On the decomposition of k-valued rational relations

Jacques Sakarovitch, Rodrigo de Souza

Symposium on Theoretical Aspects of Computer Science 2008 (Bordeaux), pp. 621-632
www.stacs-conf.org

ON THE DECOMPOSITION OF k-VALUED RATIONAL RELATIONS

JACQUES SAKAROVITCH [1] AND RODRIGO DE SOUZA [2]

[1] LTCI, ENST/CNRS, Paris (France)
E-mail address: sakarovitch@enst.fr

[2] ENST, 46, rue Barrault, 75634 Paris Cedex 13 (France)
E-mail address: rsouza@enst.fr

Abstract. We give a new, and hopefully more easily understandable, structural proof of the decomposition of a k-valued transducer into k unambiguous functional ones, a result established by A. Weber in 1996. Our construction is based on a lexicographic ordering of computations of automata and on two coverings that can be build by means of this ordering. The complexity of the construction, measured as the number of states of the transducers involved in the decomposition, improves the original one by one exponential. Moreover, this method allows further generalisation that solves the problem of decomposition of rational relations with bounded length-degree, which was left open in Weber's paper.

Extended abstract

1. Introduction

This communication is part of a complete reworking[1] and rewriting of the theory of k-valued rational relations and transducers which puts it in line with the theory of rational functions (1-valued rational relations) and functional transducers and makes it appear as a natural generalisation of the latter not only at the level of the results — as we recall in the next paragraph — but also *at the level of proofs*.

It is decidable whether a transducer is functional (originally due to Schützenberger [13]); as a consequence, the equivalence of functional transducers is decidable, and, above all, every functional transducer is equivalent to an unambiguous one [5]. These results generalise in a remarkable way to bounded valued rational relations and transducers. It is decidable whether the image of every word by a given transducer is bounded (Weber [14]), it is decidable whether it is bounded by a given integer k (Gurari and Ibarra [6]), every k-valued transducer is equivalent to the sum of k functional (and thus unambiguous) ones (Weber [15]) and the equivalence of k-valued transducers is decidable (Culik and Karhumäki [4]).

It is noteworthy that all the results just quoted for functional transducers are now (if not in the original papers) established by means of constructions conducted on the

1998 ACM Subject Classification: F.1.1, F.4.3.

Key words and phrases: rational relation, k-valued transducer, unambiguous transducer, covering of automata.

[1]A financial support of CAPES Foundation (Brazilian government) for doctoral studies is gratefully acknowledged by the second author (second in the alphabetical order, as in use in the British and French encyclopedias — not in the Lusitanian ones).

transducers themselves [2,9,11] whereas the corresponding results on k-valued transducers come, in some sense, "from outside" and, what is worse, from a different world for each of them. Gurari and Ibarra's proof for the decidability of the k-valuedness relies on a reduction to the emptiness problem for a class of counter automata, Culik and Karhumäki's one for the decidability of the equivalence appears in the context of the solution of Ehrenfeucht's conjecture on HDTOL languages, and Weber's proof of the decomposition — which we shall discuss more in detail below — is highly combinatorial and still somewhat detached from the transducers.

Our approach for those results are based on constructions which depend directly on the structure of the automata. They give back the subject a full coherence and yield systematically better complexity bounds. This will be illustrated in this paper with a new proof of the decomposition theorem which we restate below as Theorem 1.1. In [12] we give a new proof for the decidability of the k-valuedness.

Theorem 1.1 (Weber [15])**.** *Every k-valued transducer $\mathcal{T}$ can be effectively decomposed into a sum of k (unambiguous) functional transducers.*[2]

Our proof for Theorem 1.1 differs from the original one by three aspects. First, Weber's proof is generally considered as very difficulty to follow, whereas ours is hopefully simpler. Second, Weber's construction results in k transducers whose number of states is a double exponential on the number of states of $\mathcal{T}$, whereas we obtain a decomposition of single exponential size. Third and finally, our method allows to solve the problem, posed by Weber, of the decomposition of bounded length-degree rational relations with a more general statement (in Weber's question, θ is the length morphism):

Theorem 1.2. *Let $\tau : A^* \to B^*$ be a finite image rational relation and $\theta : B^* \to C^*$ a morphism such that the composition $\tau\theta$ is k-valued.*[3] *Every transducer $\mathcal{S}$ realising τ can be effectively decomposed into k transducers whose compositions with θ are functions.*

Our proof makes use twice of the notion of covering of automata. A covering of an automaton[4] $\mathcal{A}$ is an *expansion* of $\mathcal{A}$: a new automaton $\mathcal{B}$ whose states and transitions map to those of $\mathcal{A}$, preserving adjacency and labels of transitions. Moreover, the outgoing transitions of every state of $\mathcal{B}$ map one-to-one to those of the projection, which implies a bijection between the successful computations of $\mathcal{A}$ and $\mathcal{B}$. Typically, $\mathcal{B}$ is larger than $\mathcal{A}$, for several states can have the same image. This allows to choose certain subsets of the computations of $\mathcal{A}$ by erasing parts of $\mathcal{B}$.

The two coverings we are going to define are based on a lexicographic ordering on the computations. This method can be seen as a conceptual generalisation of the one used by H. Johnson in order to build a lexicographic selection of deterministic rational relations [7, 8].

The first construction, explained in Section 3.3, is what we call the *lag separation covering* $\mathcal{U}_N$ of a (real time) transducer $\mathcal{T}$. It is parameterised by an integer N, and roughly speaking allows to distinguish between computations with same input and same output and

[2]By "decomposed" we mean that the relation realised by $\mathcal{T}$ and the union of the relations realised by the k transducers are the same.

[3]We write functions and relations using a postfix notation: $x\tau$ is the image of x by the relation τ and thus the composition of relations is written by left-to-right concatenation. Let us recall that the rational relations are closed under composition [5].

[4]As we shall define in Section 2, transducers are automata of a certain kind.

whose *lag*[5] is bounded by N. If $\mathcal{T}$ is k-valued, we show that for a certain N, $\mathcal{U}_N$ contains a subtransducer $\mathcal{V}_N$ which is equivalent to $\mathcal{T}$ and input-k-ambiguous[6] (Proposition 4.2).

The second construction (Section 3.2) is what we call the *multi-skimming covering* of an $\mathbb{N}$-automaton. It proves the following *multi-skimming theorem* for $\mathbb{N}$-rational series:

Theorem 1.3. *Let $\mathcal{A}$ be a finite $\mathbb{N}$-automaton with n states realising the series s. There exists an infinite $\mathbb{N}$-covering $\mathcal{B}$ of $\mathcal{A}$ such that for every integer $k > 0$, there exists a finite $\mathbb{N}$-quotient $\mathcal{B}_k$ of $\mathcal{B}$ which satisfies: $\mathcal{B}_k$ is an $\mathbb{N}$-covering of $\mathcal{A}$ with at most $n\,(k+1)^n$ states; for every i, $0 \leq i < k$, there exists an unambiguous subautomaton $\mathcal{B}_k^{(i)}$ of $\mathcal{B}_k$ which recognises the support of $s - i$; there exists a subautomaton $\mathcal{D}_k$ of $\mathcal{B}_k$ whose behaviour is $s - k$.*

Here $s - k$ is the series obtained from s by subtracting k to every coefficient larger than k and assigning 0 to the others. In particular, Theorem 1.3 says that, if $\mathcal{A}$ is a k-ambiguous automaton, then there exists a finite covering $\mathcal{B}_k$ of $\mathcal{A}$ and unambiguous subautomata $\mathcal{B}_k^{(0)}, \ldots, \mathcal{B}_k^{(k-1)}$ of $\mathcal{B}_k$ such that the successful computations of the union $\bigcup_i \mathcal{B}_k^{(i)}$ are in bijection with those of $\mathcal{A}$. Of course, it is not new that $s - k$ is a $\mathbb{N}$-rational series when s is. This is an old result by Schützenberger which can be proved by iterated applications of Eilenberg's Cross-Section Theorem [5], or of the construction given in [11]. But all these methods yield an automaton whose size is a tower of exponentials of height k. Theorem 1.3 thus answers a problem left open in [11] with a solution which is better than the one that was conjectured there.

These coverings together give in two steps a decomposition of a k-valued transducer $\mathcal{T}$. First, the lag separation covering of $\mathcal{T}$ yields a transducer $\mathcal{V}_N$ equivalent to $\mathcal{T}$ and whose underlying input automaton, say $\mathcal{A}$, is k-ambiguous. Next, the multi-skimming covering applied to $\mathcal{A}$ yields, as stated in the discussion after Theorem 1.3, k unambiguous automata $\mathcal{B}_k^{(i)}$; the successful computations of the union of the $\mathcal{B}_k^{(i)}$ are in bijection with those of $\mathcal{A}$ and, as the transitions of every $\mathcal{B}_k^{(i)}$ map on those of $\mathcal{A}$, one can "lift" on them the output of the corresponding transitions of $\mathcal{V}_N$: one thus obtain k unambiguous functional transducers $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$ decomposing $\mathcal{T}$ (see Figure 1).

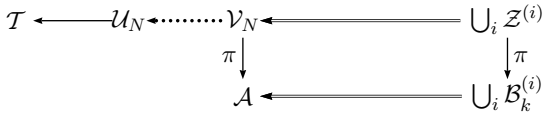


Figure 1: Decomposition of a k-valued transducer $\mathcal{T}$. The simple edge stands for a covering; the dotted one represents an input-k-ambiguous subautomaton; the double ones are immersions; π is a projection on the underlying input automaton.

Our proof goes so to speak in the opposite way that Weber's one: our first step is to build an input-k-ambiguous transducer from which the decomposition is extracted, whereas the existence of such a transducer is viewed in [15] as a consequence of the decomposition. Moreover, although both proofs have a very general idea in common — a classification of computations from which at most k successful ones can be distinguished, for every input word — the way we do this is completely different. Indeed, Weber's decomposition is

[5]To be defined in the body of the paper.

[6]When it comes to ambiguity in transducers, we distinguish between *input-ambiguity* (called ambiguity in most of the references) and ambiguity of the transducer (which allows to define ambiguity for relations).

extracted from the strongly connected components of a graph built on a preliminary decomposition of $\mathcal{T}$ into exponentially many functional transducers. We perform a selection among the computations of $\mathcal{T}$ according to a lexicographic ordering on the transitions.

A rough estimation of the complexity (number of states, as a function on the number of states of $\mathcal{T}$) of this two-step procedure gives a double exponential: one for the lag separation covering $\mathcal{U}_N$ and other for the multi-skimming covering. However, a major feature of this construction is that every computation in the newly built automata corresponds to a computation in the original transducer — this is basically what we mean by structural proof — which allows to track down the usefulness of every newly created state. Then, a careful analysis shows that restricting the constructions to the *trim* parts of the automata the number of obtained states is bounded by $2^{\mathcal{O}(hLk^4n^{k+4})}$ states, where n is the number of states of $\mathcal{T}$, h is the size of the output alphabet and L is the maximal length of the outputs of the transitions (Section 4.2). This is to be compared with the size of Weber's decomposition described in [15], 2^{2^P}, where $P = p(n + L + h + k)$ is a polynomial whose degree and coefficients do not seem to be easily derived from the arguments developed there.

The proof of Theorem 1.2 starts with the construction of k unambiguous transducers decomposing the k-valued relation $\tau\theta$. Next, we show that these transducers induce a decomposition of the set of successful computations of $\mathcal{S}$. This gives a new set of k finite transducers, not necessarily unambiguous, which decompose $\mathcal{S}$ (Section 4.3).

Finally, let us note that — as explained in [15] — the improvement in the size of the decomposition from double to single exponential yields an improvement of the same order for the complexity of the decision of the equivalence of k-valued transducers.

2. Preliminaries

We basically follow the definitions and notation in [1, 5, 10].

The semiring of the nonnegative integers is denoted by $\mathbb{N}$, the set of words over a finite alphabet A (the free monoid over A) by A^* and the empty word by 1_{A^*}. The length of $u \in A^*$ is denoted by $|u|$. The powerset of a set X is denoted by $\mathfrak{P}(X)$.

An *automaton* over a monoid M is a labelled directed graph $\mathcal{A} = (Q, M, E, I, T)$ defined by the set Q of vertices, called *states* and E of edges, called *transitions*, together with two subsets I and T of Q, the initial and final states respectively. Every transition e in E is associated with a triple (p, m, q) of $Q \times M \times Q$, specifying its origin, label and end. Note that we shall explicitly consider cases where *distinct transitions* have the *same* origin, label and end, even though we take the liberty to write $e : p \xrightarrow{m} q \in E$ meaning a transition e associated with (p, m, q). The automaton $\mathcal{A}$ 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 as $p_0 \xrightarrow[\mathcal{A}]{m_1 \dots m_l} p_l$. Its label is $m_1 \dots m_l \in M$ and its length 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. These sets are the family $\operatorname{Rat} M$ of the *rational subsets* of M.

A state of $\mathcal{A}$ is *accessible* if it can be reached by a computation starting at some state of I, and *co-accessible* if some state of T can be reached from it. The state is *useful* if is both accessible and co-accessible, and we say that $\mathcal{A}$ is *trim* if every state is useful.

If M is a free monoid A^* and the labels of transitions are letters, then $\mathcal{A}$ is a classical automaton over A; we write in this case $\mathcal{A} = (Q, A, E, I, T)$. If M is a product $A^* \times B^*$, then every transition is labelled by a pair denoted as $u|x$ and consisting of an input word

$u \in A^*$ and an output one $x \in B^*$; and $\mathcal{A}$ is a *transducer* realising a *rational relation* from A^* to B^*. The image of a word $u \in A^*$ by a transducer is the set of outputs of successful computations whose input is u. The transducer is called *k-valued*, for $k \in \mathbb{N}$, if the cardinality of the image of every input word is at most k.

By using classical constructions on automata, every transducer can be transformed into a *real-time* one: a transducer whose labels are of form $a|K$, where a is a *letter*, $K \in \operatorname{Rat} B^*$ and I and T are functions from Q to $\operatorname{Rat} B^*$ [5, 10]. For finite image relations we may suppose that the transitions read a letter and output a single word, and the image of every final state is 1_{B^*}. In this case, the transducer is denoted rather as $\mathcal{T} = (Q, A, B^*, E, I, T)$.

The *underlying input automaton* $\mathcal{A}$ of a real-time transducer $\mathcal{T}$ is the (classical) automaton obtained by forgetting the output of the transitions and replacing the functions I and T by their domains. The behaviour of $\mathcal{A}$ is the domain of the relation realised by $\mathcal{T}$.

Figure 2: A 2-valued real-time transducer $\mathcal{T}$ over $\{a\}^* \times \{b\}^*$ and its (infinitely ambiguous) underlying input automaton. The behaviour of $\mathcal{T}$ is the relation defined by $(1_{A^*})|\mathcal{T}| = 1_{B^*}$ and $(a^n)|\mathcal{T}| = \{b^n, b^{n+1}\}$ for $n > 0$.

An $\mathbb{N}$-*automaton* is an automaton labelled by letters with multiplicities in $\mathbb{N}$ attached to the transitions and to initial and final states. It realises an $\mathbb{N}$-*rational series*: a function $s : A^* \to \mathbb{N}$ which assigns to $u \in A^*$ a multiplicity given by summing the multiplicites (product of the multiplicities of transitions) of the successful computations labelled by u.

Every $\mathbb{N}$-automaton or real-time transducer can be described by a *matrix representation* (λ, μ, ν), where $\lambda \in S^Q$ ($\nu \in S^Q$) is a row (column) vector for the multiplicities of the initial (final) states, $\mu : A^* \to S^{Q \times Q}$ is a morphism, $S = \mathbb{N}$ for $\mathbb{N}$-automata and $S = \operatorname{Rat} B^*$ for transducers. The behaviour can be expressed by the function which maps every $u \in A^*$ to $\lambda \cdot u\mu \cdot \nu$. This leads to call *dimension* the set of states of an automaton.

It will be useful to consider $\mathbb{N}$-automata whose transitions are *characteristic*, that is, with multiplicity 1. Every $\mathbb{N}$-automaton can be transformed into such a one by splitting every transition with multiplicity $l > 0$ into a set of l characteristic ones (Figure 3).

Figure 3: An $\mathbb{N}$-automaton $\mathcal{C}_1$ over $\{a, b\}$, on the right-hand side with characteristic transitions obtained by splitting multiplicities. If $u \in \{a, b\}^*$ is viewed as the writing in the binary system of an integer $\overline{u}$ by interpreting a as 0 and b as 1, then $u|\mathcal{C}_1| = \overline{u}$.

A *morphism* from $\mathcal{B} = (R, M, F, J, U)$ to $\mathcal{A} = (Q, M, E, I, T)$, denoted by $\varphi : \mathcal{B} \to \mathcal{A}$, is a pair of mappings $R \to Q$ and $F \to E$, both denoted by φ, such that $J\varphi \subseteq I$, $U\varphi \subseteq T$ and for every $e \in F$, if e is associated with (p, m, q), then $e\varphi$ is associated with $(p\varphi, m, q\varphi)$. We say that φ is a *covering* if φ induces a bijection between the outgoing transitions of p and $p\varphi$, I is in bijection with J and $T\varphi^{-1} = U$. An *immersion* is by definition a subautomaton of a covering. These conditions imply that every successful computation of $\mathcal{B}$ maps to a

successful computation of $\mathcal{A}$, and thus $|\mathcal{B}| \subseteq |\mathcal{A}|$. In the case of coverings, there is indeed a bijection between the successful computations and thus $|\mathcal{B}| = |\mathcal{A}|$ [9].

A covering of the split form of an $\mathbb{N}$-automaton $\mathcal{A}$ is the split form of an $\mathbb{N}$-*covering* of $\mathcal{A}$, see [3, 10, 11] for the definition of the latter. The $\mathbb{N}$-series realised by an $\mathbb{N}$-automata and any of its $\mathbb{N}$-coverings are the same.

3. Lexicographic coverings

The idea of the two coverings we are going to define is to order lexicographically computations of automata, inasmuch as it can be done with words on some alphabet. Here, the alphabet is the set of transitions, and computations are seen as words on it.

3.1. The lexicographic ordering of computations

Let $\mathcal{A} = (Q, A, E, I, T)$ be a classical automaton. Fix a (partial) ordering $\prec$ on E such that transitions are comparable iff they have the same label and origin. This ordering is extended on E^* and thus on the computations of $\mathcal{A}$ in such a way that it can be called a *lexicographic ordering of the computations*: $c = e_1 e_2 \dots e_l e_{l+1} \dots e_n$ and $d = e'_1 e'_2 \dots e'_l e'_{l+1} \dots e'_m$ ($e_i, e'_j \in E$ for $1 \leq i \leq n$ and $1 \leq j \leq m$) are such that $c \prec d$ iff c and d have the same label (thus $m = n$) and there exists l such that $e_i = e'_i$ for $1 \leq i \leq l-1$ and $e_l \prec e'_l$.

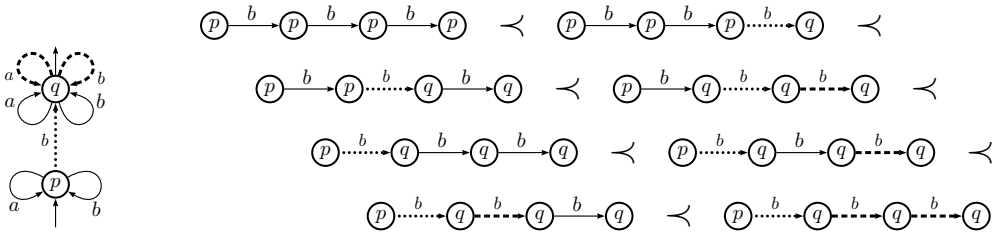


Figure 4: A lexicographic ordering between the computations of $\mathcal{C}_1$ labelled by *bbb* and starting at p. Solid transitions are smaller than the dotted and dashed ones.

The definitions for other kinds of automata are similar but, in order to give them the wanted meaning, a little bit more delicate: for $\mathbb{N}$-automata, the ordering is put on the split form, and for real-time transducers, on the underlying input automaton.[7]

3.2. The multi-skimming covering of an $\mathbb{N}$-automaton

The aim of the multi-skimming covering of an $\mathbb{N}$-automaton $\mathcal{A} = (Q, A^*, E, i, T)$ is to count, for every successful computation, the number of the smaller ones according to $\prec$.

Let $\xi : E \to \mathbb{N}^Q$ be the function from transitions to $\mathbb{N}$-vectors indexed by Q defined by $(e\xi)_r = \operatorname{card}\Big(\{f \in E \mid f : p \xrightarrow{a} r \text{ and } f \prec e\}\Big)$, for $e : p \xrightarrow{a} q \in E$ and $r \in Q$.

Definition 3.1. The multi-skimming covering of $\mathcal{A}$ is the (infinite) $\mathbb{N}$-automaton $\mathcal{B}$ of dimension $Q \times \mathbb{N}^Q$ defined as follows:

[7] In order to ease the explanation, we shall describe the constructions for automata with a single initial state. Computations starting at distinct initial states become ordered by extending $\prec$ to new transitions $i \xrightarrow{1} p$ starting at a "hidden" initial state i, for every $p \in I$. Initial multiplicities can be treated similarly.

- the initial state is $(i, \vec{0})$ (where $\vec{0}$ is the zero vector);
- the final states are $T \times \mathbb{N}^Q$;
- for every $(p, \mathsf{v}) \in Q \times \mathbb{N}^Q$ and every $e : p \xrightarrow{a} q \in E$, $(p, \mathsf{v}) \xrightarrow{a} (q, \mathsf{v} \cdot a\mu + e\xi)$ is a transition of $\mathcal{B}$ (where μ is the morphism of the matrix representation of $\mathcal{A}$). □

It follows from this definition that for every state (p, v) of $\mathcal{B}$, the outgoing transitions of (p, v) are in bijection with those of p. Thus, the projection φ of $\mathcal{B}$ on the first component is an $\mathbb{N}$-covering of $\mathcal{A}$. The property below follows by induction on the length of computations[8]:

Property 3.2. Let $C : (i, \vec{0}) \xrightarrow[\mathcal{B}]{u} (p, \mathsf{v})$ be a computation. For every $q \in Q$, v_q is the number of computations $d : i \xrightarrow[\mathcal{A}]{u} q$ such that $d \prec C\varphi$ (where $C\varphi$ is the projection of C on $\mathcal{A}$). ■

We define as above the (finite) automaton $\mathcal{B}_k$ satisfying Theorem 1.3; the difference is that it counts *until* $k-1$. Let $\mathbb{N}_k = \{0, \ldots, k-1, \omega\}$ be the quotient semiring of $\mathbb{N}$ given by the relation $k = k+1$ (ω is the class of k and plays the role of an infinity). The dimension of $\mathcal{B}_k$ is $Q \times \mathbb{N}_k^Q$; transitions and initial and final states are defined as in Definition 3.1, but the matrix operations $\mathsf{v} \cdot a\mu + e\xi$ are made in $\mathbb{N}_k$. The morphism $\mathbb{N} \to \mathbb{N}_k$ induces an $\mathbb{N}$-quotient $\mathcal{B} \to \mathcal{B}_k$, and as noted, $\mathcal{B}_k$ is an $\mathbb{N}$-covering of $\mathcal{A}$. Figure 5 shows an example.

By induction on the length of computations, we have:

Property 3.3. Let $C : (i, \vec{0}) \xrightarrow[\mathcal{B}_k]{u} (p, \mathsf{v})$ be a computation. For every $q \in Q$, v_q is the number of computations $d : i \xrightarrow[\mathcal{A}]{u} q$ such that $d \prec C\varphi$, if this number is smaller than k, or it is ω otherwise. ■

Proof of Theorem 1.3. In view of Property 3.3, we can obtain the subautomata $\mathcal{B}_k^{(i)}$ of $\mathcal{B}_k$ by erasing the condition of being final of some final states of $\mathcal{B}_k$: each $\mathcal{B}_k^{(i)}$ is defined by choosing as final only the states $(p, \mathsf{v}) \in T \times \mathbb{N}_k^Q$ such that $\sum_{q \in T} \mathsf{v}_q = i$; $\mathcal{D}_k$ is the subautomaton of $\mathcal{B}_k$ defining as final the states $(p, \mathsf{v}) \in T \times \mathbb{N}_k^Q$ such that $\sum_{q \in T} \mathsf{v}_q = \omega$. ■

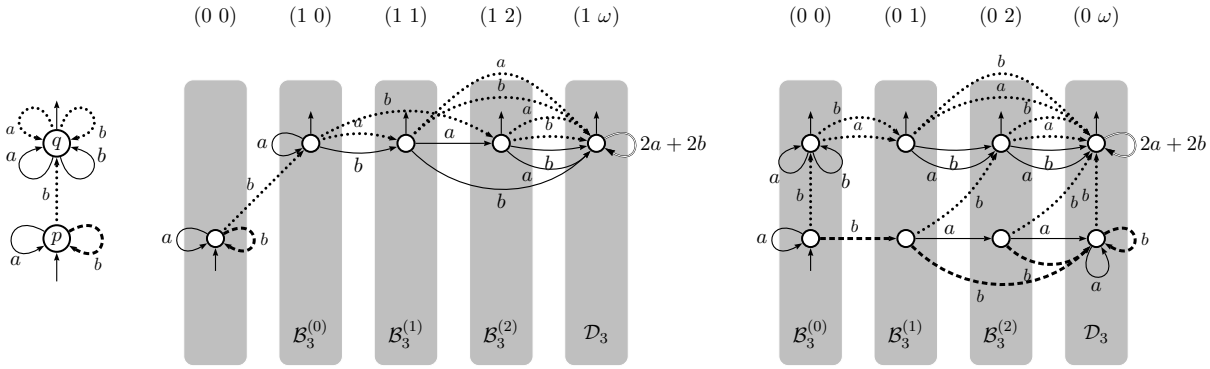
3.3. The lag separation covering of a real-time transducer

Let $\mathcal{T} = (Q, A, B^*, E, i, T)$ be a real-time transducer. We aim with the lag separation covering of $\mathcal{T}$ at a selection between computations of this transducer with *same input and same output* (stated in Property 3.9). This will be useful in Section 4 to construct a input-k-ambiguous transducer from a k-valued one.

It is not possible in general to build a finite expansion which allows to select exactly one computation for each pair of words in the relation realised by the transducer, for this would lead to an unambiguous transducer and there exist rational relations which are inherently ambiguous. The idea is to fix a parameter N and compare only computations such that *the differences of lengths of outputs along them (their "lag") are bounded by* N.

At first, let us recall the *Lead or Delay action*, defined in [2] to describe differences of words. We restate it in a slightly different form, based on the *free group* $F(B)$ generated by B: the quotient of $(B \cup \overline{B})^*$ by the relations $x\overline{x} = \overline{x}x = 1_{B^*}$ ($x \in B$), where $\overline{B}$ a disjoint copy of B. The inverse of $u \in B^*$, denoted by $\overline{u}$, is the mirror image of u with barred letters.

[8] Computations of coverings will be represented with capital letters.



(a) With the dashed transition starting at p being smaller than the dotted one...

(b) ... and with the other ordering on the outgoing transitions of p.

Figure 5: The multi-skimming covering at layer $k = 3$ for $\mathcal{C}_1$ with two different orderings of the transitions starting at p. States of the covering are pairs (r, v), where r is a state of the automaton (horizontal projection) and v is an $\mathbb{N}_3$-vector indexed by $\{p, q\}$ in that order (vertical projection). Solid transitions leaving q are smaller than the dotted ones in both coverings. The double transitions stand for four transitions. The automata $\mathcal{B}_3^{(i)}$ recognising the support of $s-i$ and $\mathcal{D}_3$ recognising $s-3$ are given by keeping as final exactly one final state at the indicated columns.

We denote by $\mathbf{1}$ the empty word of $F(B)$ (which is the class of the empty word of B). Let $\Delta = B^* \cup \overline{B}^* \cup \{\mathbf{0}\}$, where $\mathbf{0}$ is a new element, a zero, not in $F(B)$, and $\rho : F(B) \cup \{\mathbf{0}\} \to \Delta$ be the function $w\rho = w$, if $w \in \Delta$, and $w\rho = \mathbf{0}$ otherwise.

Definition 3.4. The Lead or Delay Action of $B^* \times B^*$ on Δ is defined by $w \cdot (x, y) = (\overline{x}\, w\, y)\rho$, $w \in \Delta$, $(x, y) \in B^* \times B^*$ (the product is taken with the rules $\mathbf{0}\, x = x\, \mathbf{0} = \mathbf{0}$). □

Intuitively, $\mathbf{1} \cdot (x, y)$ represents the "difference" of the words x and y, being a positive word if x is a prefix of y (the *lead* of y with respect to x), a negative word if y is a prefix of x (the *delay* of y with respect to x), and $\mathbf{0}$ if x and y are not prefixes of a common word.

Definition 3.5. Let $c : p \xrightarrow{u|x} q$ and $d : p' \xrightarrow{u|y} q'$ be two computations of $\mathcal{T}$ with the same input u. As $\mathcal{T}$ is a real-time transducer, c and d have the same length. We define their *Lead or Delay*, denoted by $\mathrm{LD}(c, d)$, as the element $\mathbf{1} \cdot (x, y)$ of Δ, and if $\mathrm{LD}(c, d) \neq \mathbf{0}$, their *lag* as the integer $\langle c, d \rangle = \mathsf{max}\{|\mathrm{LD}(c', d')| \mid c', d' \text{ prefixes of } c, d \text{ with the same length}\}$. □

Similarly to the multi-skimming covering, the states of the lag separation covering of $\mathcal{T}$ carry vectors indexed by Q. But in this case the "stored information" is the Lead or Delay between any computation and those which are smaller. Let $\xi : E \to \mathfrak{P}(\Delta)^Q$ be the function given by $(e\xi)_r = \{(\overline{x}y)\rho \mid f : p \xrightarrow{a|y} r \in E,\ f \prec e\}$, for $e : p \xrightarrow{a|x} q \in E$ and $r \in Q$.

Definition 3.6. The lag separation covering of $\mathcal{T}$ is the (infinite) real-time transducer $\mathcal{U} = (R, A, B^*, F, j, U)$ defined by

- $R = Q \times \mathfrak{P}(\Delta)^Q$;
- $j = (i, \vec{0})$ (where $\vec{0}$ is the vector whose entries are all equal to $\emptyset$);
- $U = T \times R$;
- for every $(p, \mathsf{v}) \in R$ and every $e : p \xrightarrow{a|x} q \in E$, $(p, \mathsf{v}) \xrightarrow{a|x} \big(q, (\overline{x} \cdot \mathsf{v} \cdot a\mu + e\xi)\rho\big)$ is a transition in F (where μ is the morphism of the matrix representation of $\mathcal{T}$, $\overline{x} \cdot \mathsf{v}$

is the vector obtained by multiplying on the left every entry of v by $\overline{x}$, and ρ is extended componentwise to vectors in $\mathfrak{P}(\Delta)^Q$). □

As before, for every state (p, v) of $\mathcal{U}$, there is a bijection between the outgoing transitions of (p, v) and those of p: the projection φ of $\mathcal{U}$ on the first component is a covering on $\mathcal{T}$. By induction on the length of computations, we have:

Property 3.7. Let $C : (i, \vec{0}) \xrightarrow{u|x} (p, \mathsf{v})$ be a computation of $\mathcal{U}$. For every state q of $\mathcal{T}$, v_q is the set of Lead or Delay of $C\varphi$ (the projection of C on $\mathcal{T}$) and any computation of $\mathcal{T}$ smaller than $C\varphi$ and which ends in q: $\mathsf{v}_q = \{\mathrm{LD}(C\varphi, d) \mid d : i \xrightarrow[\mathcal{T}]{u|y} q,\ d \prec C\varphi\}$. ■

In order to build the announced selection of computations of $\mathcal{T}$, we define a "bounded" lag separation covering where only the computations with lag bounded by N are compared, so that only words in $\Delta_N = B^{\leq N} \cup \overline{B}^{\leq N}$ are "stored" in the entries of the vectors v. Let $\rho_N : F(B) \to \Delta_N \cup \{\mathbf{0}\}$ be the function defined by $w\rho_N = w$, if $w \in \Delta_N$, and $w\rho_N = \mathbf{0}$ otherwise. The element $\mathbf{0}$ is intentionally omitted from Δ_N in order to simplify the writing of Property 3.8, and in the extension of ρ_N to $\mathfrak{P}(F(B))^Q$ the image of a word not in Δ_N will be seen as the empty set so that for $\mathsf{v} \in \mathfrak{P}(F(B))^Q$, $\mathsf{v}\rho_N$ is a vector in $\mathfrak{P}(\Delta_N)^Q$ (which does not contain $\mathbf{0}$ in any of its entries). We define $\mathcal{U}_N$ as the (finite) transducer constructed as in Definition 3.6, but with states and transitions given by:

$$R = Q \times \mathfrak{P}(\Delta_N)^Q, \qquad \forall\, (p, \mathsf{v}) \in R,\ \forall\, e : p \xrightarrow{a|x} q \in E \quad (p, \mathsf{v}) \xrightarrow{a|x} \big(q, (\overline{x} \cdot \mathsf{v} \cdot a\mu + e\xi)\rho_N\big) \in F.$$

Due to the fact that ρ_N is not a morphism, it is not true in general that $\mathcal{U}$ is a covering of $\mathcal{U}_N$; but $\mathcal{U}_N$ is another covering of $\mathcal{T}$. By induction we have (see Figure 6(a)):

Property 3.8. Let $C : (i, \vec{0}) \xrightarrow{u|x} (p, \mathsf{v})$ be a computation of $\mathcal{U}_N$. For every state q of $\mathcal{T}$, $\mathsf{v}_q = \{\mathrm{LD}(C\varphi, d) \mid d : i \xrightarrow[\mathcal{T}]{u|y} q,\ \ x, y \text{ prefixes of a common word},\ d \prec C\varphi,\ \langle C\varphi, d\rangle \leq N\}$. ■

The wanted selection is a consequence of Property 3.8 and can be stated as follows:

Property 3.9. Let $\mathcal{V}_N$ be the subtransducer of $\mathcal{U}_N$ obtained by removing the property of being final of every state $(p, \mathsf{v}) \in T \times R$ such that $\mathbf{1} \in \mathsf{v}_t$ for some $t \in T$. A computation C of $\mathcal{V}_N$ is successful if, and only if, $C\varphi$ is successful in $\mathcal{T}$ and for every successful computation d of $\mathcal{T}$ smaller than $C\varphi$ with (same input and) same output, $\langle C\varphi, d\rangle > N$. ■

The transducers $\mathcal{T}$ and $\mathcal{V}_N$ are equivalent: if (u, x) is in the behaviour of $\mathcal{T}$, the smallest successful computation of $\mathcal{T}$ labelled by (u, x) is the projection of a successful one in $\mathcal{V}_N$.

The following remark on the trim part of $\mathcal{V}_N$ will be useful for the evaluation of the size of the decomposition (Section 4.2).

Property 3.10. Let $\mathcal{T}$ be a trim and k-valued transducer with n states, and whose output alphabet has h letters. The number of useful states of $\mathcal{V}_N$ is bounded by 2^{2hNk^2n}.

Proof. We write $\mathfrak{P}_{(l)}(X)$ for the set of the subsets with at most l elements of a set X. Clearly, $\mathrm{card}\left(\mathfrak{P}_{(l)}(X)\right) \leq \mathrm{card}\,(X)^{l^2}$. The hypothesis that $\mathcal{T}$ is trim and k-valued together with Property 3.8 imply that the vectors in the useful states of $\mathcal{V}_N$ have in every coordinate at most k words, thus these states belong to $Q \times \mathfrak{P}_{(k)}(\Delta_N)^Q$. The cardinality of this set is at most $n \cdot \left(\mathrm{card}\,(\Delta_N)^{k^2}\right)^n \leq n \cdot \left((2h)^{Nk^2}\right)^n$. This is clearly bounded by 2^{2hNk^2n}. ■

4. Decomposing a k-valued rational relation

As said in the introduction, we first prove a result for k-valued transducers:

Theorem 4.1. *Any k-valued transducer is equivalent to an input-k-ambiguous one.*

This will be established by the lag separation covering: for some adequate N, $\mathcal{V}_N$ is input-k-ambiguous (Proposition 4.2). Next, Theorem 1.1 is proved by applying the multi-skimming covering on the underlying input automaton of $\mathcal{V}_N$ (Section 4.2).

4.1. From a k-valued transducer to an input-k-ambiguous one

Proposition 4.2. *Let $\mathcal{T}$ be a real-time transducer with n states and lengths of outputs of transitions bounded by L. If $\mathcal{T}$ is k-valued, then for $N \geq L\,n^{k+1}$ $\mathcal{V}_N$ is input-k-ambiguous.*

The crux of the proof is a combinatorial property stated in Theorem 2.2 of [15], and restated here as Lemma 4.3. In this lemma, $\mathcal{T}^{k+1}$ is the cartesian product of $\mathcal{T}$ by itself $k+1$ times, a natural generalisation of the squaring of $\mathcal{T}$ defined in [2] to establish the decidability of the functionality of transducers. In $\mathcal{T}^2$, every computation corresponds to a pair of computations of $\mathcal{T}$ *with the same input*; in $\mathcal{T}^{k+1}$, every computation corresponds then to a $(k+1)$ tuple of computations of $\mathcal{T}$ *with the same input* (this construction is heavily used in [12] to give a new proof of the decidability of k-valuedness).

Lemma 4.3 (Weber [15])**.** *If $\mathcal{T}$ is k-valued, then for every successful computation $\mathbf{c}$ of $\mathcal{T}^{k+1}$ there exists a pair i, j of coordinates such that the projections $\mathbf{c}_i$ and $\mathbf{c}_j$ satisfy $\mathrm{LD}(\mathbf{c}_i, \mathbf{c}_j) = 1$ (that is, $\mathbf{c}_i$ and $\mathbf{c}_j$ have the same output) and $\langle \mathbf{c}_i, \mathbf{c}_j \rangle < Ln^{k+1}$.* ∎

A concise proof for Lemma 4.3 can be derived from a property of the Lead or Delay action stated in Lemma 5 of [2]. Although not so long, it is omitted due to space constraints.

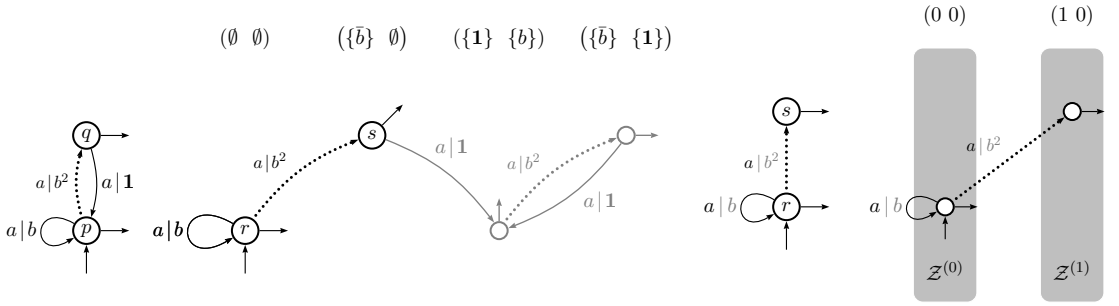
Proof of Proposition 4.2. Fix $N \geq Ln^{k+1}$. By Property 3.9, distinct successful computations of $\mathcal{V}_N$ with the same input either output distinct words or have a lag greater than Ln^{k+1}. Hence $k+1$ distinct successful computations of $\mathcal{V}_N$ with the same input word would project on a $(k+1)$-tuple of computations of $\mathcal{T}$ that contradicts Lemma 4.3. ∎

4.2. Decomposing the input-k-ambiguous transducer $\mathcal{V}_N$

As observed in Section 3.3, $\mathcal{T}$ and $\mathcal{V}_N$ are equivalent (for every N). Thus, a decomposition of $\mathcal{V}_N$ is also a decomposition of $\mathcal{T}$.

Take $N = Ln^{k+1}$ and let $\mathcal{A}$ be the underlying input automaton of $\mathcal{V}_N$. It is straightforward to decompose $\mathcal{V}_N$ by applying the multi-skimming covering on $\mathcal{A}$. By Proposition 4.2, $\mathcal{A}$ is k-ambiguous, hence the multi-skimming covering yields unambiguous automata $\mathcal{B}_k^{(0)}, \ldots, \mathcal{B}_k^{(k-1)}$ which are immersions in $\mathcal{A}$, and whose successful computations are in bijection with those of $\mathcal{A}$. By lifting to the transitions of $\mathcal{B}_k^{(0)}, \ldots, \mathcal{B}_k^{(k-1)}$ the corresponding outputs in $\mathcal{V}_N$ of the projected ones, we obtain unambiguous transducers $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$ whose union is equivalent to $\mathcal{T}$. Figure 6 shows an example with a given ordering for each covering. Other decompositions are obtained by varying these orderings.

The number of states of the decomposition depends on the following parameters of $\mathcal{T}$: n (number of states), h (cardinality of the output alphabet), L (maximal of the lengths of the outputs of transitions) and k (valuedness). We claim:



(a) A lag separation covering $\mathcal{U}_N$ with $N = 1$. The $\mathfrak{P}(\Delta_1)^Q$-vectors (vertical projection) are indexed by $\{p, q\}$, in that order. The dotted transition is larger than the solid one. The input-2-ambiguous (and equivalent to $\mathcal{T}$) subtransducer $\mathcal{V}_1$ is reduced to the states $\{r, s\}$.

(b) The lifted transducers $\mathcal{Z}^{(0)}$ and $\mathcal{Z}^{(1)}$ from a 2-skimming of the input automaton of $\mathcal{V}_1$. The choices of final states yield $|\mathcal{Z}^{(0)}|\colon a^n \mapsto b^n\,(n \geq 0)$ and $|\mathcal{Z}^{(1)}|\colon a^n \mapsto b^{n+1}\,(n > 0)$.

Figure 6: A decomposition of the 2-valued transducer $\mathcal{T}$ of Figure 2.

Property 4.4. Each transducer $\mathcal{Z}^{(i)}$ has at most $2^{\mathcal{O}(hLk^4n^{k+4})}$ useful states.

The proof is based on a fine analysis of the useful states of $\mathcal{Z}^{(i)}$ and goes as follows. Let X be the set of useful states of $\mathcal{V}_N$ (as said in Section 3.3, $X \subseteq Q \times \mathfrak{P}_{(k)}(\Delta_N)^Q$). Each transducer $\mathcal{Z}^{(i)}$ is obtained by the multi-skimming covering of the underlying input automaton of $\mathcal{V}_N$, hence its states belong to $X \times \mathbb{N}_k^X$ (assuming that $\mathcal{Z}^{(i)}$ was built on the trim part of $\mathcal{V}_N$). By the stated properties of the constructions, we can derive that if[9] (P, V) is useful in $\mathcal{Z}^{(i)}$, then V has at most kn entries different from 0. In other words, the set of coordinates of V having a nonzero value belongs to $\mathfrak{P}_{(kn)}(X)$. There are k possible nonzero values for each such coordinate, namely $\{1, \ldots, k-1, \omega\}$, thus the number of useful states of $\mathcal{Z}_i$ is at most $\operatorname{card}(X) \cdot \operatorname{card}\big(\mathfrak{P}_{(kn)}(X)\big) \cdot k^{kn}$. To conclude, it remains to use the discussion on the number of useful states of $\mathcal{V}_N$ at the end of Section 3.3: we have that $\operatorname{card}\big(\mathfrak{P}_{(kn)}(X)\big) \leq \operatorname{card}(X)^{(kn)^2}$, and by Property 3.10, $\operatorname{card}(X) \leq 2^{2hNk^2n}$. With $N = n^{k+1}L$, we obtain the bound of $2^{\mathcal{O}(hLk^4n^{k+4})}$ states.

4.3. The morphic decomposition theorem

We turn now to Theorem 1.2, the proof of which goes in four steps. First, we construct a k-valued transducer $\mathcal{T}$ realising the composition $\tau\theta$. This is done by relabelling the transitions of the transducer $\mathcal{S}$ realising τ: every transition $p \xrightarrow{a|x} q$ of $\mathcal{S}$ is replaced by $p \xrightarrow{a|x\theta} q$. Next, $\mathcal{T}$ is decomposed into k unambiguous transducers $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$. These transducers are immersions in $\mathcal{V}_N$ and, by composition of morphisms, also in $\mathcal{T}$; but it may be the case that not every successful computation of $\mathcal{T}$ is projected by some successful one in the union of the $\mathcal{Z}^{(i)}$. The third and crucial step (described more precisely below) consists, roughly speaking, to *stick* the successful computations of $\mathcal{T}$ to the transducers $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$ in order to obtain equivalent (thus functional) transducers $\mathcal{W}^{(0)}, \ldots, \mathcal{W}^{(k-1)}$, not necessarily unambiguous, whose successful computations project on the whole set of successful computations of $\mathcal{T}$. Finally, the transitions of each $\mathcal{W}^{(i)}$ are relabelled in order to construct an

[9] Capital letters are used in order to distinguish the states of $\mathcal{Z}^{(i)}$ from the states of other automata.

immersion of $\mathcal{S}$: $e : p \xrightarrow{a|y} q$ in $\mathcal{W}^{(i)}$ projects on a transition f of $\mathcal{T}$; the label of f is, by construction, of form $a|x\theta$; the output y of e is replaced by x. This yields k transducers decomposing $\mathcal{S}$, not necessarily functional, but whose compositions with θ are functional.

The definition of the transducers $\mathcal{W}^{(0)}, \ldots, \mathcal{W}^{(k-1)}$ is based on a generalisation of the property of functional transducers that the lag between every pair of successful computations with same label is bounded by some integer (this appears implicitly in a proof of [2]).

Property 4.5. Let $N = n^{k+1}L$ and $K = 2(k+1)N$. If $\mathcal{T}$ is k-valued, then for every successful computation c of $\mathcal{T}$ there exists a successful computation D in $\mathcal{Z}^{(0)} \cup \cdots \cup \mathcal{Z}^{(k-1)}$ with same input, same output and such that $\langle c, D\varphi \rangle < K$. ∎

We can obtain each $\mathcal{W}^{(i)}$ from the product of $\mathcal{T} \times \mathcal{Z}^{(i)}$ by the Lead or Delay action, see [2] for details. The part of this product restricted to states having Lead or Delay in Δ_K projects on the successful computations of $\mathcal{T}$ with lag smaller than K with some successful computation in $\mathcal{Z}^{(i)}$. The number of states of $\mathcal{W}^{(i)}$ is bounded by $n \times M \times \mathrm{card}\,(\Delta_K)$, where M is the number of states of $\mathcal{Z}(i)$. This is again of order $2^{\mathcal{O}(hLk^4n^{k+4})}$.

References

[1] J. Berstel. *Transductions and Context-Free Languages*. B. G. Teubner, 1979.

[2] M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292:45–63, 2003.

[3] M.-P. Béal, S. Lombardy, and J. Sakarovitch. Conjugacy and equivalence of weighted automata and functional transducers. In D. Grigoriev, J. Harrison, and E. A. Hirsch, editors, *Proc. of CSR'06*, volume 3967 of *Lecture Notes in Computer Science*, pages 58–69, 2006.

[4] K. Culik and J. Karhumäki. The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theoretical Computer Science*, 47(1):71–84, 1986.

[5] S. Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[6] E. Gurari and O. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Mathematical Systems Theory*, 16:61–66, 1983.

[7] J. H. Johnson. Do rational equivalence relations have regular cross-sections? In Wilfried Brauer, editor, *Proc. ICALP'85*, volume 194 of *Lecture Notes in Computer Science*, pages 300–309. Springer-Verlag, 1985.

[8] J. H. Johnson. Rational equivalence relations. *Theoretical Computer Science*, 47(3):39–60, 1986.

[9] J. Sakarovitch. A construction on finite automata that has remained hidden. *Theoretical Computer Science*, 204(1–2):205–231, 1998.

[10] J. Sakarovitch. *Éléments de théorie des automates*. Vuibert, 2003. English translation: *Elements of Automata Theory*, Cambridge University Press, to appear.

[11] J. Sakarovitch. The rational skimming theorem. In Do Long Van and M. Ito, editors, *Proc. of The Mathematical Foundations of Informatics (1999)*, World Scientific, pages 157–172, 2005.

[12] J. Sakarovitch and R. de Souza. On the decidability of finite valuedness of transducers. in preparation (preliminary version available at `http://www.infres.enst.fr/~rsouza`).

[13] M. P. Schützenberger. Sur les relations rationnelles. In H. Barkhage, editor, *Automata Theory and Formal Languages, 2nd GI Conference*, volume 33 of *Lecture Notes in Computer Science*, pages 209–213, 1975.

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

[15] A. Weber. Decomposing a k-valued transducer into k unambiguous ones. *RAIRO Informatique Théorique et Applications*, 30(5):379–413, 1996.