm{dom}(C_j)$, and in this case we let $v_i = [\![\phi]\!]([\![C_1]\!](u_i), \ldots, [\![C_k]\!](u_i))$. The value of r (which also defines the value of u) is then $\sum_{i=1}^{n} v_i$.

The (unique) sequence $(u_1, \lambda(q_0, q_1)) \ldots (u_n, \lambda(q_{n-1}, q_n))$ is denoted in the sequel by $\mathrm{dec}_C(u)$. We define the representation size of a WCA $C = (A, \lambda)$ with $A = (Q, I, F, \Delta)$ as $|C| = \sum_{p,q \in Q} n_{p,q} + |Q| + |\Delta| \ell$ where $n_{p,q}$ is the number of states of the NFA recognising $\Delta(p, q)$ and ℓ is the maximal size of expressions in the range of λ, $|\phi(C_1, \ldots, C_k)| = |\phi| + \sum_{i=1}^{k} |C_i|$.

Example 5. Let $\Sigma = \{a, b, c, d\}$ and $\bullet, \$ \notin \Sigma$. The WCA depicted below realises the function which maps any word of the form $u_1\$ \ldots u_n\$ \bullet v_1\$ \ldots v_m\$$, where $u_i, v_i \in \{a, b, c, d\}^*$ to $\sum_{i=1}^{n} \max(\#_a(u_i), \#_b(u_i)) + \sum_{i=1}^{m} \max(\#_c(v_i), \#_d(v_i))$. The automata A_σ are unambiguous WA counting the number of occurences of σ, and C_i are shortcuts for $\phi_{id}(C_i)$ where ϕ_{id} defines the identify function.


C:
$(\Sigma^*\$)^*\bullet \mid C_1$
$(\Sigma^*\$)^* \mid C_2$
$\Sigma^*\$ \mid \max\{A_a, A_b\}$
C_1:
$\Sigma^*\$ \mid \max\{A_c, A_d\}$
C_2:

Remark 3. Here WCA are unambiguous *by definition*. Note that, a fully non-deterministic version of WCA can be defined, by using a max aggregator to combine the values of all accepting runs. As we will see, the > 0-emptiness problem is already undecidable for (unambiguous) WCA, and further synchronisation restriction will be necessary to recover decidability, this is why we decided to ask for unambiguity in the definition.

Proposition 10. *The domain of any WCA is (effectively) regular.*

Proof. Let $C = (A, \lambda)$ be an m-WCA where $A = (Q, I, F, \Delta)$. We show by induction on m that $\mathrm{dom}(C)$ is (effectively) regular. If $m = 0$ then C is an unambiguous WA and its domain is (effectively) regular (given by its underlying (input) automaton). Otherwise, assume by induction hypothesis that for all $m' < m$ the domain of any m'-WCA is (effectively) regular. We construct a GNFA to recognise $\mathrm{dom}(C)$. Let $p, q \in Q$ and $\lambda(p, q) = v$. We define $\mathrm{dom}(v) = \bigcap_{j=1}^{k} \mathrm{dom}(C_j)$ if $v = \phi(C_1, \ldots, C_k)$. By induction hypothesis each $\mathrm{dom}(v)$ is (effectively) regular since all C_j is a m'-WCA with $m' < m$. By definition of $\mathrm{dom}(C)$, the domain of C is the language of the GNFA $A' = (Q, I, F, \Delta')$ where for all $q, p \in Q$ we have $\Delta'(p, q) = \Delta(p, q) \cap \mathrm{dom}(v)$. $\square$

Closure properties We now investigate the closure properties of WCA. Given two quantitative languages f_1, f_2, let us define their *split sum* $f_1 \odot f_2$ as the function mapping any word u which can be uniquely decomposed into u_1, u_2 such that $u_i \in \mathrm{dom}(f_i)$ for all i, to $f_1(u_1) + f_2(u_2)$ [2]. We also define the conditional choice $f_1 \triangleright f_2$ as the mapping of any word $u \in \mathrm{dom}(f_1)$ to $f_1(u)$, and of any word $u \in \mathrm{dom}(f_2) \setminus \mathrm{dom}(f_1)$ to $f_2(u)$ [2]. These operators may be thought of as (unambiguous) concatenation and disjunction in rational expressions. WCA are closed under these two operations, as well as Presburger combinators, (unambiguous) iterated sum and regular domain restriction, in the sense given by Proposition 11.

First, we show a property on the split sum of languages (proved in Appendix).

Lemma 4. *Let L_1, L_2 be two regular languages L_1, L_2 both realisable by some GNFA with at most n states. If $L_1 \odot L_2$ denotes the set of words u which can be uniquely decomposed into $u_1 u_2$ with $u_i \in L_i$, then there exists $2n$ regular languages N_i, M_i such that $N_i \subseteq L_1$ and $M_i \subseteq L_2$, and $L_1 \odot L_2 = \bigcup_{i=1}^{n} N_i M_i$.*

We now turn to the closure properties of WCA.

Proposition 11. ***Closure Properties of WCA*** *The class of quantitative languages defined by WCA is closed under split sum, conditional choice, Presburger combinators, iterated sum and regular domain restriction.*

More precisely, let $C, C_1, \dots, C_k$ be WCA, ϕ be a Presburger combinator of arity k and $L \subseteq \Sigma^*$ a regular language. One can construct WCA respectively denoted by $\phi(C_1, \dots, C_k)$, $C^{\circledast}$, $C_1 \odot C_2$, $C_1 \rhd C_2$ and $C|_L$ such that

– $\mathrm{dom}(\phi(C_1, \dots, C_k)) = \bigcap_{i=1}^k \mathrm{dom}(C_i)$ *and for all* $u \in \bigcap_{i=1}^k \mathrm{dom}(C_i)$,

$$[\![\phi(C_1, \dots, C_n)]\!](u) = [\![\phi]\!]([\![C_1]\!](u), \dots, [\![C_n]\!](u))$$

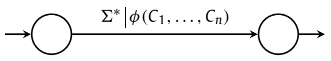
– $[\![C^{\circledast}]\!] = [\![C]\!]^{\circledast}$, $[\![C \odot D]\!] = [\![C]\!] \odot [\![D]\!]$ *and* $[\![C \rhd D]\!] = [\![C]\!] \rhd [\![D]\!]$, $[\![C|_L]\!] = [\![C]\!]|_L$

Proof. We prove the closure under each operator one by one.

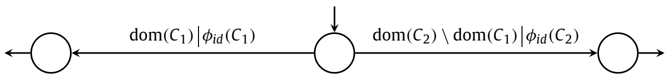
Closure under star Let $C^{\circledast} = (A, \lambda)$ defined as follows. The only difficulty is that it should be unambiguous (because we want decomposition to be unique), hence it is not correct to add some ε-transition from accepting states of C to its initial states. However, it is possible to define an unambiguous NFA $B = (Q, I, F, \Delta)$ with a set of special states $S \subseteq Q$ such that $L(B) = \mathrm{dom}(C)^{\circledast}$ and such that for all $u \in L(B)$, the occurrences of special states in the accepting run of B on u decomposes (uniquely) u into factors that belong to $\mathrm{dom}(C)$. Then, the states of $C^{\circledast}$ are the states $S \cup I$, the initial states I, final states $S \cap F$, and $\Delta(s, s')$ for all $s \in S \cup I$ and $s' \in S$, is the set of words on which there is a run of B from s to s' that does not pass by any state of S (except at the end and beginning). This set is easily shown to be regular. Finally, $\lambda(s, s') = \phi_{id}(C)$ where ϕ_{id} defines the identity function.

Closure under regular domain restriction If $C = (A, \lambda)$, then it suffices to take the product of A with any DFA B such that $L(B) = L$. Since A is a GNFA, the product is a bit different than the usual automata product. Assume $A = (Q, I, F, \Delta)$ and $B = (P, p_0, F', \delta')$ (a classical DFA). Then $A \times B = (Q \times P, I \times \{p_0\}, F \times F', \Delta \times \delta')$ where for all (q, p), (q', p'), $(\Delta \times \delta')((q, p), (q', p'))$ is the set of words in $\Delta(q, q')$ such that there exists a run of B from state p to state p'. This set is effectively regular. The resulting GNFA is unambiguous since B was taken to be deterministic and A is unambiguous.

Closure under Presburger combinators The WCA $\phi(C_1, \dots, C_n)$ is defined as:

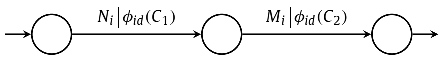


Closure under conditional choice The WCA $C_1 \rhd C_2$ is defined as


$\mathrm{dom}(C_1) | \phi_{id}(C_1)$
$\mathrm{dom}(C_2) \setminus \mathrm{dom}(C_1) | \phi_{id}(C_2)$

Note that $\mathrm{dom}(C_i)$ are regular by Proposition 10.

Closure under split sum By Lemma 4, $\mathrm{dom}(C_1) \odot \mathrm{dom}(C_2) = \bigcup_{i=1}^n N_i M_i$ for some regular languages $N_i \subseteq \mathrm{dom}(C_1)$ and $M_i \subseteq \mathrm{dom}(C_2)$. For all $i = 1, \dots, n$, we define the WCA $(C_1 \odot C_2)|_{N_i M_i}$ as depicted below:



Note that it is unambiguous since $N_i M_i \subseteq \mathrm{dom}(C_1) \odot \mathrm{dom}(C_2)$, $N_i \in \mathrm{dom}(C_1)$ and $M_i \in \mathrm{dom}(C_2)$. Futhermore, for all $u \in N_i M_i$ such that $u = u_1 u_2$ with $u_1 \in \mathrm{dom}(C_1)$ and $u_2 \in \mathrm{dom}(C_2)$ we have $[\![(C_1 \odot C_2)|_{N_i M_i}]\!](u) = [\![C_1]\!](u_1) + [\![C_2]\!](u_2)$. Finally, we let $C_1 \odot C_2 = \rhd_{i=1}^n (C_1 \odot C_2)|_{N_i M_i}$ (the way this expression is parenthesised, as well as the order in which the index i are taken, does not change the semantics). $\square$

A direct consequence of the closure properties of WCA is that any iterated-sum expression can be encoded into an equivalent WCA.

Corollary 2. *Given an iterated-sum expression E we can construct a WCA C such that $[\![E]\!] = [\![C]\!]$.*

The undecidability of WCA is implied by Theorem 2 and Corollary 2. Furthermore, the iterated-sum expression of Theorem 2 can be encoded by a 1-WCA.

Corollary 3. *Emptiness, universality, inclusion and equivalence for WCA are undecidable problems, even for 1-WCA.*

5.2. Synchronised weighted chop automata

As for iterated-sum expressions, we present a subclass of WCA with a notion of synchronisation into which synchronised expressions can be compiled. Then, in the next section, we show the decidability of synchronised WCA.

Definition 7. The notion of synchronisation of WCA is inductively defined:

– Two terms $\phi_1(C_1, \dots, C_k)$ and $\phi_2(C'_1, \dots, C'_{k'})$ are synchronised if C_i is synchronised with C'_j for all i, j.
– Two WCA C_1, C_2 are synchronised, denoted by $C_1||C_2$, if they are either both 0-WCA, or $C_1 = (A_1, \lambda_1)$ and $C_2 = (A_2, \lambda_2)$, and the following holds: for all $u \in L(A_1) \cap L(A_2)$, if $\mathrm{dec}_{C_1}(u) = (u_1, E_1), \dots, (u_n, E_n)$ and $\mathrm{dec}_{C_2}(u) = (v_1, F_1), \dots, (v_m, F_m)$, then $n = m$ and for all $1 \le i \le n$, we have $u_i = v_i$ and E_i is synchronised with F_i.

We write $\mathrm{Sync}(\{C_1, \dots, C_n\})$ if $C_i||C_j$ for all $i, j \in \{1, \dots, n\}$. Now, a WCA C is synchronised if it is an unambiguous WA, or it is of the form (A, λ), and any expression $\phi(C_1, \dots, C_k)$ in the range of λ satisfies $\mathrm{Sync}(\{C_1, \dots, C_k\})$.

E.g., the WCA of Example 5 is trivially synchronised since any Presburger combinators of C have an arity 1 and C_1, C_2 are unambiguous WA.

Remark 4. Let C_1 and C_2 be two WCA. As a consequence of the definition of synchronisation, if $C_1||C_2$, then both C_1 and C_2 are m-WCA for the same m. It is because of the base case of the definition of synchronisation: WA can be synchronised with WA only. Then, the property goes by induction on m.

Proposition 12. *Let C_1, C_2 be two WCA. Deciding whether $C_1||C_2$ holds can be done in* PTime.

Proof. We provide a recursive algorithm which takes two chop automata C_1, C_2 and checks whether $C_1||C_2$ in PTime. If C_1 and C_2 are both unambiguous WA, then the algorithm returns true. If one of them is an unambiguous WA and the other not, then the algorithm returns false.

Now, consider the case where we have chop automata $C_1 = (A_1, \lambda_1)$ and $C_2 = (A_2, \lambda_2)$. We first show how to decide the following (weaker) property: for all $u \in L(A_1) \cap L(A_2)$, if $\mathrm{dec}_{C_1}(u) = (u_1, E_1), \dots, (u_n, E_n)$ and $\mathrm{dec}_{C_2}(u) = (v_1, F_1), \dots, (v_m, F_m)$, then $n = m$ and for all $i \in \{1, \dots, n\}$, we have $u_i = v_i$.

As in Proposition 9, the idea is to construct a transducer T, which defines a function from Σ^* to 2^{Σ^*}, whose domain is $L(A_1) \cap L(A_2)$. Given $u \in \mathrm{dom}(T)$, if $\mathrm{dec}_{C_1}(u) = (u_1, E_1), \dots, (u_n, E_n)$ and $\mathrm{dec}_{C_2}(u) = (v_1, F_1), \dots, (v_m, F_m)$, then T returns the set of words $\{u_1\#u_2\#\dots\#u_n, v_1\#v_2\#\dots\#v_n\}$. By definition of a WCA decomposition $u_1 \dots u_n = v_1 \dots v_n = u$, then the latter set is a singleton i.e. $u_1\#u_2\#\dots\#u_n = v_1\#v_2\#\dots\#v_n$ iff the two decompositions are equal. Hence, it suffices to decide whether T defines a function (i.e. is functional), which can be done in PTime in the size of T (see for instance [4]).

It remains to show how to construct T. T is the disjoint union of two transducers T_1 and T_2, which respectively output $u_1\#u_2\#\dots\#u_n$ and $v_1\#v_2\#\dots\#v_n$. Basically T_i can be seen as a copy of A_i where each transitional NFA are replaced by transducer which output the input with # as an ending maker. Consider T_1. Let $A_1 = (Q, q_0, F, \Delta)$ and $(A_{p,q})_{p,q \in Q}$ be NFA that recognise $\Delta(p, q)$. Whenever a transition (α, σ, β) of $A_{q,q'}$ is fired, T_1 makes several choices. Either q' is not final in $A_{q,q'}$ and T_1 moves to state β and write σ on the output. Either q' is final, in that case T_1 may move to state β while writing σ on the output, or move to the initial state of some automaton $A_{q',q''}$ for some non-deterministically chosen state $q'' \in Q$, and write $\sigma\#$ on the output. Clearly, T_1 has a polynomial size in the size of C_1.

In order to decide the synchronisation between C_1 and C_2, we also need to check that $E_i||F_i$ for all sub-expressions E_i, F_i that occur at the same position in some decomposition. Formally, let S be the set of expressions E, F such that there exists $u \in L(A_1) \cap L(A_2)$, such that $\mathrm{dec}_{C_1}(u) = (u_1, E_1), \dots, (u_n, E_n)$ and $\mathrm{dec}_{C_2}(u) = (v_1, F_1), \dots, (v_m, F_m)$ and there exists i such that $E_i = E$ and $F_i = F$. We will show that S can be computed in PTime. Once S has been computed, for every pair $(\phi(C_1, \dots, C_n), \phi'(C'_1, \dots, C'_m)) \in S$, it suffices to call this algorithm on each pair (C_i, C'_j) for all i, j.

It remains to show that S can be computed in polynomial time. Again, for all expressions E, F occurring in the range of λ_1 and λ_2 respectively, one could define some automaton $A_{E,F}$ (of polynomial size) which accepts a word $u \in L(A_1) \cap L(A_2)$ iff E, F occurs together in the respective decomposition of u, i.e. $E \in \mathrm{dec}_{C_1}(u)$ and $F \in \mathrm{dec}_{C_2}(u)$. Then, for all these pairs, if $L(A_{E,F}) \neq \varnothing$ (which can be checked in PTime), then we add (E, F) to S. To construct $A_{E,F}$, the idea is to simulate, via a product construction, an execution of C_1 and an execution of C_2 in parallel (by also simulating the smaller automata defining the regular languages on the transitions of C_1 and C_2). In this product construction, one bit of memory is used to remember whether a pair of states (p_1, p_2) of C_1 and C_2 respectively, such that $E = \lambda_1(p_1)$ and $F = \lambda_2(p_2)$, was reached. The automaton accepts if such a pair was found, and the simulation of the two runs accept (meaning that $u \in L(A_1) \cap L(A_2)$). □

Since a WCA is synchronised iff any of its Presburger combinators are parameterised by a synchronised set of WCA, Proposition 12 implies the following.

Corollary 4. *Deciding whether a WCA is synchronised is in* PTIME.

6. Decidability of synchronised weighted chop automata and iterated sum expressions

In this section, we prove that emptiness, universality, inclusion and equivalence problems are decidable for synchronised WCA. Then, we show that synchronised iterated sum expressions can be effectively converted into synchronised WCA, thus proving their decidability (Theorem 3).

Overview of the proof of synchronised WCA decidability The cornerstone of the proof is proving the semi-linearity of the range of synchronised WCA. Let us give the intuitive ideas on how this can be shown. Assume for the time being that we consider a synchronised WCA $C = (A, \lambda)$ with a single state q, and hence a single transition from q to q on any word of $\Delta(q, q)$. Assume that $\lambda(q, q) = \phi(C_1, \dots, C_n)$. Then, in order to prove that the range of $[\![C]\!]$ is semi-linear, we have to show that $\{([\![C_1]\!](u), \dots, [\![C_n]\!](u)) \mid u \in \Delta(q, q) \cap \bigcap_{i=1}^{n} \mathrm{dom}(C_i)\}$ is a semi-linear set. For a general synchronised WCA, if one wants to prove the result by induction on its structure, a stronger statement is needed, namely to consider tuples of WCA instead of a single one. In particular, we show (Lemma 5) the following result: For all $C_1, \dots, C_n$ WCA such that $\mathrm{Sync}\{C_1, \dots, C_n\}$, the following set is (effectively) semi-linear:

$$\left\{ ([\![C_1]\!](u), \dots, [\![C_n]\!](u)) \mid u \in \bigcap_{i=1}^{n} \mathrm{dom}(C_i) \right\}$$

Similarly to the case where the C_i are unambiguous WA, for which semi-linearity was shown by converting the tuples of automata into a single unambiguous weighted automaton weighted in $\mathbb{Z}^n$ (through a product construction), we can as well, and thanks to synchronisation, define the product of n m-WCA, for all m, as a WCA weighted in $\mathbb{Z}^n$, what we call a *generalised WCA* (GWCA). Then we show semi-linearity of the range of synchronised GWCA, by induction on its nesting level (level 0 corresponds to unambiguous WA with values in $\mathbb{Z}^n$). We first define formally GWCA.

Definition 8 *(GWCA).* The class of *generalised WCA* is inductively defined as follows. A 0-GWCA is an unambiguous WA with values in $\mathbb{Z}^d$. For $m > 0$, an m-GWCA is defined as a WCA (A, λ), except that λ returns d-tuples of terms of the form $\phi(C_1, \dots, C_k)$, where C_i are m'-GWCA with $m' < m$ and ϕ is a Presburger combinator of arity $d \times k$ that returns a d-tuple of values.

We define the representation size of a GWCA $C = (A, \lambda)$ of dimension d with $A = (Q, I, F, \Delta)$ as $|C| = \sum_{p,q \in Q} n_{p,q} + |Q| + |\Delta| d\ell$ where $n_{p,q}$ is the number of states of the NFA recognising $\Delta(p, q)$ and ℓ is the maximal size of expressions in the range of λ, $|\phi(C_1, \dots, C_k)| = |\phi| + \sum_{i=1}^{k} |C_i|$. The semantics and the notion of synchronisation are both defined the same way as for WCA. Just to make it clear for synchronisation, $C_1 = (A_1, \lambda_1)$ and $C_2 = (A_2, \lambda_2)$ are synchronised if for all $u \in \mathrm{dom}(C_1) \cap \mathrm{dom}(C_2)$, we let:

$$\mathrm{dec}_{C_1}(u) = (u_1, (E_{1,1}, \dots, E_{1,d})), \dots, (u_n, (E_{n,1}, \dots, E_{n,d}))$$
$$\mathrm{dec}_{C_2}(u) = (u'_1, (E'_{1,1}, \dots, E'_{1,d'})), \dots, (u'_n, (E'_{n',1}, \dots, E'_{n',d'}))$$

then $n = n'$, for each $i \in \{1, \dots, n\}$, $j \in \{1, \dots, d\}$ and $j' \in \{1, \dots, d'\}$ we have $u_i = u'_i$ and if $E_{i,j}$ and $E'_{i,j'}$ are of the respective form $\phi(C_1, \dots, C_k)$ and $\phi'(C'_1, \dots, C'_{k'})$, then $C_\ell || C'_{\ell'}$ for all $\ell \in \{1, \dots, k\}$ and $\ell' \in \{1, \dots, k'\}$.

We now define the product of C_1, C_2 two m-GWCA. If $m = 0$ then C_1, C_2 are unambiguous WA with values in $\mathbb{Z}^{d_1}$ and $\mathbb{Z}^{d_2}$ respectively and the product construction is a classical state product construction, whose transitions are valued in $\mathbb{Z}^{d_1+d_2}$. Else if $m > 0$ such that $C_i = (Q_i, I_i, F_i, \Delta_i, \lambda_i)$ for each i, we define $C_1 \times C_2 = (Q_1 \times Q_2, I_1 \times I_2, F_1 \times F_2, \Delta, \lambda)$ where $\Delta((p_1, p_2), (q_1, q_2)) = \Delta_1(p_1, q_1) \cap \Delta_2(p_2, q_2)$ and $\lambda((p_1, p_2), (q_1, q_2)) = (\lambda_1(p_1, q_1), \lambda_2(p_2, q_2))$ for all $p_1, q_1 \in Q_1$ and $p_2, q_2 \in Q_2$.

As a direct consequence of the construction and the definition of synchronisation, we have:

Proposition 13. *Given C_1, C_2 two WCA. We have* $\mathrm{dom}(C_1 \times C_2) = \mathrm{dom}(C_1) \cap \mathrm{dom}(C_2)$ *and if* $\mathrm{Sync}\{C_1, C_2\}$ *then* $[\![C_1 \times C_2]\!](u) = ([\![C_1]\!](u), [\![C_2]\!](u))$ *for all* $u \in \mathrm{dom}(C_1) \cap \mathrm{dom}(C_2)$.

6.1. Decidability of synchronised WCA

We show the semi-linearity of the range of a synchronised GWCA, by induction on its nesting level.

Lemma 5. *Given a tuple $(C_1, \dots, C_n)$ of GWCA such that* $\mathrm{Sync}(\{C_1, \dots, C_n\})$ *then* $\{([\![C_1]\!](u), \dots, [\![C_n]\!](u)) \mid u \in \bigcap_{i=1}^{n} L(C_i)\}$ *is effectively semi-linear.*

Proof. We recall that if $\text{Sync}\{C_1,\dots,C_n\}$, then the C_i are all m-GWCA for some m by Remark 4.

If $m=0$, then the C_i are all unambiguous WA whose transitions are valued by tuples of integers, and we can take their product, whose range can be shown to be semi-linear by Lemma 3.

Now, suppose that $m>0$. Let us construct the product $C=(\prod_{i=1}^n C_i)$ of GWCA C_i. By Proposition 13 we have $\text{dom}(C)=\bigcap_i \text{dom}(C_i)$ and for all $u\in\text{dom}(C)$, $[\![C]\!](u)=([\![C_1]\!](u),\dots,[\![C_n]\!](u))$. Therefore, it suffices to show that $\text{Range}(C)=\{[\![C]\!](u)\mid u\in\text{dom}(C)\}$ is semi-linear to prove the lemma.

Suppose that $C=(A,\lambda)$ where $A=(Q,I,F,\Delta)$ and λ maps any pair $p,q\in Q$ to some n-ary tuple of expressions $(\phi_1(C^1_1,\dots,C^1_{k_1}),\dots,\phi_n(C^n_1,\dots,C^n_{k_n}))$. W.l.o.g, we can assume that $\Delta(p,q)=\text{dom}(C^i_j)$ for all $1\le i\le n$ and all $1\le j\le k_i$. Otherwise, if $L=\bigcap_{1\le i\le n}\bigcap_{1\le j\le k_i}\text{dom}(C^i_j)$ (which is regular by Proposition 10), we restrict the domain of any C^i_j to L and replace $\Delta(p,q)$ by $\Delta(p,q)\cap L$, this does not change the semantics of C. Closure under regular domain restriction was shown for (non-generalised) WCA in Proposition 11, but the same proof works for synchronised GWCA. With this assumption, we get $\text{dom}(C)=L(A)$. We also define the range of $\lambda(p,q)$ as follow:

$$S_{p,q}=\{[\![(\phi_1(C^1_1,\dots,C^1_{k_1})]\!](u),\dots,[\![\phi_n(C^n_1,\dots,C^n_{k_n})]\!](u))\mid u\in\Delta(p,q)\}$$

By synchronisation of C, all the C^j_i are all m'-WCA for some $m'<m$ and $\text{Sync}(\{C^1_1,\dots,C^1_{k_1},\dots,C^n_1,\dots,C^n_{k_n}\})$ holds. Then the induction hypothesis on the tuple $(C^1_1,\dots,C^1_{k_1},\dots,C^n_1,\dots,C^n_{k_n})$ gives the semi-linearity of $S_{p,q}$ since $\phi_1,\dots,\phi_n$ are semi-linear maps.

Now, sets of n-tuple of integers have the structure of a monoid $(2^{\mathbb{Z}^n},+,\mathbb{0}_n)$ where $+$ is defined by $S+S'=\{s+s'\mid s\in S,s'\in S'\}$ and $\mathbb{0}_n=\{(0,\dots,0)\}$ (tuple of arity n). Consider the free monoid over $Q\times Q$ and the morphism μ from this monoid to $(2^{\mathbb{Z}^n},+,\mathbb{0}_n)$, defined by $\mu((p,q))=S_{p,q}$ for all $(p,q)\in Q\times Q$. It is easily shown that for any regular language $N\subseteq(Q\times Q)^*$, $\mu(N)$ is semi-linear. It is because the $S_{p,q}$ are semi-linear, and semi-linear sets are (effectively) closed under sum, union, and Kleene star [11]. Thus, by taking $N\subseteq(Q\times Q)^*$ as the set of words for the form $(p_1,p_2)(p_2,p_3)\dots(p_{k-1},p_k)$ such that $p_1\in I$, $p_k\in F$, and for all $i\in\{1,\dots,k-1\}$, $\Delta(p_i,p_{i+1})\neq\varnothing$ then the semantics of WCA which sum values along runs gives the desired result. Details of the regularity of N and the equivalence between $\text{Range}(C)$, $\mu(N)$ are given in Appendix. □

A direct consequence of the latter result is the semi-linearity of the range of synchronised WCA:

Corollary 5. *Given C a synchronised WCA, $\{[\![C]\!](u)\mid u\in\text{dom}(C)\}$ is semi-linear and effectively computable.*

The following theorem is a direct consequence of Corollary 5, Remark 1 and Proposition 11.

Theorem 4. *The emptiness and universality problems are decidable for synchronised WCA. The inclusion and equivalence problems for WCA C_1,C_2 such that $\text{Sync}\{C_1,C_2\}$ holds are decidable.*

6.2. Decidability of synchronised expressions

We conclude this section by showing that any synchronised expression can be converted into a synchronised WCA. This conversion is effective and thus the decidability of synchronised expressions (Theorem 3) becomes a consequence of Theorem 4.

Main ideas of the proof Let us illustrate the intuition of this proof on an example. Take some synchronised expression $E=\phi(A,B^{\circledast})$ for some unambiguous WA A,B, and some Presburger combinator ϕ. The difficulty for converting this kind of expression into WCA, comes from the fact that A is applied on the whole input word, while B is applied iteratively on factors of it. The unambiguous WA A,B are both 0-WCA. Then, according to the construction of Proposition 11, the expression $B^{\circledast}$ can be converted into a 1-WCA which we also denote by $B^{\circledast}$. Therefore, A and $B^{\circledast}$ are not synchronised, since n-WCA are synchronised with n-WCA only by Remark 4.[13] So, if we let $C_{\phi(A,B^{\circledast})}$ be the WCA obtained by applying the Presburger combinator ϕ to the 0-WCA A and the 1-WCA $B^{\circledast}$ (as defined in Proposition 11), we have that $C_{\phi(A,B^{\circledast})}$ is equivalent to E, but it is not a synchronised WCA in general.

To synchronise A and $B^{\circledast}$, the idea is to artificially chop runs of A into smaller pieces, synchronised with B. To do that, we construct a 1-WCA consisting to a GNFA which calls A for partial computations from any state p to any state q. Due to synchronisation, these partial computations are required to accept words which are in $\text{dom}(B)$. More precisely, for all states p,q of A we define $A_{p,q}$ to be the WA A with initial state p, final state q, and whose domain is restricted to $\text{dom}(B)$. Then, all the smaller automata $A_{p,q}$ are combined into a single 1-WCA which simulates successive applications of the automata $A_{p,q}$, by taking care of the fact that the words it accepts must be uniquely decomposable into factors of $\text{dom}(B)$. This

[13] In this sense, the definition of synchronisation may seem a bit strong, unlike that of synchronisation of iterated-sum expressions, but it makes the decidability (and in particular the semi-linearity property Lemma 5) way easier to prove, because it allows from a product construction, just as in the non-iterated case (monolithic expressions).

resulting 1-WCA, say C_A, is necessarily synchronised with $B^{\circledast}$, and we can return the single synchronised 2-WCA $C_{\phi(C_A,B^{\circledast})}$ obtained by applying ϕ on C_A and $B^{\circledast}$ (again Proposition 11), which is equivalent to expression E. In order to be able to convert any synchronised expression into a synchronised WCA by structural induction, one needs a stronger statement, namely on tuples of synchronised expressions:

Theorem 5. *For all tuples of iterated-sum expressions $\mathbb{E} = (E_1, \dots, E_n)$ such that* $\mathrm{Sync}(\mathbb{E})$, *one can construct a tuple of WCA $\mathbb{C} = (C_1, \dots, C_n)$ such that the following condition hold:*

1. *for all $i \in \{1, \dots, n\}$*, $\mathrm{dom}(C_i) = \bigcap_{j=1}^{n} \mathrm{dom}(E_j)$
2. *for all $i \in \{1, \dots, n\}$, for all $u \in \bigcap_{j=1}^{n}$* $\mathrm{dom}(E_j)$, $[\![E_i]\!](u) = [\![C_i]\!](u)$
3. $\mathrm{Sync}(\mathbb{C})$

Proof. The proof goes by induction on $||\mathbb{E}|| = \sum_{i=1}^{n} ||E_i||$ where for any iterated-sum expression E, we define $||E|| \in \mathbb{N}$ inductively by $||A|| = 0$, $||E^{\circledast}|| = ||E|| + 1$, $||\phi(E_1, E_2)|| = 1 + ||E_1|| + ||E_2||$.

So, if $||\mathbb{E}|| = 0$, then all expressions in $\mathbb{E}$ are unambiguous WA $A_1, \dots, A_n$, and hence are 0-WCA. However, the conditions in the lemma requires that the domains of all WCA are equal to $D = \bigcap_{i=1}^{n} \mathrm{dom}(A_i)$. It suffices to restrict $A_1, \dots, A_n$ to D, which is always possible since unambiguous WA are closed under regular domain restriction, and D is regular. One obtains unambiguous WA $A'_1, \dots, A'_n$, i.e. 0-WCA, which satisfy the requirements since unambiguous WA are always mutually synchronised.

The case $||\mathbb{E}|| > 0$ is more involved and is a disjunction of the following cases.

Case 1 There exists i such that $E_i = \phi(F_1, F_2)$. Then we define the $(n+1)$-tuple[14] $\mathbb{E}' = (E_1, \dots, E_{i-1}, F_1, F_2, E_{i+1}, \dots, E_n)$ which satisfies $\mathrm{Sync}(\mathbb{E}')$ by definition of synchronisation for iterated-sum expressions. We also have $||\mathbb{E}'|| < ||\mathbb{E}||$, hence we can apply the induction hypothesis on $\mathbb{E}'$, and obtain a tuple $\mathbb{C}' = (C_1, \dots, C_{i-1}, C_i^1, C_i^2, C_{i+1}, \dots, C_n)$ of WCA such that:

i for all $j \neq i$, $\mathrm{dom}(C_j) = \mathrm{dom}(C_i^1) = \mathrm{dom}(C_i^2) = D'$
ii for all $j \neq i$, $[\![E_j]\!]|_{D'} = [\![C_j]\!]$, $[\![F_1]\!]|_{D'} = [\![C_i^1]\!]$, $[\![F_2]\!]|_{D'} = [\![C_i^2]\!]$
iii $\mathrm{Sync}(\mathbb{C}')$ (in particular all WCA in $\mathbb{C}'$ have the same chope level m)

where $D' = \bigcap_{\alpha \in \mathbb{E}'} \mathrm{dom}(\alpha)$. We return the tuple of WCA

$$\mathbb{C} = (\phi_{id}(C_1), \dots, \phi_{id}(C_{i-1}), \phi(C_i^1, C_i^2), \phi_{id}(C_{i+1}), \dots, \phi_{id}(C_n))$$

where ϕ_{id} is a Presburger combinator which realises the identity function, and the Presburger operations on WCA have been defined in Proposition 11. Note that, we return $\phi_{id}(C_j)$ instead of C_j for all $j \neq i$, to force each WCA of $\mathbb{C}$ to have the same chop level $m+1$ and therefore get synchronisation of $\mathbb{C}$.

Case 2 There are only stars in the tuple i.e. $\mathbb{E}$ is of the form $(F_1^{\circledast}, \dots, F_n^{\circledast})$. In this case we apply our induction hypothesis on $(F_1, \dots, F_n)$, which is synchronised since $(F_1^{\circledast}, \dots, F_n^{\circledast})$ is synchronised. Then we obtain a tuple of WCA $(C_1, \dots, C_n)$, and return $(C_1^{\circledast}, \dots, C_n^{\circledast})$ where the operation $^{\circledast}$ on WCA has been defined in Proposition 11.

Case 3 Only unambiguous WA and iterated-sum expressions are mixed. Let us explain the construction with only a single automaton and a single star expression, i.e. $\mathbb{E} = (A, F^{\circledast})$. The case where there are arbitrarily many automata and star expressions is more technical but not more difficult, the main difficulties being found in this special case. The difficulty comes from the fact that A and F do not apply at the same level of decomposition, as explained in the overview of the proof: A runs on the whole input word, while F runs on factors of it.

Hence we artificially chop runs of A into run factors on words of $\mathrm{dom}(F)$. We assume w.l.o.g that A is trim,[15] and for all states p_1, p_2 of A, define A_{p_1,p_2} to be A with only initial state p_1 and only accepting state p_2. Since A is unambiguous and trim, so are the WA A_{p_1,p_2}.

Let B be a DFA accepting the set of words u that are uniquely decomposed into factors in $\mathrm{dom}(F)$ (see the proof of Proposition 5 for a construction of B). For all states q_1, q_2 of B, let L_{q_1,q_2} be the set of words in $\mathrm{dom}(F)$ such that there exists a run of B from state q_1 to state q_2. Clearly:

$$\mathrm{dom}(F^{\circledast}) = \bigcup_{m \in \mathbb{N}} \left\{ L_{q_0,q_1} L_{q_1,q_2} \dots L_{q_{m-1},q_m} \;\middle|\; \begin{array}{l} q_0, q_1, \dots, q_m \text{ are states of } B, \\ q_0 \text{ is initial and } q_m \text{ final} \end{array} \right\}$$

[14] Note that this case exhibits already the need to consider tuples of expressions rather a single expression, to prove Theorem 5 inductively.

[15] Trim mean that, all its states are accessible from an initial state and co-accessible from a final state. It is well-known that any automaton can be trimmed in polynomial time.

Now, for all states p_1, p_2 of A and all states q_1, q_2 of B, define A_{p_1,p_2,q_1,q_2} as the WA A_{p_1,p_2} restricted to the domain L_{q_1,q_2}. It can be assumed to be unambiguous as well, since A_{p_1,p_2} is unambiguous and B is deterministic. Then, apply the induction hypothesis on the pairs (A_{p_1,p_2,q_1,q_2}, F) which is synchronised to get equivalent synchronised WCA $(C^1_{p_1,p_2,q_1,q_2}, C^2_{p_1,p_2,q_1,q_2})$.

We now explain how to combine these WCA into a pair of WCA (C_1, C_2) equivalent to $(A, F^{\circledast})$ on $\mathrm{dom}(A) \cap \mathrm{dom}(F^{\circledast})$. Let C_{p_1,p_2,q_1,q_2} be a GWCA defined as the product $C^1_{p_1,p_2,q_1,q_2} \otimes C^2_{p_1,p_2,q_1,q_2}$ (hence with values in $\mathbb{Z}^2$). We now have to combine all these WCA into a single one C running on the whole input word. We construct C by taking the union of all WCA $C_{p,p',q,q'}$, and by "merging", for all states p_1, p_2, p_3 of A and all states q_1, q_2, q_3 of B, the accepting states of C_{p_1,p_2,q_1,q_2} with the initial states of C_{p_2,p_3,q_2,q_3}. The merging operation is non-deterministic, because we may need to stay in the automaton C_{p_1,p_2,q_1,q_2} even if we have already seen one of its accepting state: when C_{p_1,p_2,q_1,q_2} triggers a transition to one of its accepting state, it either go to it, or to some initial state of C_{p_2,p_3,q_2,q_3}. The initial states of C are the initial states of any $C_{p,p',q,q'}$ such that p is initial in A, q is initial in B. The accepting states of C are the accepting states of any $C_{p,p',q,q'}$ such that p' is accepting in A and q' is accepting in B. That way, we have $\mathrm{dom}(C) = \mathrm{dom}(A) \cap \mathrm{dom}(F^{\circledast})$. The WCA C_1, C_2 are obtained from C by projecting the pairs of expressions occurring in C to their first and second component respectively.

Finally, let us sketch how to proceed if there are more than one automaton $A_1, \dots, A_n$ in $\mathbb{E}$ and more than one star expression $F_1^{\circledast}, \dots, F_m^{\circledast}$ in $\mathbb{E}$. The idea is very similar, but we consider an automaton B that accepts all words that are uniquely decomposed according to $\bigcap_j \mathrm{dom}(F_j)$. Since the star expressions are synchronised, they all decompose the input word the same way, making this construction sound. Then, in the sub-weighted automata $A_{p,p',q,q'}$, p and p' are instead tuples of states of each automata A_j. $\square$

Finally, the announced direct corollary of the previous lemma is:

Corollary 6. *Any synchronised expression E is (effectively) equivalent to some synchronised weighted chop automaton C_E, i.e. $[\![E]\!] = [\![C_E]\!]$.*

7. Discussion

First, instead of iterating sum, one could iterate a max operator. As observed in Remark 2, this also yields to undecidability. Second, the decidability of synchronised expressions goes by the model weighted chop automata, which "chop" the input word into factors on which sub-automata are applied. Any synchronised expression can be converted into a synchronised chop automaton (Theorem 5). We conjecture that the converse of Theorem 5 is not true, i.e. synchronised WCA are strictly more expressive than synchronised expressions. In particular, we conjecture that synchronised expressions are not closed under split sum, unlike synchronised WCA (Proposition 11). The quantitative language of Example 5 does not seem to be definable by any synchronised expression.

It turns out that extending iterated-sum expressions with split sum $\odot$ and conditional choice $\triangleright$, with a suitable notion of synchronisation, gives a formalism equivalent to synchronised WCA. We were however not able to come up with an elegant and simple notion of synchronisation for such extended expressions, so we decided not to include it.

An expression formalism with unambiguous iterated sum, conditional choice and split sum, whose atoms are constant quantitative languages (any word from a regular language is mapped to a same constant value), was already introduced by Alur et al. [2]. It is shown that this formalism is equivalent to unambiguous WA. Our goal was to go much beyond this expressivity, by having a formalism closed under Presburger combinators. Adding such combinators to the expressions of [2] would immediately yield an undecidable formalism (as a consequence of Theorem 2). This extension would actually correspond exactly to the extension we discussed in the previous paragraph, and one could come up with a notion of synchronisation allowing decidability. We did not do it in this paper, for the reason explained before, but an interesting further direction would be to define a simple notion of synchronisation for the extension of [2] with Presburger combinators. More generally, our notion of synchronisation is semantical (but decidable). This raises the question of whether another weighted expression formalism with a purely syntactic notion of synchronisation could be defined.

Also, Proposition 7 highlights the fact that WCA, which are unambiguous by definition, are strictly more expressive than finitely ambiguous WA. It turns out that WCA can also define quantitative languages which seem to require exponential ambiguity of WA, for instance the function f of Example 2, given by $f(u_1 s_1 \dots u_n s_n) = \sum_{i=1}^n \max(\#_a(u_i), \#_b(u_i))$. As a matter of fact, we conjecture that WA and WCA have incomparable expressive power. We have shown in Proposition 2 that simple expressions (which are strictly less expressive than WCA) can express quantitative languages which are not definable by WA (over $(\mathbb{Z}, \max, +)$), for instance the function $w \mapsto \min(\#_a(w), \#_b(w))$. We conjecture that the following quantitative language "max prefix", which is trivially definable by some linearly ambiguous WA, is not definable by any WCA.

$$w \in \{a, b\}^* \mapsto \max\{\#_a(w') - \#_b(w') \mid w' \text{ prefix of } w\}$$

Indeed, and intuitively, chop automata decompose the input word unambiguously into independent, non-overlapping factors, which are themselves decomposed recursively. Implementing the function "max prefix" requires to consider linearly many overlapping factors (the prefixes). Of course, this function would be definable by an extension of WCA where the unambiguity condition is dropped, however this would yield a model strictly more expressive than WA and, as a consequence, undecidability (for the inclusion problem for instance).

Finally, Chatterjee et al. have introduced a recursive model of WA [7]. They are incomparable to weighted chop automata: they can define QL whose ranges are not semi-linear, but the recursion depth is only 1 (a master WA calls slave WA). Alur et al. have recently introduced a very general model of weighted automata for stream processing, which is hierarchical, as chop automata, and is parametrised by a set of aggregators [3]. Their goal was to provide efficient evaluation algorithms for their model. It could be interesting to see if notion of synchronisation could be defined on their model to obtain decidability results with respect to quantitative verification questions.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are very grateful to Ismaël Jecker and Nathan Lhote for fruitful discussions on this work, and for their help in establishing the undecidability result.

Appendix

Proof of Proposition 1. To prove that simple expressions define Lipschitz continuous functions, we need to show that for all simple expression E, there exists $k \in \mathbb{N}$ such that for all words $u, v \in \Sigma^*$:

$$|E(u) - E(v)| \leq k \cdot d(u, v) \tag{1}$$

Remember that $d(u, v) = |u| + |v| - 2|\sqcap(u, v)|$. We reason by induction on the structure of the simple expressions.

First, let us consider the base case where $E = D$. As D is deterministic, the partial sum on $u = w\dot{u}'$ and $v = w\dot{v}'$ on their common prefix $w = \sqcap(u, v)$ is equal in the two cases to some value s_w then on the two different suffixes u' and v', their sum may differ but at most by the following amount: $|u'| \times \ell + |v'| \times \ell$ where ℓ is the maximum of the set of absolute value of weights appearing in the automaton D. It is clear that the inequality 1 is true when we take $k = \ell$.

Second, we consider the operation min for the inductive case, i.e. $E = \min(E_1, E_2)$. The other operators are treated similarly. By induction hypothesis, E_1 and E_2 defines Lipschitz continuous functions, and we note k_1 and k_2 their respective Lipschitz constants. We claim that $k = \max(k_1, k_2)$ is an adequate constant to show the Lipschitz continuity of E, i.e.: for all words $u, v \in \Sigma^*$

$$|\min(E_1(u), E_2(u)) - \min(E_1(v), E_2(v))| \leq k \cdot d(u, v)$$

Let us assume that $\min(E_1, E_2)(u) = E_1(u)$ and $\min(E_1, E_2)(v) = E_2(v)$, and that $E_1(u) \geq E_2(v)$. Thus $|E_1(u) - E_2(v)| = E_1(u) - E_2(v)$. Furthermore we have $E_1(u) - E_2(v) \leq E_1(u) - E_1(v) \leq k_1 \cdot d(u, v) \leq \max(k_1, k_2) \cdot d(u, v)$. So finally, $\max(k_1, k_2) \cdot d(u, v) = k \cdot d(u, v)$. All the other cases are treated similarly. □

Proof of Proposition 2. We define an unambiguous WA A that realises f a function, which associates to any word of the form $a^{n_k}ba^{n_{k-1}}b \dots ba^{n_0}$, the value n_0, i.e. the length of the last block of a. Note that, f is non-Lipschitz continuous since for all K we have $|f(a^{K+1}) - f(a^{K+1}b)| > K \cdot d(a^{K+1}, a^{K+1}b)$. This establishes that simple expressions are not as expressive as finite valued WA (which extends unambiguous WA). When reading the first a or the first a after a b, A uses its non-determinism to guess whether this a belongs to the last block or not.

To prove the second statement, it was shown in [17] (Section 3.6) that the function $f : u \mapsto \min(\#_a(u), \#_b(u))$ is not definable by any WA. Clearly, $u \mapsto \#_\sigma(u)$ for $\sigma \in \{a, b\}$ is definable by a deterministic WA A_σ, hence f is definable by the simple expression $\min(A_a, A_b)$. □

Proof of Corollary 1. Let E be an monolithic expression. W.l.o.g we can assume that E is of the form $\phi(W_1, \dots, W_d)$ using Lemma 1. Since ϕ is a Presburger formula, the set of tuples T which satisfy it is semi-linear. We consider $W = W_1 \otimes \cdots \otimes W_d$ the synchronised product of the WA W_i (which is a WA valued in $\mathbb{Z}^d$). Applying Lemma 3, the range of W is semi-linear and can be effectively represented by a Presburger formula $\psi(x_1, \dots, x_d)$. Finally, the range of E is defined by $\exists y, \phi(x_1, \dots, x_d, y) \wedge \psi(x_1, \dots, x_d)$. □

Proof of Lemma 4. Let A_1, A_2 be two NFA recognising L_1, L_2 respectively. We can construct an automaton A that accepts the words in L_1L_2 which admits at least two different factorisations with respect to L_1, L_2. It suffices to simulate two runs of A_1 in parallel, whenever a copy of A_1 goes to an accepting state, this copy either stays in A_1 or, thanks to an added

ε-transition, goes to some initial state of A_2. We also add one bit of memory to check that ε-transitions have been taken at two different moments in the two simulated runs. The accepting states are states $(q_2, q_2', 1)$ where q_2, q_2' are accepting states of A_2. By complementing A, we obtain an automaton, say B, recognising $L_1 \odot L_2$.

Now, we make a product between B and a disjunct union between A_1 and A_2. If Q are the states of B, Q_1 of A_1, Q_2 of A_2 (with initial states I_2), the set of states of this product is $Q \times (Q_1 \uplus Q_2 \uplus \overline{I_2})$ where $\overline{I_2}$ is a copy of I_2. B initially runs in parallel of A_1 and, when A_1 enters an accepting state, i.e. the product is in state (q, q_1) where $q_1 \in F_1$, then we add some ε-transition to any state $(q, \overline{q_2})$ where $q_2 \in I_2$. Then, from states of this form, the product continues its simulation of B and simulates in parallel A_2 (in normal states Q_2, so that the copy $\overline{I_2}$ is only met once, when the product switches to A_2). Let denote by $B \boxtimes (A_1 A_2)$ this product. We set its accepting states to be any pair (q, q_2) or $(q, \overline{q_2})$ where q_2 and q are accepting.

For all states (q, p) of $B \boxtimes (A_1 A_2)$, we denote by $L_{q,p}$ and $R_{q,p}$ the left language of (q, p) and the right language of (q, p) respectively. We claim that

$$L_1 \odot L_2 = L(B) = \bigcup_{(q,\overline{q_2}) \in Q \times \overline{I_2}} L_{q,q_2} R_{q,q_2}$$

Clearly, $\bigcup_{(q,\overline{q_2}) \in Q \times \overline{I_2}} L_{q,q_2} R_{q,q_2} \subseteq L(B)$ since the product also checks that the input words are accepted by B. Conversely, if $u \in L(B)$, then it is uniquely decomposed into $u_1 u_2$ where $u_i \in L(A_i)$. From $r = q_1 \dots q_{n+1} p_1 \dots p_{m+1}$ an accepting run on u (where $n = |u_1|$ and $m = |u_2|$), an accepting run $r_1 = \alpha_1 \dots \alpha_{n+1}$ of A_1 on u_1, and a accepting run $r_2 = \beta_1 \dots \beta_{m+1}$ of A_2, we can construct the following accepting run of $B \boxtimes (A_1 A_2)$:

$$(q_1, \alpha_1) \dots (q_{n+1}, \alpha_{n+1})(p_1, \overline{\beta_1})(p_2, \beta_2) \dots (p_{m+1}, \beta_{m+1}) \qquad \square$$

Proof details of Lemma 5. We prove the following statements:

1. N is regular
2. $\mathrm{Range}(C) = \mu(N)$

1. Let $A_{p,q}$ be the NFA recognising $\Delta(p, q)$. It is simple to combine the automata $A_{p,q}$ such that $\Delta(p, q) \neq \varnothing$ into a single automaton recognising N. For instance, one can take the disjoint union of all $A_{p,q}$ such that $\Delta(p, q) \neq \varnothing$, add the states of A, with I the set of initial states and F the set of final states, and add the following ε-transitions: from any state $p \in Q$, add ε-transitions to the initial states of any NFA $A_{p,q}$, and from any final state of any automaton $A_{p,q}$, add some ε-transition to q.
2. $\subseteq$: Let $\overline{x} \in \mathrm{Range}(C)$. Hence, there exists $u \in \mathrm{dom}(C)$ such that $[\![C]\!](u) = \overline{x}$. Let $p_1 \xrightarrow{u_1} p_2 \xrightarrow{u_2} p_3 \dots p_k \xrightarrow{u_k} p_{k+1}$ be the accepting run of A and u, i.e. $u_i \in \Delta(p_i, p_{i+1})$ for all $i = 1, \dots, k$. We have $\alpha = (p_1, p_2)(p_2, p_3) \dots (p_k, p_{k+1}) \in N$. By the semantics of WCA, $\overline{x} = \overline{x_1} + \dots + \overline{x_k}$ where $\overline{x_i} = [\![\lambda(p_i, p_{i+1})]\!](u_i)$ for all $i = 1, \dots, k$. Hence $\overline{x_i} \in S_{p_i, p_{i+1}}$ and $\overline{x} \in \mu(\alpha) \subseteq \mu(N)$.
   $\supseteq$: Let $\overline{x} \in \mu(N)$. There exists $(p_1, p_2) \dots (p_k, p_{k+1}) \in N$ such that $\overline{x} \in S_{p_1,p_2} + \dots + S_{p_k,p_{k+1}}$, by definition of μ and N. Hence $\overline{x} = \sum_{i=1}^{k} \overline{x_i}$ for some $\overline{x_i} \in S_{p_i,p_{i+1}}$. By definition of the $S_{p_i,p_{i+1}}$, there exists $u_1, \dots, u_k \in \Sigma^*$ such that $u_i \in \Delta(p_i, p_{i+1})$ and $\overline{x_i} = [\![\lambda(p_i, p_{i+1})]\!](u_i)$. Moreover, by definition of N, p_1 is initial and p_{k+1} is final, hence $u_1 \dots u_k \in \mathrm{dom}(C)$, and by the semantics of C, $[\![C]\!](u) = \sum_i \overline{x_i}$, i.e. $\overline{x} \in \mathrm{Range}(C)$. $\square$

References

[1] S. Almagor, U. Boker, O. Kupferman, What's decidable about weighted automata, in: ATVA'11, 2011, pp. 482–491.
[2] R. Alur, A. Freilich, M. Raghothaman, Regular combinators for string transformations, in: CSL, 2014, pp. 1–10.
[3] R. Alur, M. Raghothaman, Decision problems for additive regular functions, in: ICALP, 2013, pp. 37–48.
[4] M.-P. Béal, O. Carton, C. Prieur, J. Sakarovitch, Squaring transducers: an efficient procedure for deciding functionality and sequentiality, TCS 292 (1) (2003).
[5] K. Chatterjee, L. Doyen, H. Edelsbrunner, T.A. Henzinger, P. Rannou, Mean-payoff automaton expressions, in: CONCUR, 2010, pp. 269–283.
[6] K. Chatterjee, L. Doyen, T.A. Henzinger, Quantitative languages, ACM Trans. Comput. Log. 11 (4) (2010).
[7] K. Chatterjee, T.A. Henzinger, J. Otop, Nested weighted automata, in: LICS, 2015.
[8] L. Daviaud, P. Guillon, G. Merlet, Comparison of max-plus automata and joint spectral radius of tropical matrices, CoRR, arXiv:1612.02647, 2016.
[9] M. Droste, P. Gastin, Weighted automata and weighted logics, Theor. Comput. Sci. 380 (2007) 69–86.
[10] M. Droste, W. Kuich, H. Vogler, Handbook of Weighted Automata, 2009.
[11] S. Eilenberg, M.-P. Schützenberger, Rational sets in commutative monoids, J. Algebra 13 (1969) 173–191.
[12] D. Figueira, L. Libkin, Path logics for querying graphs: combining expressiveness and efficiency, in: 30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015, 2015, pp. 329–340.
[13] E. Filiot, R. Gentilini, J.-F. Raskin, Finite-valued weighted automata, in: FSTTCS, 2014, pp. 133–145.
[14] E. Filiot, R. Gentilini, J.-F. Raskin, Quantitative languages defined by functional automata, LMCS 11 (3) (2015).
[15] E. Filiot, N. Mazzocchi, J.-F. Raskin, Decidable weighted expressions with Presburger combinators, in: FCT, 2017, pp. 11–13.
[16] E.M. Gurari, O.H. Ibarra, The complexity of decision problems for finite-turn multicounter machines, ICALP (1981) 495–505.
[17] I. Klimann, S. Lombardy, J. Mairesse, C. Prieur, Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton, TCS 327 (3) (2004).

[18] F. Klaedtke, H. Rueß, Monadic second-order logics with cardinalities, ICALP (2003) 681–696.
[19] D. Krob, The equality problem for rational series with multiplicities in the tropical semiring is undecidable, Int. Jour. of Alg. and Comp. 4 (3) (1994) 405–425.
[20] A.W. Lin, Model Checking Infinite-State Systems: Generic and Specific Approaches, PhD thesis, U. Edinburgh, 2010.
[21] B. Scarpellini, Complexity of subcases of presburger arithmetic, Transactions of the American Mathematical Society 284 (1984) 203–218.
[22] Y. Velner, The complexity of mean-payoff automaton expression, in: ICALP, 2012.