From Two-Way to One-Way Finite State Transducers

Emmanuel Filiot
LACL
University Paris-Est Créteil

Olivier Gauwin
LaBRI,
University of Bordeaux

Pierre-Alain Reynier
LIF, Aix-Marseille Univ.
& CNRS, UMR 7279

Frédéric Servais
Hasselt University and
Transnational University of Limburg

***Abstract*—Any two-way finite state automaton is equivalent to some one-way finite state automaton. This well-known result, shown by Rabin and Scott and independently by Shepherdson, states that two-way finite state automata (even non-deterministic) characterize the class of regular languages. It is also known that this result does not extend to finite string transductions: (deterministic) two-way finite state transducers strictly extend the expressive power of (functional) one-way transducers. In particular deterministic two-way transducers capture exactly the class of MSO-transductions of finite strings.**

In this paper, we address the following definability problem: given a function defined by a two-way finite state transducer, is it definable by a one-way finite state transducer? By extending Rabin and Scott's proof to transductions, we show that this problem is decidable. Our procedure builds a one-way transducer, which is equivalent to the two-way transducer, whenever one exists.

I. Introduction

In formal language theory, the importance of a class of languages is often supported by the number and the diversity of its characterizations. One of the most famous example is the class of regular languages of finite strings, which enjoys, for instance, computational (automata), algebraic (syntactic congruence) and logical (monadic second order (MSO) logic with one successor) characterizations. The study of regular languages has been very influential and several generalizations have been established. Among the most notable ones are the extensions to infinite strings [1] and trees [2]. On finite strings, it is well-known that both deterministic and non-deterministic finite state automata define regular languages. It is also well-known that the expressive power of finite state automata does not increase when the reading head can move left and right, even in presence of non-determinism. The latter class is known as non-deterministic *two-way finite state automata* and it is no more powerful than (one-way) finite state automata. The proof of this result was first shown in the seminal paper of Rabin and Scott [3], and independently by Shepherdson [4].

The picture of automata models over finite strings changes substantially when, instead of languages, string *transductions*, i.e. relations from strings to strings, are considered. *Transducers* generalize automata as they are equipped with a one-way output tape. At each step they read an input symbol, they can append several symbols to the output tape. Their transition systems can be either deterministic or non-deterministic. *Functional* transducers are transducers that define functions instead of relations. For instance, deterministic transducers are always functional. In this paper, we are interested in transducers that define functions, but that can be non-deterministic.

As for automata, the reading head of transducers can move one-way (left-to-right) or two-way. *(One-way) finite state transducers* have been extensively studied [5], [6]. Non-deterministic (even functional) one-way transducers (*NFTs*) strictly extend the expressive power of deterministic one-way transducers (*DFTs*), because non-determinism allows one to express local transformations that depend on properties of the future of the input string.

Two-way finite state transducers define regular transformations that are beyond the expressive power of one-way transducers [7]. They can for instance reverse an input string, swap two substrings or copy a substring. The transductions defined by two-way transducers have been characterized by other logical and computational models. Introduced by Courcelle, monadic second-order definable transductions are transformations from graphs to graphs defined with the logic MSO [8]. Engelfriet and Hoogeboom have shown that the monadic second-order definable functions are exactly the functions definable by deterministic two-way finite state transducers (*2DFTs*) when the graphs are restricted to finite strings [9]. Recently, Alur and Černý have characterized *2DFT*-definable transductions by a deterministic one-way model called *streaming string transducers* [10] and shown how they can be applied to the verification of list-processing programs [11]. Streaming string transducers extend *DFTs* with a finite set of output string variables. At each step, their content can be reset or updated by either prepending or appending a finite string, or the content of another variable, in a copyless manner. Extending *2DFTs* with non-determinism does not increase their expressive power when they define functions: non-deterministic two-way finite state transducers (*2NFTs*) that are functional define exactly the class of functions definable by *2DFTs* [9], [12]. To summarize, there is a strict hierarchy between *DFT*-, functional *NFT*- and *2DFT*-definable transductions.

Several important problems are known to be decidable for one-way transducers. The *functionality* problem for *NFT*, decidable in PTime [13], [14], asks whether a given *NFT* is functional. The *determinizability* problem, also decidable in PTime [15], [14], asks whether a given functional *NFT* can be determinized, i.e. defines a *subsequential* function. Subsequential functions are those functions that can be defined by *DFTs* equipped with an additional output function from final states to finite strings, which is used to append a last string

This work has been partly supported by the project ECSPER funded by the french agency for research (ANR-09-JCJC-0069), by the project SOSP funded by the CNRS, and by the Faculty of Sciences of University Paris-Est Créteil.

to the output when the computation terminates successfully in some final state. Over strings that always end with a unique end marker, subsequential functions are exactly the functions definable by *DFTs*. For *2NFTs*, the functionality problem is known to be decidable [16]. Therefore the determinizability problem is also decidable for *2NFTs*, since functional *2NFTs* and *2DFTs* have the same expressive power. In the same line of research, we address a definability problem in this paper. In particular we answer the fundamental question of *NFT*-definability of transductions defined by functional *2NFTs*.

Theorem 1. *For all functional 2NFTs T, it is decidable whether the transduction defined by T is definable by an NFT.*

The proof of Theorem 1 extends the proof of Rabin and Scott [3] from automata to transducers[1]. The original proof of Rabin and Scott is based on the following observation about the runs of two-way automata. Their shapes have a nesting structure: they are composed of many zigzags, each zigzag being itself composed of simpler zigzags. Basic zigzags are called z-motions as their shapes look like a Z. Rabin and Scott prove that for automata, it is always possible to replace a z-motion by a single pass. Then from a two-way automaton A it is possible to construct an equivalent two-way automaton B (called the squeeze of A) which is simpler in the following sense: accepting runs of B are those of A in which some z-motions have been replaced by single pass runs. Last, they argue[2] that after a number of applications of this construction that depends only on the number of states of A, every zigzag can be removed, yielding an equivalent one-way automaton.

The extension to *2NFTs* faces the following additional difficulty: it is not always possible to replace a z-motion of a transducer by a single pass. Intuitively, this is due to the fact that *2NFTs* are strictly more expressive than *NFTs*. As our aim is to decide when a *2NFT* T is *NFT*-definable, we need to prove that the *NFT*-definability of T implies that of every z-motion of T, to be able to apply the squeeze construction. The main technical contribution of this paper is thus the study of the *NFT*-definability of z-motions of transducers. We show that this problem is decidable, and identify a characterization which allows one to prove that the *NFT*-definability of T implies that of every z-motion of T.

This characterization expresses requirements about the output strings produced along loops of z-motions. We show that when z-motions are *NFT*-definable, the output strings produced by the three passes on a loop are not arbitrary, but conjugates. This allows us to give a precise characterization of the form of these output strings. We show that it is decidable to check whether all outputs words have this form. Last, we present how to use this characterization to simulate an *NFT*-definable z-motion by a single pass.

Applications By Theorem 1 and since functionality is decidable for *2NFTs*, it is also decidable, given a *2NFT*, whether the transduction it defines is definable by a functional *NFT*. Another corollary of Theorem 1 and the fact that functionality of *2NFTs* and determinizability of *NFTs* are both decidable is the following theorem:

Theorem 2. *For all 2NFTs T, it is decidable whether the transduction defined by T is a subsequential function.*

A practical application of this result lies in the static analysis of memory requirements for evaluating (textual and functional) document transformations in a streaming fashion. In this scenario, the input string is received as a left-to-right stream. When the input stream is huge, it should not be entirely loaded in memory but rather processed on-the-fly. Similarly, the output string should not be stored in memory but produced as a stream. The remaining amount of memory needed to evaluate the transformation characterizes its streaming space complexity. *Streamable* transformations are those transformations for which the required memory is bounded by a constant, and therefore is independent on the length of the input stream. It is known that streamable transformations correspond to transformations definable by subsequential (functional) *NFTs* [18]. The *streamabability* problem asks, given a transformation defined by some transducer, whether it is streamable. Therefore for transformations defined by functional *NFTs*, streamability coincides with determinizability, and is decidable in PTime [15], [14]. Theorem 2 is a generalization of this latter result to regular transformations, i.e. transformations defined by functional *2NFTs*, MSO transducers or streaming string transducers [10]. Other streamability problems have been studied for XML validation [19], [20], XML queries [21] and XML transformations [18]. However the XML tree transformations of [18] are incomparable with the regular string transformations studied in this paper.

Related work Most of the related work has already been mentioned. To the best of our knowledge, it is the first result that addresses a definability problem between two-way and one-way transducers. In [22], two-way transducers with a two-way output tape are introduced with a special output policy: each time a cell at position i of the input tape is processed, the output is written in the cell at position i of the output tape. With that restriction, it is shown that two-way and one-way transducers (*NFTs*) define the same class of functions. In [23], the result of Rabin and Scott, and Shepherdson, is extended to two-way automata with multiplicities. In this context, two-way automata strictly extend one-way automata.

Organization of the paper Section II introduces necessary preliminary definitions. In Section III, we describe the general decision procedure for testing *NFT*-definability of functional *2NFTs*. We introduce z-motion transductions induced by *2NFTs* and show that their *NFT*-definability is necessary. The decidability of this necessary condition as well as the construction from z-motion transducers to *NFTs* are the most technical results of this paper and are the subject of Section IV. We finally discuss side results and further questions in Section V.

[1] Shepherdson [4] and then Vardi [17] proposed arguably simpler constructions for automata. It is however not clear to us how to extend these constructions to transducers.

[2] To our knowledge, there is no published proof of this result, thus we prove it in this paper as we use it for transducers.

II. One-Way and Two-Way Finite State Machines

Words, Languages and Transductions Given a finite alphabet Σ, we denote by Σ^* the set of finite words over Σ, and by ϵ the empty word. The length of a word $u \in \Sigma^*$ is its number of symbols, denoted by $|u|$. For all $i \in \{1, \dots, |u|\}$, we denote by $u[i]$ the i-th letter of u. Given $1 \leq i \leq j \leq |u|$, we denote by $u[i..j]$ the word $u[i]u[i+1]\dots u[j]$ and by $u[j..i]$ the word $u[j]u[j-1]\dots u[i]$. We say that $v \in \Sigma^*$ is a *factor* of u if there exist $u_1, u_2 \in \Sigma^*$ such that $u = u_1 v u_2$. By $\overline{u}$ we denote the *mirror* of u, i.e. the word of length $|u|$ such that $\overline{u}[i] = u[|u| - i + 1]$ for all $1 \leq i \leq |u|$.

The *primitive root* of $u \in \Sigma^*$ is the shortest word v such that $u = v^k$ for some integer $k \geq 1$, and is denoted by $\mu(u)$. Two words u and v are *conjugates*, denoted by $\sim$, if there exist $x, y \in \Sigma^*$ such that $u = xy$ and $v = yx$, i.e. u can be obtained from v by a cyclic permutation. Note that $\sim$ is an equivalence relation. We will use this fundamental lemma:

Lemma 1 ([24]). *Let $u, v \in \Sigma^*$. If there exists $n \geq 0$ such that u^n and v^n have a common factor of length at least $|u| + |v| - gcd(|u|, |v|)$, then $\mu(u) \sim \mu(v)$.*

Note that if $\mu(u) \sim \mu(v)$, then there exist $x, y \in \Sigma^*$ such that $u \in (xy)^*$ and $v \in (yx)^*$.

A *language* over Σ is a set $L \subseteq \Sigma^*$. A *transduction* over Σ is a relation $R \subseteq \Sigma^* \times \Sigma^*$. Its domain is denoted by $\mathrm{dom}(R)$, i.e. $\mathrm{dom}(R) = \{u \mid \exists v,\ (u, v) \in R\}$, while its image $\{v \mid \exists u,\ (u, v) \in R\}$ is denoted by $img(R)$. A transduction R is *functional* if it is a function.

Automata A *non-deterministic two-way finite state automaton*[3] (*2NFA*) over a finite alphabet Σ is a tuple $A = (Q, q_0, F, \Delta)$ where Q is a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is a set of final states, and Δ is the transition relation, of type $\Delta \subseteq Q \times \Sigma \times Q \times \{+1, -1\}$. It is *deterministic* if for all $(p, a) \in Q \times \Sigma$, there is at most one pair $(q, m) \in Q \times \{+1, -1\}$ such that $(p, a, q, m) \in \Delta$. In order to see how words are evaluated by A, it is convenient to see the input as a right-infinite input tape containing the word (starting at the first cell) followed by blank symbols. Initially the head of A is on the first cell in state q_0 (the cell at position 1). When A reads an input symbol, depending on the transitions in Δ, its head moves to the left (-1) if the head was not in the first cell, or to the right $(+1)$ and changes its state. A stops as soon as it reaches a blank symbol (therefore at the right of the input word), and the word is accepted if the current state is final.

A *configuration* of A is a pair $(q, i) \in Q \times (\mathbb{N} - \{0\})$ where q is a state and i is a position on the input tape. A *run* ρ of A is a finite sequence of configurations. The run $\rho = (p_1, i_1) \dots (p_m, i_m)$ is a run on an input word $u \in \Sigma^*$ of length n if $p_1 = q_0$, $i_1 = 1$, $i_m \leq n + 1$, and for all $k \in \{1, \dots, m-1\}$, $1 \leq i_k \leq n$ and $(p_k, u[i_k], p_{k+1}, i_{k+1} - i_k) \in \Delta$. It is *accepting* if $i_m = n + 1$ and $p_m \in F$. The language of a *2NFA* A, denoted by $L(A)$, is the set of words u such that there exists an accepting run of A on u.

A *non-deterministic (one-way) finite state automaton* (*NFA*) is a *2NFA* such that $\Delta \subseteq Q \times \Sigma \times Q \times \{+1\}$, therefore we will often see Δ as a subset of $Q \times \Sigma \times Q$. Any *2NFA* is effectively equivalent to an *NFA*. It was first proved by Rabin and Scott, and independently by Shepherdson [3], [4].

Transducers *Non-deterministic two-way finite state transducers* (*2NFTs*) over Σ extend *NFAs* with a one-way left-to-right output tape. They are defined as *2NFAs* except that the transition relation Δ is extended with outputs: $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q \times \{-1, +1\}$. If a transition (q, a, v, q', m) is fired on a letter a, the word v is appended to the right of the output tape and the transducer goes to state q'. Wlog we assume that for all $p, q \in Q$, $a \in \Sigma$ and $m \in \{+1, -1\}$, there exists at most one $v \in \Sigma^*$ such that $(p, a, v, q, m) \in \Delta$. We also denote v by $\mathrm{out}(p, a, q, m)$.

A run of a *2NFTs* is a run of its underlying automaton, i.e. the *2NFAs* obtained by ignoring the output. A run ρ may be simultaneously a run on a word u and on a word $u' \neq u$. However, when the underlying input word is given, there is a unique sequence of transitions associated with ρ. Given a *2NFT* T, an input word $u \in \Sigma^*$ and a run $\rho = (p_1, i_1) \dots (p_m, i_m)$ of T on u, the output of ρ on u, denoted by $\mathrm{out}^u(\rho)$, is the word obtained by concatenating the outputs of the transitions followed by ρ, i.e. $\mathrm{out}^u(\rho) = \mathrm{out}(p_1, u[i_1], p_2, i_2 - i_1) \cdots \mathrm{out}(p_{m-1}, u[i_{m-1}], p_m, i_m - i_{m-1})$. If ρ contains a single configuration, we let $\mathrm{out}^u(\rho) = \epsilon$. When the underlying input word u is clear from the context, we may omit the exponent u. The transduction defined by T is the relation $R(T) = \{(u, \mathrm{out}^u(\rho)) \mid \rho \text{ is an accepting run of } T \text{ on } u\}$. We may often just write T when it is clear from the context. A *2NFT* T is *functional* if the transduction it defines is functional. The class of functional *2NFTs* is denoted by *f2NFT*. In this paper, we mainly focus on *f2NFTs*. The *domain* of T is defined as $\mathrm{dom}(T) = \mathrm{dom}(R(T))$. The domain $\mathrm{dom}(T)$ is a regular language that can be defined by the *2NFA* obtained by projecting away the output part of the transitions of T, called the *underlying input automaton*. A *deterministic two-way finite state transducer* (*2DFT*) is a *2NFT* whose underlying input automaton is deterministic. Note that *2DFTs* are always functional, as there is at most one accepting run per input word. A *non-deterministic (one-way) finite state transducer* (*NFT*) is a *2NFT* whose underlying automaton is an *NFA*[4]. It is deterministic (written *DFT*) if the underlying automaton is a *DFA*.

We say that two transducers T, T' are equivalent, denoted by $T \equiv T'$, whenever they define the same transduction, i.e. $R(T) = R(T')$. For all transducer classes $\mathcal{C}$, we say that a transduction $R \subseteq \Sigma^* \times \Sigma^*$ is $\mathcal{C}$-definable if there exists $T \in \mathcal{C}$ such that $R = R(T)$. Given two classes $\mathcal{C}, \mathcal{C}'$ of transducers, and

[3] We follow the definition of Vardi [17], but without stay transitions. This is without loss of generality though.

[4] This definition implies that there is no ϵ-transitions that can produce outputs, which may cause the image of an input word to be an infinite language. Those *NFTs* are sometimes called *real-time* in the literature.

a transducer $T \in \mathcal{C}$, we say that T is (effectively) $\mathcal{C}'$-definable if one can construct an equivalent transducer $T' \in \mathcal{C}'$.

The $(\mathcal{C}, \mathcal{C}')$-*definability problem* takes as input a transducer $T \in \mathcal{C}$ and asks to decide whether T is $\mathcal{C}'$-definable. If so, one may want to construct an equivalent transducer $T' \in \mathcal{C}'$. In this paper, we prove that (*f2NFT*, *NFT*)-definability is decidable.

It is known that whether an *NFT* T is functional can be decided in PTime [13]. The class of functional *NFTs* is denoted by *fNFT*. Functional *NFTs* are strictly more expressive than *DFTs*. For instance, the function that maps any word $u \in \{a,b\}^+$ to $a^{|u|}$ if $u[|u|] = a$, and to $b^{|u|}$ otherwise, is *fNFT*-definable but not *DFT*-definable. This result does not hold for *2NFTs*: functional *2NFTs* and *2DFTs* define the same class of transductions (Theorem 22 of [9]).

Examples Let $\Sigma = \{a,b\}$ and $\# \notin \Sigma$, and consider the transductions

1) $R_0 = \{(u, a^{|u|}) \mid u \in \Sigma^+, u[|u|] = a\}$
2) $R_1 = \{(u, b^{|u|}) \mid u \in \Sigma^+, u[|u|] = b\} \cup R_0$
3) $R_2 = \{(\#u\#, \#\overline{u}\#) \mid u \in \Sigma^*\}$.

R_0 is *DFT*-definable: it suffices to replace each letter by a and to accept only if the last letter is a. Therefore it can be defined by the *DFT* T_0=$(\{q_a, q_b\}, q_b, \{q_a\}, \{(q_x, y, a, q_y) \mid x, y \in \Sigma\})$.

R_1 is *fNFT*-definable but not *DFT*-definable: similarly as before we can define a *DFT* $T_0' = (\{p_a, p_b\}, p_a, \{p_b\}, \{(p_x, y, b, p_y) \mid x, y \in \Sigma\})$ that defines the transduction $\{(u, b^{|u|}) \mid u \in \Sigma^+, u[|u|] = b\}$, and construct an *NFT* T_1 as follows: its initial state is some fresh state p_0, and when reading $x \in \Sigma$ the first time, it non-deterministically goes to T_0 or T_0' by taking the transition (p_0, x, a, q_x) or (p_0, x, b, p_x), and proceeds in either T_0 or T_0'. Even if R_1 is functional, it is not *DFT*-definable, as the transformation depends on the property of the last letter, which can be arbitrarily far away from the beginning of the string.

R_2 is *2DFT*-definable: it suffices to go to the end of the word by producing ϵ each time a letter is read, to go back to the beginning while copying each input letter, and return to the end without outputting anything, and to accept. Hence it is defined by $T_2 = (\{q_0, q_1, q_2, q_3, q_f\}, q_0, \{q_f\}, \delta_2)$ where states q_1, q_2, q_3 denote passes, and δ_2 is made of the transitions $(q_0, \#, \epsilon, q_1, +1)$, $(q_1, x{\in}\Sigma, \epsilon, q_1, +1)$ (during the first pass, move to the right), $(q_1, \#, \epsilon, q_2, -1)$, $(q_2, x{\in}\Sigma, x, q_2, -1)$, $(q_2, \#, \#, q_3, +1)$, $(q_3, x{\in}\Sigma, \epsilon, q_3, +1)$, $(q_3, \#, \#, q_f, +1)$.

Crossing Sequences, Loops and Finite-Crossing *2NFTs* The notion of crossing sequence is a useful notion in the theory of two-way automata [4], [25], that allows one to pump runs of two-way automata. Given a *2NFA* A, a word $u \in \Sigma^*$ and a run ρ of A on u, the *crossing sequence* at position i, denoted by $\mathrm{CS}(\rho, i)$ is given by the sequence of states q such that (q, i) occurs in ρ. The order of the sequence is given by the order in which the pairs of the form (q, i) occur in ρ. E.g. if $\rho = (q_1,1)(q_2,2)(q_3,1)(q_4,2)(q_5,1)(q_6,2)(q_7,3)$ then $\mathrm{CS}(\rho,1) = q_1q_3q_5$, $\mathrm{CS}(\rho,2) = q_2q_4q_6$ and $\mathrm{CS}(\rho,3) = q_7$. We write $\mathrm{CS}(\rho)$ the sequence $\mathrm{CS}(\rho,1), \ldots, \mathrm{CS}(\rho,|u|+1)$.

Crossing sequences allow one to define the loops of a run. Given a run ρ of the *2NFA* A on some word u of length n, a pair of positions (i,j) is a *loop* [5] in ρ if (i) $1 \le i \le j \le n$, (ii) $\mathrm{CS}(\rho,i) = \mathrm{CS}(\rho,j)$ and (iii) $u[i] = u[j]$. Let $u_1 = u[1..(i-1)]$, $u_2 = u[i..(j-1)]$ and $u_3 = u[j..n]$. If (i,j) is a loop in ρ and $u \in L(A)$, then $u_1(u_2)^k u_3 \in L(A)$ for all $k{\geq}0$. We say that a loop (i,j) is *empty* if $i = j$, in this case we have $u_2 = \varepsilon$. The notions of crossing sequence and loop carry over to transducers through their underlying input automata.

Given a *2NFT* T, $N \in \mathbb{N}$ and a run ρ of T on a word of length n, ρ is said to be N-crossing if $|\mathrm{CS}(\rho,i)| \le N$ for all $i \in \{1,\ldots,n\}$. The transducer T is *finite-crossing* if there exists $N \in \mathbb{N}$ such that for all $(u,v) \in R(T)$, there is an accepting N-crossing run ρ on u such that $\mathrm{out}(\rho) = v$. In that case, T is said to be N-crossing. It is easy to see that if T is N-crossing, then for all $(u,v) \in R(T)$ there is an accepting run ρ on u such that $\mathrm{out}(\rho) = v$ and no states repeat in $\mathrm{CS}(\rho,i)$ for all $i \in \{1,\ldots,|u|\}$. Indeed, if some state q repeats in some $\mathrm{CS}(\rho,i)$, then it is possible to pump the subrun between the two occurrences of q on $\mathrm{CS}(\rho,i)$. This subrun has an empty output, otherwise T would not be functional.

Proposition 1. *Any f2NFT with N states is N-crossing.*

III. From Two-way to One-way Transducers

In this section, we prove the main result of this paper, i.e. the decidability of (*f2NFT*, *NFT*)-definability.

A. Rabin and Scott's Construction for Automata

The proof of Theorem 1 relies on the same ideas as Rabin and Scott's construction for automata [3]. It is based on the following key observation: Any accepting run is made of many zigzags, and those zigzags are organized by a nesting hierarchy: zigzag patterns may be composed of simpler zigzag patterns. The simplest zigzags of the hierarchy are those that do not nest any other zigzag: they are called z-motions. Rabin and Scott described a procedure that removes those zigzags by iterating a construction that removes z-motions.

A *one-step sequence* is an indexed sequence $s = a_1, \ldots, a_n$ of positions such that $a_i \in \{1, 2\ldots, m\}$, $a_1 = 1$, $a_n = m$, and $|a_{i+1} - a_i| = 1$.The sequence s is N-*crossing* if for all $x \in \{1, 2\ldots, m\}$ we have $|\{i \mid a_i = x\}| \le N$. The reversals of s are the indexes $1 < r_1 < r_2 < \cdots < r_l < n$ such that $a_{r_i+1} = a_{r_i-1}$. In the sequel we let $r_0 = 1$ and $r_{l+1} = n$.

A *z-motion* z in s is a subsequence $a_e, a_{e+1}, \ldots a_f$ such that there is $0 < i < l$ with $r_{i-1} \le e < r_i < r_{i+1} < f \le r_{i+2}$, and $a_e = a_{r_{i+1}}$ and $a_f = a_{r_i}$. We may denote z by the pair of reversals (r_i, r_{i+1}). E.g. the sequences $z_1 = 1,2,3,2,1,2,3$ and $z_2 = 4,3,2,3,4,3,2$ are z-motions. The *shape* of a run ρ is defined as the second projection of ρ, written $\mathrm{shape}(\rho)$. A run ρ is a *z-motion run* if $\mathrm{shape}(\rho)$ is a z-motion. When there is no ambiguity, z-motion runs are just called z-motions.

If T is a *2NFA*, it is possible to construct a new automaton denoted by $\mathrm{squeeze}(T)$ such that, for all accepting runs ρ of T

[5]Observe that we include the input letter in the notion of loop. We use this to avoid technical difficulties due to backward transitions (which do not read the local symbol, but its successor).

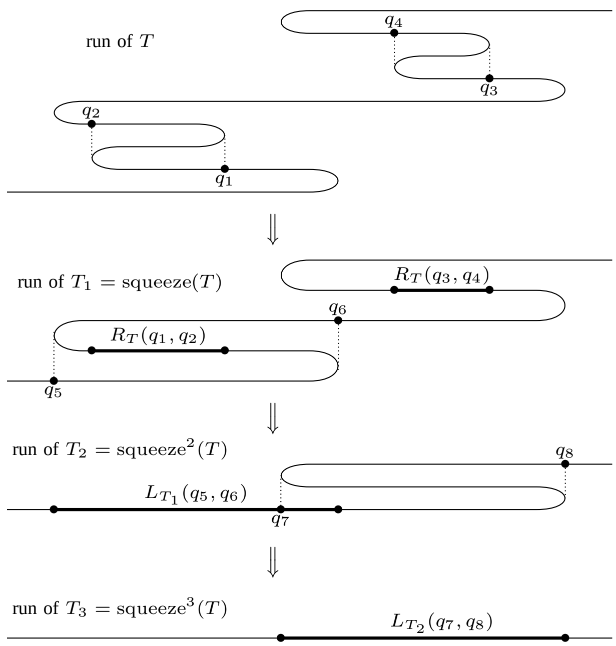

Fig. 1. Zigzags removal by applications of squeeze.

on some input word u, there exists a "simpler" accepting run of $\text{squeeze}(T)$ on u, obtained from ρ by replacing some z-motions by one-way runs that simulate three passes in parallel. It is illustrated by Fig. 1. For instance at the first step, there are two z-motions from q_1 to q_2 and from q_3 to q_4 respectively. Applying $\text{squeeze}(T)$ consists in non-deterministically guessing those z-motions and simulating them by one-way runs. This is done by the *NFA* $R_T(q_1, q_2)$ and $R_T(q_3, q_4)$ respectively. Depending on whether the z-motions enter from the left or the right, z-motions are replaced by runs of *NFAs* $R_T(.,.)$ (that read the input backwardly) or $L_T(.,.)$, as illustrated by the second iteration of squeeze on Fig. 1.

An N-crossing run ρ can be simplified into a one-way run after a constant number of applications of squeeze. This result is unpublished so we prove it in this paper. In particular, we show that if ρ is N-crossing, then its zigzag nesting depth decreases after N steps. Moreover, if ρ is N-crossing, then its zigzag nesting depth is also bounded by N. Therefore after N^2 applications of squeeze, ρ is transformed into a simple one-way run. It is sufficient to prove those results at the level of integer sequences. In particular, one can define $\text{squeeze}(s)$ the set of sequences obtained from a one-step sequence s by replacing *some* z-motions of s by strictly increasing or decreasing subsequences. The following is formalized and shown in Appendix:

Lemma 2. *Let s be an N-crossing one-step sequence over $\{1, \dots, m\}$. Then $1, 2, \dots, m$ is in $\text{squeeze}^{N^2}(s)$.*

At the automata level, it is known that for all words u accepted by a *2NFA* T with N states, there exists an N-crossing accepting run on u. Therefore it suffices to apply squeeze N^2 times to T. One gets an equivalent *2NFA* T^* from which the backward transitions can be removed while preserving equivalence with T^*, and so T.

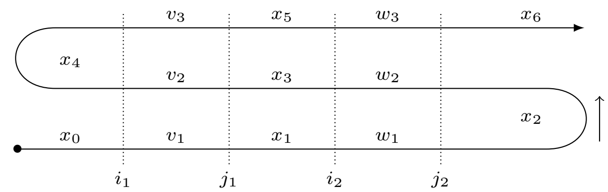

Fig. 2. Output decomposition in property $\mathcal{P}$.

B. Extension to transducers: overview

The construction used to show decidability of *NFT*-definability of *f2NFT* follows the same ideas as Rabin and Scott's construction. The main difference relies in the transformation of the local transducers defined by z-motion runs (that we call *ZNFTs*) into *NFTs*. Our procedure is built over a *ZNFT*-to-*NFT* procedure. It is seen as a black-box in this section, but is the subject of the next section.

Compared to two-way automata, one faces an extra difficulty caused by the fact that *2NFTs* (and *ZNFTs*) are not always *NFT*-definable. Therefore one defines a necessary condition that has to be tested each time we want to apply squeeze. Let us consider again Fig. 1 when T is a *2NFT*. One defines from T the transductions induced by local z-motion runs from a starting state q_1 to an ending state q_2, and show that those local transductions must be *NFT*-definable.

Once this necessary condition is satisfied, the construction squeeze can be applied and works as for Rabin and Scott's construction: the new transducer $\text{squeeze}(T)$ simulates T and non-deterministically may guess that the next zigzag of T is a z-motion run from some state q_1 to some state q_2, and thus can be simulated by a run of some *NFT* $R_T(q_1, q_2)$ or $L_T(q_1, q_2)$, depending on whether it enters from the left or the right. Then $\text{squeeze}(T)$ switches to $R_T(q_1, q_2)$ (if it entered from the right) and once $R_T(q_1, q_2)$ reaches an accepting state, it may come back to its normal mode.

C. z-motion transducers

z-motion transducers are defined like *2NFTs* except that they must define **functions** and to be accepting, a run on a word of length n must be of the form $\rho.(q_f, n+1)$ where ρ is a z-motion run and q_f is an accepting state. Note that it implies that $\text{shape}(\rho)$ is always of the form $1, \dots, n, n-1, \dots, 1, \dots, n$. The class of z-motion transducers is denoted by *ZNFTs*. Note that z-motion transducers are incomparable with *f2NFTs*. Indeed, z-motion transducers can define the transduction $u \in \Sigma^* \mapsto \overline{u}$, which is not *f2NFT*-definable as there are no end markers.

Let $T \in$ *ZNFT* and $\rho = (p_1, 1) \dots (p_n, n) (q_{n-1}, n-1) \dots (q_1, 1)(r_2, 2) \dots (r_{n+1}, n+1)$ be a run of T on a word of length n. We let $q_n = p_n$ and $r_1 = q_1$ and define the following shortcuts: for $1 \le i \le j \le n$, $\text{out}_1[i, j] = \text{out}((p_i, i) \dots (p_j, j))$, and $\text{out}_2[i, j] = \text{out}((q_j, j) \dots (q_i, i))$ and $\text{out}_3[i, j] = \text{out}((r_i, i) \dots (r_j, j))$, and $\text{out}_3[i, n+1] = \text{out}((r_i, i) \dots (r_{n+1}, n+1))$.

We characterize the *NFT*-definability of a *ZNFT* by a property that we prove to be decidable. Intuitively, this property requires that the outputs produced by loops can be produced by a single forward pass:

Definition 1 ($\mathcal{P}$-property)**.** *Let T be a ZNFT. We say that T satisfies the property $\mathcal{P}$, denoted by $T \models \mathcal{P}$, if for all words $u \in \mathrm{dom}(T)$, for all accepting runs ρ on u, and for all pairs of loops (i_1, j_1) and (i_2, j_2) of ρ such that $j_1 \leq i_2$, there exist $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5 \in \Sigma^*$, $f, g : \mathbb{N}^2 \to \Sigma^*$ and constants $c_1, c_1', c_2, c_2' \geq 0$ such that $c_1, c_2 \neq 0$ and for all $k_1, k_2 \geq 0$,*

$$f(k_1, k_2) x_0 v_1^{\eta_1} x_1 w_1^{\eta_2} x_2 w_2^{\eta_2} x_3 v_2^{\eta_1} x_4 v_3^{\eta_1} x_5 w_3^{\eta_2} x_6 g(k_1, k_2) = \beta_1 \beta_2^{k_1} \beta_3 \beta_4^{k_2} \beta_5$$

where $\eta_i = k_i c_i + c_i'$, $i \in \{1, 2\}$, and, x_i's, v_i's and w_i's are words defined as depicted in Fig. 2.

The following key lemma is proved in Section IV.

Lemma 3. *Let $T \in$ ZNFT. $T \models \mathcal{P}$ iff T is NFT-definable. Moreover, $\mathcal{P}$ is decidable and if $T \models \mathcal{P}$, one can (effectively) construct an equivalent NFT.*

Definition 2 (z-motion transductions induced by a *f2NFT*)**.** *Let $T = (Q, q_0, F, \Delta)$ be a f2NFT and $q_1, q_2 \in Q$. The transduction $\mathcal{L}_T(q_1, q_2)$ (resp. $\mathcal{R}_T(q_1, q_2)$) is defined as the set of pairs (u_2, v_2) such that there exist $u \in \Sigma^*$, two positions $i_1 < i_2$ (resp. $i_2 < i_1$), an accepting run ρ of T on u which can be decomposed as $\rho = \rho_1(q_1, i_1)\rho_2(q_2, i_2)\rho_3$ such that $u_2 = u[i_1 \dots i_2]$ and*

- $(q_1, i_1)\rho_2(q_2, i_2)$ *is a z-motion run*
- $\mathrm{out}((q_1, i_1)\rho_2(q_2, i_2)) = v_2$

z-motions can be of two forms: either they start from the left and end to the right, or start from the right and end to the left. In order to avoid considering these two cases each time, we introduce the notation $\overline{T}$ that denotes the mirror of T: it is T where the moves $+1$ are replaced by -1 and the moves -1 by $+1$. Moreover, the way $\overline{T}$ reads the input tape is slightly modified: it starts in position n and a run is accepting if it reaches position 0 in some accepting state. All the notions defined for *2NFTs* carry over to their mirrors. In particular, $(u, v) \in R(T)$ iff $(\overline{u}, v) \in R(\overline{T})$. The z-motion transductions $\mathcal{R}_T(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2)$ are symmetric in the following sense: $\mathcal{R}_T(q_1, q_2) = \mathcal{L}_{\overline{T}}(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2) = \mathcal{R}_{\overline{T}}(q_1, q_2)$.

Proposition 2. *The transductions $\mathcal{R}_T(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2)$ are ZNFT-definable.*

Proof: We only consider the case $\mathcal{L}_T(q_1, q_2)$, the other case being solved by using the equality $\mathcal{R}_T(q_1, q_2) = \mathcal{L}_{\overline{T}}(q_1, q_2)$. We first construct from T a *ZNFT* $Z_T'(q_1, q_2)$ which is like T but its initial state is q_1, and it can move to an accepting state whenever it is in q_2. However $Z_T'(q_1, q_2)$ may define input/output pairs (u_2, v_2) that cannot be embedded into some pair $(u, v) \in R(T)$ as required by the definition of $\mathcal{L}_T(q_1, q_2)$. Based on Shepherdson's construction, we modify $Z_T'(q_1, q_2)$ in order to take this constraint into account. The full proof is in Appendix. ∎

In the next subsection, we show that $\mathcal{R}_T(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2)$ must necessarily be *NFT*-definable for T to be *NFT*-definable. For that purpose, it is crucial in Definition 2 to make sure that the z-motion $(q_1, i_1)\rho_2(q_2, i_2)$ can be embedded into a global accepting run of T. Without that restriction, it might be the case that $\mathcal{L}_T(q_1, q_2)$ or $\mathcal{R}_T(q_1, q_2)$ is not *NFT*-definable although the *2NFT* T is. Indeed, the domain of $\mathcal{L}_T(q_1, q_2)$ or $\mathcal{R}_T(q_1, q_2)$ would be too permissive and accept words that would be otherwise rejected by other passes of global runs of T. This is another difficulty when lifting Rabin and Scott's proof to transducers, as for automata, the context in which a z-motion occurs is not important.

D. Decision procedure and proof of Theorem 1

We show that the construction $\mathrm{squeeze}(T)$ can be applied if the following necessary condition is satisfied.

Lemma 4. *If T is NFT-definable, then so are the transductions $\mathcal{R}_T(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2)$ for all states q_1, q_2. Moreover, it is decidable whether the transductions $\mathcal{R}_T(q_1, q_2)$ and $\mathcal{L}_T(q_1, q_2)$ are NFT-definable.*

Sketch of proof: We have seen in Lemma 3 that *NFT*-definability of an *ZNFT* is characterized by Property $\mathcal{P}$. Let $Z \in$ *ZNFT* that defines $\mathcal{L}_T(q_1, q_2)$ for some q_1, q_2, we thus sketch the proof that $Z \models \mathcal{P}$.

Consider two loops (i_1, j_1), (i_2, j_2) of a run ρ of Z on some word u, as in the premises of Property $\mathcal{P}$. They induce a decomposition of u as $u = u_1 u_2 u_3 u_4 u_5$ with $u_2 = u[i_1 \dots j_1 - 1]$ and $u_4 = u[i_2 \dots j_2 - 1]$. By definition of the transduction $\mathcal{L}_T(q_1, q_2)$, any word in $\mathrm{dom}(Z)$ can be extended into a word in $\mathrm{dom}(T)$. By hypothesis, T is *NFT*-definable, thus there exists an equivalent *NFT* T'. As T' has finitely many states, it is possible, by iterating the loops (i_1, j_1) and (i_2, j_2), to identify an input word of the form

$$u' = \alpha u_1 u_2^{c_1} u_2^{c_2} u_2^{c_3} u_3 u_4^{c_1'} u_4^{c_2'} u_4^{c_3'} u_5 \alpha'$$

and a run ρ' of T' on this word which has two loops on the input subwords $u_2^{c_2}$ and $u_4^{c_2'}$. It is then easy to conclude. ∎

Construction of $\mathrm{squeeze}(T)$ Assuming that the necessary condition is satisfied, we now explain how to construct the *f2NFT* $\mathrm{squeeze}(T)$. By hypothesis, the transductions $\mathcal{L}_T(q_1, q_2)$ and $\mathcal{R}_T(q_1, q_2)$ are *NFT*-definable for all q_1, q_2 by *NFT* $L_T(q_1, q_2)$ and $R_T(q_1, q_2)$ respectively (they exist by Proposition 2 and Lemma 3). As already said before, the main idea to define $\mathrm{squeeze}(T)$ is to non-deterministically (but repeatedly) apply $L_T(q_1, q_2)$, $R_T(q_1, q_2)$, or T, for some $q_1, q_2 \in Q$. However when applying $R_T(q_1, q_2)$, the head of $\mathrm{squeeze}(T)$ should move from the right to the left, so that we have to mirror the transitions of $R_T(q_1, q_2)$.

The transducer $\mathrm{squeeze}(T)$ has two modes, *Z*-mode or *T*-mode. In *T*-mode, it works as T until it non-deterministically decides that the next zigzag is a z-motion from some state q_1 to some state q_2. Then it goes in *Z*-mode and runs $L_T(q_1, q_2)$ or $\overline{R_T(q_1, q_2)}$, in which transitions to an accepting state have been replaced by transitions from q_2 in T, so that $\mathrm{squeeze}(T)$ returns in *T*-mode. From those transitions we also add transitions from the initial states of $L_T(q_2, q_3)$ and

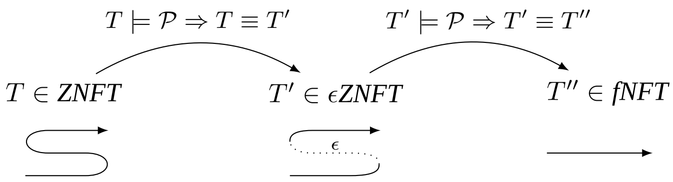

Fig. 3. From *ZNFT* to *NFT*.

$\overline{R_T(q_2, q_3)}$ for all $q_3 \in Q$, in case $\mathrm{squeeze}(T)$ guesses that the next z-motion starts immediately at the end of the previous z-motion. We detail the construction of $\mathrm{squeeze}(T)$ in Appendix.

Proposition 3. *Let $T \in$ f2NFT such that T is NFT-definable. Then $\mathrm{squeeze}(T)$ is defined and equivalent to T.*

Let $T \in$ *f2NFT*. If T is *NFT*-definable, then the operator $\mathrm{squeeze}$ can be iterated on T while preserving equivalence with T, by the latter proposition. By Proposition 1 T is N-crossing, and therefore, based on Lemma 2, it suffices to iterate $\mathrm{squeeze}$ N^2 times to remove all zigzags from accepting runs of T, as stated by the following lemma:

Lemma 5. *Let T be a f2NFT with N states. If T is fNFT-definable, then $\mathrm{squeeze}^{N^2}(T)$ is defined and equivalent to T, and moreover, for all $(u, v) \in R(T)$, there exists an accepting run ρ of $\mathrm{squeeze}^{N^2}(T)$ on u such that $\mathrm{out}(\rho) = v$ and ρ is made of forward transitions only.*

Proof of Theorem 1 In order to decide whether a *f2NFT* T is *NFT*-definable, it suffices to test whether $\mathrm{squeeze}$ can be applied N^2 times. More precisely, it suffices to set T_0 to T, i to 0, and, while T_i satisfies the necessary condition (which is decidable by Lemma 4) and $i \leq N^2$, to increase i and set T_i to $\mathrm{squeeze}(T_{i-1})$. If the procedure exits the loops before reaching N^2, then T is not *NFT*-definable, otherwise it is *NFT*-definable by the *NFT* obtained by removing from T_{N^2} all its backward transitions.

IV. From Elementary Zigzags to Lines

This section is devoted to the proof of Lemma 3 that characterizes *NFT*-definable *ZNFT* by the property $\mathcal{P}$ and states its decidability. Moreover, we give a *ZNFT*-to-*NFT* construction when $\mathcal{P}$ is satisfied.

We first prove that Property $\mathcal{P}$ is a necessary condition for *NFT*-definability. To prove the converse, we proceed in two steps. First, we define a procedure that tests whether a given *ZNFT* T is equivalent to a *ZNFT* that does not output anything on its backward pass (called ϵ*ZNFT*), and then define another procedure that tests whether the latter *ZNFT* is equivalent to an *NFT*. We show that it is always true whenever $T \models \mathcal{P}$. This approach is depicted in Fig. 3. The two steps are similar, therefore we mainly focus on the first step.

A. Property $\mathcal{P}$ is a necessary condition

We show that Property $\mathcal{P}$ only depends on transductions.

Lemma 6. *Let $T, T' \in$ZNFT. If $T \models \mathcal{P}$ and $T \equiv T'$ then $T' \models \mathcal{P}$.*

Proof: Consider two loops (i_1, j_1), (i_2, j_2) as in Property $\mathcal{P}$ in a run of T' on some word u. They induce a decomposition

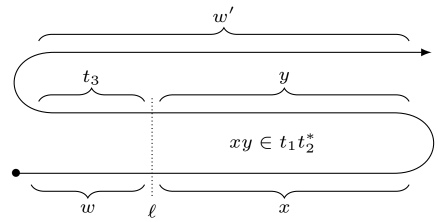

Fig. 4. Decomposition of the output according to Property $\mathcal{P}_1$.

of u as $u = u_1 u_2 u_3 u_4 u_5$ where $u_2 = u[i_1 \dots (j_1 - 1)]$ and $u_4 = u[i_2 \dots (j_2 - 1)]$, with $u_1 u_2^{k_1} u_3 u_4^{k_2} u_5 \in \mathrm{dom}(T')$ for all $k_1, k_2 \geq 0$.

As T is equivalent to T' and has finitely many states, there exist iterations of the loops on u_2 and u_4 which constitute loops in T on powers of u_2 and u_4. Formally, there exist integers $d_1, e_1, h_1, d_2, e_2, h_2$ with $e_1, e_2 > 0$ such that T has a run ρ on the input word $u_1 u_2^{d_1} u_2^{e_1} u_2^{h_1} u_3 u_4^{d_2} u_4^{e_2} u_4^{h_2} u_5$ which contains a loop on the input subwords $u_2^{e_1}$ and $u_4^{e_2}$.

We conclude easily by using the fact that $T \models \mathcal{P}$. ■

As a consequence, we obtain that Property $\mathcal{P}$ is a necessary condition for *NFT*-definability.

Lemma 7. *Let $T \in$ ZNFT. If T is NFT-definable, then $T \models \mathcal{P}$.*

Proof: Let T' be an *NFT* equivalent to T. It is easy to turn T' into a *ZNFT* T'' that performs two additional backward and forward passes which output ε. Consider two loops (i_1, j_1) and (j_1, j_2) in a run of T'', and let us write the output of this run as depicted on Fig. 2. These loops are also loops of T', and thus we can define β_1 (resp. β_2, β_3, β_4 and β_5) as x_0 (resp. v_1, x_1, w_1 and x_2), and f, g as the constant mappings equal to ϵ. Hence $T'' \models \mathcal{P}$, and we conclude by Lemma 6. ■

B. From ZNFT to ϵZNFT

The goal is to devise a procedure that tests whether the first and second passes (forward and backward) of the run can be done with a single forward pass, and constructs an *NFT* that realizes this single forward pass. Then, in order to obtain an ϵ*ZNFT*, it suffices to replace the first pass of T by the latter *NFT* and add a backward pass that just comes back to the beginning of the word and outputs ϵ all the time. The procedure constructs an ϵ*ZNFT*, and tests whether it is equivalent to T. It is based on the following key property that characterizes the form of the output words of the two first passes of any *ZNFT* satisfying $\mathcal{P}$. Intuitively, when these words are long enough, they can be decomposed as words whose primitive roots are conjugate.

Definition 3 ($\mathcal{P}_1$-property)**.** *Let $T \in$ ZNFT with m states, and let $(u, v) \in R(T)$ where u has length n. Let $K = 2.o.m^3.|\Sigma|$ where $o = max\{|v| \mid (p, a, v, q, m) \in \Delta\}$. The pair (u, v) satisfies the property $\mathcal{P}_1$, denoted by $(u, v) \models \mathcal{P}_1$, if for all accepting runs ρ on u, there exist a position $1 \leq \ell \leq n$ and $w, w', t_1, t_2, t_3 \in \Sigma^*$ such that $v \in w t_1 t_2^* t_3 w'$ and:*

$$\begin{array}{lll} \mathrm{out}_1[1,\ell] = w & \mathrm{out}_2[1,\ell] = t_3 & \mathrm{out}_1[\ell,n]\mathrm{out}_2[\ell,n] \in t_1 t_2^* \\ \mathrm{out}_3[1,n+1] = w' & & |t_i| \leq 2K, \forall i \in \{1,2,3\} \end{array}$$

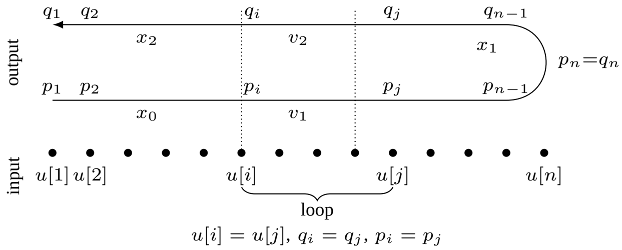

Fig. 5. Decomposition of the two first passes of a z-motion run with loop.

This decomposition is depicted in Fig. 4. T satisfies property $\mathcal{P}_1$, denoted $T \models \mathcal{P}_1$, if all $(u, v) \in R(T)$ satisfy it.

Proposition 4. *Let $T \in$ ZNFT. If $T \models \mathcal{P}$, then $T \models \mathcal{P}_1$.*

Proof: • If $|\mathrm{out}_2[1, n-1]| \leq K$, then clearly, it suffices to take $\ell = n$, $t_1 = \mathrm{out}_2[n-1, n]$, $t_2 = \varepsilon$, $t_3 = \mathrm{out}_2[0, n-1]$, $w = \mathrm{out}_1[1, n]$ and $w' = \mathrm{out}_3[1, n+1]$.

• Otherwise, $|\mathrm{out}_2[1, n-1]| > K$. Therefore u is of length $2.m^3.|\Sigma|$ at least and there exists a (non-empty) loop (i, j) in ρ. We can always choose this loop such that $|\mathrm{out}_2[1, i]| \leq K$ and $1 \leq |\mathrm{out}_2[i, j]| \leq K$ (see Lemma 16 in Appendix).

The loop partitions the input and output words into factors that are depicted in Fig. 5 (only the two first passes are depicted). Formally, let $u = u_1u_2u_3$ such that $u_2 = u[i \ldots (j-1)]$. Let $x_0 = \mathrm{out}_1[1, i]$, $v_1 = \mathrm{out}_1[i, j]$, $x_1 = \mathrm{out}_1[j, n]\mathrm{out}_2[j, n]$, $v_2 = \mathrm{out}_2[i, j]$, $x_2 = \mathrm{out}_1[1, i]$, $x_3 = \mathrm{out}_3[1, i]$, $v_3 = \mathrm{out}_3[i, j]$ and $x_4 = \mathrm{out}_3[j, n+1]$. In particular, we have $|x_2| \leq K$, $1 \leq |v_2| \leq K$ and $x_0v_1x_1v_2x_2x_3v_4x_4 \in T(u)$. Since (i, j) is a loop we also get $x_0v_1^kx_1v_2^kx_2x_3v_3^kx_4 \in T(u_1u_2^ku_3)$ for all $k \geq 0$. We then distinguish two cases:

1) If $v_1 \neq \epsilon$. We can apply Property $\mathcal{P}$ by taking the second loop empty. We get that for all $k \geq 0$

$$f(k)x_0v_1^{kc+c'}x_1v_2^{kc+c'}x_2x_3v_3^{kc+c'}x_4g(k) = \beta_1\beta_2^k\beta_3$$

where $f, g : \mathbb{N} \to \Sigma^*$, $c \in \mathbb{N}_{>0}$, $c' \in \mathbb{N}$, and $\beta_1, \beta_2, \beta_3 \in \Sigma^*$. Since the above equality holds for all $k \geq 0$, we can apply Lemma 1 and we get $\mu(v_1) \sim \mu(\beta_2)$ and $\mu(\beta_2) \sim \mu(v_2)$, and therefore $\mu(v_1) \sim \mu(v_2)$. So there exist $x, y \in \Sigma^*$ such that $v_1 \in (xy)^*$ and $v_2 \in (yx)^*$. One can show (see Lemma 17 in Appendix) that $v_1x_1v_2 \in x(yx)^*$. Then it suffices to take $\ell = i$, $w = x_0$, $t_1 = x$, $t_2 = yx$ and $t_3 = x_2$.

2) The second case ($v_1 = \epsilon$) is more complicated as it requires to use the full Property $\mathcal{P}$, using two non-empty loops. First, we distinguish two cases whether $|\mathrm{out}_1[j, n]| \leq K$ or not. For the latter case, we identify a second loop and then apply Property $\mathcal{P}$. Details can be found in the Appendix B. ■

Construction of an ϵ*ZNFT* from a *ZNFT* We construct an ϵ*ZNFT* T' from a *ZNFT* T such that $R(T') = \{(u, v) \in R(T) \mid (u, v) \models \mathcal{P}_1\}$. Intuitively, the main idea is to perform the two first passes in a single forward pass, followed by a non-producing backward pass, and the final third pass is exactly as T does. Therefore, T' guesses the words t_1, t_2 and t_3 and makes sure that the output v is indeed of the form characterized by $\mathcal{P}_1$. This can be done in a one-way fashion while simulating the forward and backward passes in parallel and by guessing non-deterministically the position ℓ. In addition, the output mechanism of T' exploits the special form of v: the idea is to output powers of t_2 while simulating the two first passes.

First, let us describe how T' simulates the forward and backward passes in parallel during the first forward pass. It guesses both the state of the backward pass, and the current symbol (this is needed as the symbol read by the backward transition is the next symbol). The first state (q^*) guessed for the backward pass needs to be stored, as the last (forward) pass should start from q^*. The transducer can go from state (p, q, σ) to state (p', q', σ') if the current symbol is σ and there is a (forward) transition $(p, \sigma, x, p', +1)$ and a (backward) transition $(q', \sigma', y, q, -1)$. Therefore if Q is the set of states of T, T' uses, on the first pass, elements of $Q \times Q \times \Sigma$ in its states. The transducer T' can non-deterministically decide to perform the backward and non-producing backward pass whenever it is in some state (q, q, σ) and the current symbol is σ. This indeed happens precisely when the forward and backward passes are in the same state q. If the current symbol is not the last of the input word, then the whole run of T' is not a z-motion and therefore it is not accepting.

Second, we describe how the ϵ*ZNFT* T', with the guess of t_1, t_2, t_3, verifies during its first forward pass that the output has the expected form, and how it produces this output. During the first pass, T' can be in two modes: In mode 1 (before the guess ℓ), T' verifies that the output on the simulated backward pass is t_3 and proceeds as T in the first forward pass (it outputs what T outputs on the forward pass). Mode 2 starts when the guess ℓ has been made. In this mode, T' first outputs t_1 and then verifies that the output of the forward/backward run from and to position ℓ is of the form $t_1t_2^*$. It can be done by using pointers on t_1 and t_2. There are two cases (guessed by T'): either t_1 ends during the forward pass or during the backward pass (using notations of Fig.4, either t_1 is a prefix of x, or x is a prefix of t_1).

In the first case, T' needs a pointer on t_1 to make sure that the output of T in the forward pass starts with t_1. It also needs a pointer on t_2, initially at the end of t_2, to make sure that the output of T on the simulated backward pass is a suffix of t_2^* (the pointer moves backward, coming back to the last position of t_2 whenever it reaches the first position of t_2). Once the verification on t_1 is done, T' starts, by using a pointer initially at the first position in t_2, to verify that the output of T in the forward pass is a prefix of t_2^*. Once the forward and the simulated backward passes merge, the two pointers on t_2 must be at the same position, otherwise the run is rejected.

During this verification, T' also has to output a power of t_2 (remind that it has already output t_1). However the transitions of T may not output exactly one t_2, nor a power of t_2, but may cut t_2 before its end. Therefore T' needs another pointer h to know where it is in t_2. Initially this pointer is at the first position of t_2 ($h = 1$). Suppose that T' simulates T using the (forward) transition $(p, \sigma, x, p', +1)$ and the (backward) transition $(q', \sigma', y, q, -1)$. If this step occurs before the end

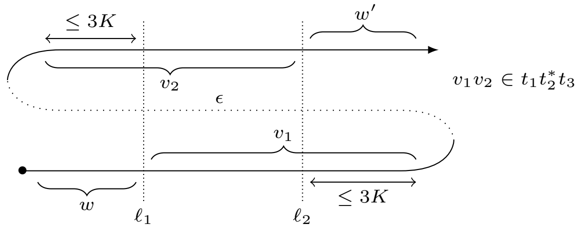

Fig. 6. Decomposition of the output according to Property $\mathcal{P}_2$.

of t_1, then T' outputs $t_2^\omega[h \dots (h + |y|)]$ (t_2^ω is the infinite concatenation of t_2), and the pointer h is updated to $1 + ((h + |y| - 1) \ mod \ |t_2|)$. Otherwise,$T'$ outputs $t_2^\omega[h \dots (h+|x|+|y|)]$ and h is updated to $1 + ((h + |x| + |y| - 1) \ mod \ |t_2|)$.

The second case (when T' guesses that t_1 ends during the backward pass) is similar. T' has to guess exactly the position in the output where t_1 ends. On the first pass it verifies that the output is a prefix of t_1, and on the simulated backward pass, it checks that the output is a suffix of t_2^* (and outputs as many t_2 as necessary, like before), until the end of t_1 is guessed to occur. From that moment it enters a verification mode on both passes.

The main property of this construction is that no wrong output words are produced by T', due to the verification and the way the output words are produced, i.e. for all $(u, v) \in R(T')$, we have $(u, v) \in R(T)$.

Proposition 5. *Let* $T \in$ *ZNFT.* $R(T') = \{(u, v) \in R(T) \mid (u, v) \models \mathcal{P}_1\}$.

Lemma 8. *Let $T \in$ ZNFT. If $T \models \mathcal{P}$, then T is equivalent to the ϵZNFT T'. Moreover, the latter is decidable.*

Proof: If $T \models \mathcal{P}$, then by Proposition 4, $T \models \mathcal{P}_1$. Therefore by Proposition 5, T and T' are equivalent.

We know that $R(T') \subseteq R(T)$, and since T and T' are both functional, they are equivalent iff $\mathrm{dom}(T) \subseteq \mathrm{dom}(T')$. Both domains can be defined by *NFAs*. Those *NFAs* simulate the three passes in parallel and make sure that those passes define a z-motion. Therefore testing the equivalence of T and T' amounts to test the equivalence of two *NFAs*. ∎

C. From ϵZNFT to NFT

We have seen how to go from a *ZNFT* to an *ϵZNFT*. We now briefly sketch how to go from an *ϵZNFT* to a (functional) *NFT*. Given an *ϵZNFT* T', we define an *fNFT* T'' such that T' and T'' are equivalent as soon as $T' \models \mathcal{P}$. The ideas are very similar to the previous construction therefore we do not give all the details here.

We exhibit a property on the form of output words produced by an *ϵZNFT* that verifies $\mathcal{P}$. Intuitively, apart from the beginning of the first pass, and the end of the second pass, if the two passes produce long enough outputs, then these outputs can be decomposed so as to exhibit conjugate primitive roots.

Definition 4 ($\mathcal{P}_2$-property)**.** *Let $T' \in \epsilon$ZNFT with m states, and let $(u, v) \in R(T')$ where u has length n. Let $K = 2om^3|\Sigma|$ where $o = max\{|v| \mid (p, a, v, q, m) \in \Delta\}$. The pair (u, v) satisfies the property $\mathcal{P}_2$, denoted by $(u, v) \models \mathcal{P}_2$, if for all accepting runs ρ on u, there exist two positions $1 \leq \ell_1 \leq \ell_2 \leq n$ and $w, w', t_1, t_2, t_3 \in \Sigma^*$ such that:*

$$\begin{array}{ll}
\mathrm{out}_1[1, \ell_1] = w & |t_i| \leq 3.K, \ \forall i \in \{1, 2, 3\} \\
\mathrm{out}_3[\ell_2, n+1] = w' & |\mathrm{out}_1[\ell_2, n]| \leq 3.K \\
\mathrm{out}_1[\ell_1, n]\mathrm{out}_3[1, \ell_2] \in t_1 t_2^* t_3 & |\mathrm{out}_3[1, \ell_1]| \leq 3.K
\end{array}$$

This decomposition is depicted in Fig. 6. T' satisfies property $\mathcal{P}_2$, denoted $T \models \mathcal{P}_2$, if all $(u, v) \in R(T')$ satisfy it.

The proof of the following proposition uses the same structure and techniques as that of Proposition 4. Using a (long) case analysis, we identify loops in runs, and apply Property $\mathcal{P}$ to show that output words have the expected form.

Proposition 6. *Let $T' \in \epsilon$ZNFT. If $T' \models \mathcal{P}$, then $T' \models \mathcal{P}_2$.*

We can now sketch the construction of an *fNFT* T'' which recognizes the subrelation of T' defined as $\{(u, v) \in R(T') \mid (u, v) \models \mathcal{P}_2\}$. Again, the construction is rather similar and uses the same techniques to that of T' starting from T.

The transducer T'' simulates, in a single forward pass, the three passes of T'. Hence it also checks that the run of the *ZNFT* T' it simulates is a z-motion run, which is a semantic restriction of accepting runs of *ZNFTs*. The *fNFT* T'' also guesses positions ℓ_1 and ℓ_2, and uses three modes accordingly. It also guesses the words t_1, t_2 and t_3, and words for $\mathrm{out}_3[1, \ell_1]$ and $\mathrm{out}_1[\ell_2, n]$, which are all of bounded length (see Property $\mathcal{P}_2$). The output of T'' is produced according to the mode, using pointers to check the guesses, similarly to T'.

If all the guesses happen to be verified, it outputs the correct output word, otherwise the input word is rejected. As a consequence, T'' recognizes a subrelation of T' and thus checking the equivalence of T' and T'' amounts to checking the equivalence of their domains (as the two transducers are functional), which is decidable. From Proposition 6 we get:

Lemma 9. *Let $T' \in \epsilon$ZNFT. If $T' \models \mathcal{P}$, then T' is equivalent to the fNFT T''. Moreover, the latter property is decidable.*

Proof of Lemma 3. Lemma 7 states that if T is *NFT*-definable, then $T \models \mathcal{P}$. Conversely, if $T \models \mathcal{P}$, then by Lemma 8, the first construction outputs an equivalent *ϵZNFT* T'. By Lemma 6, we have $T' \models \mathcal{P}$. By Lemma 9, the second construction outputs an equivalent *NFT* T''. Therefore T is *NFT*-definable by T''. In order to decide whether $T \models \mathcal{P}$, it suffices to construct T', check that T and T' are equivalent, and then construct T'' and check whether T' and T'' are equivalent. Both problems are decidable by Lemma 8 and 9.

V. Discussion

Complexity The procedure to decide (*f2NFT*, *NFT*)-definability is non-elementary exponential time and space. This is due to the *ZNFT*-to-*NFT* construction which outputs an *NFT* of doubly exponential size. Indeed, the first step of this construction transforms any *ZNFT* with n states into an *ϵZNFT* with at least $|\Sigma|^{4on^3|\Sigma|}$ states, as the *ϵZNFT* has to guess words of length $4on^3|\Sigma|$, where o is the maximal length of an output word of a transition. The *ϵZNFT*-to-*NFT* construction also outputs an exponentially bigger transducer. Therefore the $\mathrm{squeeze}$ operation outputs a transducer which is doubly exponentially larger. Since this operation has to be iterated N^2 times in the worst case, where N is the number of states of the initial *f2NFT*, this leads to a non-elementary procedure. On the other hand, the best lower bound we have for this problem is PSpace (by a simple proof that reduces the emptiness problem of the intersection of n *DFAs* is given in Appendix).

Succinctness It is already known that *2DFAs* are exponentially more succinct than *NFAs* [26]. Therefore this result carries over to transducers, already for transducers defining identity relations on some particular domains. However we show here a stronger result: the succinctness of *2NFTs* also comes from the transduction part and not only from the domain part. We can indeed exhibit a family of *NFT*-definable transductions $(R_n)_n$ that can be defined by *2DFTs* that are exponentially more succinct than their smallest equivalent *NFT*, and such that the family of languages $(\mathrm{dom}(R_n))_n$ does not show an exponential blow up between *2DFAs* and *NFAs*.

For all $n \geq 0$, we define R_n whose domain is the set of words $\#u\#$ for all $u \in \{a,b\}^*$ of length n, and the transduction is the mirror transduction, i.e. $R_n(\#u\#) = \#\overline{u}\#$.

Clearly, R_n is definable by a *2DFT* with $O(n)$ states that counts up to n the length of the input word by a forward pass, and then mirrors it by a backward pass. It is also definable by an *NFT* with $O(2^n)$ states: the *NFT* guesses a word u of length n (so it requires $O(2^n)$ states), outputs its reverse, and then verifies that the guess was correct. It is easy to prove that any *NFT* defining R_n needs at least 2^n states by a pumping argument. On the other hand, the domain of R_n can be defined by a *DFA* with $O(n)$ states that counts the length of the input word up to n. Note that the alphabet does not depend on n.

Further Questions We have shown that $($*f2NFT*$,$ *NFT*$)$-definability is decidable, however with a non-elementary procedure. We would like to characterize precisely the complexity of this problem. Our procedure works for functional *2NFTs*, which are equivalent to *2DFTs*. Therefore we could have done our proof directly for *2DFTs*. However (functional) non-determinism was added with no cost in the proof so we rather did it in this more general setting. The extension of our results to relations instead of functions is still open.

Our proof is an adaptation of the proof of Rabin and Scott [3] to transducers. Alternative constructions based on the proofs of Shepherdson [4] or Vardi [17], and alternative models such as streaming string transducers [10] or MSO transformations [8], [9], could lead to better complexity results or refined results. In particular, we believe that our results are highly related to the problem of minimizing the number of variables in a streaming string transducer.

Finally, we plan to study extensions of our results to infinite string tranformations, defined for instance by streaming string transducers [27], and to tree transformations, following our initial motivation from XML applications.

Acknowledgements We warmly thank Sebastian Maneth and Julien Tierny for interesting discussions.

REFERENCES

[1] J. R. Büchi, “On a decision method in restricted second order arithmetic,” in *Proceedings of the International Congress on Logic, Methodology, and Philosophy of Science*. Stanford University Press, 1962, pp. 1–11.
[2] J. W. Thatcher and J. B. Wright, “Generalized finite automata theory with an application to a decision problem of second-order logic,” *Mathematical Systems Theory*, vol. 2, no. 1, pp. 57–81, 1968.
[3] M. O. Rabin and D. Scott, “Finite automata and their decision problems,” *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 114–125, 1959.
[4] J. C. Shepherdson, “The reduction of two-way automata to one-way automata,” *IBM Journal of Research and Development*, vol. 3, no. 2, pp. 198–200, 1959.
[5] J. Berstel, *Transductions and context-free languages*. Teubner, 1979.
[6] J. Sakarovich, *Elements of Automata Theory*. Cambridge University Press, 2009.
[7] B. Courcelle, “The expression of graph properties and graph transformations in monadic second-order logic,” in *Handbook of Graph Transformation*. World Scientific, 1996, vol. I, Foundations.
[8] ——, “Monadic second-order definable graph transductions: a survey,” *Theoretical Computer Science*, vol. 126, no. 1, pp. 53–75, 1994.
[9] J. Engelfriet and H. J. Hoogeboom, “MSO definable string transductions and two-way finite-state transducers,” *ACM Transactions on Computational Logic (TOCL)*, vol. 2, no. 2, pp. 216–254, 2001.
[10] R. Alur and P. Černý, “Expressiveness of streaming string transducers,” in *FSTTCS*, vol. 8. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010, pp. 1–12.
[11] ——, “Streaming transducers for algorithmic verification of single-pass list-processing programs,” in *POPL*, 2011, pp. 599–610.
[12] R. de Souza, “Uniformisation of two-way transducers,” in *LATA*, ser. LNCS, vol. 7810. Springer, 2013, pp. 547–558.
[13] E. M. Gurari and O. H. Ibarra, “A note on finite-valued and finitely ambiguous transducers,” *Mathematical Systems Theory*, vol. 16, no. 1, pp. 61–66, 1983.
[14] M.-P. Béal, O. Carton, C. Prieur, and J. Sakarovitch, “Squaring transducers: an efficient procedure for deciding functionality and sequentiality,” *Theoretical Computer Science*, vol. 292, no. 1, pp. 45–63, 2003.
[15] A. Weber and R. Klemm, “Economy of description for single-valued transducers,” *Information and Computation*, vol. 118, no. 2, pp. 327–340, 1995.
[16] K. Culik and J. Karhumaki, “The equivalence problem for single-valued two-way transducers (on NPDT0L languages) is decidable,” *SIAM Journal on Computing*, vol. 16, no. 2, pp. 221–230, 1987.
[17] M. Y. Vardi, “A note on the reduction of two-way automata to one-way automata,” *Information Processing Letters*, vol. 30, no. 5, pp. 261–264, 1989.
[18] E. Filiot, O. Gauwin, P.-A. Reynier, and F. Servais, “Streamability of nested word transductions,” in *FSTTCS*, vol. 13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2011, pp. 312–324.
[19] L. Segoufin and C. Sirangelo, “Constant-memory validation of streaming XML documents against DTDs,” in *ICDT*, ser. LNCS, vol. 4353. Springer, 2007, pp. 299–313.
[20] V. Bárány, C. Löding, and O. Serre, “Regularity problems for visibly pushdown languages,” in *STACS*, ser. LNCS, vol. 3884. Springer, 2006, pp. 420–431.
[21] O. Gauwin, J. Niehren, and S. Tison, “Queries on XML streams with bounded delay and concurrency,” *Information and Computation*, vol. 209, no. 3, pp. 409–442, 2011.
[22] O. Carton, “Two-way transducers with a two-way output tape,” in *DLT*, ser. LNCS, vol. 7410. Springer, 2012, pp. 263–272.
[23] M. Anselmo, “Two-way automata with multiplicity,” in *ICALP*, ser. LNCS. Springer, 1990, vol. 443, pp. 88–102.
[24] C. Choffrut and J. Karhumäki, *Combinatorics on words*. Springer-Verlag, 1997, vol. 1, pp. 329–438.
[25] J. Hopcroft and J. Ullman, *Introduction to Automata Theory*. Addison-Wesley, 1979.
[26] J.-C. Birget, “State-complexity of finite-state devices, state compressibility and incompressibility,” *Mathematical Systems Theory*, vol. 26, no. 3, pp. 237–269, 1993.

[27] R. Alur, E. Filiot, and A. Trivedi, “Regular transformations of infinite strings,” in *LICS*. IEEE, 2012, pp. 65–74.

Appendix A
Complements to Section III

A. Iterative z-motions removal (proof of Lemma 2)

We define the crossing number of the position $x \in \{1, \dots, m\}$ as the number $|\{i \mid a_i = x\}|$. Hence the sequence s is K-crossing if all its positions $x \in \{1, 2 \dots, m\}$ have a crossing number less or equal than K.

We say that two z-motions $z_1 = (r_i, r_{i+1})$, $z_2 = (r_j, r_{j+1})$ are *consecutive*, resp. *positionally disjoint*, if $j = i + 2$, resp. $\max(a_{r_i}, a_{r_{i+1}}) < \min(a_{r_j}, a_{r_{j+1}})$ (or $\max(a_{r_j}, a_{r_{j+1}}) < \min(a_{r_i}, a_{r_{i+1}})$). Moreover we say that z_1 and z_2 are *disjoint* if they are not consecutive or if they are positionally disjoint. Equivalently, the z-motions $z_1 = a_{k_1}, a_{k_1+1}, \dots, a_{k_2}$ and $z_2 = a_{k_3}, a_{k_3+1}, \dots, a_{k_4}$ are disjoint if and only if $k_2 < k_3$ or $k_4 < k_1$.

Lemma 10. *If s is K-crossing, then for all $z_1, z_2, \dots, z_t$ consecutive z-motions, for all $i \leq t - K$, z_i and z_{i+K} are positionally disjoint.*

Proof: Let $j \in \{1, \dots, l\}$ such that $z_1 = (r_j, r_{j+1}), z_2 = (r_{j+2}, r_{j+3}), \dots$ and, wlog, assume $a_{r_j} < a_{r_{j+1}}$. As a consequence of the definition of z-motions, consecutive z-motions form a stair, that is, we have $a_{r_{j+2i}} \leq a_{r_{j+2(i+1)}}$ and $a_{r_{j+2i+1}} \leq a_{r_{j+2(i+1)+1}}$. If z_i and z_{i+K} are not positionally disjoint, all z_k for $i \leq k \leq i + K$ share the leftmost position of z_i, i.e. they share $a_{r_{j+2(i-1)}}$. Therefore s is not K-crossing. ■

We say that a position x is *in between* the positions y and z whenever $y \leq x \leq z$ or $z \leq x \leq y$. We say that the pair of reversals (or a z-motion) (r, s) is *nested* into the pair (r', s') if a_r and a_s are in between $a_{r'}$ and $a_{s'}$.

Lemma 11. *Let (r_i, r_j), with $i < j$, be a pair of reversals and $z = (r, r')$ be a z-motion. If a_r is in between a_{r_i} and a_{r_j}, and $r \in \{r_{i+1}, r_{j+1}\}$, or if $a_{r'}$ is in between a_{r_i} and a_{r_j}, and $r' \in \{r_{i-1}, r_{j-1}\}$, then z is nested in (r_i, r_j).*

Proof: Suppose that $z = (r_{i-2}, r_{i-1})$ (the other cases are proved similarly). Wlog assume $a_{r_j} \leq a_{r_i}$, so by hypothesis we have $a_{r_j} \leq a_{r_{i-1}} \leq a_{r_i}$. Then, as a consequence of basic properties of reversals, $a_{r_{i-1}} \leq a_{r_{i-2}}$ (because $a_{r_{i-1}} \leq a_{r_i}$). Moreover as (r_{i-2}, r_{i-1}) is a z-motion we have $a_{r_{i-2}} \leq a_{r_i}$. Therefore we have the inequalities: $a_{r_j} \leq a_{r_{i-1}} \leq a_{r_{i-2}} \leq a_{r_i}$, which means that z is nested in (r_i, r_j). ■

The one-step sequence s' is obtained from $s = a_1, \dots, a_n$ by removing the z-motion $z = a_{k_1}, \dots, a_{k_2}$ also defined by its reversals as $z = (r_i, r_{i+1})$, if $s' = a_1, \dots, a_{r_i}, a_{k_2+1}, \dots, a_n$. Note that the sequence s' is a one-step sequence because s is one and because $a_{r_i} = a_{k_2}$. The sequence s' has exactly 2 less reversals than s and each reversal of s not in z corresponds to one of the reversals of s', each z-motion $z' = (r_j, r_{j+1})$ of s such that $r_j, r_{j+1} \notin \{r_i, r_{i+1}\}$ is also a z-motion in s' (up to an index shift). Note also that positionally disjoint z-motions in s are still positionally disjoint in s'.

We define the function $\mathrm{squeeze}(s)$ as the function that associates to a one-step sequence s the set of one-step sequences that can be obtained from s by removing *some* pairwise disjoint z-motions of s.

We say that a set Z of z-motions of s is *consistent* if no two z-motions of Z share a reversal, that is, if $(r, r'), (s, s') \in Z$, then $r, r' \neq s$ and $r, r' \neq s'$. The consistent set Z is *maximal* if it is not strictly contained into any other consistent set of z-motions of s.

Lemma 12. *Let s be a K-crossing one-step sequence. If Z is a consistent set of z-motions of s then there is some $s' \in \mathrm{squeeze}^K(s)$ that contains no z-motion of Z.*

Proof: Let $Z = \{z_1, z_2, \dots\}$, where the z-motion are ordered, i.e., if $z_i = (r, r+1)$ and $z_{i+1} = (s, s+1)$ then $r+1 < s$. We define $s_0 = s$, and, for all $0 < i \leq K$, s_i is obtained from s_{i-1} by removing $z_i, z_{i+K}, z_{i+2K} \dots$. Clearly $s_{i+1} \in \mathrm{squeeze}^{i+1}(s)$ if z_{i+jK} and $z_{i+j'K}$ are disjoint in s_i. We consider two cases. Either z_{i+jK} and $z_{i+j'K}$ belong to a sequence of z-motions in Z that are consecutive in s, in that case we can apply Lemma 10 which shows that they are disjoint. Otherwise, z_{i+jK} and $z_{i+j'K}$ do not belong to such a sequence of z-motions in Z, that is, there exists a reversal r that does not appear in any z-motion of Z and which is between the second reversal of z_{i+jK} and the first reversal of $z_{i+j'K}$, but then they cannot be consecutive in s' (they are also separated by r in s'), so, by definition, they are disjoint. ∎

Proof of Lemma 2: Let $s_1 = s$, and for all $i \geq 1$, let Z_i be a maximal consistent set of z-motions of s_i, and s_{i+1} be the one-step sequence obtained from s_i by removing Z_i. We show that each z-motion z in Z_i has one of its positions whose crossing in s is at least $i + i'$ where i' is the crossing of the corresponding (some shift might be applied) position in s_{i+1}. This trivially holds for $s_1 = s$, so suppose it holds for i and let us show it also holds for $i+1$. Let $r'_1, r'_2, \dots, r'_{l'}$ be the reversals of s_i, let $z = (r'_{i'}, r'_{j'})$ be a z-motion in Z_{i+1} (recall that we abuse notation and refer to the reversals of s_{i+1} using the reversals of s_i though there is a shift of index for some of them). As Z_i is maximal, z is not a z-motion in s_i, so there is a z-motion $z' = (r'_k, r'_{k+1}) \in Z_i$ such that one of the following holds:

- $k = i' + 1$ or $k = j' + 1$ and $a_{r'_k}$ is in between $a_{r'_{i'}}$ and $a_{r'_{j'}}$
- $k = i' - 2$ and $a_{r'_{k+1}}$ is in between $a_{r'_{i'}}$ and $a_{r'_{j'}}$

Intuitively the above property states that one of the z-motions, z', in Z_i must prevent z to be a z-motion in s_i, that is, z' is somehow 'in' z. In each of these two cases we can apply Lemma 11 which states that z' is nested in z. By induction hypothesis, one of the position of z' has a crossing number in s of at least $i + i'$, where i' is the crossing number of the corresponding position in s_{i+1}. As s_{i+2} is obtained from s_{i+1} after removing z, we have $i' \geq 1 + i''$ where i'' is the crossing number of the corresponding position in s_{i+2}. So we have proved that the crossing number of this position is at least $(i+1) + i''$.

To conclude, as s is K-crossing, all positions are at most K-crossing, therefore the property we just proved implies that s_i for $i > K$ has no z-motion, that is $s_i = 1, 2, \dots, m$. By Lemma 12, K applications of squeeze are sufficient to remove a consistent set of z-motions, therefore $1, 2, \dots, m$ is in $\mathrm{squeeze}^{K^2}(s)$. ∎

B. Proof of Proposition 2

A crossing sequence s is *repetition-free* if each state occurs at most once in s. If Q is the set of states of A, we denote by $CS(Q)$ the set of repetition-free crossing sequences of A.

Based on Shepherdson's construction, it is possible to construct a one-way automaton whose states are sequences of states, such that any run ρ of A maps to the sequence of crossing sequences of ρ, and conversely any sequence of crossing sequences of this automaton maps to a run of A. This automaton may have infinitely many states, but it is well-known it is sufficient to consider repetition-free crossing sequences of states only [25].

Lemma 13 ([25])**.** *For all 2NFAs A with set of states Q, it is possible to construct an equivalent NFA $CS(A)$ whose set of states is $CS(Q)$, and such that for all accepting runs ρ' of $CS(A)$ on u, there exists an accepting run ρ of A on u such that $CS(\rho) = \rho'$.*

Lemma 14. *Let A be a 2NFA with set of states Q, and $q_1, q_2 \in Q$. Let M_{q_1,q_2} be the language of words u_2 such that there exists a word $u \in L(A)$, an accepting run ρ of A on u such that $\rho = \rho_1(q_1, i_1)\rho_2(q_2, i_2)\rho_3$ and $u_2 = u[i_1..i_2]$. Then M_{q_1,q_2} is regular.*

Proof: Given two sequences of states s_1 and s_2, the language Acc_{s_1,s_2} is defined as the set of words $u_2 \in \Sigma^*$ such that there exist a word $u \in \Sigma^*$, two positions $i_1 \leq i_2$ such that $u_2 = u[i_1..i_2]$, and an accepting run ρ on u such that $CS(\rho, i_1) = s_1$ and $CS(\rho, i_2) = s_2$. In other words, s_2 is accessible from s_1 by u_2. It is easy to show that for all s_1, s_2, there exists repetition-free sequences s'_1, s'_2 such that $Acc_{s_1,s_2} = Acc_{s'_1,s'_2}$. Therefore one can consider repetition-free sequences only. We have seen (Lemma 13) that one can construct an *NFA* whose states are the repetition-free crossing sequences of the runs of T. An easy reachability analysis of this *NFA* allows one to construct an *NFA* A_{q_1,q_2} whose states are repetition-free crossing sequences of T and such that $M_{q_1,q_2} = \bigcup\{Acc_{s_1,s_2} \mid q_1 \in s_1, q_2 \in s_2,\ s_1, s_2 \text{ are repetition-free}\}$. ∎

Proof: The transduction $\mathcal{L}_T(q_1, q_2)$ is a function, otherwise T would not be functional.

We define an intermediate *ZNFT* $Z'_T(q_1, q_2)$ that mimics T but starts initially in the state q_1 and whenever it reaches the state q_2, it non-deterministically decides to go to a fresh accepting state q'_f. Formally, $Z'_T(q_1, q_2) = (Q \cup \{q'_f\}, q_1, \{q'_f\}, \Delta')$ where $\Delta' = \Delta \cup \{(q_2, a, \epsilon, q'_f, +1) \mid a \in \Sigma\}$. Clearly, to any accepting run of $Z'_T(q_1, q_2)$ on a word $u_2 \in \Sigma^*$ corresponds a z-motion run of T on u_2 of the form $\rho'_2 = (q_1, 1)\rho_2(q_2, |u_2|)$ and conversely. However $Z'_T(q_1, q_2)$ is too permissive as it does not check that ρ'_2 can be embedded into a global accepting run of T. We now show how to restrict the domain of $Z'_T(q_1,q_2)$ to take this further constraint into account.

By a simple adaptation of Shepherdson's construction (see Lemma 14), the language M_{q_1,q_2} of words u_2 such that there exists $u \in \mathrm{dom}(T)$ and an accepting run ρ of T on u such that $\rho = \rho_1(q_1,i_1)\rho_2(q_2,i_2)\rho_3$ and $u_2 = u[i_1..i_2]$, can be defined by an *NFA* A_{q_1,q_2}.The transducer $Z_T(q_1,q_2)$ is finally defined as $Z'_T(q_1,q_2)$ where during the third and last pass, it also checks that the input word is in M_{q_1,q_2} by running A_{q_1,q_2} in parallel via a product construction.

Let us briefly explain why this construction is correct. Suppose that $(u_2,v) \in Z_T(q_1,q_2)$. We have $u_2 \in M_{q_1,q_2}$, therefore there exist $u \in \Sigma^*$ and two positions $i_1 < i_2$ such that $u_2 = u[i_1..i_2]$, and an accepting run ρ of T of the form $\rho_1(q_1,i_1)\rho_2(q_2,i_2)\rho_3$. The subrun $(q_1,i_1)\rho_2(q_2,i_2)$ is not necessarily a z-motion, and it does not necessarily outputs v. However since $(u_2,v) \in Z_T(q_1,q_2)$, we also have that $(u_2,v) \in Z'_T(q_1,q_2)$, and therefore there exists a z-motion run ρ' of T from q_1 to q_2 on u_2. One can therefore substitute $(q_1,i_1)\rho_2(q_2,i_2)$ by ρ' in ρ (modulo a shift of the positions occurring in ρ'), and one gets a new run $\gamma = \rho_1\rho'\rho_2$. The run γ is still an accepting run of T on u, and therefore $(u_2,v) \in L_T(q_1,q_2)$. The converse is easy by applying the definitions. ∎

C. Proof of Lemma 4

Proof: As in the proof of Proposition 2, we consider only the transductions $\mathcal{L}_T(q_1,q_2)$, the other case being solved by using the equality $\mathcal{R}_T(q_1,q_2) = \mathcal{L}_{\overline{T}}(q_1,q_2)$. Let $Z \in$ *ZNFT* that defines $\mathcal{L}_T(q_1,q_2)$ for some q_1,q_2 and suppose that T is *NFT*-definable. By Lemma 3 we have to show that $Z \models \mathcal{P}$. Let $u \in \mathrm{dom}(Z)$ of length n and $\rho = (p_1,1)\dots(p_n,n)(q_{n-1},n-1)\dots(q_1,1)(r_2,2)\dots(r_{n+1},n+1)$ an accepting run of Z on u. Let (i_1,j_1) and (i_2,j_2) be two loops of ρ such that $j_1 \le i_2$. These loops induce a decomposition of the input word u as $u = u_1u_2u_3u_4u_5$ with $u_2 = u[i_1..j_1-1]$ and $u_4 = u[i_2..j_2-1]$.

As (i_1,j_1) and (i_2,j_2) are loops in ρ, for any $k_1,k_2 \ge 0$, we have $u_1u_2^{k_1}u_3u_4^{k_2}u_5 \in \mathrm{dom}(Z)$. By definition of the transduction $\mathcal{L}_T(q_1,q_2)$, any word in $\mathrm{dom}(Z)$ can be extended into a word in $\mathrm{dom}(T)$. Thus, for any $k_1,k_2 \ge 0$, there exists $\alpha_{k_1,k_2},\alpha'_{k_1,k_2} \in \Sigma^*$ such that $u(k_1,k_2) = \alpha_{k_1,k_2}u_1u_2^{k_1}u_3u_4^{k_2}u_5\alpha'_{k_1,k_2} \in \mathrm{dom}(T)$.

In addition, by assumption, T is *NFT*-definable and thus there exists an *NFT* T' such that $T \equiv T'$. We consider such an *NFT* T', and denote by N its number of states. Let us consider $k_1 = k_2 = N+1$. There exists an accepting run ρ' of T' on the word $u(k_1,k_2)$. Consider the state in which is this run just before the i-th iteration of the word u_2, for $i \in \{1,\dots,k_1\}$. As $k_1 = N+1$, two of these states must be equal. A similar reasoning can be done for the powers of the word u_4. As a consequence, there exist constants $c_i,c'_i \ge 0$ with $i \in \{1,2,3\}$ such that $c_2,c'_2 > 0$ and the word $u(k_1,k_2)$ can be decomposed as follows:

$$u(k_1,k_2) = \alpha_{k_1,k_2}u_1u_2^{c_1}u_2^{c_2}u_2^{c_3}u_3u_4^{c'_1}u_4^{c'_2}u_4^{c'_3}u_5\alpha'_{k_1,k_2}$$

with the property that ρ' contains two loops on the input subwords $u_2^{c_2}$ and $u_4^{c'_2}$.

To conclude, we let β_1 (resp. β_2, β_3, β_4, β_5) be the output produced by ρ' on the input subword $u_1u_2^{c_1}$ (resp. $u_2^{c_2}$, $u_2^{c_3}u_3u_4^{c'_1}$, $u_4^{c'_2}$, $u_4^{c'_3}u_5$), and $f(k_1,k_2)$ (resp. $g(k_1,k_2)$) be the output produced by ρ' on the input subword $\alpha(k_1,k_2)$ (resp. α'_{k_1,k_2}). ∎

D. Definition of $\mathrm{squeeze}(T)$

We let $L_T(q_1,q_2) = (Q^{q_1,q_2},q_0^{q_1,q_2},F^{q_1,q_2},\Delta^{q_1,q_2})$ and $R_T(q_1,q_2) = (P^{q_1,q_2},p_0^{q_1,q_2},G^{q_1,q_2},\Gamma^{q_1,q_2})$ for all $q_1,q_2 \in Q$.

We let $\mathrm{squeeze}(T) = (Q',Q'_0,F',\Delta')$ and show formally how to construct it. For more convenience here we assume that $\mathrm{squeeze}(T)$ can have a set of initial states. It will be easy to transform it into a (usual) *2NFT*. We let $Q' = Q \uplus \biguplus\{Q^{q_1,q_2} \uplus P^{q_1,q_2} \mid q_1,q_2 \in Q\}$, $Q'_0 = \{q_0\} \cup \{q_0^{q_1,q_2} \mid q_1,q_2 \in Q\} \cup \{p_0^{q_1,q_2} \mid q_1,q_2 \in Q\}$, $F' = F$ and Δ' is the least set satisfying for all $q_1,q_2 \in Q$:

- $\Delta \uplus \biguplus_{q_1,q_2\in Q}\Delta^{q_1,q_2} \subseteq \Delta'$;
- $\forall (p,a,v,q_1,m)\in\Delta$, $(p,a,v,q_0^{q_1,q_2},m)\in\Delta'$;
- $\forall q \in F^{q_1,q_2}$, $\forall (p,a,v,q,+1) \in \Delta^{q_1,q_2}$, $\forall (q_2,a,v',q_3,m) \in \Delta$, $(p,a,vv',q_3,m) \in \Delta'$
- $\forall q \in F^{q_1,q_2}$, $\forall (p,a,v,q,+1) \in \Delta^{q_1,q_2}$, $\forall (q_2,a,v',q_3,m) \in \Delta$, for all $q_4 \in Q$, $(p,a,vv',q_0^{q_3,q_4},m) \in \Delta'$
- $\forall q \in F^{q_1,q_2}$, $\forall (p,a,v,q,+1) \in \Delta^{q_1,q_2}$, $\forall q_3 \in Q$, $\forall (q_0^{q_2,q_3},a,v',q',m) \in \Delta^{q_2,q_3} \cup \Gamma^{q_2,q_3}$, $(p,a,vv',q',m) \in \Delta'$

and similarly:

- $\biguplus_{q_1,q_2\in Q}\Gamma^{q_1,q_2} \subseteq \Delta'$;
- $\forall (p,a,v,q_1,m)\in\Delta$, $(p,a,v,p_0^{q_1,q_2},m)\in\Delta'$;
- $\forall q \in G^{q_1,q_2}$, $\forall (p,a,v,q,-1) \in \Gamma^{q_1,q_2}$, $\forall (q_2,a,v',q_3,m) \in \Delta$, $(p,a,vv',q_3,m) \in \Delta'$
- $\forall q \in G^{q_1,q_2}$, $\forall (p,a,v,q,-1) \in \Gamma^{q_1,q_2}$, $\forall (q_2,a,v',q_3,m) \in \Delta$, for all $q_4 \in Q$, $(p,a,vv',q_0^{q_3,q_4},m) \in \Delta'$
- $\forall q \in G^{q_1,q_2}$, $\forall (p,a,v,q,-1) \in \Gamma^{q_1,q_2}$, $\forall q_3 \in Q$, $\forall (q_0^{q_2,q_3},a,v',q',m) \in \Delta^{q_2,q_3} \cup \Gamma^{q_2,q_3}$, $(p,a,vv',q',m) \in \Delta'$

E. Proof of Proposition 3

Proof: Since $\mathrm{squeeze}(T)$ contains T as a subtransducer, we have $R(T) \subseteq R(\mathrm{squeeze}(T))$. Let us show that $R(\mathrm{squeeze}(T)) \subseteq R(T)$. Let $(u,v) \in R(\mathrm{squeeze}(T))$. Therefore there exists an accepting run ρ of $\mathrm{squeeze}(T)$ on u that outputs v. We are going to construct an accepting run of T on u that outputs v, this can be done by induction on the number of times ρ goes in *Z*-mode. If it never does so, ρ is accepting run of T and we are done. Otherwise suppose that ρ goes at least once in *Z*-mode for some $q_1,q_2 \in Q$. Note that the set Δ' consists of Δ, the sets $\Delta^{p,q}$ and $\Gamma^{p,q}$ for all

$p, q \in Q$, and new transitions of three kinds (of the form (p, a, vv', q_3, m), $(p, a, vv', q_0^{q_3,q_4}, m)$ and (p, a, vv', q', m) in the definition). Consider the first use of such a transition t in ρ. One can decompose ρ as $\rho_1\rho_2 t\rho_3$ where ρ_1 is in T-mode, ρ_2 in Z-mode, and assume that $\rho_2 t$ is a forward run on a factor u_2 of u (the case of a backward run is symmetric).

Let us inspect the case where $t = (p, a, vv', q_3, m)$. The other two cases (depending on the form of t) are proved similarly. Suppose that $p \in Q^{q_1,q_2}$. Then it means that $(u_2, v) \in \mathcal{L}_T(q_1, q_2)$, and therefore one can easily reconstruct a z-motion run ρ_2' of T on u_2 from q_1 to q_2 that outputs v. Then by definition of Δ', we know that there exists a transition from q_2 to q_3 that produces v'. By induction we can also transform ρ_3 into a run ρ_3' of T that ends in an accepting state and outputs the same word. Therefore $\rho_1'\rho_2'(q_2, a, v', q_3, m)\rho_3'$ is an accepting run of T on u that outputs the same word as ρ. Therefore $(u, v) \in R(T)$. ■

Appendix B
Complements to Section IV

A. Technical results

Lemma 15. *Let Σ, Γ, Λ be three finite alphabets, Ψ a morphism from Γ to Σ^* and Φ a morphism from Γ to Λ. Let $M = max\{|\Psi(\gamma)| \mid \gamma \in \Gamma\}$. For all words $u \in \Gamma^*$, if $|\Psi(u)| > (|\Lambda| + 1).M$, then there exist two positions $1 \le k_1 < k_2 \le |u|$ such that*[6]*:*

1) $|\Psi(u[1..(k_1 - 1)])| \le (|\Lambda| + 1).M$
2) $1 \le |\Psi(u[k_1..(k_2 - 1)])| \le (|\Lambda| + 1).M$
3) $\Phi(u[k_1]) = \Phi(u[k_2])$.

Proof: Let $L(u)$ be the set of *loops* that are strictly contained in u, i.e. $L(u) = \{(i, j) \mid 1 \le i < j \le |u|,\ (i \ne 1) \vee (j \ne |u|),\ \Phi(u[i]) = \Phi(u[j])\}$. We first show the following by induction on $|u|$:

$$(i) \begin{cases} |\Psi(u)| > (|\Lambda| + 1).M \\ \Longrightarrow \\ \exists (i, j) \in L(u),\ 1 \le |\Psi(u[i..j])| \le (|\Lambda| + 1).M \end{cases}$$

If $|u| = 0$ (resp. $|u| = 1$) then $|\Psi(u)| = 0$ (resp. $|\Psi(u) \le M$) and therefore the above implication is obviously satisfied. Otherwise suppose that $|u| > 0$ and $|\Psi(u)| > (|\Lambda| + 1).M$. Therefore we have $|u| > |\Lambda|+1 \ge 2$, and $|u[2..|u|]| > |\Lambda|$, and so by the pigeon-hole principle there exist two positions $i < j$ in $u[2..|u|]$ such that $\Phi(u[i]) = \Phi(u[j])$, so that $L(u) \ne \varnothing$.

Suppose that for all $(i, j) \in L(u)$, $\Psi(u[i..(j-1)]) = \epsilon$. If we remove maximally from u all the factors of u from position i to position $(j-1)$ for all $(i, j) \in L(u)$, one obtains a word v such that $L(v) = \varnothing$ and $|\Psi(v)| = |\Psi(u)| > (|\Lambda|+1).M$. Moreover $|v| \le |\Lambda| + 1$ since $L(v) = \varnothing$, but this contradicts $|\Psi(v)| > (|\Lambda| + 1).M$ by definition of M. Since $L(u) \ne \varnothing$, we get the existence of $(i_0, j_0) \in L(u)$ such that $\Psi(u[i_0..(j_0 - 1)]) \ne \epsilon$. If $|\Psi(u[i_0..(j_0-1)])| \le (|\Lambda|+1).M$ we are done. Otherwise, since $|u[i_0..(j_0 - 1)]| < |u|$, by induction hypothesis we get the existence of a pair $(i^*, j^*) \in L(u[i_0..(j_0 - 1)])$ such that $1 \le |\Psi(u[i_0..(j_0 - 1)][i^*..(j^* - 1)])| \le (|\Lambda| + 1).M$, from which we can conclude by taking $i = i^* + i_0 - 1$ and $j = j^* + i_0 - 1$ (note that $(i, j) \in L(u)$).

This shows items (2) and (3) of the Lemma. Again by induction on $|u|$ and by using (i), we prove the lemma. If $|u| = 0$ or $|u| = 1$, then the implication obviously holds. Otherwise assume that $|\Psi(u)| > (|\Lambda| + 1).M$. By (i) there exists $(k_1, k_2) \in L(u)$ that satisfies (2) and (3). If $|\Psi(u[1..(k_1 - 1)])| \le (|\Lambda| + 1).M$ we are done, otherwise by induction hypothesis, there exists $(k_1', k_2') \in L(u[1..(k_1 - 1)])$ which satisfies (1), (2) and (3), from which we can conclude. ■

Lemma 16. *Let $T \in$ ZNFT with m states. Let o be the maximal length of an output word in a transition of T and $K = 2.o.m^3.|\Sigma|$. Let ρ be a run on a word u of length n. We write ρ as the sequence $(p_1, 1) \dots (p_n, n)(q_{n-1}, n - 1) \dots (q_1, 1)(r_2, 2) \dots (r_{n+1}, n + 1)$ and let $q_n = p_n$ and $r_1 = q_1$. Let $1 \le k < \ell \le n$ such that $|\mathrm{out}_2[k, \ell]| > K$. There exists a loop (i, j) in ρ such that $k \le i < j \le \ell$ and*

1) $|\mathrm{out}_2[k, i]| \le K$
2) $1 \le |\mathrm{out}_2[i, j]| \le K$.

Proof: We show this result by using Lemma 15.

We consider the alphabet $\Delta^3 \times \Sigma$, where Δ denotes the set of transitions of T. Given a triple of transitions $\theta = ((s_\ell, a_\ell, u_\ell, s_\ell')_{1\le \ell \le 3})$, and a letter $a \in \Sigma$, we define the mappings Ψ and Φ as $\Psi(\theta, a) = u_2$ and $\Phi(\theta, a) = (s_1, s_2, s_3, a)$. Then, we associate to the run ρ, considered between positions k and ℓ, a word over this alphabet of length $\ell - k$, indexed from k to $\ell - 1$, and defined as $\eta = (\sigma_m)_{k \le m \le \ell-1}$, where σ_m is composed of the three transitions used respectively to go from configuration (p_m, m) to configuration $(p_{m+1}, m + 1)$, from configuration $(q_{m+1}, m + 1)$ to configuration (q_m, m), and from configuration (r_m, m) to configuration $(r_{m+1}, m + 1)$, and of the letter $u[m]$.

Using these definitions, we have $\Psi(\eta) = \mathrm{out}_2[k, \ell]$, and, for any $k \le m \le \ell - 1$, $\Phi(\sigma_k) = (p_k, q_k, r_k, u[k])$. Then it suffices to apply Lemma 15 to get the result. ■

Lemma 17. *Let $x, y, z, t \in \Sigma^*$ such that $x \ne \epsilon$ and $y \ne \epsilon$. Suppose that for all $i \ge 0$, $x^i y z^i$ is a prefix of t^ω. Then there exists $\alpha_1, \alpha_2 \in \Sigma^*$ such that $x \in (\alpha_1\alpha_2)^*$, $z \in (\alpha_2\alpha_1)^*$ and $xyz \in \alpha_1(\alpha_2\alpha_1)^*$.*

Proof: By Lemma 1 $\mu(x) \sim \mu(t)$ and $\mu(z) \sim \mu(t)$, therefore $\mu(x) \sim \mu(z)$, i.e. there exists α_1, α_2 with $x \in (\alpha_1\alpha_2)^*$ and $z \in (\alpha_2\alpha_1)^*$. Moreover as x^i is a prefix of t^ω for all $i > 0$, clearly $\mu(t) = \mu(x) = \alpha_1\alpha_2$.

Now let $xyz = (\alpha_1\alpha_2)^k\alpha$ a prefix of $(\alpha_1\alpha_2)^\omega$ and let us show that $\alpha = \alpha_1$. So suppose $\alpha\beta = \alpha_1$ (the other case when $\alpha_1\beta = \alpha$ is proved similarly). Therefore $z = (\alpha_2\alpha_1)^a = (\alpha_2\alpha\beta)^a$ but also $xyz = (\alpha_1\alpha_2)^k\alpha$ implies that $z = (\beta\alpha_2\alpha)^a$. So $\beta\alpha_2\alpha = \alpha_2\alpha\beta$ which means $\alpha_1\alpha_2$ is not primitive if $\beta \ne \epsilon$. ■

[6] In this Lemma, if $k_1 = 1$ then we let $u[1..(k_1 - 1)] = \epsilon$

B. Proof of Proposition 4

Proof: • If $|\mathrm{out}_2[1,n-1]| \leq K$, then clearly, it suffices to take $\ell = n$, $t_1 = \mathrm{out}_2[n-1,n]$, $t_2 = \varepsilon$, $t_3 = \mathrm{out}_2[0,n-1]$, $w = \mathrm{out}_1[1,n]$ and $w' = \mathrm{out}_3[1,n+1]$.

• Otherwise, $|\mathrm{out}_2[1,n-1]| > K$. Therefore u is of length $2.m^3.|\Sigma|$ at least and there exists necessarily a (non-empty) loop (i,j) in ρ. We can always choose this loop such that $|\mathrm{out}_2[1,i]| \leq K$ and $1 \leq |\mathrm{out}_2[i,j]| \leq K$ (see Lemma 16).

The loop partitions the input and output words into factors that are depicted in Fig. 5 (only the two first passes are depicted). Formally, let $u = u_1u_2u_3$ such that $u_2 = u[i..(j-1)]$. Let $x_0 = \mathrm{out}_1[1,i]$, $v_1 = \mathrm{out}_1[i,j]$, $x_1 = \mathrm{out}_1[j,n]\mathrm{out}_2[j,n]$, $v_2 = \mathrm{out}_2[i,j]$, $x_2 = \mathrm{out}_1[1,i]$, $x_3 = \mathrm{out}_3[1,i]$, $v_3 = \mathrm{out}_3[i,j]$ and $x_4 = \mathrm{out}_3[j,n+1]$. In particular, we have $|x_2| \leq K$, $1 \leq |v_2| \leq K$ and $x_0v_1x_1v_2x_2x_3v_4x_4 \in T(u)$. Since (i,j) is a loop we also get $x_0v_1^kx_1v_2^kx_2x_3v_3^kx_4 \in T(u_1u_2^ku_3)$ for all $k \geq 0$.

We then distinguish two cases:

1) If $v_1 \neq \epsilon$. We can apply Property $\mathcal{P}$ by taking the second loop empty. We get that for all $k \geq 0$

$$f(k)x_0v_1^{kc+c'}x_1v_2^{kc+c'}x_2x_3v_3^{kc+c'}x_4g(k) = \beta_1\beta_2^k\beta_3$$

where $f,g : \mathbb{N} \to \Sigma^*$, $c \in \mathbb{N}_{>0}$, $c' \in \mathbb{N}$, and $\beta_1,\beta_2,\beta_3 \in \Sigma^*$. Since the above equality holds for all $k \geq 0$, we can apply Lemma 1 and we get $\mu(v_1) \sim \mu(\beta_2)$ and $\mu(\beta_2) \sim \mu(v_2)$, and therefore $\mu(v_1) \sim \mu(v_2)$. So there exist $x,y \in \Sigma^*$ such that $v_1 \in (xy)^*$ and $v_2 \in (yx)^*$. By Lemma 17, we obtain that $v_1x_1v_2 \in x(yx)^*$. Then it suffices to take $\ell = i$, $w = x_0$, $t_1 = x$, $t_2 = yx$ and $t_3 = x_2$ to conclude the proof.

2) Otherwise, we have $v_1 = \epsilon$. We decompose x_1 as $x_1 = y_1y_2$ where $y_1 = \mathrm{out}_1[j,n]$ and $y_2 = \mathrm{out}_2[j,n]$.
We again distinguish two cases:
   a) We first consider the case when $|y_1| = |\mathrm{out}_1[j,n]| > K$. In this case, we can as before decompose the input word $u[j..n]$ to identify a loop. More precisely, there exists a loop (r,s) in ρ such that $r \geq j$, $|\mathrm{out}_1[j,r]| < K$ and $1 \leq |\mathrm{out}_1[r,s]| \leq K$. This loop gives a decomposition of u_3 as $u_4u_5u_6$. We will then apply Property $\mathcal{P}$ to the two loops (i,j) and (r,s). The loop (r,s) gives a decomposition of y_1 as $z_0w_1z_1$, y_2 as $z_2w_2z_3$ and x_4 as $z_4w_3z_5$. By Property $\mathcal{P}$, there exist words β_i, $i \in \{1,\ldots,5\}$, and c_1,c_1',c_2,c_2',f,g such that, for all $k_1,k_2 \geq 0$,

$$f(k_1,k_2)x_0v_1^{\eta_1}z_0w_1^{\eta_2}z_1z_2w_2^{\eta_2}z_3v_2^{\eta_1}x_2$$
$$x_3v_3^{\eta_1}z_4w_3^{\eta_2}z_5g(k_1,k_2) = \beta_1\beta_2^{k_1}\beta_3\beta_4^{k_2}\beta_5$$

   where $\eta_i = k_ic_i+c_i'$, $i \in \{1,2\}$. Recall that $w_1 \neq \varepsilon$ and $v_2 \neq \varepsilon$. As a consequence, we can, using sufficiently large values of k_1 and k_2 and applying Lemma 1, prove that $\mu(w_1) \sim \mu(\beta_4)$, that $\mu(v_2) \sim \mu(\beta_4)$, and thus deduce that $\mu(w_1) \sim \mu(v_2)$. Therefore there exist x,y such that $v_2 \in (yx)^*$ and $w_1 \in (xy)^*$ from which we deduce that $w_1z_1z_2w_2z_3v_2 \in x(yx)^*$. Recall that by the choice of the loop (r,s) we have $|z_0| \leq K$. We can thus define $\ell = i$, $w = x_0$, $t_1 = z_0x$, $t_2 = yx$ and $t_3 = x_2$ to obtain the result.
   b) The last case is when $|y_1| = |\mathrm{out}_1[j,n]| \leq K$. We consider the length of $y_2 = \mathrm{out}_2[j,n]$. First observe that if we have $|y_2| \leq K$ then we are done. Indeed, we can define $\ell = j$, $t_1 = y_1$, $t_2 = y_2$ and $t_3 = v_2x_2$. It is routine to verify that the conditions of Property $\mathcal{P}_1$ are fulfilled.
   We thus suppose that $|y_2| > K$. In this case, we can as before identify a loop (r,s) in the run ρ such that $r \geq j$, $\mathrm{out}_2[s,n] \leq K$ and $1 \leq \mathrm{out}_2[r,s] \leq K$. We do not give the details, but one can apply Property $\mathcal{P}$ to the two loops (i,j) and (r,s) and use the fact that $\mathrm{out}_2[i,j] \neq \varepsilon$ and $\mathrm{out}_2[r,s] \neq \varepsilon$ to prove that $\mu(\mathrm{out}_2[i,j]) \sim \mu(\mathrm{out}_2[r,s])$. Then, there exist x,y such that $\mathrm{out}_2[r,s] \in (xy)^*$ and $\mathrm{out}_2[i,j] \in (yx)^*$ from which we deduce that $\mathrm{out}_2[i,s] \in (xy)^*x$.
   Finally, we let $\ell = i$, $w = x_0$, $t_1 = \mathrm{out}_1[i,n]\mathrm{out}_2[s,n]$, $t_2 = xy$ and $t_3 = xx_2$ to obtain the result.

■

C. From ϵZNFT to NFT

We state the following Lemma whose proof is similar to that of Lemma 16:

Lemma 18. *Let $T \in$ ZNFT with m states. Let o the maximal length of an output word in a transition of T and $K = 2.o.m^3.|\Sigma|$. Let ρ be a run on a word u of length n. We write ρ as the sequence $(p_1,1)\ldots(p_n,n)(q_{n-1},n-1)\ldots(q_1,1)(r_2,2)\ldots(r_{n+1},n+1)$ and let $q_n = p_n$ and $r_1 = q_1$. Let two indices $1 \leq i \leq j \leq n$. Then, we have:*

1) *if $|\mathrm{out}_3[i,j]| > K$, there exists a loop (k_1,k_2) in ρ with $i \leq k_1 < k_2 \leq j$ such that*
   a) $|\mathrm{out}_3[i,k_1]| \leq K$
   b) $1 \leq |\mathrm{out}_3[k_1,k_2]| \leq K$
2) *if $|\mathrm{out}_3[i,j]| > K$, there exists a loop (k_1,k_2) in ρ with $i \leq k_1 < k_2 \leq j$ such that*
   a) $|\mathrm{out}_3[k_2,j]| \leq K$
   b) $1 \leq |\mathrm{out}_3[k_1,k_2]| \leq K$
3) *if $|\mathrm{out}_1[i,j]| > K$, there exists a loop (k_1,k_2) in ρ with $i \leq k_1 < k_2 \leq j$ such that*
   a) $|\mathrm{out}_1[i,k_1]| \leq K$
   b) $1 \leq |\mathrm{out}_1[k_1,k_2]| \leq K$
4) *if $|\mathrm{out}_1[i,j]| > K$, there exists a loop (k_1,k_2) in ρ with $i \leq k_1 < k_2 \leq j$ such that*
   a) $|\mathrm{out}_1[k_2,j]| \leq K$
   b) $1 \leq |\mathrm{out}_1[k_1,k_2]| \leq K$

Proof of Proposition 6

Proof: We let $T = (Q,q_0,F,\Delta)$ and $K = 2.o.m^3.|\Sigma|$. Recall that as $T' \in \epsilon$*ZNFT*, we have $\mathrm{out}_2[1,n] = \epsilon$.

Let us define the position ℓ as the largest positive integer less than or equal to n such that $\mathrm{out}_1[\ell,n] = \epsilon$.

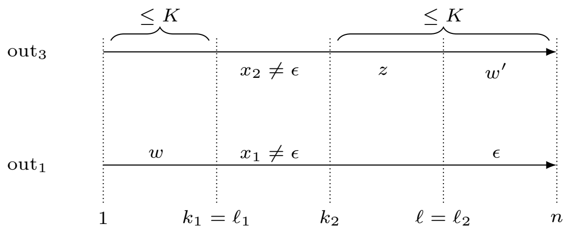

Fig. 7. Decomposition of the output for case I.1)

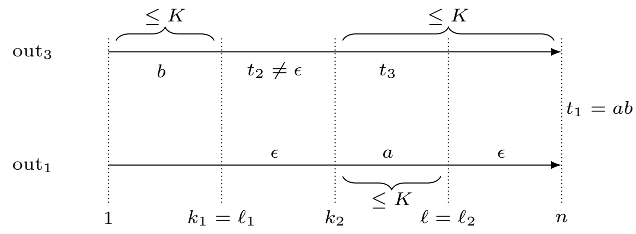

Fig. 8. Decomposition of the output for case I.2)

We first observe that if $|\mathrm{out}_3[1,\ell]| \leq K$, then we are done, by considering $\ell_1 = \ell_2 = \ell$. Indeed, we then consider $w = \mathrm{out}_1[1,\ell]$, $w' = \mathrm{out}_3[\ell, n+1]$, $t_1 = \mathrm{out}_3[1,\ell]$, and $t_2 = t_3 = \epsilon$.

Thus, we now suppose that we have $|\mathrm{out}_3[1,\ell]| > K$. In this case, we can apply Lemma 18, case 1): there exists a loop (k_1, k_2) such that $|\mathrm{out}_3[1,k_1]| \leq K$ and $1 \leq |\mathrm{out}_3[k_1,k_2]| \leq K$.

We again distinguish two cases:

Case I: $|\mathrm{out}_3[k_2,\ell]| \leq K$. For this case, we again distinguish three cases, depending on the value of $\mathrm{out}_1[k_1,k_2]$ and on the length of $|\mathrm{out}_1[k_2,\ell]| \leq K$:

1) if we have $\mathrm{out}_1[k_1,k_2] \neq \epsilon$. We will prove that the output word $\mathrm{out}_1[k_1,n]\mathrm{out}_3[1,\ell]$ has the expected form ($t_1 t_2^* t_3$). Therefore we use the $\mathcal{P}$-property on the loop (k_1,k_2) with an additional empty loop. We define:

$$\begin{aligned} w &= \mathrm{out}_1[1,k_1] \\ x_1 &= \mathrm{out}_1[k_1,k_2] \\ y &= \mathrm{out}_1[k_2,n]\mathrm{out}_3[1,k_1] \\ x_2 &= \mathrm{out}_3[k_1,k_2] \\ z &= \mathrm{out}_3[k_2,\ell] \\ w' &= \mathrm{out}_3[\ell,n+1] \end{aligned}$$

Property $\mathcal{P}$ entails that there exist $\beta_1,\beta_2,\beta_3,f,g,c,c'$ such that, for all $k \geq 0$,

$$f(k)wx_1^{kc+c'}yx_2^{kc+c'}zw'g(k) = \beta_1\beta_2^k\beta_3$$

As we have $x_1 \neq \epsilon$, and $x_2 \neq \epsilon$, this entails, thanks to the fundamental lemma (Lemma 1), that $\mu(x_1) \sim \mu(x_2)$. Let t_2 be $\mu(x_1)$. We can write $t_2 = z_1 z_2$ and $\mu(x_2) = z_2 z_1$. As a consequence, we obtain that $x_1 y x_2$ is of the form $t_2^*.z_1$ by Lemma 17. We can thus set $t_1 = \epsilon$, $t_3 = z_1.z$, $\ell_1 = k_1$ and $\ell_2 = \ell$. It is routine to verify that words w, w', t_1, t_2, t_3 verify the conditions of $\mathcal{P}_2$-property.

This case is depicted on Figure 7.

2) if we have $\mathrm{out}_1[k_1,k_2] = \epsilon$ and $|\mathrm{out}_1[k_2,\ell]| \leq K$. We will show that the result is easy. Indeed, consider $\ell_1 = k_1$, $\ell_2 = \ell$, $t_1 = \mathrm{out}_1[k_1,n]\mathrm{out}_3[1,k_1]$, $t_2 = \mathrm{out}_3[k_1,k_2]$, and $t_3 = \mathrm{out}_3[k_2,\ell]$. It is routine to verify that all the requirements of $\mathcal{P}_2$-property are met.

This case is depicted on Figure 8.

3) last, if we have $\mathrm{out}_1[k_1,k_2] = \epsilon$ and $|\mathrm{out}_1[k_2,\ell]| > K$. In this case, we will have to identify a loop in this part ($[k_2,\ell]$) of the input word, to prove the expected form of

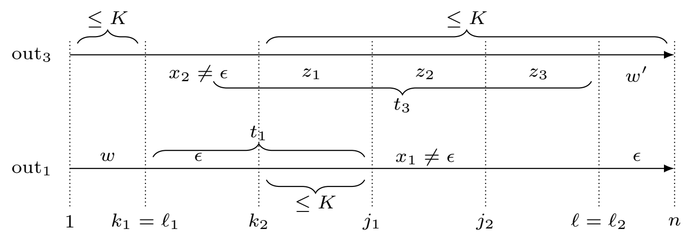

Fig. 9. Decomposition of the output for case I.3)

the output words. Formally, we apply Lemma 18 as we did before, except that we are interested in the output produced in the first pass of the *ZNFT*, and not in that produced in the third pass. We thus apply case 3) of Lemma 18. We can thus exhibit a loop (j_1,j_2) with $k_2 \leq j_1 < j_2 \leq \ell - 1$ such that $|\mathrm{out}_1[k_2,j_1]| \leq K$ and $1 \leq |\mathrm{out}_1[j_1,j_2]| \leq K$.

We are now ready to prove that the output word $\mathrm{out}_1[k_1,n]\mathrm{out}_3[1,\ell]$ has the expected form ($t_1 t_2^* t_3$). To this aim, we define:

$$\begin{aligned} u_1 &= u[1,k_1-1] & w &= \mathrm{out}_1[1,k_1] \\ u_2 &= u[k_1,k_2-1] & t_1 &= \mathrm{out}_1[k_1,j_1] \\ u_3 &= u[k_2,j_1-1] & x_1 &= \mathrm{out}_1[j_1,j_2] \\ u_4 &= u[j_1,j_2-1] & y &= \mathrm{out}_1[j_2,n]\mathrm{out}_3[1,k_1] \\ u_5 &= u[j_2,\ell-1] & x_2 &= \mathrm{out}_3[k_1,k_2] \\ u_6 &= u[\ell,n] & z_1 &= \mathrm{out}_3[k_2,j_1] \\ & & z_2 &= \mathrm{out}_3[j_1,j_2] \\ & & z_3 &= \mathrm{out}_3[j_2,\ell] \\ & & w' &= \mathrm{out}_3[\ell,n+1] \end{aligned}$$

As (k_1,k_2) and (j_1,j_2) are loops, we can apply Property $\mathcal{P}$. Using the fundamental lemma, we can deduce that $\mu(x_1) \sim \mu(x_2)$, using a reasoning similar to that of the proof of Proposition 4. Thus, we can set $t_2 = \mu(x_1)$, and write $t_2 = \alpha_1\alpha_2$ such that $\mu(x_2) = \alpha_2\alpha_1$, from which we deduce $x_1 y x_2 \in t_2^*\alpha_1$ (Lemma 17). Finally, we let $z = z_1 z_2 z_3$, $t_3 = \alpha_1 z$, $\ell_1 = k_1$ and $\ell_2 = \ell$. The reader can verify that all the requirements of $\mathcal{P}_2$-property are met.

This case is depicted on Figure 9.

Case II: we have $|\mathrm{out}_3[k_2,\ell]| > K$. We distinguish three cases, according to the length of the word $\mathrm{out}_1[k_2,\ell]$, and to the value of $\mathrm{out}_1[k_1,k_2]$:

1) if we have $|\mathrm{out}_1[k_2,\ell]| \leq K$, we distinguish two cases:
   a) We first consider the case when $\mathrm{out}_1[k_1,k_2] = \epsilon$.

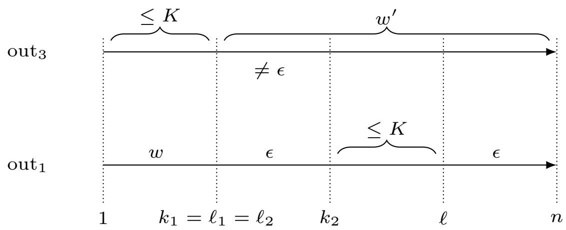

Fig. 10. Decomposition of the output, case II.1).a)

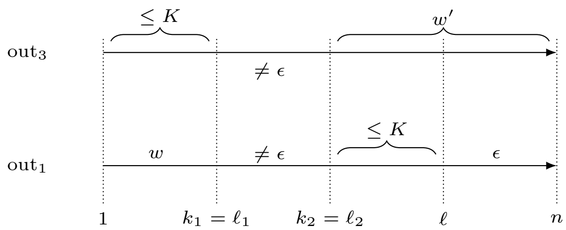

Fig. 11. Decomposition of the output for case II.1).b)

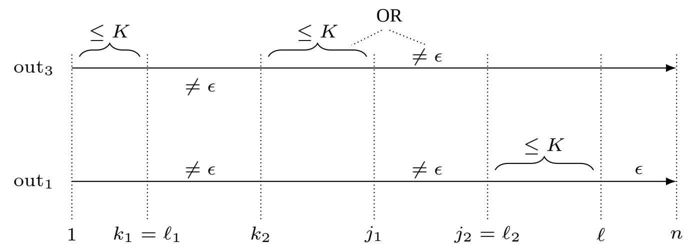

Fig. 12. Decomposition of the output for case II.2).a) and b)

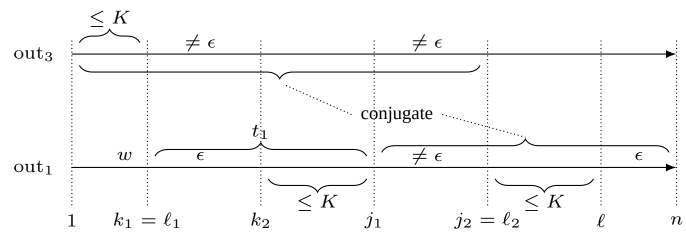

Fig. 13. Decomposition of the output for case II.3).a), $\text{out}_3[j_1, j_2] \neq \epsilon$

In this case, we can simply define $\ell_1 = \ell_2 = k_1$, and verify that the conditions of the $\mathcal{P}_2$-property are met. This case is depicted on Figure 10.

b) The second case is when $\text{out}_1[k_1, k_2] \neq \epsilon$. This case is easy as we can show that $\mu(\text{out}_1[k_1, k_2]) \sim \mu(\text{out}_3[k_1, k_2])$, and deduce the expected form for the output words, by setting $\ell_2 = k_2$. This case is depicted on Figure 11.

2) if we have $|\text{out}_1[k_2, \ell]| > K$ and $\text{out}_1[k_1, k_2] \neq \epsilon$. As $|\text{out}_1[k_2, \ell]| > K$, we can apply Lemma 18, case 4), to identify a loop (j_1, j_2) such that $|\text{out}_1[j_2, \ell]| \leq K$ and $1 \leq |\text{out}_1[j_1, j_2]| \leq K$. In this case, we set $\ell_1 = k_1$ and $\ell_2 = j_2$.

There are three cases, according to $\text{out}_3[j_1, j_2]$ and $\text{out}_3[k_2, j_1]$:

a) We first consider the case when $\text{out}_3[j_1, j_2] \neq \epsilon$. In this case, using Property $\mathcal{P}$, we can show that $\mu(\text{out}_1[k_1, k_2]) \sim \mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[j_1, j_2])$. This allows to prove the expected form of the output words.

b) Second, we suppose that $\text{out}_3[j_1, j_2] = \epsilon$ and that $\text{out}_3[k_2, j_1] \leq K$. In this case, we can use the word t_3 to cover the output word $\text{out}_3[k_2, j_1]$. Last, using a reasoning on word combinatorics, we can prove that $\mu(\text{out}_1[k_1, k_2]) \sim \mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2])$ and conclude.

Cases a) and b) are depicted on Figure 12.

c) Last, we consider the case $\text{out}_3[j_1, j_2] = \epsilon$ and $\text{out}_3[k_2, j_1] > K$. By Lemma 18, case 2), there exists a loop (p_1, p_2) included in the interval $[k_2, j_1]$ such that $|\text{out}_3[p_2, j_1]| \leq K$ and $1 \leq |\text{out}_3[p_1, p_2]| \leq K$. We claim that the result holds. The only difficult property is the fact the output word has the expected form $(t_1 t_2^* t_3)$. This can be proven using word combinatorics, by showing, using the Property $\mathcal{P}$, that $\mu(\text{out}_1[k_1, k_2]) \sim \mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[p_1, p_2])$.

3) last, if we have $|\text{out}_1[k_2, \ell]| > K$ and $\text{out}_1[k_1, k_2] = \epsilon$. We first let $\ell_1 = k_1$. As we have $|\text{out}_1[k_2, \ell]| > K$, we can apply Lemma 18, case 3), to identify a loop (j_1, j_2) included in the interval (k_2, ℓ) such that $|\text{out}_1[k_2, j_1]| \leq K$ and $1 \leq |\text{out}_1[j_1, j_2]| \leq K$. We distinguish two cases:

a) if $|\text{out}_1[j_2, \ell]| \leq K$. We define $\ell_2 = j_2$. We consider the value of $\text{out}_3[j_1, j_2]$.

If we have $\text{out}_3[j_1, j_2] \neq \epsilon$, then we can conclude. Indeed, using word combinatorics, we can prove $\mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[j_1, j_2])$ and prove that the output word $\text{out}_1[k_1, n]\text{out}_3[1, j_2]$ has the expected form. This case is depicted on Figure 13.

Otherwise, we have $\text{out}_3[j_1, j_2] = \epsilon$. For this case we distinguish two cases:

i) if $\text{out}_3[k_2, j_1] \leq K$: we can conclude directly. Indeed, it is easy to show that $\mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2])$. The word $\text{out}_3[k_2, j_2]$ is not necessarily conjugated with the previous words, but its length is less than K by hypothesis, thus we can use the word t_3 to handle this part of the output. This case is depicted on Figure 14.

ii) if $\text{out}_3[k_2, j_1] > K$: we will apply Lemma 18, case 2), to identify a loop (p_1, p_2) included in the interval (k_2, j_1) such that $|\text{out}_3[p_2, j_1]| \leq K$ and $1 \leq |\text{out}_3[p_1, p_2]| \leq K$. Then we can prove that $\mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[p_1, p_2])$ and conclude.

b) if $|\text{out}_1[j_2, \ell]| > K$. We can apply Lemma 18, case 4), to identify a loop (p_1, p_2) included in the

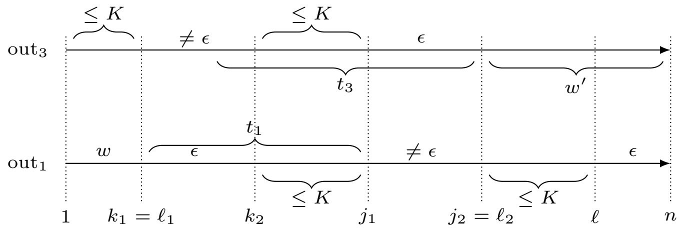

Fig. 14. Decomposition of the output for case II.3).a).i)

interval (j_2, ℓ) such that $|\text{out}_1[p_2, \ell]| \leq K$ and $1 \leq |\text{out}_1[p_1, p_2]| \leq K$. In the sequel, we let ℓ_2 be p_2 and ℓ_1 be k_1. We let $\alpha = \text{out}_3[p_1, p_2]$ and $\alpha' = \text{out}_3[j_1, j_2]$. The situation is depicted on Figure 15. We distinguish five cases:

i) if $\alpha \neq \epsilon$, we conclude easily by showing that $\mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_1[p_1, p_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[p_1, p_2])$.
ii) if $\alpha = \epsilon$ and $|\text{out}_3[j_2, p_1]| > K$: we can identify a loop in ρ, included in the interval $[j_2, p_1]$, such that out_3 is non-empty on this loop. We can then derive the result.
iii) if $\alpha = \epsilon$, $|\text{out}_3[j_2, p_1]| \leq K$ and $\alpha' \neq \epsilon$, then we can show that $\mu(\text{out}_1[j_1, j_2]) \sim \mu(\text{out}_1[p_1, p_2]) \sim \mu(\text{out}_3[k_1, k_2]) \sim \mu(\text{out}_3[j_1, j_2])$, and conclude as the output $\text{out}_3[j_2, p_2]$ has length less than K (t_3 can be defined so as to cover these words).
iv) if $\alpha = \epsilon$, $|\text{out}_3[j_2, p_1]| \leq K$, $\alpha' = \epsilon$ and $|\text{out}_3[k_2, j_1]| > K$, we can identify a loop inside the interval $[k_2, j_1]$. This loop can be used to prove the result, as we know that the length of the word $\text{out}_3[j_1, p_2]$ is less than K.
v) else, *i.e.* if $\alpha = \epsilon$, $|\text{out}_3[j_2, p_1]| \leq K$, $\alpha' = \epsilon$ and $|\text{out}_3[k_2, j_1]| \leq K$, then we are done as t_3 can be defined as $\text{out}_3[k_2, p_2]$.

■

Construction of T'' from T'

We provide here some additional details for the definition of the *NFT* T'' from the ϵ*ZNFT* T'.

First, the transducer T'' should, in a single forward pass, simulate the three passes (forward, backward, and forward) of T'. Therefore it maintains a triple of states of T' and the current symbol.

Second, it uses three modes: before the guess of position ℓ_1, between positions ℓ_1 and ℓ_2, and after position ℓ_2.

Third, it should guess the words of bounded length t_1, t_2 and t_3, and two additional words x and y of bounded length ($\leq 3.K$) which intuitively correspond to words $\text{out}_3[1, \ell_1]$ and $\text{out}_1[\ell_2, n]$ (see property $\mathcal{P}_2$).

Last, it verifies in the different modes that the output has the expected form, and produces in a forward manner the overall output word. Therefore it distinguishes between different cases, whether t_1 is a prefix of $\text{out}_1[\ell_1, \ell_2]$ or whether t_1 also covers $\text{out}_1[\ell_2, n]$ or $\text{out}_3[1, \ell_1]$, or even $\text{out}_3[\ell_1, \ell_2]$. It manipulates pointers in the different words of bounded length it has guessed to verify the form of the output, and to produce the correct output, as we did in the construction of T'.

Appendix C
Lower Bound

Lemma 19. *(2DFT, NFT)-definability is PSpace-Hard.*

Proof: Consider n *DFAs* $A_1, \ldots, A_n$. Let us define the following transduction (where $\# \notin \Sigma$):

$$T \quad : \quad u \quad \mapsto \quad \begin{cases} \overline{u_1} \text{ if } u = \#u_1\#u_2\# \text{ and } u_2 \in \bigcap_i L(A_i) \\ \text{undefined otherwise.} \end{cases}$$

Clearly, T is definable by a *2DFT*. It suffices to first perform n back and forth non-producing passes on u to determine whether $u_2 \in \bigcap_i L(A_i)$, and then a last backward pass to reverse u_1.

Then, T is *NFT*-definable iff $\text{dom}(T) = \emptyset$ iff $\bigcap_i L(A_i) = \emptyset$. Indeed, if $\text{dom}(T) = \emptyset$ then T is obviously *NFT*-definable. Otherwise, there exists $u_2 \in \bigcap_i L(A_i)$, and therefore $\#\Sigma^*\#u_2\# \subseteq \text{dom}(T)$. If T is *NFT*-definable, then so would be the reverse operation. Contradiction. ■

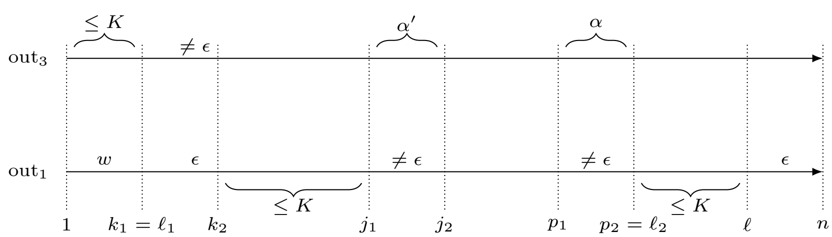

Fig. 15. Decomposition of the output for case II.3).b)