Finite-Valued Weighted Automata*

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

1 **Université Libre de Bruxelles**
2 **Universitá di Perugia**

Abstract

Any weighted automaton (WA) defines a relation from finite words to values: given an input word, its set of values is obtained as the set of values computed by each accepting run on that word. A WA is k-valued if the relation it defines has degree at most k, i.e., every set of values associated with an input word has cardinality at most k. We investigate the class of quantitative languages defined by k-valued automata, for all parameters k. We consider several measures to associate values with runs: sum, discounted-sum, and more generally values in groups.

We define a general procedure which decides, given a bound k and a WA over a group, whether this automaton is k-valued. We also show that any k-valued WA over a group, under some general conditions, can be decomposed as a union of k unambiguous WA. While inclusion and equivalence are undecidable problems for arbitrary sum-automata, we show, based on this decomposition, that they are decidable for k-valued sum-automata, and k-valued discounted sum-automata over inverted integer discount factors. We finally show that the quantitative Church problem is undecidable for k-valued sum-automata, even given as finite unions of deterministic sum-automata.

1 Introduction

Finite-state acceptor machines have found many applications in computer science. One of the most famous and studied example is the class of finite-state automata, which enjoys good algorithmic and closure properties. For instance, the decision problems of inclusion and equivalence, are decidable for finite-state automata. Finite-state automata and some of their variants have been successfully applied, for instance, to the theory of model-checking [6]. Their behaviour is however purely *Boolean* (either they accept their input or not). In many applications, this abstraction is not sufficient to accurately model systems where *quantitative* aspects are important. Weighted automata (WA) have been introduced to overcome this modelling weakness, as they can define *quantitative languages*, i.e. functions from words to values in some arbitrary set. WA have been studied for long [9] but recently, applications in computer-aided verification have been considered and new theoretical questions have been addressed [3]. In contrast with finite-state automata, the inclusion problem is however undecidable for some classes of WA, such as automata over the tropical semiring. One of the main goal of this paper is to recover decidability for some expressive classes of WA.

Weighted automata (on finite words) extend finite-state automata with values which, in general, are taken in a semiring $(S, \oplus, \otimes)$. In this paper however, we focus on sum (weighted automata over the tropical semiring) and discounted sum automata, but generalise our results whenever it can be done. The value of an accepting run of a sum (resp. discounted sum) automaton A on a finite word is the (left-to-right) sum (resp. discounted sum for a discount factor $\lambda \in]0, 1[$) of all the values occurring along that run. The value $A(w)$ of a word w is defined only if there exists an accepting run on w, as the maximum of all the values of the

* This work is partially supported by FNRS (Mobilité 2014) and by INdAM-GNCS (Project 2014). We thank the anonymous reviewers for their careful reading of our manuscript and their many insightful comments and suggestions.

accepting runs on w. As an example, consider the quantitative language L which associates with each word w over the alphabet $\{a, b\}$, the value $L(w) = max(\#_a(w), \#_b(w))$, where $\#_x(w)$ is the number of occurrences of x in w. This language L can be defined by a sum automaton A defined as the union of two disjoint deterministic WA A_a and A_b: For all $x \in \{0, 1\}$, A_x has a single (accepting and initial) state with a 1-weighted loop on reading x, and a 0-weighted loop on reading the other symbol.

The (quantitative) *inclusion problem* generalizes the classical inclusion problem for Boolean languages. It is the problem of deciding, given two WA A and B, whether for all words u, if u is accepted by A, then it is accepted by B and $A(u) \leq B(u)$. Even for the class of sum automata, this problem is know to be undecidable [15]. In [10], we have introduced the class of *functional WA*, i.e. WA such that every accepting runs on the same input word have same value. We have shown, for several measures (sum, discounted sum, and ratio), that this class is decidable, and has decidable inclusion problem.

Contributions In this paper, we generalise the class of functional WA to k-valued WA i.e., WA such that, for all input words w, the set of values computed by all the accepting runs on w has cardinality at most k. For instance, the WA A defining the language L is k-valued for all $k \geq 2$. We show, for the measures sum and discounted sum (over inverted integer discount factor $1/n$), that they have decidable inclusion (and therefore decidable equivalence). Given $k \in \mathbb{N}$ and a sum-automaton (resp. a discounted-sum automaton) A, we prove that checking whether A is k-valued is decidable. We also show that if A is k-valued, then it can be decomposed as finite union of functional (i.e. 1-valued) sum-automata (resp. discounted-sum automata).

The last two results are more generally shown for *group automata (GA)*. Group automata extend finite state automata with weights over an *infinitary* group, i.e. a group $(G, \otimes)$ that satisfies some condition called the infinitary condition. This condition implies that two runs of a WA, on the same input, that synchronize on loop with different delays (i.e. the difference between their output values before and after taking the loop are non equal), can generate infinitely many different delays when iterating that loop. The semantics of a group automaton is a function from finite words w to the set of values (in G) of all the accepting runs on w. It is k-valued if any set of values associated with w has cardinality at most k. As we show, sum- and discounted-sum automata can be treated as group automata, as far as the max operation that combines the values of a word is not relevant. Real-time string-to-string transducers are also group automata (over the free group generated by the output alphabet), as they satisfy the infinitary condition. Therefore, our results (decidability of k-valuedness and decomposition) do not only apply to sum- and discounted-sum automata, but more generally to group automata, and as a particular case, we recover some results that were already known for string transducers. We now detail our contributions:

k-valuedness problem First, we show that given $k \in \mathbb{N}$ and a GA A, checking whether A is k-valued is decidable. The proof of decidable k-valuedness for GA relies on a pumping argument that allows us to bound the size of non k-valuedness witnesses. Applied to our measures, this general procedure for testing k-valuedness is in PSpace for both sum and discounted sum automata (seen as GA), when k is part of the input. The k-valuedness problem is shown to be PSpace-hard for sum automata, when k is part of the input. When k is fixed, we also show that testing k-valuedness can be done in PTime for sum automata, based on PTime complexity result for checking the existence of a zero-circuit in a multi-weighted graph [17]. Still for a fixed parameter k, we give another general k-valuedness checking procedure for GA which, applied to discounted sum, provides us with a PTime complexity. Applied to sum, this also yields PTime complexity with a worst constant than

the ad-hoc PTime procedure. This general procedure extends to GA a procedure that was defined in [18] to decide k-valuedness of string-to-string transducers. Besides generalizing the procedure of [18], we show here how to simplify it and provide a simpler correctness proof based on the small witness property for non k-valuedness.

Decomposition of k-valued group automata We show that any k-valued GA is equivalent to a union of k functional GA. Our decomposition technique non-trivially extends to GA a procedure that has been introduced for string-to-string transducers [19].

Inclusion, equivalence, and synthesis problems Based on the decomposition result, we give a general scheme to decide inclusion of two k-valued sum (resp. discounted sum) WA. This scheme reduces the inclusion problem to checking the existence of a path with strictly positive sum (resp. discounted sum) on all dimensions of a multi-weighted graph. We show that this latter problem is decidable for sum (in PTime), and discounted sum (in PSpace) with inverted discount factors $1/n$, $n \in \mathbb{N}$. Therefore, it yields decidable inclusion and equivalence problems for these two classes of k-valued WA. If the two WA are given as unions of deterministic automata and known to have included domains, this procedure is PTime for sum automata, and PSpace for discounted sum automata. For general rational discount factors, the problem is still open and related to other open problems identified in, e.g., [4] and [2]. Finally, we consider the quantitative synthesis problem: the alphabet is partitioned into two sets that are controlled respectively by two players. The winning objective of the protagonist is given by a sum automaton A. The problem is to decide whether the protagonist has a strategy to choose his letters such that whatever letters the opponent chooses, the outcome of their interaction (which is a word) has strictly positive value by A. This problem was shown to be undecidable for functional sum-automata and decidable for deterministic sum-automata in [10]. Here, we show that it is undecidable for the union of p deterministic sum-automata, for $p \geq 4$.

Related Works Comparisons with [10, 18, 19] have already been mentioned before. The notion of k-valuedness originates from the theory of string-to-string transducers. For general (non-deterministic) transducers, inclusion and equivalence are undecidable [11], but decidable for k-valued transducers [12, 21, 7]. This latter result has then been extended to tree transducers [20]. The k-valuedness problem for string-to-string transducers has been shown to be decidable in several papers [12, 18, 21, 8], in PTime when k is fixed. Any sum automaton A can be transformed into a string-to-string transducer T_A over a unary output alphabet, by adding m to all the weights of A (where m is the minimal weight occurring on the transitions of A). Then A is k-valued iff T_A is. While this encoding allows us to reuse existing results on string-to-string transducers, the complexity results are not optimal, because the weights of A are encoded in binary, and their translation into strings is unary. Moreover, there is no such encoding for discounted sum automata and therefore we rather give general procedures at the level of infinitary groups.

Sum-automata over $\mathbb{N}$ (called distance automata) have been considered in several papers, and known results on the distance problem (deciding whether there exists an upper bound on the value of every input word) are nicely summarised in [22]. In particular in [22], the class of finite-valued distance automata is considered. Finite-valuedness (deciding whether there exists k such that the distance automaton is k-valued) is shown to be decidable by a direct reduction to the string transducer case, and therefore the complexity bound is not optimal when the weights are encoded in binary. The finite-valuedness problem is not adressed in this paper. We leave it however as future work and it seems, again, that the technique of [19] would nicely generalise to infinitary groups.

Finitely ambiguous sum automata are sum automata such that there exists a bound b

on the number of accepting runs on the same input word. They are known to be equivalent to union of unambiguous sum automata [13, 14, 19, 22]. The best bound on the size of the union (which matches exactly the degree of ambiguity of the automaton) is obtained in [19]. Our decomposition of k-valued WA as unions of unambiguous automata directly uses these results as an intermediate step. In [13], it is shown that the equivalence problem for finitely ambiguous sum-automata is decidable. There is however no precise complexity result.

Finally, let us mention that for the strictly positive discounted sum problem on multi-weighted graphs, if one requires the discounted sum to be greater than or equal to 0 on all the dimensions, then it corresponds to an open problem, which is at least as difficult as the exact value (open) problem in 1-dimensional graph (i.e. decide in a weighted graph, whether there exists a path with discounted sum exactly 0), as shown in [4].

2 Weighted Automata

Let W be a set (called weight-set) and Σ be a finite alphabet. An *automaton* over Σ and W is a tuple $A{=}(Q, q_I, F, \delta, \gamma)$ where Q is a finite set of states, F is a set of final states, $q_I \in Q$ is the initial state, $\delta \subseteq Q{\times}\Sigma{\times}Q$ is the transition relation, and $\gamma : \delta \to W$ is an edge-labelling function, mapping δ onto a weight-set W. A run ρ of A over a word $w = \sigma_0 \dots \sigma_n \in \Sigma^*$ is a sequence $\rho = q_0\sigma_0q_1 \dots \sigma_nq_{n+1}$ such that $q_0 = q_I$ and for all $i \in \{0, \dots, n\}$, $(q_i, \sigma_i, q_{i+1}) \in \delta$. We write $\rho : q_0 \xrightarrow{w|m} q_{n+1}$ to denote that ρ is a run on w starting at q_0 and ending in q_{n+1}, where $m = \gamma(q_0, \sigma_0, q_1) \cdot \gamma(q_1, \sigma_1, q_2) \cdot \dots \cdot \gamma(q_n, \sigma_n, q_{n+1})$. We write $|\rho|$ to denote the length of ρ, where $|\rho| = n + 1$. The run ρ is *accepting* if $q_{n+1} \in F$. A state q in $A = (Q, q_I, F, \delta, \gamma)$ is said *co-accessible* (resp. *accessible*) by some word $w \in \Sigma^*$ if there exists some run $\rho : q \xrightarrow{w|m} q_f$ for some $q_f \in F$ (resp. some run $\rho : q_I \xrightarrow{w|m} q$). If such a word exists, we say that q is co-accessible (resp. accessible). A is said to be *trim* if all its states are both accessible and co-accessible. It is well-known that an automaton can be trimmed in polynomial time [9].

A pair of states (q, q') is co-accessible if there exists a word w such that q and q' are co-accessible by w. The domain of A, denoted by $\mathrm{dom}(A)$, is defined as the set of words $w \in \Sigma^+$ on which there exists some accepting run of A. Note that (Q, q_I, F, δ) is a classical finite state automaton over Σ. We say that A is *deterministic* if (Q, q_I, F, δ) is deterministic. We say that A is *unambiguous* if (Q, q_I, F, δ) admits at most one accepting run for each word.

Give a run $\rho = q_0\sigma_0 \dots \sigma_nq_{n+1}$ and $0 \le \ell \le n + 1$, we define $\rho[\ell] = q_\ell$, $\rho[{:}\ell] = q_0\sigma_0 \dots \sigma_{\ell-1}q_\ell$ and $\rho[\ell{:}] = q_\ell\sigma_\ell \dots q_{n+1}$. Given a run $\rho' = q'_0\sigma_0 \dots \sigma_nq'_{n+1}$ on the same input as ρ, we define the *synchronized product* $\rho \otimes \rho'$ as the sequence $(q_0, q'_0)\sigma_0 \dots \sigma_n(q_{n+1}, q'_{n+1})$. A pair (ℓ_1, ℓ_2) such that $0 \le \ell_1 < \ell_2 \le n + 1$ is called a *cycle* (or loop) in ρ if $\rho[\ell_1] = \rho[\ell_2]$. The synchronized operator $\otimes$ can be inductively extended to sequences of runs as follows: $\rho_1 \otimes \dots \otimes \rho_m = (\rho_1 \otimes \dots \otimes \rho_{m-1}) \otimes \rho_m$. A cycle (ℓ_1, ℓ_2) in $\rho_1 \otimes \dots \otimes \rho_m$ is sometimes called a *synchronizing cycle* to emphasise that we consider product of runs.

If W is equipped with an operation $\cdot$, we define the *value* $V(\rho)$ of a run $\rho = q_0\sigma_0q_1 \dots \sigma_nq_{n+1}$ in $A{=}(Q, q_I, F, \delta, \gamma)$ as: $V(\rho) = \gamma(q_0, \sigma_0, q_1) \cdot \gamma(q_1, \sigma_1, q_2) \cdot \dots \cdot \gamma(q_n, \sigma_n, q_{n+1})$ if ρ is accepting and $V(\rho) = \bot$, otherwise[1]. The relation $R_A = \{(w, V(\rho)) \mid w \in \Sigma^+, \rho \text{ is a run of } A \text{ on } w\}$ is called[2] the *relation induced by A*. For all $k \in \mathbb{N}$, R_A is k-valued if for all words $w \in \Sigma^+$, we

[1] Here $\bot \notin W$ is a fresh symbol used to represent the fact that the value of ρ undefined

[2] As in [3], we do not consider the empty word as our weighted automata do not have initial and final weight functions. This eases our presentation but all our results carry over to the more general setting with initial and final weight function [9].

have $|\{v \mid (w,v) \in R_A, v \neq \bot\}| \leq k$. In this case we say that A is *k-valued* (and functional if $k = 1$).

In this paper we consider weight sets having the algebraic structure of a group $(W, \cdot, \mathbb{1})$. Recall that a group is a structure $(S, \cdot, \mathbb{1})$, where S is a set, $\cdot : S \times S \to S$ is an associative operation, $\mathbb{1} \in S$ is a two sided identity element for $\cdot$ over S, and each element $s \in S$ admits an inverse $s^{-1} \in S$, such that $s^{-1} \cdot s = s \cdot s^{-1} = \mathbb{1}$ (the inverse is unique).

Functional automata over groups where studied in [10], where a polynomial functionality test was given based on the notion of *delay* between two runs [5, 9] (cfr. Definition 1). Moreover, [10] provided a determinization test for automata over *infinitary groups*, i.e. groups satisfying the *infinitary condition* (c.f.r. Definition 2, below). Intuitively, the infinitary condition ensures that iterating two runs on a parallel loop induces infinitely many delays.

▶ **Definition 1** (Delay). Let $A = (Q, q_I, F, \delta, \gamma)$ be an automaton over a weight-set group $(W, \cdot, \mathbb{1})$. Given two elements $d_1, d_2 \in W$, the *delay* between d_1 and d_2 is defined by $\mathsf{delay}(d_1, d_2) = d_1^{-1} \cdot d_2$. Let $p, q \in Q$. A value $d \in W$ is a *delay* for (p, q) if A admits two runs $\rho : q_I \xrightarrow{w} p$, $\rho' : q_I \xrightarrow{w} q$ on some $w \in \Sigma^*$ s.t. $\mathsf{delay}(\rho, \rho') =_{def} \mathsf{delay}(V(\rho), V(\rho')) = d$.

▶ **Definition 2** (Infinitary Condition). A group $(W, \cdot, \mathbb{1})$ is said *infinitary* if it satisfies the *infinitary condition* stating that for all $v_1, w_1, v_2, w_2 \in W$, if $v_1^{-1} w_1 \neq v_2^{-1} v_1^{-1} w_1 w_2$, then:

$$|\{v_2^{-h} v_1^{-1} w_1 w_2^h \mid h \geq 0\}| = \infty$$

A *group automaton (GA)* over Σ, is an automaton over Σ and an infinitary group. A *weighted automaton (WA)* over Σ is an automaton over Σ and a semiring (we refer the reader to [9] for a definition of semiring). If A is a weighted automaton over Σ and a semiring $(W, \oplus, \cdot)$, the *quantitative language* L_A defined by A is a mapping $L_A : \Sigma^* \to W \cup \{\bot\}$, where $\bot \notin W$, defined for all $w \in \Sigma^*$ by $L_A(w) = \bot$ if $w \notin \mathrm{dom}(A)$, and $L_A(w) = \bigoplus R_A(w)$ otherwise. Note that any weighted automaton over a semiring $(W, \oplus, \cdot)$ such that can be seen as a group automaton, provided $(W, \cdot, \mathbb{1})$ is an infinitary group, where $\mathbb{1}$ is the neutral element for $\cdot$.

Remarkable subclasses of WA, such as e.g. Sum- and Dsum-automata [3], can be seen as group automata, as outlined in the following remark (cfr. also Proposition 1).

▶ Remark (Sum- and Dsum-automata). Sum- (resp. Dsum-) automata are usually defined as WA $A = (Q, q_I, F, \delta, \gamma)$, where the weight set $\gamma : \delta \to \mathbb{Z}$ maps edges onto integers and the value of a run ρ is simply the *sum* (resp. *discounted sum*) of the weights along its edges. More precisely, given a run $\rho = q_0 \sigma_0 q_1 \ldots \sigma_n q_{n+1}$ in a Sum-automaton A, $V(\rho)$ is defined by $V(\rho) = \sum_{i=0}^{i=n} \gamma(q_i, \sigma_i, q_{i+1})$ if ρ is accepting, and by $V(\rho) = \bot$ otherwise. If A is a Dsum-automaton with discount factor $\lambda \in]0, 1[$, then $V(\rho)$ is defined by $\sum_{i=0}^{i=n} \lambda^i \gamma(q_i, \sigma_i, q_{i+1})$ if ρ is accepting, and by $\bot$ otherwise (we assume that $\bot < m$ for all $m \in \mathbb{Z}$).

Clearly, Sum-automata can be seen as group automata over the group $(\mathbb{Z}, +, 0)$. For Dsum-automata, consider the group $(W, \cdot, \mathbb{1})$, where $W = \mathbb{Q}^2$, $\cdot$ is defined by $(a, x) \cdot (b, y) = (\frac{1}{y} a + b, xy)$, $\mathbb{1} = (0, 1)$, and given $(a, x) \in W$, the inverse $(a, x)^{-1}$ is given by $(a, x)^{-1} = (-xa, x^{-1})$. Given $\lambda \in \mathbb{Q} \cap]0, 1[$, a Dsum-automaton A on Σ can be seen as a group automaton over $(W, \cdot, \mathbb{1})$, by replacing each weight a in A with the pair $(a, \lambda), a \in \mathbb{Z}$. Let $w = w_0 \ldots w_n \in \Sigma$, and consider a run $\rho : q_0 \xrightarrow{w} q_{n+1}$ on w in A. Then, ρ is valued by the pair $(a, x) = (\frac{1}{\lambda^n} \gamma(q_0, w_0, q_1) + \cdots + \gamma(q_n, w_n, q_{n+1}), \lambda^{n+1})$. Hence, (a, x) codes the value $\frac{ax}{\lambda} = \mathsf{Dsum}(\rho)$. It can be shown that groups involved in the above definitions satisfy the infinitary condition.

▶ Proposition 1. Sum and Dsum automata can be coded as group automata (over an infinitary group).

3 The k-Valuedness Problem

In this section, we consider the problem of determining whether a group automaton (for fixed and unfixed k). As a first result, we provide a pumping argument that can be turned into a simple k-valuedness test.

▶ **Lemma 3** (Pumping). *Let $A = (Q, q_I, F, \delta, \gamma)$ be a group automaton over $(W.\cdot, \mathbb{1})$. A is not k-valued iff there exists a word $w \in \Sigma^*$ having length $|w| \leq |Q|^{k+1} + |Q|^{k+1} \cdot (\frac{k(k-1)}{2})^2$ such that A admits $k+1$ accepting runs with pairwise distinct values on w.*

Lemma 3 yields immediately a decidability procedure for deciding if a given group automaton is k-valued: It is sufficient to guess $k+1$ runs of bounded length and check if they are accepting and the outputs are pairwise distinct. Such a procedure turns out to be PSPACE if the accumulated values along the guessed runs can be stored using polynomial space. In particular, this is the case for the classes of Sum- and Dsum-automata. Therefore,

▶ **Theorem 4.** *The k-valuedness problem is decidable for the class of group automata. When k is part of the input, it is PSPACE-complete on the subclass of* Sum*-automata and PSPACE-easy on the subclass of* Dsum*-automata.*

▶ Remark. Lemma 3 can be put in parallel with Theorem 3 in [8], establishing that k-valuedness for string to string transducers can be decided by analizing the image of a finite set of words, namely those having length $|Q|^{k+1}F(k)$, where $F(k)$ in [8] is an exponential function expressed in terms of a recurrence relation.

3.1 The k-Valuedness Problem when k is a Fixed Constant

In this subsection, we study the k-valuedness problem on group automata assuming k as a fixed constant. This assumption does not lower the complexity of the pumping based procedure of the latter section (for both measures), however in this section we give other, new procedures which provide better upper bounds for a fixed k. These new procedures, for a non-fixed k, have worse complexities than the pumping based procedure (Exptime).

The Case of Sum-Automata Theorem 6 below shows that Sum-automata can be tested for k-valuedness in PTIME for fixed k. The polynomial k-valuedness algorithm designed within Theorem 6 relies onto a reduction to the problem of finding a strictly positive path (from the initial state to a final state) in a multi-weighted graph.

A k-weighted graph $G = (Q, q_I, E, F, w)$ is simply a weighted graph, where Q is the set states, q_I is the inital state, F is the set of final states, $E \subseteq Q \times Q$ is the edge relation and $w : E \mapsto \mathbb{Z}^k$ assigns to each edge a vector of integers of dimension k. A path $\pi = q_0, \ldots, q_n$ in G is strictly positive if and only if $\sum_{i=0}^{n-1} w(q_i, q_{i+1}) > 0_{\mathbb{Z}^k}$. Lemma 5 below shows that deciding whether a k-weighted graph G contains a strictly positive path to a final state can be done in polynomial time.

▶ **Lemma 5.** *Determining if a k-weighted graph contains a strictly positive path from the inital state to a final state can be done in polynomial time.*

Intuitively, the multi-weighted graph G built in Theorem 6 to decide if a given Sum-automaton A is k-valued has dimension k. The vertices of G are $(k+1)$-tuples of states of A and G admits an edge e from $(q_1, \ldots, q_{k+1})$ to $(p_1, \ldots, p_{k+1})$ if and only if there exists $a \in \Sigma$ such that for all $1 \leq i \leq k+1$ $q_i \xrightarrow{a|n_i} p_i$ is an edge in A. Moreover the weight $w(e)$ of e is given by $(n_2 - n_1, n_3 - n_2, \ldots, n_{k+1} - n_k)$. Therefore, A admits $k+1$

accepting runs $q_I \xrightarrow{w|v_1} p_1, \dots, q_I \xrightarrow{w|v_{k+1}} p_{k+1}$ on $w \in \Sigma^*$ with pairwise distinct values $v_1 < v_2 < \dots < v_{k+1}$ if and only if the associated multi-weighted graph G admits a strictly positive path to $(p_1, \dots, p_{k+1})$.

▶ **Theorem 6.** *The k-valuedness problem on* Sum*-automata can be solved in polynomial time, when k is a fixed constant.*

▶ Remark. Note that for string transducers the k-valuedness problem can be reduced in polynomial time to the emptiness problem on one reversal nondeterministic multicounter machines[3] [12]. Although a similar reduction could be easily designed for Sum-automata, the overall procedure for testing k-valuedness would be only pseudopolynomial, as it turns out that emptiness of multicounter one-reversal machines where counters can be incremented by arbitrary constants is NP-complete.

The Case of Group Automata In this subsection we show that the whole class of group automata can be tested for k-valuedness in polynomial time, assuming that k is an input fixed constant. The main ingredients of the polynomial test designed are the same as the ones used for the k-valuedness test on string transducers in [18], generalized from strings to groups. However, the pumping result stated in the previous section allows to significatively simplify the overall procedure of [18]. The first ingredient that we will use is the notion of *pairwise delays*, that is a natural extension of the concept of delay between two runs at the core of the functionality tests in [18]

▶ **Definition 7** (k-Pairwise Delay)**.** Let $(W, \cdot, \mathbb{1})$ be an infinitary group, let $k \in \mathbb{N}$. A k-pairwise delay on W is a function $\delta : \{\langle i, j\rangle \mid 1 \le i < j \le k\} \to W$.

We denote by $\Delta_k(\mathcal{G})$ the set of all k-pairwise delays on $\mathcal{G} = (W, \cdot, \mathbb{1})$ (omitting $\mathcal{G}$ and/or k when the latter are clear from the context).

▶ **Definition 8.** A group automaton $A = (Q, q_I, F, \delta, \gamma)$ on $(W, \cdot, \mathbb{1})$ admits a k-pairwise delay δ on $(q_1, \dots, q_k) \in Q^k$ iff there exists $w \in \Sigma^*$ such that for each pair $\langle i, j\rangle, 1 \le i < j \le k$, A admits two runs $q_I \xrightarrow{w|n} q_i$, $q_I \xrightarrow{w|m} q_j$ such that $n^{-1} \cdot m = \delta(\langle i, j\rangle)$. In this case, we say that w witnesses δ on $(q_1, \dots, q_k)$.

Given $A = (Q, q_I, F, \delta, \gamma)$ on $\mathcal{G} = (W, \cdot, \mathbb{1})$, and $k, i \in \mathbb{N}$, we define the function $D^i_k : Q^k \mapsto 2^{\Delta_k(\mathcal{G})}$ where $D^i_k(q_1, \dots, q_k)$ is the set of k-pairwise delay on $(q_1, \dots, q_k)$ witnessed by some word w of length $|w| \le i$. Our pumping result ensures the existence of a bound b, polynomial in $|Q|$, such that D^b_{k+1} retains enough information to decide whether A is k-valued:

▶ **Lemma 9.** *Let $A = (Q, q_I, F, \delta, \gamma)$ be a group automaton over $(W, \cdot, \mathbb{1})$, let $b = |Q|^{k+1} + |Q|^{k+1} . (\frac{k(k-1)}{2})^2$. A is not k-valued if and only if it admits a tuple of final states $(q_1, \dots, q_{k+1}) \in F^{k+1}$ such that $D^b_{k+1}(q_1, \dots, q_{k+1})$ contains a $(k+1)$-pairwise delay δ satisfying the following condition: for all $0 \le i < j \le k+1$, $\delta(\langle i, j\rangle) \ne \mathbb{1}$.*

Unfortunately, given $(q_1, \dots, q_{k+1}) \in Q^{k+1}$, $D^b_{k+1}(q_1, \dots, q_{k+1})$ could contain exponentially many delays w.r.t. b, since each word w admits $\mathcal{O}(2^{|w|})$ runs on it. However, such delays can be properly abstracted using a generalization from strings to groups of the very powerful and elegant notion of *minimal traverse*, introduced in [18] to give a compact representation of the delays generated by a k-valued string transducer. Formally,

[3] The emptiness problem of reversal-bounded multicounter machines can be solved in polynomial time [12]

▶ **Definition 10** (k-Pairwise Partial Delay)**.** Let $(W, \cdot, \mathbb{1})$ be an infinitary group, let $k \in \mathbb{N}$. A k-pairwise partial delay on W is a function $\delta : \{\langle i, j\rangle \mid 1 \leq i < j \leq k\} \to W \cup \perp$.

We denote by $\Delta_k^{\perp}(\mathcal{G})$ the set of all k-pairwise partial delays on $\mathcal{G} = (W, \cdot, \mathbb{1})$. Given $\tau, \tau' \in \Delta_k^{\perp}(\mathcal{G})$, we say that τ is smaller (i.e. "more abstract") than τ' ($\tau \sqsubseteq \tau'$) if and only if for all pairs $1 \leq i < j \leq k$, if $\tau(\langle i, j\rangle) \neq \perp$ then $\tau(\langle i, j\rangle) = \tau'(\langle i, j\rangle)$. Note that the lattice defined by $\sqsubseteq$ on $\Delta_k^{\perp}(\mathcal{G})$ is not complete, namely τ, τ' admit a least upper bound if and only if they are compatible on the defined components.

Definition 11, below, formally introduces the notion of minimal traverse for a set of k-pairwise delays D. Intuitively, a minimal traverse τ for a set of k-pairwise delays D is so called for two reasons. First, it is a k-pairwise partial delay τ that traverses i.e. *intersects* each delay $\delta \in D$ on a pair of components (that is, $\tau(\langle i, j\rangle) = \delta(\langle i, j\rangle)$ for some $1 \leq i < j \leq k$). Second, it is minimal w.r.t. $\sqsubseteq$.

▶ **Definition 11** (Minimal Traverse)**.** Let D be a set of k-pairwise delays on $(W, \cdot, \mathbb{1})$. A *traverse* for D is a k-pairwise partial delay $\tau \in \Delta_k^{\perp}(\mathcal{G})$ such that:

- for each $\delta \in D$, there exists a pair $\langle i, j\rangle, 1 \leq i < j \leq k$ such that $\delta(\langle i, j\rangle) = \tau(\langle i, j\rangle) \neq \perp$
- for each pair $\langle i, j\rangle, 1 \leq i < j \leq k$, if $\tau(\langle i, j\rangle) \neq \perp$, then there exists $\delta \in D$ such that $\delta(\langle i, j\rangle) = \tau(\langle i, j\rangle)$.

A *minimal traverse* for D is a traverse for D minimal w.r.t. $\sqsubseteq$.

The set of minimal traverses for (a set of delays) D can obviously be empty, but more importantly its *maximum size depends only on* k, as stated in Lemma 12 (i.e. it is a constant, if k is assumed to be a constant). Given a set of delays D, we denote by $\alpha(D)$ the set of minimal traverses for D.

▶ **Lemma 12.** *[18] For each set of k-pairwise delays D on $(W, \cdot, \mathbb{1})$, $|\alpha(D)| \leq 2^{k^4}$.*

We are now ready to present the announced abstraction for D_k^i. Given $D_k^i : Q^k \mapsto 2^{\Delta_k(\mathcal{G})}$, we denote by $\alpha(D_k^i) : Q^k \mapsto 2^{\Delta_k^{\perp}(\mathcal{G})}$ the function that associates with each tuple of states $(q_1, \ldots, q_k) \in Q^k$, the set of minimal traverses $\alpha(D_k^i(q_1, \ldots, q_k))$ for $D_k^i(q_1, \ldots, q_k)$.

Lemma 13 below states that the abstraction α retains enough information to decide whether A is k-valued by inspecting $\alpha(D_{k+1}^b)$, where $b = |Q|^{k+1} + |Q|^{k+1} \cdot (\frac{k(k-1)}{2})^2$ is the bound given by our pumping lemma.

▶ **Lemma 13.** *Let $A = (Q, q_I, F, \delta, \gamma)$ be a group automaton over $(W, \cdot, \mathbb{1})$, let $b = |Q|^{k+1} + |Q|^{k+1} \cdot (\frac{k(k-1)}{2})^2$. A is not k-valued if and only if it admits a tuple of final states $(q_1, \ldots, q_{k+1}) \in F^{k+1}$ such that $\alpha(D_{k+1}^b)(q_1, \ldots, q_{k+1})$ does not contain a minimal traverse τ satisfying the following condition: $\tau$is equal to one on each pair of components on which it is not $\perp$, i.e. $\tau \sqsubseteq 1^{k+1}$.*

Lemma 14 states that $\alpha(D_k^{i+1})$ can be computed from $\alpha(D_k^i)$ (in PTIME).

▶ **Lemma 14.** *Let $A = (Q, q_I, F, \delta, \gamma)$ be a group automaton. For all $i > 0$, $\alpha(D_k^i)$ can be computed in polynomial time wrt $|Q|$ and the size of $\alpha(D_k^{i-1})$.*

We are finally ready to state our polynomial result for testing k-valuedness:

▶ **Theorem 15.** *Let A be a group automaton over $(W, \cdot, \mathbb{1})$. If the delays accumulated along each polynomially bounded path of A can be computed in polynomial time, then k-valuedness can be tested in polynomial time for A, when k is a fixed constant.*

▶ **Corollary 16.** *The k-valuedness problem on* Dsum*-automata can be solved in polynomial time, when k is a fixed constant.*

4 Decomposition of k-valued group automata

In this section, we show that any k-valued group automaton A (over an infinitary group) is equivalent to a union of unambiguous group automata. Beside providing more insights toward expressivity issues, the equivalence established throughout this section will be used in Section 5 to provide positive results w.r.t. the decidability of the quantitative language inclusion problem on k-valued WA. The proof of this equivalence goes in two steps. First, we prove that A is equivalent to a k-ambiguous group automaton (for every input string, there are at most k accepting runs). The construction generalizes to groups that of [19] established for string-to-string transducers. Then, it is know that any k-ambiguous group automaton is equivalent to a union of unambiguous ones. This result has been proved in [19] for string transducers, based on a notion of lexicographic covering of k-ambiguous automata that can be directly applied to group automata.

From k-valued to k-ambiguous group automata We assume that $G = (W, \cdot, \mathbb{1})$ is a group that satisfies the infinitary condition. Recall that $\mathsf{delay}(u, v) = u^{-1} \cdot v$ for all $u, v \in W$. The construction we present in this section generalizes to groups the construction of [19], proved for string-to-string transducers. In [19], the k-ambiguous equivalent transducer non-deterministically guesses, when it reads an input string u, an output string v and a run ρ that produces v such that all the runs on u that are strictly smaller than ρ (according to a lexicographic order on runs) either (i) produce a different value, or (ii) have a delay that exceeds some bound N on some prefix of u. For a sufficiently large N (that depends on the transducer only), there are at most k such runs ρ if the transducer is k-valued.

In order to check properties (i) and (ii), it suffices to store the delays between the current chosen run ρ with all the smaller runs. This is done by keeping track of all the states q the smaller runs can reach, together with all the possible delays associated with q. If some delay exceeds the bound N, it can safely be replaced by the value ∞, and therefore one only needs to store delays of length N at most. The main difficulty, when generalising this construction to groups, is that for groups, there is no notion of "long" delay since in general, the group is not equipped with a metric. Instead, the bound N we consider is on the number of different delays between the prefixes of the current chosen run ρ and the prefixes of the smaller runs.

Given two runs ρ, ρ' on the same input word $u \in \Sigma^*$, we denote by $\mathsf{lag}(\rho, \rho')$ the set $\mathsf{lag}(\rho, \rho') = \{\mathsf{delay}(\rho[:i], \rho'[:i]) \mid 0 \leq i \leq |u| + 1\}$. Our construction is based on the following key lemma, that generalizes over group a similar lemma for strings proved in [19].

▶ **Lemma 17.** *Let A be a trim group automaton over G with n states. If A is k-valued, then for all tuples of runs $(\rho_1, \ldots, \rho_{k+1})$ on the same input, there exists $1 \leq i < j \leq k+1$ such that:*

1. *ρ_i and ρ_j have the same output, i.e. $V(\rho_i) = V(\rho_j)$*
2. $|\mathsf{lag}(\rho_i, \rho_j)| \leq n^{k+1}$

Based on Lemma 17, we show that any k-valued trim GA $A = (Q, q_I, F, \Delta, \gamma)$ over G is equivalent to some k-ambiguous GA $A' = (Q', q'_I, F', \Delta', \gamma')$. We let $N = n^{k+1}$, where n is the number of states of A. The idea is to order the transitions of A with a total order $\prec_A$ and to extend it to a lexicographic order over runs. Then, the construction non-deterministically guesses a run ρ of A, and checks that all the runs that are smaller than ρ produce either a different value than ρ, or have a lag with ρ whose size exceeds N. Lemma 17 will ensure that there are at most k such runs ρ on the same input. In order to check these properties, A' guesses the transitions of ρ and for each state $q \in Q$, it stores all the lags (which are sets of delays) between the current prefix of ρ and the runs that end-up in q and are smaller than this prefix. If the size of a lag exceeds N, then A' replaces this lag by the extra value ∞.

We will see that doing so, any lag that has to be stored by A' is a lag that can be generated by two runs of length n^{2k+3} at most, and therefore the state space will be finite.

Given a lag between two runs ρ_1 and ρ_2 and two transitions on the same symbol that extend these two runs respectively, in order to compute the lag of the extended runs, A' also needs to know the delay between ρ_1 and ρ_2. Therefore, A' also needs to store, for each lag, the "current" delay contained in this lag.

Construction of A'. Let us now define formally the state set Q' of A'. We let $\mathcal{D}$ be the set of all delays that can be generated by pairs of runs of length at most N^2n^4 (we will show later why we need to take such a value). Note that $\mathcal{D}$ is a finite set. Let also $\mathcal{P}_N(\mathcal{D})$ be all the subsets of $\mathcal{D}$ of cardinality at most N. Then, we let $Q' = Q \times (Q \to 2^{(\mathcal{P}_N(\mathcal{D})\times\mathcal{D})\cup\{\infty\}})$. After reading an input string $u \in \Sigma^*$, A' is in state $(q,\ell) \in Q'$ iff q is the state of the current guessed run ρ of A on u, and for all $p \in Q$, $(L,d) \in \ell(p)$ iff there exists a run ρ' on u ending in p, smaller than ρ, and such that $\mathsf{lag}(\rho,\rho') = L$, $|L| \leq N$, and $\mathsf{delay}(\rho,\rho') = d$. Moreover, $\infty \in \ell(p)$ iff there exists a run ρ' on u ending in p, smaller than ρ, such that $|\mathsf{lag}(\rho,\rho')| > N$.

We accept a run iff it ends in a state (q,ℓ) such that q is accepting in A ($q \in F$), and there is no state $p \in Q$ such that $(L,\mathbb{1}) \in \ell(p)$ for some L (otherwise it would mean that there exists a smaller run which gives the same output than the guessed run, on the same input). So, $F' = \{(q,\ell) \in Q' \mid \forall p \in Q, \forall (L,d) \in \ell(p),\ d \neq \mathbb{1}\}$. The initial state of A' is $q'_I = (q_I, \ell_0)$ where q_I is the initial state of A, and for all $q \in Q$, $\ell_0(q) = \varnothing$ if $q \neq q_I$ and $\{(\{\mathbb{1}\},\mathbb{1})\}$ if $q = q_I$. The transitions in Δ' are defined by: $(p,\ell) \xrightarrow{a|u} (p',\ell') \in \Delta'$ iff $p \xrightarrow{a|u} p' \in \Delta$ and for all $q' \in Q$, $\ell'(q') = \mathsf{prune}(\mathsf{update}(\ell,p,p',a,u,q'))$ where $\mathsf{update}(\ell,p,p',a,u,q')$ is the union of

$$\begin{array}{ll}
L_1 & \{(L \cup \{u^{-1}\cdot d\cdot v\}, u^{-1}\cdot d\cdot v) \mid \exists q.(L,d) \in \ell(q) \wedge q \xrightarrow{a|v} q'\}, \\
L_2 & \{(\{u^{-1}\cdot v\}, u^{-1}\cdot v) \mid \exists p \xrightarrow{a|v} q' \prec_A p \xrightarrow{a|u} p'\}, \mathit{and}, \\
L_3 & \{\infty \mid \exists q.\infty \in \ell(q) \wedge \exists q \xrightarrow{a|v} q' \in \Delta\}
\end{array} \quad .$$

and prune replaces each element (L,d) by ∞ if $|L| > N$, and $\mathsf{prune}(\infty) = \infty$.

It is easy to show that A' is well-defined, in the sense that the transition function is defined over states of Q' only, i.e. all the delays computed by A' can be generated by pairs of runs of length at most N^2n^4.

▶ **Lemma 18.** *A and A' are equivalent. Moreover, A' is k-ambiguous.*

On the ground of the decomposition of k-valued GA as k-ambiguous GA, and the known decomposition of k-ambiguous automata into union of unambiguous automata, one gets finally the following equivalence theorem:

▶ **Theorem 19.** *Any k-valued GA over G is equivalent to the union of k unambiguous GA over G.*

5 The Inclusion and Synthesis Problems for k-Valued WA

Inclusion problem In this section we consider k-valued V-automata, $V \in \{\mathsf{Sum}, \mathsf{Dsum}\}$ and we provide positive decidability results w.r.t. their inclusion problem. Given two V-automata on the alphabet Σ, the inclusion problem ($A \leq B$?) asks whether for all $w \in \Sigma^+$, $L_A(w) \leq L_B(w)$. This problem is undecidable for general Sum-automata, open for general Dsum-automata and PSPACE-c for functional Sum- and Dsum-automata [3, 10]. Relying on the results in the previous sections, we provide an inclusion test applying to k-valued Sum and to the class of Dsum-automata having a discount factor λ that is an inverted integer, i.e. we consider discount factors of the form $\frac{1}{n}, n \in \mathbb{N}^*$. In more details, we show how to encode the inclusion problem for Sum-automata (resp. Dsum-automata) into the problem of determining a strictly positive path (resp. a path with strictly positive discounted sum) in a multi-weighted graph.

Let A, B be two k-valued weighted V-automata, $V \in \{\mathsf{Sum}, \mathsf{Dsum}\}$. The encoding proceeds in three steps. First, we check whether $\mathrm{dom}(A) \subseteq \mathrm{dom}(B)$. Then, by Theorem 19, A and B can be effectively decomposed as unions of k unambiguous V-automata $\bigcup_{i=1}^{k} G_i$ and $\bigcup_{i=1}^{k} H_i$ respectively. We can assume that $dom(G_i) = dom(A)$ for all i (resp. $dom(H_i) = dom(B)$ for all i)[4]. Clearly, $A \not\leq B$ iff there exist $w \in \Sigma^*$ and $i \in \{1, \dots, k\}$ such that $G_i(w) > B(w)$. For all $i \in \{1, \dots, k\}$, we finally check whether $G_I(w) > B(w)$ for some w (i.e. $G_i(w) > H_j(w)$ for some w and all j) by a reduction to a multi-dimensional graph problem, as follows.

Let $i \in \{1, \dots, k\}$, Q_i be the set of states of G_i, and P_j the set of states of H_j for all $j \in \{1, \dots, k\}$. We construct a k-weighted graph $\Phi_i = (V_i, q_i^0, F_i, E_i, w_i)$ where the vertices are $V_i = Q_i \times P_1 \times \cdots \times P_k$, the weight function has type $w_i : E_i \to \mathbb{Z}^k$, and the weighted transitions are defined by $(q, p_1, \dots, p_k) \xrightarrow{(v_1,\dots,v_k)} (q', p_1', \dots, p_k') \in E_i$ iff $q \xrightarrow{a|m} q'$ is a transition of G_i and for all $j \in \{1, \dots, k\}$, $p_j \xrightarrow{a|v_j+m} p_j'$ is a transition of H_j. Final vertices (resp. the initial vertex) in Φ_i are those whose components are all final (resp. initial) states.

Then, the following holds: $A \not\leq B$ if and only if there exists $i \in \{1, \dots, k\}$ and a path π to a final state in Φ_i with strictly positive sum (resp. stricly positive discounted sum) on all components. By Lemma 5, it is possible to check in polynomial time whether a multi-weighted graph of dimension k contains a strictly positive path to a target vertex. Therefore, the inclusion problem is decidable for k-valued Sum-automata.

Lemma 20 below provides a PSPACE procedure to check whether a multi-weighted graph has a path to a given target state with strictly positive *discounted* sum w.r.t. a discounted factor λ that is an inverted integer $\frac{1}{n}, n \in \mathbb{N}^*$. This leads to decidability of the inclusion also for the class of Dsum-automata, when the discount factor λ is an inverted integer[5].

▶ **Lemma 20.** *Let $(V, v_0, E, w : E \mapsto \mathbb{Z}^k)$ be a multi-weighted graph of dimension k, $t \in V$, and consider $\lambda = \frac{1}{n}, n \in \mathbb{N}^*$. The problem of deciding whether G admits a path from v_0 to t with strictly positive discounted sum on all components w.r.t. λ is in PSPACE.*

▶ **Theorem 21.** *The inclusion problem is decidable for k-valued* Sum*-automata and k-valued* Dsum*-automata associated with a discount factor λ such that $\lambda = \frac{1}{n}, n \in \mathbb{N}^*$.*

▶ Remark. If A and B are given in input as union of k_1 and k_2 unambiguous WA with the same domain, then checking whether $A \leq B$ can be done in polynomial time (resp. PSPACE) for Sum-automata (resp. Dsum-automata w.r.t. inverted integer discount factors) in virtue of Lemma 20, Lemma 5 and Theorem 21. Finally, note that the encoding into WA over groups povided by Proposition 1 allows to associate a different discount factor with each edge of the given automaton. The results in Theorem 21 can be further generalized in this sense: In particular, Proposition 1 guarantees the correctness of the encoding into the problem of determining a path with strictly positive discounted sum in a multi-weighted graph. The proof of Lemma 20, solving the latter problem, can be easily generalized to deal with different inverted integer discount factors.

[4] Otherwise, we opportunely complete G_i.

[5] Note that inverted integer discounted factors where considered in [1] to provide a determinization procedure applying to *complete* Dsum-automata (all states are accepting). However, for Dsum-automata with final states no special form of discount factor can guarantee determinization [1].

Quantitative Synthesis We consider *quantitative realizability* problem. The realizability problem is better understood as a game between two players: "Player input" (the environment, also called Player I) and "Player output" (the controller, also called Player O). Player I (resp. Player O) controls the letters of a finite alphabet Σ_I (resp. Σ_O). We assume that $\Sigma_O \cap \Sigma_I = \varnothing$ and that Σ_O contains a special symbol $\#$ whose role is to stop the game. We let $\Sigma = \Sigma_O \cup \Sigma_I$. Formally, the realizability game is a turn-based game played on an arena defined by a weighted automaton $A = (Q = Q_O \uplus Q_I, q_0, F, \delta = \delta_I \cup \delta_O, \gamma)$, whose set of states is partitioned into two sets Q_O and Q_I, $\delta_O \subseteq Q_O \times \Sigma_O \times Q_I$, $\delta_I \subseteq Q_I \times \Sigma_I \times Q_O$, and such that $\mathrm{dom}(A) \subseteq (\Sigma \backslash \{\#\})^* \#$. Player O starts by giving an initial letter $o_0 \in \Sigma_O$, Player I responds providing a letter $i_0 \in \Sigma_I$, then Player O gives o_1 and Player I responds i_1, and so on. Player O has also the power to stop the game at any turn with the distinguishing symbol $\#$. In this case, the game results in a finite word $(o_0 i_0)(o_1 i_1)\dots(o_j i_j)\# \in \Sigma^*$, otherwise the outcome of the game is an infinite word $(o_0 i_0)(o_1 i_1)\dots \in \Sigma^\omega$.

The *quantitative realizability problem* asks whether Player O has the strategy $\lambda_O : (\Sigma_O \Sigma_I)^* \to \Sigma_O$ such that for all Player I's strategies $\lambda_I : \Sigma_O(\Sigma_I \Sigma_O)^* \to \Sigma_I$, the outcome of this two strategie, which is a finite word w, satisfies $A(w) > 0$, in which case we say that A is realizable. We refer the reader to [10] for formal definitions of strategies and outcomes. In [10], we have shown that the realizability problem is undecidable for functional Sum WA (and for other measures), while a positive decidability results applies to deterministic Sum WA (and for other measure). As a corollary, we know that the problem is undecidable for k-valued Sum WA. We will strengthen this negative result here and show that even if the problem is decidable for deterministic WA, it is undecidable for Sum WA defined as union of k deterministic automata, for $k \geq 4$. In particular, we show that the halting problem for deterministic 2-counter Minsky machines [16] can be reduced to the quantitative language realizability problem for the union of 4-deterministic Sum-automata.

▶ **Theorem 22.** *The realizability problem for a* 4*-valued* Sum*-automata defined as a union of* 4 *deterministic* Sum*-automata is undecidable.*

References

1 U. Boker and T. A. Henzinger. Determinizing discounted-sum automata. In *CSL*, pages 82–96, 2011.

2 V. Bruyère, N. Meunier, and J.-F. Raskin. Secure equilibria in weighted games. In *LICS*, 2014. to appear.

3 K. Chatterjee, L. Doyen, and T. A. Henzinger. Quantitative languages. *ACM Trans. Comput. Log*, 11(4), 2010.

4 K. Chatterjee, V. Forejt, and D. Wojtczak. Multi-objective discounted reward verification in graphs and mdps. In *LPAR*, pages 228–242, 2013.

5 C. Choffrut. Une caractérisation des fonctions séquentielles et des fonctions sous-séquentielles en tant que relations rationnelles. *Theor. Comput. Sci.*, 5(3):325–337, 1977.

6 E. Clarke, O. Grumberg, and D. Peled. *Model Checking.* The MIT Press, Cambridge, Massachusetts, 1999.

7 R. de Souza. On the decidability of the equivalence for k-valued transducers. In *Developments in Language Theory*, pages 252–263, 2008.

8 R. de Souza and N. Kobayashi. A combinatorial study of k-valued rational relations. *Journal of Automata, Languages and Combinatorics*, 3/4(13):207–231, 2008.

9 M. Droste, W. Kuich, and H. Vogler. *HandBook of Weighted Automata.* Springer, 2009.

10 E. Filiot, R. Gentilini, and J.-F. Raskin. Quantitative languages defined by functional automata. In *CONCUR*, pages 132–146, 2012.

11 T. V. Griffiths. The unsolvability of the equivalence problem for -free nondeterministic generalized machines. *Journal of the ACM*, 1968.

12 E. Gurari and O. Ibarra. A note on finitely-valued and finitely ambiguous transducers. *Mathematical Systems Theory*, 16(1):61–66, 1983.

13 K. Hashiguchi, K. Ishiguro, and S. Jimbo. Decidability of the equivalence problem for finitely ambiguous finance automata. *IJAC*, 12(3):445, 2002.

14 I. Klimann, S. Lombardy, J. Mairesse, and C. Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *TCS*, 327(3):349–373, 2004.

15 D. Krob. The equality problem for rational series with multiplicities in the tropical semiring is undecidable. *Int. Journal of Algebra and Computation*, 4(3):405–425, 1994.

16 M. L. Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

17 K. S. Rao and G. Sullivan. Detecting cycles in dynamic graphs in polynomial time. STOC '88, pages 398–406. ACM, 1988.

18 J. Sakarovitch and R. de Souza. On the decidability of bounded valuedness for transducers. MFCS. Springer-Verlag, 2008.

19 J. Sakarovitch and R. de Souza. Lexicographic decomposition of k -valued transducers. *Theory Comput. Syst*, 47(3):758–785, 2010.

20 H. Seidl. Equivalence of finite-valued tree transducers is decidable. *Mathematical Systems Theory*, 27(4):285–346, 1994.

21 A. Weber. On the valuedness of finite transducers. *Acta Informatica*, 27(8):749–780, 1989.

22 A. Weber. Finite-valued distance automata. *TCS*, 134(1):225–251, 7 Nov. 1994.