On the Decidability of Bounded Valuedness for Transducers

Jacques Sakarovitch[1] and Rodrigo de Souza[2] *

[1] LTCI, ENST/CNRS, Paris (France) sakarovitch@enst.fr
[2] ENST, 46, rue Barrault, 75634 Paris Cedex 13 (France) rsouza@enst.fr

Abstract. We give a new and conceptually different proof for the decidability of k-valuedness of transducers (a result due to Gurari and Ibarra), without resorting to any other kind of machines than transducers. In contrast with the previous proof, our algorithm takes into account the structure of the analysed transducers and yields better complexity bounds. With the same techniques, we also present a new proof, hopefully more easily understandable, for the decidability of bounded valuedness (a result due to Weber).

Extended abstract

1 Introduction

This communication is part of a complete reworking of the theory of k-valued rational relations and transducers which makes it appear as a natural generalisation of the theory of rational functions (the 1-valued ones) and functional transducers, not only at the level of results but *also at the level of proofs*.

In one word, it is decidable whether a finite transducer is functional (Schützenberger [1]), the equivalence of functional transducers is decidable (consequence of the previous result), and every functional transducer is equivalent to an unambiguous one (Eilenberg [2]). These results generalise in a remarkable way to bounded valued transducers: it is decidable whether the cardinality of the image of every word by a given transducer is bounded (Weber [3]) and whether it is bounded by a given integer k (Gurari and Ibarra [4], by reduction to the emptiness problem for a class of multi-counter automata); the equivalence of k-valued transducers is decidable (Culik and Karhumäki [5] in the context of the study of Ehrenfeucht's conjecture, and Weber [6]), and every k-valued transducer is equivalent to the sum of k functional and unambiguous ones (Weber [7]).

In [8], we have given a new and shorter proof for this last result, with a gain of one exponential in the size of the result with respect to the original proof. It is based on a construction that we call the *lag separation covering* (of real-time transducers). This construction itself uses the *Lead or Delay Action* (LDA for short) introduced in [9] to describe an efficient construction for the decidability of the functionality of transducers.

In this communication we present a new proof for the following result:

* A financial support of CAPES Foundation (Brazilian government) for doctoral studies is gratefully acknowledged by this author.

Theorem 1 (Gurari-Ibarra [4]). *Let $\mathcal{T}$ be a transducer and k a positive integer. It is decidable in polynomial time whether $\mathcal{T}$ is k-valued.*

We also present here a new proof for the decidability of the bounded valuedness, which comes very naturally together with the proof of Theorem 1:

Theorem 2 (Weber [3]). *Let $\mathcal{T}$ be a transducer. It is decidable in polynomial time whether there exists an integer k such that $\mathcal{T}$ is k-valued.*

In a third part [10], we tackle the decidability of the equivalence of k-valued transducers by using together the methods we present here and in [8].

In the original proof of Theorem 1 by Gurari and Ibarra, a nondeterministic $k(k+1)$-counter 1-turn automaton $\mathcal{A}$ (see [11] for definitions) is built. A computation in $\mathcal{A}$ corresponds to $k+1$ computations of $\mathcal{T}$ with the same input, and each pair of these computations of $\mathcal{T}$ is associated with two counters. The counters are incremented by the lengths of the outputs until a position, guessed nondeterministically, where these outputs become different, and a computation of $\mathcal{A}$ is successful iff the outputs of its $k+1$ projections are pairwise distinct. Theorem 1 follows then from the decidability, in polynomial time, of the emptiness of a finite turn r-counter automaton, another result due to Ibarra [11].

If this theoretical scheme is clear, the actual complexity of the corresponding procedure is difficult to estimate beyond the fact that "it is polynomial". This is particularly true for the procedure which decides of the emptiness of a multi-counter automaton, for it is based on general arguments of complexity theory: if the r-counter automaton accepts some input, then there exists a constant c such that it accepts an input of length bounded by $(rm)^{cr}$ (where m is the number of transitions); it is possible to test these bounded inputs with a nondeterministic Turing machine working in space proportional to $cr\log(rm)$; for each nondeterministic Turing machine working in space $f(n)$ there exists an equivalent deterministic one working in time $d^{f(n)}$, for some constant d (cf. [12]). It is not clear how these two constants c and d can be effectively computed and if their actual values have any relationship with the transducer under inspection.

Our proof of Theorem 1 (Section 3) stems from a generalisation of the characterisation of functional transducers with the Lead or Delay Action (LDA) $\mathcal{G}$ in [9]. Roughly speaking, a computation in the product $\mathcal{T}\times\mathcal{T}=\mathcal{T}^2$ projects on two computations with equal inputs in $\mathcal{T}$, thus $\mathcal{T}$ is functional iff every successful computation in $\mathcal{T}^2$ outputs a pair of equal words. Differences between words are witnessed by the LDA $\mathcal{G}$, and $\mathcal{T}$ is functional iff the product $\mathcal{T}^2\times\mathcal{G}$ is isomorph to $\mathcal{T}^2$ (being hence finite) and assigns the empty word to the final states of $\mathcal{T}^2$.

At first we generalise the LDA to an action $\mathcal{G}_{k+1}$ which measures the differences between the outputs in the $(k+1)$-tuples of computations of $\mathcal{T}$. It is not difficult to get a necessary and sufficient condition on $\mathcal{T}^{k+1}\times\mathcal{G}_{k+1}$ for $\mathcal{T}$ be k-valued (Proposition 1). The problem is that this condition *is not effective anymore* for $\mathcal{T}^{k+1}\times\mathcal{G}_{k+1}$ may be infinite for $k>1$ even if $\mathcal{T}$ is k-valued. The core of our method – and this is of course more complicated – is the proof that it is possible to attach to every state of $\mathcal{T}^{k+1}$ a finite set of information, effectively computable from $\mathcal{T}$, which retains all the useful information from $\mathcal{T}^{k+1}\times\mathcal{G}_{k+1}$

to decide whether $\mathcal{T}$ is k-valued (Theorem 3). These sets are what we call the *Lead or Delay Valuation* of $\mathcal{T}^{k+1}$ (LDV). We explain in Section 3.4 how the LDV can be constructed in $\mathcal{O}(\ell n^{k+1} m^{k+1})$, where n and m are the numbers of states and transitions of $\mathcal{T}$ and ℓ is the maximal length of outputs of transitions. By comparison with the complexity of the procedure to decide the functionality in [9], this is probably the best that can be hoped for. On the other hand it is to be acknowledged that the constant hidden in the "big O" is handed by a function which grows exponentially fast with the valuedness k, namely, $2^{5(k+1)^4}$.

Weber's proof of Theorem 2 is somewhat similar to the classical characterisation of bounded $\mathbb{N}$-automata of Mandel and Simon [13] (made more explicit in [14]): $\mathcal{T}$ is bounded valued iff $\mathcal{T}$ does not contain certain *forbidden* computations (Theorem 4). Weber gives in [3] an algorithm to detect these computations.

We give another proof for Theorem 2, which uses a construct, the *lag separation covering*, that we have defined in [8] in order to establish the decomposition resulted quoted above. We first describe the forbidden computations in a slightly different way (Theorem 5). With the help of the lag separation covering, the proof that the absence of these computations implies the bounded valuedness is straightforward: if this holds for $\mathcal{T}$, then the covering has an equivalent subtransducer whose underlying input automaton is finitely ambiguous; in other words, every input word can be read by a bounded number of computations in $\mathcal{T}$, thus $\mathcal{T}$ is bounded valued. We explain in Section 4.2 how this characterisation can be tested in a certain subtransducer of the product of $\mathcal{T}^3$ by the LDA in complexity $\mathcal{O}(\ell n^3(n^3+m^3))$. To some extent the complexity claimed in [3] is of the same order as it is in $\mathcal{O}(\ell^2 n^9)$ but the proof is indeed difficult to follow.

Due to space constraints most of the proofs have not been included, but they can be found in [15] and hopefully in a forthcoming paper which is in preparation. We have tried our best to give here the ideas underlying the proofs. This is anyway a highly technical matter of which it would be futile to disguise the intrinsic complexity.

2 Preliminaries

We follow the definitions and notation in [16,2,17]. The set of words over a finite alphabet A (the free monoid over A) is denoted by A^*, and the empty word by 1_{A^*}, or simply 1 in figures. The length of a word u in A^* is denoted by $|u|$.

Let M be a monoid. An automaton $\mathcal{A} = (Q, M, E, I, T)$ is a directed graph given by sets Q of states, I, $T \subseteq Q$ of initial and final states, respectively, and $E \subseteq Q \times M \times Q$ of transitions labelled by M. It is finite if Q and E are finite.

A *computation* in $\mathcal{A}$ is a sequence of transitions $c : p_0 \xrightarrow{m_1} p_1 \xrightarrow{m_2} \dots \xrightarrow{m_l} p_l$, also denoted by $c : p_0 \xrightarrow{m_1 \dots m_l} p_l$. Its *label* is the element $m_1 \dots m_l$ of M and its length is $|c| = l$. It is *successful* if $p_0 \in I$ and $p_l \in T$. The *behaviour* of $\mathcal{A}$ is the set $|\mathcal{A}| \subseteq M$ of labels of successful computations. The behaviour of finite automata over M coincide with the family $\operatorname{Rat} M$ of *rational subsets* of M [2].

If M is a free monoid A^* and the labels of transitions are letters, then $\mathcal{A}$ is a (boolean) automaton over A. If M is a product $A^* \times B^*$, then every transition is

labelled by an input word $u \in A^*$ and an output one $x \in B^*$ — this is denoted by $u|x$ — and $\mathcal{A}$ is a transducer realising a *rational relation* from A^* to B^*.

The image of a word u by a transducer $\mathcal{T}$ is the set of outputs of the successful computations reading u; $\mathcal{T}$ is called *k-valued* (for an integer $k > 0$) if the cardinalities of these images are at most k, and *bounded valued* if there exists such k.

We shall only consider *real-time* transducers: their labels are pairs $a|K$ formed by a letter a and a set $K \in \operatorname{Rat} B^*$, and I and T are functions from Q to $\operatorname{Rat} B^*$. By using classical constructions on automata, every transducer can be transformed into a real-time one. For bounded valued relations we may suppose that the transitions output a single word, and in order to avoid inessential details we can also suppose that the image of every initial or final state is the empty word.[3] In this case, the transducer is denoted rather as $\mathcal{T} = (Q, A, B^*, E, I, T)$.

We shall make systematic use of product of automata. For real-time transducers, this operation is defined in the same way as for boolean automata, with the difference that the outputs have to be taken into account. Formally, the *square* of $\mathcal{T} = (Q, A, B^*, E, I, T)$ is the transducer $\mathcal{T}^2 = (Q^2, A, {B^*}^2, E^{(2)}, I^2, T^2)$ where $(p, q) \xrightarrow{a|(u,v)} (p', q')$ is in $E^{(2)}$ iff both $p \xrightarrow{a|u} p'$ and $q \xrightarrow{a|v} q'$ are in E (see [9] for details). We define likewise the product of $\mathcal{T}$ by itself l times: a transducer $\mathcal{T}^l$ labelled by $A \times {B^*}^l$ whose state set is Q^l, and the set of transitions is $E^{(l)}$.

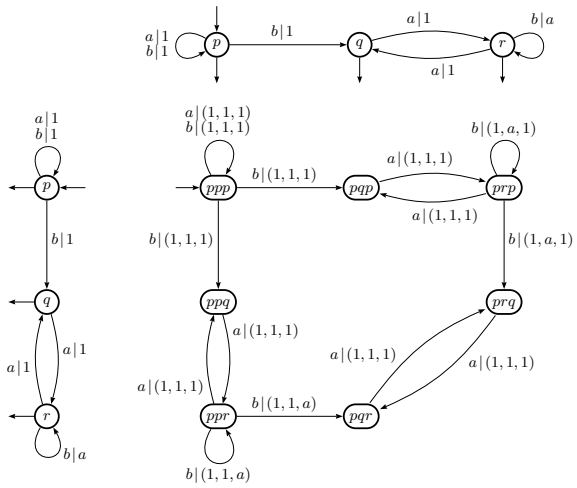

Fig. 1: A transducer $\mathcal{T}_1$ (drawn on the left and above), and the part of $\mathcal{T}_1^3$ accessible from (p, p, p) and co-accessible to (p, q, r). All states are final.

A last, and useful, convention: all automata (or transducers) considered or built by the various algorithms described here are implicitly assumed to be *accessible*. In particular, we write that Q^l is the set of states of $\mathcal{T}^l$, but indeed when we say that $\mathbf{q}$ is a state[4] of $\mathcal{T}^l$, we mean that $\mathbf{q}$ is accessible in $\mathcal{T}^l$.

[3] Such transducers are also called *nondeterministic generalised sequential machines*.

[4] We write tuples of states, or of words, with bold letters.

3 Deciding k-valuedness

Our proof for Theorem 1 consists in testing the k-valuedness of a transducer $\mathcal{T}$ in the cartesian product $\mathcal{T}^{k+1}$ in the same way as the functionality may be witnessed in the product of $\mathcal{T}^2$ by the Lead or Delay Action (LDA) $\mathcal{G}$ [9].

At first, the road to the generalisation seems easy: $(k+1)$-tuples of distinct computations in $\mathcal{T}$ with the same input are seen as computations in $\mathcal{T}^{k+1}$ and $\mathcal{T}$ is k-valued if in such computations at least two of the outputs are equal. To that end, the LDA is generalised to a *Pairwise Lead or Delay Action*, denoted $\mathcal{G}_{k+1}$, and the wanted property is expressed in $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$ (Proposition 1).

The difficulty arises with the fact that $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$ *may be infinite for* $k > 1$, *even if* $\mathcal{T}$ *is* k*-valued* (as in Figure 2). Here comes the crux of the proof: with the definition of *partially defined pairwise differences*, or PDPD (Section 3.3), we are able to attach to every state $\mathbf{q}$ of $\mathcal{T}^{k+1}$ a *finite set* $\mathsf{m}(\mathbf{q})$ of PDPDs. This $\mathsf{m}(\mathbf{q})$ subsumes the essential information contained in the states of $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$ that map onto $\mathbf{q}$ and makes it possible to characterise the k-valuedness within a finite object, the *Lead or Delay Valuation* (LDV) of $\mathcal{T}^{k+1}$ (Theorem 3). As we explain in Section 3.4, the LDV can be built in polynomial time with a traversal of $\mathcal{T}^{k+1}$.

3.1 The Lead or Delay Action

Let B be an alphabet and $\overline{B}$ a disjoint copy of B. The underlying structure of the LDA is the free group $F(B)$ generated by B, that is, the quotient of $(B \cup \overline{B})^*$ by the relations $x\overline{x} = \overline{x}x = 1_{B^*}$, for every x in B. The inverse of an element u in $F(B)$ is denoted by $\overline{u}$ (for example, $\overline{x\,x\,y} = \overline{y}\,\overline{x}\,\overline{x}$). We write $\Delta = B^* \cup \overline{B}^* \cup \{\mathbf{0}\}$, where $\mathbf{0}$ is a new element, a zero, not in $F(B)$, and define a function $\rho : F(B) \cup \{\mathbf{0}\} \to \Delta$ by $w\rho = w$, if $w \in \Delta$, and $w\rho = \mathbf{0}$ otherwise.[5]

Definition 1 ([9,17]). *The Lead or Delay Action (LDA) of* $B^* \times B^*$ *on* Δ*, denoted by* $\mathcal{G}$*, is defined as follows: for every* $w \in \Delta$ *and* $(u, v) \in B^* \times B^*$*,* $w \cdot (u, v) = (\overline{u}\, w\, v)\rho$ *(where the product is taken with the rules* $\mathbf{0}u = u\mathbf{0} = \mathbf{0}$*).*

Intuitively, $1_{B^*} \cdot (u, v)$ represents the "difference" of the words u and v, being a positive word if u is a prefix of v (the *lead* of v with respect to u), a negative word if v is a prefix of u (the *delay* of v with respect to u), and $\mathbf{0}$ if u and v are not prefixes of a common word. In [9], an effective characterisation of the functionality is made with the product $\mathcal{T}^2 \times \mathcal{G}$ (cf. Definition 3), which shows the differences between pairs of computations of $\mathcal{T}$: $\mathcal{T}$ is functional iff $\mathcal{T}^2 \times \mathcal{G}$ assigns an unique value of $\Delta - \{\mathbf{0}\}$ to *every useful state* of $\mathcal{T}^2$ and 1_{B^*} to the final ones.

3.2 The Pairwise Lead or Delay Action

In order to deal with the differences between the outputs of an arbitrary number l ($l > 1$) of computations in parallel, we generalise the LDA as follows. Let us

[5] We use a postfix notation for relations: $x\tau$ is the image of x by the relation τ.

write $D_l = \{(i,j) \mid 1 \leq i < j \leq l\}$. We write Δ_l for Δ^{D_l}, that is, the set of vectors of dimension D_l with entries in Δ, which we call *pairwise differences* or PD for short. The entry at the coordinate (i,j) of a PD δ is denoted by $\delta_{i,j}$. The PD with all entries equal to the empty word is denoted by η.

Definition 2. *For every integer $l > 1$, the Pairwise Lead or Delay Action of B^{*l} on Δ_l is the function $\mathcal{G}_l : \Delta_l \times B^{*l} \to \Delta_l$ which maps every $(\delta, \mathbf{u})$ in $\Delta_l \times B^{*l}$ to the PD γ in Δ_l such that, for every (i,j) in D_l, $\gamma_{i,j} = \delta_{i,j} \cdot (\mathbf{u}_i, \mathbf{u}_j)$.*

($\mathcal{G}_l$ is indeed an action for the LDA is applied independently to each coordinate.)

Definition 3. *For every integer $l > 1$, the product of $\mathcal{T}^l$ by $\mathcal{G}_l$ is the (accessible) transducer $\mathcal{T}^l \times \mathcal{G}_l = (Q^l \times \Delta_l, A, B^{*l}, F, I^l \times \{\eta\}, T^l \times \Delta_l)$ where $(\mathbf{p}, \delta) \xrightarrow{a|\mathbf{u}} (\mathbf{q}, \delta')$ is a transition in F iff $\mathbf{p} \xrightarrow{a|\mathbf{u}} \mathbf{q}$ is a transition in $E^{(l)}$ and $\delta' = \delta \cdot \mathbf{u}$.*

The k-valuedness of $\mathcal{T}$ is witnessed by the final states of $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$:

Proposition 1. *A transducer $\mathcal{T}$ is k-valued iff for every final state $(\mathbf{q}, \delta)$ of $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$, δ has at least one entry equal to 1_{B^*}.* □

This condition is not however effective. For every state $\mathbf{q}$ of $\mathcal{T}^{k+1}$, let us write $X(\mathbf{q})$ for the set of PDs in the states of $\mathcal{T}^{k+1} \times \mathcal{G}_{k+1}$ projecting on $\mathbf{q}$: $X(\mathbf{q}) = \{\delta \in \Delta_{k+1} \mid (\mathbf{q}, \delta) \text{ state of } \mathcal{T}^{k+1} \times \mathcal{G}_{k+1}\}$. Contrary to the characterisation of the functionality in [9], $X(\mathbf{q})$ may be infinite, even if $\mathcal{T}$ is k-valued (as in Figure 2).

3.3 A Finite Characterisation of k-valuedness

The main concept for the definition of the Lead or Delay Valuation is that of *traverse* of a set of PDs. Intuitively, a traverse for $X \subseteq \Delta_l$ is a PD γ in Δ_l such that for every δ in X, there exists a coordinate (i,j) satisfying $\delta_{i,j} \neq \mathbf{0}$ and $\gamma_{i,j} = \delta_{i,j}$. In other words, each PD in X has a non null "intersection" with γ.

It may well exists some (i,j) in which no intersection arises. Such coordinates are not really useful. For this reason, we embed Δ_l in a larger set $\mathrm{H}_l = [\Delta \cup \{\perp\}]^{D_l}$ of *partially defined pairwise differences* (PDPD), where $\perp$ fills undefined entries. Now, we say that a traverse for a set $X \subseteq \mathrm{H}_l$ of PDPDs is a PDPD $\gamma \in \mathrm{H}_l$ satisfying: for every $\delta \in X$, there exists a coordinate (i,j) such that $\delta_{i,j} \neq \mathbf{0}$, $\delta_{i,j} \neq \perp$, and $\gamma_{i,j} = \delta_{i,j}$; for every (i,j) such that $\gamma_{i,j} \neq \perp$, there exists at least one δ in X such that $\delta_{i,j} \neq \mathbf{0}$ and $\gamma_{i,j} = \delta_{i,j}$. A traverse has at least one defined entry, and has no entry equal to $\mathbf{0}$. We denote by $\mathrm{tv}(X)$ the set of traverses for X. As before, $\mathrm{tv}(X)$ may be infinite or empty.

The set H_l is naturally ordered by $\beta \sqsubseteq \gamma$ iff γ coincides with β on the defined entries of β. We denote by $\mathsf{m}(X) = \min(\mathrm{tv}(X))$ the set of minimal traverses for X, and for a state $\mathbf{q}$ of $\mathcal{T}^{k+1}$ we write $\mathsf{m}(\mathbf{q}) = \mathsf{m}(X(\mathbf{q}))$. The set $\mathsf{m}(\mathbf{q})$ is what we call the *value* of $\mathbf{q}$, the family of these sets is the *Lead or Delay Valuation* (LDV) of $\mathcal{T}^{k+1}$. It is not difficult to restate Proposition 1 in terms of this concept:

Theorem 3. *A transducer $\mathcal{T}$ is k-valued iff for every final state $\mathbf{q}$ of $\mathcal{T}^{k+1}$ there exists at least one γ in $\mathsf{m}(\mathbf{q})$ whose defined entries are all equal to 1_{B^*}.* □

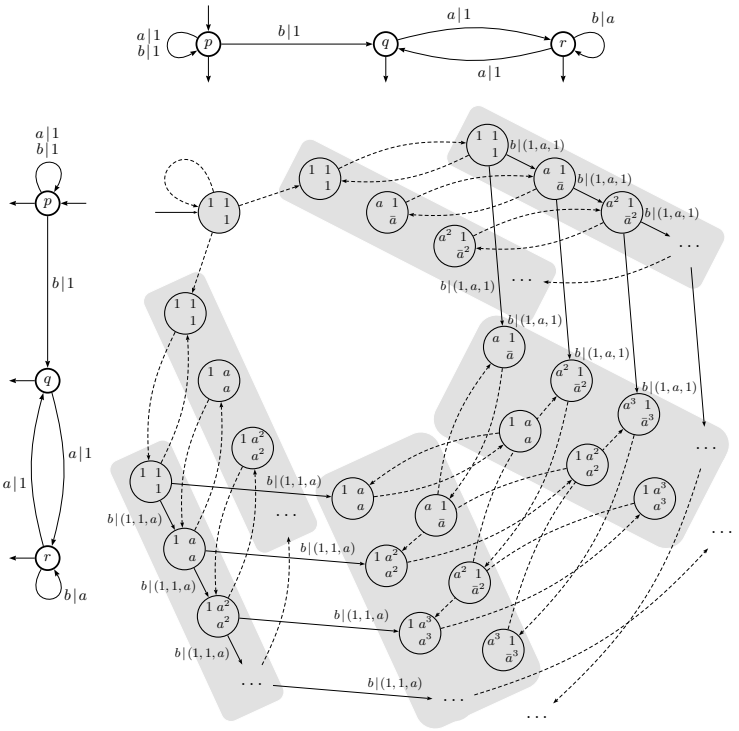

Fig. 2: The product by $\mathcal{G}_3$ of the part of $\mathcal{T}_1^3$ in Figure 1. Gray regions gather states which project on a same state of $\mathcal{T}^3$. The output of the dotted transitions is $(1,1,1)$.

Theorem 3 is the finite characterisation of the k-valuedness we are aiming at because the sets $\mathsf{m}(\mathbf{q})$ are finite and computable. The finiteness holds indeed for the set of minimal traverses of every set of PDs:

Proposition 2. *For every $l > 1$, for every $X \subseteq \Delta_l$, $\mathrm{card}(\mathsf{m}(X)) \leq 2^{l^4}$.* □

3.4 Making the Characterisation Effective

The effective construction of the LDV is based on two properties. The first one is a stability property in the strongly connected components (SCCs) of $\mathcal{T}^{k+1}$. The second one states that every $\mathsf{m}(\mathbf{q})$ depends uniquely on the values of the states which precede the SCC of $\mathbf{q}$.

We say that a PDPD γ is *stable* in $\mathbf{q}$ if for every circuit $\mathbf{q} \xrightarrow{f|\mathbf{u}} \mathbf{q}$, $\gamma \cdot \mathbf{u} = \gamma$.

Proposition 3. *Every $\gamma \in \mathsf{m}(\mathbf{q})$ is stable in $\mathbf{q}$. Thus, for every $\mathbf{p}$ in the same SCC as $\mathbf{q}$ and every computation $\mathbf{q} \xrightarrow{f|\mathbf{u}} \mathbf{p}$, $\mathsf{m}(\mathbf{p}) = \mathsf{m}(\mathbf{q}) \cdot \mathbf{u}$.* □

(Here $\mathcal{G}_l$ is extended to sets of PDPDs: for every $X \subseteq \mathrm{H}_l$ and every l-tuple of words $\mathbf{u}$, $X \cdot \mathbf{u} = \{\delta \cdot \mathbf{u} \mid \delta \in X\}$, where undefined entries of δ remains undefined in $\delta \cdot \mathbf{u}$). For $X \subseteq \mathrm{H}_{k+1}$, we denote $\mathrm{st}_{\mathbf{q}}(X) = \{\gamma \in X \mid \gamma \text{ stable in } \mathbf{q}\}$.

In order to explain the second property, we define a commutative and associative operation between sets of PDPDs. Given β and γ in (the partially ordered set) H_l, let $\beta \vee \gamma$ be their least upper bound (which exists iff β and γ are compatible on the defined coordinates). For $X, Y \subseteq \mathrm{H}_l$, we define $X \oplus Y = \min(\{\beta \vee \gamma \mid \beta \in X,\ \gamma \in Y\})$. If X and Y are finite, then $X \oplus Y$ is clearly finite. Let us also fix a notation. For every SCC C of $\mathcal{T}^{k+1}$, let $I(C)$ be the set of transitions incoming in C: $I(C) = \{\mathbf{p} \xrightarrow{a|\mathbf{u}} \mathbf{r} \mid \mathbf{p} \notin C, \mathbf{r} \in C\}$. For every $e : \mathbf{p} \xrightarrow{a|\mathbf{u}} \mathbf{r}$ in $I(C)$ and every state $\mathbf{q}$ in C, let $\mathbf{v}_{e,\mathbf{q}}$ be the output of an arbitrary but fixed computation from $\mathbf{r}$ to $\mathbf{q}$, and $X_{e,\mathbf{q}} = \mathrm{st}_{\mathbf{q}}(\mathsf{m}(\mathbf{p}) \cdot (\mathbf{u}\mathbf{v}_{e,\mathbf{q}}))$.

Proposition 4. *For every* $\mathbf{q}$ *(in the SCC* C *of* $\mathcal{T}^{k+1}$*),* $\mathsf{m}(\mathbf{q}) = \oplus_{e \in I(C)} X_{e,\mathbf{q}}$. □

Propositions 3 and 4 yield a construction of the LDV of $\mathcal{T}^{k+1}$ with a topological traversal of the SCCs of $\mathcal{T}^{k+1}$. It starts at a *hidden* initial state $\mathbf{i}$ with outgoing transitions labelled by 1_{B^*} ending in the initial states of $\mathcal{T}^{k+1}$; $\mathsf{m}(\mathbf{i})$ is the set of PDPDs having exactly one defined entry which is equal to 1_{B^*}.

We express the complexity of the algorithm on the following parameters of $\mathcal{T}$: n (number of states), m (number of transitions) and ℓ (maximal length of the outputs of transitions)[6]. The analysis depends on the following:

Proposition 5. *For every* $\gamma \in \mathsf{m}(\mathbf{q})$*, if* $\gamma_{i,j}$ *is defined, then*[7] $|\gamma_{i,j}| \leq \ell n^{k+1}$. □

Testing whether a PDPD is stable in a SCC with s transitions can be made in time $\mathcal{O}(k^2 \ell n^{k+1} s)$. By Proposition 2, the cardinality of every $\mathsf{m}(\mathbf{q})$ is finite and does not depend on the transducer $\mathcal{T}$. Therefore, the construction of each set $X_{e,\mathbf{q}}$ can be made in $\mathcal{O}(k^2 \ell n^{k+1} s)$ and each operation $\oplus$ in $\mathcal{O}(k^2 \ell n^{k+1})$. It follows that the overall complexity of our algorithm is $\mathcal{O}(\ell n^{k+1} m^{k+1})$. The multiplicative constant hidden in the "big O" comes from the bound established in Proposition 2 and is thus at most $2^{(k+1)^4}$.

Example 1. In this example and in the figures, PDs are represented as upper triangular matrices indexed by $\{p, q\} \times \{q, r\}$ (in this order).

Let $\mathbf{q} = (p, q, r)$ be a state of $\mathcal{T}_1^3$ (Figure 1). The set of PDs in $\mathcal{T}_1^3 \times \mathcal{G}_3$ attached to $\mathbf{q}$ is $X(\mathbf{q}) = \left\{ \left(\begin{smallmatrix} 1 & a^t \\ & a^t \end{smallmatrix}\right) \mid t > 0 \right\} \cup \left\{ \left(\begin{smallmatrix} a^t & 1 \\ & \overline{a}^t \end{smallmatrix}\right) \mid t > 0 \right\}$ (see Figure 2). The set of traverses of $X(\mathbf{q})$ is $\mathrm{tv}(X(\mathbf{q})) = \left\{ \left(\begin{smallmatrix} 1 & 1 \\ & a^t \end{smallmatrix}\right) \mid t > 0 \right\} \cup \left\{ \left(\begin{smallmatrix} 1 & 1 \\ & \overline{a}^t \end{smallmatrix}\right) \mid t > 0 \right\} \cup \left\{ \left(\begin{smallmatrix} 1 & 1 \\ & \bot \end{smallmatrix}\right) \right\}$. There is only one minimal one: $\mathsf{m}(\mathbf{q}) = \left\{ \left(\begin{smallmatrix} 1 & 1 \\ & \bot \end{smallmatrix}\right) \right\}$. This is the value of $\mathbf{q}$, also obtained by applying the operation $\oplus$ to the PDPDs $\left(\begin{smallmatrix} 1 & \bot \\ & \bot \end{smallmatrix}\right)$ and $\left(\begin{smallmatrix} \bot & 1 \\ & \bot \end{smallmatrix}\right)$ incoming in the SCC of $\mathbf{q}$ (see Figure 3).

4 Deciding Finite Valuedness

Weber's proof for Theorem 2 [3] is in two steps: first, the bounded valuedness is characterised by three conditions on the computations of the transducer $\mathcal{T}$

[6] Recall that the valuedness k is considered as a constant.

[7] Recall that $\gamma_{i,j}$, if defined, is a word in $B^* \cup \overline{B}^*$; $|\gamma_{i,j}|$ is as usual the length of it.

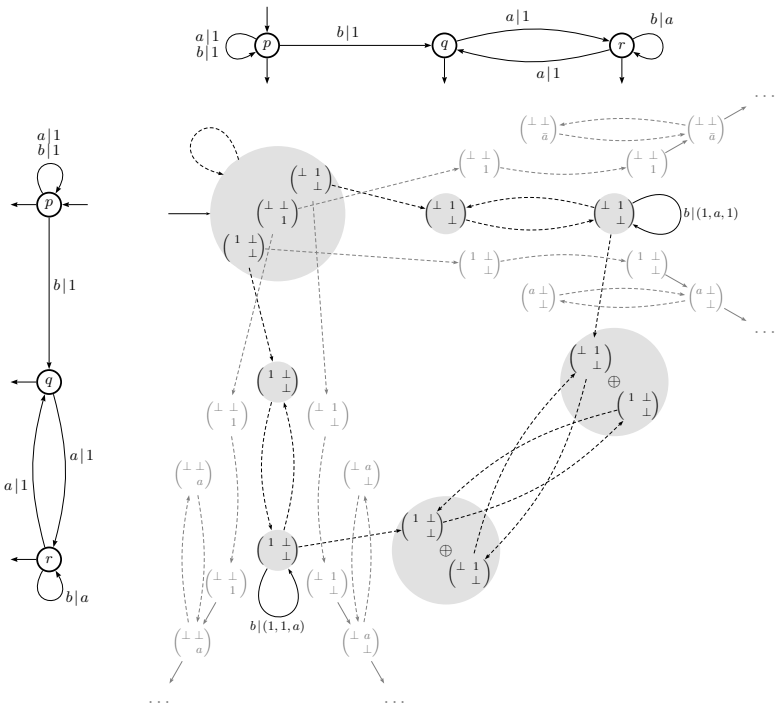

Fig. 3: The values $\mathsf{m}(\mathbf{q})$ for the states of $\mathcal{T}_1^3$ (filled regions) accessible from (p, p, p) and co-accessible to (p, q, r). Dashed transitions have output equal to $(1_{B^*}, 1_{B^*}, 1_{B^*})$.

(Theorem 4); next, it is shown that these conditions can be tested by means of a construction with the underlying graph of $\mathcal{T}$. The proof is difficult due in part to the fact that besides the decidability it gives an upper bound for the valuedness.

Our proof is akin to Weber's one, but on the other hand is different in both steps. We first describe other conditions, C1 and C2 (Theorem 5), and prove that they characterise the bounded valuedness. That stating these new conditions is useful comes from the fact that they are well-fitted with the use of a construction for transducers which we defined in [8], the *lag separation covering*. This construction together with the characterisation of bounded ambiguity for $\mathbb{N}$-automata due to Mandel and Simon (Theorem 7) yields a straightforward proof that C1 and C2 imply the bounded valuedness of $\mathcal{T}$: if $\mathcal{T}$ satisfies C1 and C2, then with the construction of a lag separation covering on $\mathcal{T}$ we obtain a transducer which is equivalent to $\mathcal{T}$, and whose underlying input automaton, say $\mathcal{A}$, satisfies the conditions S1 and S2 in Theorem 7; thus, $\mathcal{A}$ realises a series which is bounded by some integer k; as the number of outputs in $\mathcal{T}$ for every input word u is at most the number of successful computations in $\mathcal{A}$ labelled by u, the valuedness of $\mathcal{T}$ is at most k. The condition C1 is easily testable. In order to test C2, the LDA will be useful again, and we describe in Section 4.2 a condition on $\mathcal{T}^3 \times \mathcal{G}$ equivalent to C2 which can be tested in polynomial time. It turns out that the product $\mathcal{T}^3 \times \mathcal{G}$ captures the technicalities of the constructions underlying Weber's algorithm (Section 3 of [3]).

4.1 A Characterisation of Bounded Valuedness

Weber's conditions for bounded valuedness are better described with the help of Figure 4. Let us call a computation $p \to p \to q \to q$ with $p \neq q$ a *dumbbell-computation*, and a computation such as on the right of Figure 4 a *W-computation*.

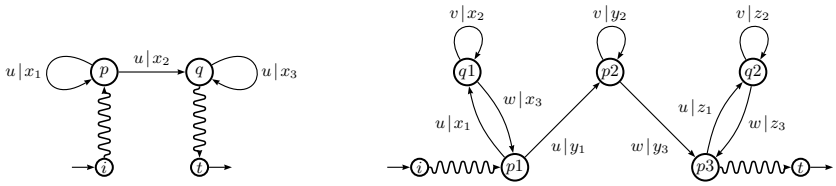

Fig. 4: A dumbbell-computation and a W-computation.

Theorem 4 (Weber [3]). *A trim transducer $\mathcal{T}$ is bounded valued iff:*[8] ***W1)*** *$\mathcal{T}$ does not contain co-terminal*[9] *circuits with same input and distinct outputs;* ***W2)*** *$\mathcal{T}$ does not contain a dumbbell-computation $p \xrightarrow{u|x_1} p \xrightarrow{u|x_2} q \xrightarrow{u|x_3} q$ with $x_1x_2 \neq x_2x_3$;* ***W3)*** *$\mathcal{T}$ does not contain a W-computation with $|x_1| \neq |y_2|$.* □

It is not so difficult to see that these three conditions are necessary for the bounded valuedness. The substance of the theorem is that they are sufficient.

The idea of our new conditions is to adjoin a restriction on the *lag* between the computations which allows to capture W2 and W3 on a single statement.[10]

Theorem 5. *A trim transducer $\mathcal{T}$ with n states and output lengths bounded by ℓ is bounded valued iff:* ***C1)*** *$\mathcal{T}$ does not contain a circuit which contain co-terminal transitions $p \xrightarrow{a|u} q$ and $p \xrightarrow{a|v} q$ such that $u \neq v$;* ***C2)*** *$\mathcal{T}$ does not contain a dumbbell-computation $c_1 : p \xrightarrow{u|x_1} p, c_2 : p \xrightarrow{u|x_2} q, c_3 : q \xrightarrow{u|x_3} q$ where either $x_1x_2 \neq x_2x_3$ or $x_1x_2 = x_2x_3$ and $\langle c_1c_2,\, c_2c_3\rangle > \ell n^3$.*

If the valuedness of $\mathcal{T}$ is bounded, clearly the condition C1 must hold, and every $c_1c_2c_3$ in C2 must satisfy $x_1x_2 = x_2x_3$. The proof that every dumbbell-computation must satisfy $\langle c_1c_2,\, c_2c_3\rangle \leq \ell n^3$ is a pumping argument showing that such a lag would imply a W-computation which does not satisfies W3.

The proof of the sufficiency of the conditions C1 and C2 is straightforward with the use of two tools. The first one is the lag separation covering of $\mathcal{T}$, a

[8] Weber's conditions are slightly different (but equivalent), for W1 and W2 are stated together. We chose other presentation in order to make clear the comparison with the statements to come.

[9] With same origin and same end.

[10] The notation $\langle c,\, d\rangle$ in this statement stands for the *lag* between two computations c and d and is defined in [8].

construction parametrised by an integer $N > 0$ of a new and larger transducer $\mathcal{U}_N$ with a morphism from $\mathcal{U}_N$ to $\mathcal{T}$ inducing a bijection between their successful computations.[11] This covering allows to avoid pairs of computations such that the differences of lengths of outputs along them (their "lag") are bounded by N:

Theorem 6 ([8]). *For every $N > 0$, the lag separation covering $\mathcal{U}_N$ contains a subtransducer $\mathcal{V}_N$ equivalent to $\mathcal{T}$ where distinct successful computations with same label have lag larger than N.* □

The second tool is a classical characterisation of boundedness of $\mathbb{N}$-automata:

Theorem 7 (Mandel-Simon [13], Seidl-Weber [14]). *A trim $\mathbb{N}$-automaton $\mathcal{A}$ realises a bounded $\mathbb{N}$-series iff:*[12] ***S1)*** *$\mathcal{A}$ does not contain a circuit which contains a transition with multiplicity greater than 1;* ***S2)*** *$\mathcal{A}$ does not contain a dumbbell-computation.* □

The idea is to show, starting from Theorem 6, that if $\mathcal{T}$ satisfies C1 and C2, then the underlying input automaton $\mathcal{A}$ of $\mathcal{V}_N$ (with $N = \ell n^3$) satisfies S1 and S2. Now, by Theorem 7, $\mathcal{A}$ is of bounded ambiguity, and thus the valuedness of $\mathcal{V}_N$ (and that of $\mathcal{T}$, for $\mathcal{V}_N$ is equivalent to it) is bounded.

4.2 Testing the Characterisation in $\mathcal{T}^3 \times \mathcal{G}$

The condition C1 in Theorem 5 is easily testable. The substance of our algorithm is a characterisation of C2 within the product of $\mathcal{T}^3$ by the LDA $\mathcal{G}$. In Section 3.2, we defined $\mathcal{T}^3 \times \mathcal{G}_3$, the LDA applied to every pair of projections of $\mathcal{T}^3$. The product $\mathcal{T}^3 \times \mathcal{G}$ is defined likewise but in this case $\mathcal{G}$ acts on a single pair of projections, the first and the second one. Let $\mathcal{W}$ be the part of $\mathcal{T}^3 \times \mathcal{G}$ consisting of states accessible from some state of form $((p, p, q), 1_{B^*})$ and co-accessible to some state of form $((p, q, q), v)$ (where p and q are distinct states of $\mathcal{T}$).

Lemma 1. *The transducer $\mathcal{T}$ (with n states and lengths of outputs bounded by ℓ) satisfies C2 iff for every state $((r, s, t), w)$ of $\mathcal{W}$, $w \neq \mathbf{0}$ and $|w| \leq \ell n^3$.* □

Thus, $\mathcal{W}$ allows to test C2. But this subtransducer seems to be very large, for there are exponentially many words of length at most ℓn^3. In order to obtain a polynomial time complexity, the idea is to consider only the *harder* part of $\mathcal{W}$, the subtransducer $\mathcal{W}'$ consisting of states which are co-accessible to some circuit whose output (x, y, z) is such that either $x \neq 1_{B^*}$ or $y \neq 1_{B^*}$:

Lemma 2. *If all the states of $\mathcal{W}'$ satisfy the conditions in Lemma 1, then the same is true for the states of $\mathcal{W}$.* □

It turns out that this part is not so large due to the following lemma[13]:

[11] For the definition of *covering of automata* see [18].

[12] The original statement in [13] reads instead of S1 that S1': $\mathcal{A}$ contains neither a circuit with multiplicity greater than 1 nor distinct co-terminal circuits with the same input. In the presence of S2 both formulations are equivalent.

[13] Let us note the similarity of this lemma with a critical property to establish the polynomial complexity of the procedure given in [9] to test the sequentiality of transducers.

Lemma 3. *In states of $\mathcal{W}'$ projecting on a same state of $\mathcal{T}^3$ the words in the second component must be prefix of a common word or C2 is not satisfied.* □

Thus, in order to construct $\mathcal{W}'$ we can maintain for every state (r, s, t) of $\mathcal{T}^3$ only two words of length at most ℓn^3, a positive and a negative one, whose prefixes represent the states of $\mathcal{W}'$ already constructed. Each prefix implies a traversal of $\mathcal{T}^3$, thus the complexity of our algorithm to test the bounded valuedness of $\mathcal{T}$ is $\mathcal{O}(\ell n^3(n^3 + m^3))$ (where m is the number of transitions of $\mathcal{T}$).

References

1. Schützenberger, M.P.: Sur les relations rationnelles. In: Automata Theory and Formal Languages, 2nd GI Conference. Volume 33 of Lecture Notes in Computer Science. (1975) 209–213
2. Eilenberg, S.: Automata, Languages, and Machines. Volume A. Academic Press (1974)
3. Weber, A.: On the valuedness of finite transducers. Acta Informatica **27**(8) (1989) 749–780
4. Gurari, E., Ibarra, O.: A note on finite-valued and finitely ambiguous transducers. Mathematical Systems Theory **16** (1983) 61–66
5. Culik, K., Karhumäki, J.: The equivalence of finite valued transducers (on HDT0L languages) is decidable. Theoretical Computer Science **47**(1) (1986) 71–84
6. Weber, A.: Decomposing finite-valued transducers and deciding their equivalence. SIAM Journal on Computing **22**(1) (1993) 175–202
7. Weber, A.: Decomposing a k-valued transducer into k unambiguous ones. RAIRO Informatique Théorique et Applications **30**(5) (1996) 379–413
8. Sakarovitch, J., de Souza, R.: On the decomposition of k-valued rational relations. In Albers, S., Weil, P., eds.: Proceedings of STACS 2008. (2008) 621–632 arXiv:0802.2823v1, available from `http://stacs-conf.org` (to appear in Theory of Computing Systems).
9. Béal, M.P., Carton, O., Prieur, C., Sakarovitch, J.: Squaring transducers: an efficient procedure for deciding functionality and sequentiality. Theoretical Computer Science **292** (2003) 45–63
10. de Souza, R.: On the decidability of the equivalence of k-valued transducers. Submitted
11. Gurari, E., Ibarra, O.: The complexity of decision problems for finite-turn multicounter machines. Journal of Computer and System Sciences **22**(2) (1981) 220–229
12. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley (2001)
13. Mandel, A., Simon, I.: On finite semigroups of matrices. Theoretical Computer Science **5**(2) (October 1977) 101–111
14. Weber, A., Seidl, H.: On the degree of ambiguity of finite automata. Theoretical Computer Science **88** (1991) 325–349
15. Sakarovitch, J., de Souza, R.: On the decidability of finite valuedness of transducers. Manuscript, `http://www.infres.enst.fr/~rsouza/DFV.pdf`.
16. Berstel, J.: Transductions and Context-Free Languages. B. G. Teubner (1979)
17. Sakarovitch, J.: Éléments de théorie des automates. Vuibert (2003) English translation: *Elements of Automata Theory*, Cambridge University Press, to appear.
18. Sakarovitch, J.: A construction on finite automata that has remained hidden. Theoretical Computer Science **204**(1–2) (1998) 205–231