THÈSE

PRÉSENTÉE À

L'UNIVERSITÉ DE BORDEAUX

ÉCOLE DOCTORALE DE MATHÉMATIQUES ET INFORMATIQUE

par **Félix BASCHENIS**

pour obtenir le grade de

DOCTEUR

SPÉCIALITÉ: INFORMATIQUE

Minimizing resources for regular word transductions

Soutenue le **05 Décembre 2017** devant un jury composé de:

Anca MUSCHOLL	Professeur, Université de Bordeaux	Directrice
Olivier GAUWIN	Maitre de conférences, Université de Bordeaux ..	Co-encadrant
Gabriele PUPPIS	Chargé de recherches, CNRS	Co-encadrant
Olivier CARTON	Professeur, Université Paris Diderot	Rapporteur
Pierre-Alain REYNIER	Professeur Université de Aix-Marseille	Rapporteur
Emmanuel FILIOT	Chercheur qualifié, Université Libre de Bruxelles	Examinateur
Sylvain LOMBARDY .	Professeur, Université de Bordeaux	Président du jury
Sylvain SALVATI	Professeur, Université de Lille	Examinateur

Laboratoire d'acceuil

Laboratoire Bordelais de Recherche en Informatique (LaBRI)
Unité mixte de recherche CNRS (UMR 5800)
351, cours de la libération F-33405 Talence cedex,
France

Abstract

The goal of this thesis was to study definability questions about finite-state transducers and in particular two-way transducers. It is known that two-way transducers cover a larger class of transductions than one-way transducers. Then the first question we tackled is the *one-way definability problem*: is it possible to realize a given two-way transduction by a one-way transducer? This problem was shown to be decidable for functional transducers[1] but the decision procedure had non-elementary complexity [39]. We proposed a characterization of *one-way definability* that allows us to decide it in double-exponential space, and provide an equivalent one-way transducer of triple-exponential size. We first studied this question for a restricted class, namely *sweeping transducers*, for which the decision procedure and the construction of the one-way transducer take one less exponential. For such transducers, our procedure is optimal in the sense that we have shown that there exists a family of functions that are one-way definable and for which an equivalent one-way transducer requires doubly exponential size.

The study of sweeping transducers raised other definability questions: Is a given transducer equivalent to some sweeping transducer? And to some sweeping transducer that performs at most k passes? We showed that those questions are decidable and the decision procedure, as well as the equivalent transducer have the same complexity as in the one-way case. Moreover, as we have shown that there exists a bound on the number of passes required to realize a transduction by a sweeping transducer, we managed to obtain a procedure to minimize the number of passes of a sweeping transducer.

Finally we tried to characterize sweeping transducers in other models for regular transductions such as *Streaming String transducers* (*SST*) and *MSO* transductions on graphs. As we obtained an equivalence between the

[1]We also show as a side result that one-way definability becomes undecidable for non-functional transducers.

number of passes of a sweeping transducer and the number of registers of the equivalent *SST* we provided a minimization procedure for the number of registers of a large class of *SST*'s. To conclude, our work allowed us to provide a good overall understanding of the definability questions between the models for regular transductions and in particular regarding the resources, whether it is the number of passes (and of course one-way definability is crucial in that aspect) or the number of registers.

Keywords: Automata, Logic, Transduction, One-way and two-way transducers, *SST*, *MSO* transduction

Résumé

Cette thèse a eu pour objectif d'étudier des questions naturelles de définissabilité autour des transducteurs bidirectionnels. Il est bien connu que les transducteurs bidirectionnels forment une plus grande classe de transductions que celles définissables par les transducteurs unidirectionnels. La première question que nous avons étudiée est donc de décider si un transducteurs bidirectionnel est définissable par un transducteur unidirectionnel. Il a été montré en 2013 [39] que cette question est décidable pour des transducteurs fonctionnels [2] mais la complexité de la procédure de décision était non-élémentaire.

Nous proposons une caractérisation de la "définissabilité par transducteur unidirectionnel" décidable en espace doublement exponentiel. Cette caractérisation est effective en ce sens qu'elle produit en temps triplement exponentiel le transducteur équivalent. De plus, nous avons commencé à étudier ce problème pour les transducteurs "sweeping", pour lesquels la procédure de décision et la construction du transducteur équivalent requièrent une exponentielle de moins. Comme nous avons par ailleurs montré qu'il existe des familles de fonctions réalisables de façon unidirectionnelle avec au minimum deux sauts exponentiels, notre procédure est optimale dans le cas "sweeping".

Le fait d'avoir particulièrement étudié les transducteurs "sweeping" nous a poussé à étudier d'autres questions de définissabilité : est-ce qu'un transducteur donné est réalisable par un transducteur sweeping ? Et par un transducteur sweeping réalisant au maximum k passages ? Nous montrons que ces questions sont décidables avec les mêmes complexités obtenues précédemment. Comme nous avons montré qu'il existe une borne sur le nombre de passages nécéssaires pour réaliser avec un transducteur sweeping une transduction donnée, cela nous permet de minimiser le nombre de passages

[2]Nous montrons aussi en paralèlle que cette question devient indécidable si les transducteurs ne sont plus fonctionnels.

d'un transducteur sweeping.

Enfin nous avons cherché à caractériser la classe des transductions sweeping dans d'autres modèles de transductions, les *Streaming String Transducers* (*SST*) et les transductions *MSO*. Cela a en autres permis, en établissant une correspondance entre le nombre de passages des transducteurs sweeping et le nombre de registres d'une sous-classe de *SST*, de minimiser le nombre de registres pour une classe intéressante de *SST*. Dans l'ensemble, notre travail a permis de couvrir l'ensemble des relations entre ces modèles, et les questions de définissabilité qui se posent naturellement.

Mots-clés: Automates, Logique, Transduction, Transducteur unidirectionnel et bidirectionnel, *SST*, Transduction *MSO*

Description de la thèse en français.

Théorie des languages.

Cette thèse s'inscrit dans le cadre de l'étude des langages formels. Un langage est un ensemble de mots, c'est à dire de séquences de lettres. C'est un outil théorique pour définir et étudier les langages naturels, les langages de programmation, ou les langages des propriétés mathématiques de certains objects. La communauté scientifique a proposé au cours du vingtième siècle plusieurs modèles de machines à calculer permettant de représenter des langages (comme les machines de Turing, qui furent introduites avant les automates) mais celui qui nous occupera dans cette thèse sont les automates: des machines à mémoire finie, ce qui est une contrainte naturelle. Les automates sont des machines avec un ensemble fini d'états, un alphabet décrivant les entrées possibles du calcul, et une fonction de transition permettant de passer d'un état à un autre lors de la lecture (qui se fait de gauche à droite) de l'entrée. Très vite les chercheurs ont étudiés des variantes de ce modèle, en rajoutant du non-déterminisme, la possibilité de bouger la tête de lecture sur l'entrée dans les deux directions (de gauche à droite et de droite à gauche) ainsi que la possibilité d'étiquetter des transitions par le mot vide ϵ. Tout ces modèles se sont trouvés être équivalents (en particulier, l'équivalence entre les automates unidirectionnels et bidirectionnels a été montré de façon indépendante par Shepherdson [78] d'une part, et part Rabin et Scott [66] d'autre part, et ce de deux façon différente). De plus, des résultats comme la description logique des langages acceptés par les automates (par *MSO* sur les graphes) ou l'équivalence avec la présentation plus algébriques en terme d'expression régulières ont montrés que cette classe de langage était robuste et stable (par exemple en tant stable par union, concatenation ou produit), et elle fut appelée *Langages réguliers.*

Transducteurs.

Nous nous sommes plus particulièrement intéressés aux transducteurs qui sont une façon d'étendre les automates afin qu'ils reconnaissent des fonctions ou des relations sur les mots, et non plus des ensemble de mots. Ces machines furent introduites à la même période que les automates, et ont de nombreuses applications pratiques (en base de données, traitement du langage, reconaissance visuelle, etc). Par rapport aux automates, la notion de fonction ou de relation régulière est beaucoup plus compliquées à dégager que celle de langage régulier. En effet les transducteurs non-deterministe ne sont pas équivalents aux transducteurs déterministes, de la même façon que les transducteur unidirectionnel ne sont pas équivalents aux transducteurs bidirectionnels. Des questions de définissabilité découlent de ces obversations: Etant donné un transducteur bidirectionnel ou non-déterministe, est-il possible de réaliser la même transduction avec un transducteur unidirectionnel ou déterministe? La procédure de décision pour décider la définissabilité par transducteur déterministe fut trouvé relativement tôt [20], mais le seul résultat avant cette thèse sur la définissabilité par transducteur unidirectionnel date de 2013 et la preuve de la décidabilité de cette question [39], avec une complexité non-élémentaire. Notre premier objectif fut de chercher à améliorer cette complexité.

Motivations.

Il est bien sur important d'un point de vue théorique de connaitre les relations entre différents modèles de transducteurs, mais une telle procédure de décision aurait aussi des implications pratiques. Avec l'accroissement de la vitesse des connexions et du coût du stockage des données, être capable de traiter l'entrée d'une machine sans avoir à l'enregistrer est utile en pratique. Un transducteur unidirectionnel déterministe, comme il lit l'entrée de gauche à droite sans jamais y revenir n'a pas besoin de la garder en mémoire. Ce n'est pas le cas d'un transducteur bidirectionnel qui a besoin d'accéder à la totalité de l'entrée à n'importe quel moment de son calcul. C'est pourquoi il est parfois plus intéressant d'utiliser des transducteurs unidirectionnels utilisant plus de mémoire qu'un transducteur bidirectionnel, mais ne dépendant pas de la taille de l'entrée (qui ne doit plus être stockée).

Ces questions nous ont amenées à réfléchir à la notion de coût en mémoire pour les machines bidirectionneles. Nous avons étudié le nombre de retours en arrière des transducteurs bidirectionnels (il se trouve, comme nous l'avons montré dans [9] que c'est la même chose que le nombre de passages d'un transducteur à balayage). Nous avons donc cherché à résoudre la définissabilité par transducteur à balayage réalisant k passages (cette question, quand $k = 1$, est exactement la définissabilité par transducteur unidirectionnel), puis la minimisation du nombre de passages. Enfin, il était logique de finir en étudiant d'éventuelles correspondances avec d'autres notions de mémoire dans d'autres modèles.

Chapitre 2: Présentation des modèles.

Ce chapitre introductif est dédié aux définitions des modèles utilisés, et sert à illustrer les différences entre ces modèles ainsi que les questions naturelles qui se posent. Nous présentons trois grands modèles de transduction: les transducteurs à état finis, les transductions MSO sur les mots, et les SST. Nous nous sommes concentrés sur la présentation des outils techniques utilisés dans cette thèse, comme les *crossing sequences*, et des relations importantes permettant de dégager la notion de transduction régulière (Figure 2.18). L'on pourra remarquer que dans chacun des modèles, les questions sont souvent les mêmes (déterminisation, fonctionalité, définissabilité par un modèle plus simple, minimisation de ressources), et quelques fois, les idées mises en place pour y répondre également. Bien que ce chapitre ait surtout pour but de synthétiser la problématique autour des modèles de transductions, quelques unes de nos contributions sont présentées ici car elles donnent une bonne compréhension globale du sujet.

Chapitre 3: définissabilité par transducteur unidirectionnel.

La complexité non élémentaire obtenue dans [39] repose en partie sur le fait que les auteurs ont adaptés la transformation d'un automate bidirectionnel en automate unidirectionnel de Rabin et Scott [66], qui élimine des parties de runs dont la forme ressemble à un zigzag, en les contractant. Dans le cas des transducteurs, il faut analyser chacun de ces zigzag afin de savoir s'ils sont réalisables avec un seul passage de gauche à droite. Pour fournir une complexité élémentaire à ce problème nous avons adaptés la méthode de Shepherdson, qui permet de produire directement un automate équivalent en regardant des tranches de run. C'est ce qui nous occupe dans le chapitre 3 de cette thèse. Il est séparé en deux parties. Nous avons d'abord étudié le cas des transducteurs à balayage, pour ensuite étudier le cas général.

Notre preuve repose sur le fait que des productions périodiques réalisés lors de deux passages consécutifs peuvent être produite de façon unidirectionnel en produisant le nombre de lettre produit par chaque tranche, dans l'ordre de la période. Cela nous permet d'introduire une notion de décomposition d'un run en parties bornées (donc devinables en utilisant le non-déterminisme), en parties de gauche à droite, et en parties périodiques. Si chaque calcul acceptant admet une telle décomposition, alors nous pouvons construire un transducteur qui devine une telle décomposition et produit sa sortie de façon unidirectionnelle, selon cette décomposition. Ce transducteur, qui devine un nombre fini de mots de taille exponentielle, a une taille doublement exponentielle. La partie délicate de ce théorème est de s'assurer qu'un transducteur définissable par transducteur unidirectionnel admet toujours une telle décomposition.

Nous réalisons cela en utilisant de la combinatoire sur les mots. Pour obtenir des productions périodiques, nous introduisons la notion d'inversion, c'est à dire des paires de positions dans le calcul, dont l'ordre horizontal est inverse à l'ordre vertical: l'une est à gauche et en haut de l'autre. Ainsi, l'ordre des mots produits est inversé par rapport à l'ordre dans l'entrée, ce qui nous permet d'obtenir une équation de mots aux propriétés remarquables. Une analyse combinatoire de cette équation permet enfin d'obtenir la périodicité de la sortie entre les deux positions considérées.

Le cas général est étudié dans la deuxième partie du chapitre 3. Les notions d'inversions et de décompositions sont identiques, mais la présence de retours en arrière au sein du calcul rend compliqué l'étude des boucles de celui-ci. Leur étude a nécessité beaucoup de travail et d'attention, afin de comprendre quelles propriétés étaient nécessaires pour obtenir les mêmes équations sur les mots que dans le cas des transducteurs à balayage. La formalisation de ces propriétés s'est révélée technique et a nécessitée d'utiliser des outils issus de la théorie des Ramsey afin d'obtenir les objets adéquats.

Finalement nous avons réussi à obtenir une complexité élémentaire en espace doublement exponentiel pour décider cette question, et ce de façon effective puisque nous somme capables de fournir un transducteur unidirectionnel équivalent (lorsque c'est possible, évidemment) de taille triplement exponentiel. Il est à noter que l'effort fait pour dénouer les boucles des transducteurs bidirectionnels nous coute une exponentielle, et que notre construction est optimale dans le cas des transducteurs à balayage puisque nous avons montré l'existence de fonctions définissable par transducteurs unidirectionnels, dont la taille est nécessairement doublement exponentielle.

Chapitre 4: définissabilité par transducteurs à balayage et minimisation.

Nous avons d'abord remarqué que les idées utilisées pour la définissabilité par transducteurs unidirectionnels fonctionnaient de la même manière pour la définissabilité par transducteurs à balayage réalisant k passages. De plus, nos travaux sur les boucles bidirectionnelles nous ont permis de trouver une borne sur le nombre maximal de passages réalisé par un transducteur à balayage équivalent à un transducteur bidirectionnel donné T. Cela nous a permis de minimiser le nombre de passages à la fois pour les transducteurs à balayage et pour les transducteurs bidirectionnels.

Enfin, nous avons identifiés une équivalence entre les transducteurs à k passages et une sous-classe de SST possédant $2k$ registres. Cela nous a donc permis de minimiser le nombre de registres dans cette sous-classe de SST.

Publications.

La preuve de définissabilité pour les transducteurs à balayage du Chapitre 3 a été publiée dans [8] ainsi que plusieurs résultats du Chapitre 2. Dans [9] nous traitons principalement des questions de minimization du Chapitre 4. Cet article contient également des résultats sur les transducteurs à balayage, comme la sous-classe des transductions MSO qui leur correspond, ainsi que leur équivalence avec les transducteurs bidirectionnels à allers-retours bornés. Ce travail a été originellement écrit pour les transducteurs à balayage, mais comme nous avons plus tard résolu la question de la définissabilité par transducteur unidirectionnel dans le cas général, nous avons adapté cet article pour produire le Chapitre 2. Enfin, nous avons terminé ce travail par la publication du cas général de la définissabilité par transducteur unidirectionnel dans [10], qui correspond à la deuxième partie du Chapitre 3

Remerciements

Cette thèse n'aurait pu voir le jour sans la présence d'un certain nombre de personnes à mes cotés que je me dois de remercier ici. Tout d'abord je remercie grandement Olivier Carton et Pierre-Alain Reynier pour avoir accepté de rapporter mon travail. Je suis également honoré de compter Emmanuel Filiot, Sylvain Lombardy et Sylvain Salvati parmi mon jury.

Un parcours scolaire est toujours marqué par des rencontres, sources d'inspiration, de confiance, ou de satisfaction intellectuelle. Au secondaire, Monsieur Ravassard, rapidement suivi par Madame Veyre, ont su cultiver ma curiosité, respectivement en mathématiques et en physique. Mes deux professeurs de mathématiques en classes préparatoires à Janson-de-Sailly, Shaley Mohan et Luc Abergel ont énormément compté pour moi (je n'oublie pas Marc-Antoine Blain, un professeur exceptionnel de physique, discipline que j'avais malheureusement abandonné). Aussi différents dans leur approche que passionnés et pédagogues, ils m'ont inculqué deux façons complémentaires de comprendre et de faire des mathématiques qui coexistent encore en moi et dans ce travail. Si toute l'équipe enseignante du département d'informatique de l'ENS Cachan est de qualité, je tiens à remercier particulièrement Serge Haddad, qui a été d'une patience et d'une compréhension exemplaire à mon égard, ainsi que Sylvain Schmitz dont la gentillesse et le talent oratoire ne sont plus à démontrer. Au cours de mes années à l'ENS, j'ai eu la chance de faire des stages avec d'excellents encadrants, qui m'ont beaucoup appris: Christophe Paul, Paul Blain Levy et Amélie Gheerbrant. Il me faut saluer aussi mes directeurs de Master, Alexis Saurin et Arnaud Durand, que je ne remercierai jamais assez de m'avoir mis le pied à l'étrier, que ce soit avec Amélie (avec qui ce fut un réel plaisir de travailler!) ou avec cette thèse. Je salue également les jeunes chercheurs que j'ai eu la chance de rencontrer (certains sont des amis, d'autres j'espère le deviendront avec le temps) que ce soit pendant mes études (Alice, Edon, Nathan, Louis-Marie) en conférence

(Michael, Ismaël, Guillermo) ou aux EJC (Florent, Bruno, Manon, Elise, Florian, Hamza, Anastasia, et j'en oublie) et qui m'ont aidé à ressentir mon appartenance à ce milieu particulier qu'est la recherche.

Enfin, je ne saurai quoi dire pour évoquer mes responsables lors de ces trois années Anca Muscholl, Olivier Gauwin et Gabriele Puppis (qui, bien que n'en n'ayant pas le titre, ont été tous les deux de réels directeurs de thèse à mes yeux). Je ne peux imaginer meilleur encadrement pour une thèse, que ce soit au niveau de la connaissance du sujet et des perspectives associées, qu'au niveau du soutien personnel et de l'apprentissage du métier de chercheur. Ce travail et mon avenir leur doivent (presque) tout.

Pour le reste, ma famille, mes amis, celles et ceux qui sont dans mon coeur le savent (mis à part peut-être Anne, qui j'espère s'en rappellera si elle tombe un jour sur cet ouvrage). J'ai beau tenir à mon indépendance je sais tout ce que je dois à mes proches et ne l'oublie pas, mais n'utiliserai pas ces lignes pour leur dire des choses que je tais habituellement.

"L'homme a beau étendre le cercle de ses idées,
sa lumière n'est toujours qu'une étincelle
promenée dans la nuit immense qui l'enveloppe."

Pierre-Joseph Proudhon, *Mélanges.*

À Laetitia, puisse-t-elle à son tour
promener son étincelle où bon lui semble.

Contents

Chapter 1

Introduction

General considerations. Computer science is a recent domain of modern science, and is a consequence of the developments in logic and the foundations of mathematics in the first half of the twentieth century, as well as a theoretical tool needed to support the birth of modern computing in its second half. This dichotomy still holds nowadays and most of the time, the motivations behind a topic lie in a practical technological question (for instance about networks or connected objects), and the theoretical aspects involved in its resolution require some advanced mathematical notions. This makes it difficult for a non-specialist (but very stimulating for the computer scientist) to understand what computer science is about, that is the notion of information, which can be studied from different points of view and with different concerns in mind. What is a valid information and how do we express it? How can we process it and what is exactly a computation? How is this information obtained, modified, and exchanged? All those questions and many others required new frameworks and gave birth to different areas of computer science. The domain that will interest us rose from the following issue: what is the simplest device that provides a general computation for any instance of a given problem?

After we understood that any finite information can be modeled by a sequence of 0 and 1 (or as a mathematician would say, an element of the free semigroup over a finite set) the study of formal languages (that is sets of such sequences) became an efficient way to verify syntactical properties of objects. As a scientist, my appeal for formal language theory, and in particular automata theory, is that it is a symbol of the specificity of computer science. For instance the concept of automata, as a machine performing a sequence of elementary operations using some instructions and a finite memory, is natural, practical, and visual. However the study of automata and regular languages in general involves some very deep algebra. I hope this thesis will be a good example of how intuitive ideas can sometimes hide behind advanced mathematics.

General context of formal language theory. A language is a set of words, that is sequences of letters. It can be a framework for reasoning about natural languages, programming languages, or mathematical properties of some model of concrete objects. As the number of words is infinite, we might have infinite languages and be unable to provide a list of the words that belong to those. Then a first question arises: how can we use a language if we cannot decide if a word belongs to that language ? This question, that is representing an infinite set by a finite object is at the core of formal language theory.

Of course one can provide several models of computation (for instance Turing machines were introduced before automata) but deterministic finite-state automata are the simplest object when we want to use a finite amount of memory (which is a natural requirement given the current state of computer technologies). In particular, contrary to Turing machines, many decision problems are decidable thanks to this finite memory requirement (for instance, model checking). Quickly the scientific community studied slight variations of this model, for instance the introduction of non-determinism [66], the possibility to move the head of the input tape in both directions [66, 78], or the existence of ϵ-transitions, and those models have been proved to be equivalent (in particular the equivalence between two-way and one-way automata have been shown independently by Shepherdson, and by Rabin and Scott [66, 78]). Moreover, some links with the languages described by logic (in particular *MSO* on linear graphs [18, 34, 83]) or algebraic presentations (what we called regular expressions [55]) were discovered at the same time. Those equivalences implied that such languages were stable by union, concatenation, and product, and have been called *regular* languages as they form a very robust class of languages.

Automata and regular languages have found a lot of applications in formal verification, programming language theory in general and compilers in particular but it remained a domain of theoretical research in itself, for instance concerning the memory usage of the machines. Even if it is a natural path to explore that many have tried to take, some problems such as the cost of turning a non-deterministic automaton into a deterministic two-way one (asked in 1978 by Sakoda and Sipser [74]) are open since almost forty years. Such topics are now sometimes studied under the name minicomplexity [53].

In parallel, automata were enhanced either to read more complex inputs (for instance trees [67], or infinite words [17]), or to provide more information than accepting or not an input word (for instance weighted automata [75] or more recently cost-register automata [3]). One of the most natural way to extend automata is to provide them with an output tape so that they define functions (or relations, in the non-deterministic setting) over the free semigroup instead of subsets. We call such machines transducers and such relations transductions. Transducers were introduced at the same time as automata [63] and are used in a variety of fields of computer science including database theory, image and language processing, and machine learning. It is after all quite natural: computers are essentially machines that transform information or provide a result. The study of finite transducers, the relations they compute, how they compare to other models of transductions, and some definability questions that they raise is the global goal of this thesis.

Overview of finite transducers. The notion of regularity became quickly important and turned automata theory into one of the major fields of computer science. It also became clear that the situation was much more complex for transducers.

Indeed, two-way transducers, as well as non-deterministic ones, are not equivalent to one-way deterministic transducers. The decision procedure for checking if a one-way transducer can be turned into a deterministic one was found quite early [20] but this area remained badly known for a long time. For instance, the equivalence of transducers with a logical characterization was only obtained in 2001 [36] after the introduction of the more general model of graph transductions by Courcelle in the early 90's [25]. Unfortunately this equivalence is only true in the deterministic case, but this result raised new interests for transducers. In particular, Alur introduced streaming string transducers: a model of machines that are equivalent to the logical characterization both in the deterministic and non-deterministic cases [2, 4]. One of the results following this renewed interest can be seen as the starting point of this thesis: the decidability of the possibility to turn a two-way transducer into a non-deterministic one-way transducer [39].

Motivations. The above result is quite important, and was only proved more than fifty years after we understood that the equivalence between one-way and two-way automata did not lift to transducers. The fact that the demonstration of [39] yields a non-elementary complexity of the decision procedure (while the best lower bound was in PSpace) led us to think that there is something that we still did not understand about this problem. This intractable complexity lies in the fact that the authors adapted the translation from two-way to one-way automata by Rabin and Scott [66], which tries to eliminate factors of runs that are shaped like *zigzags* by *squeezing* them. However in the transducer case, one does not know *a priori* which zigzag will proved to be impossible to simulate with a single pass, which results in a lot of applications of the squeezing procedure. The main goal of this thesis was to provide an elementary effective procedure for deciding this problem, that provides an equivalent transducer of reasonable size.

It is of course important to cover the relations between the different models of transducers, as those two models are natural ways to represent transductions, but such an efficient procedure is also interesting from a concrete point of view. With the increasing speed of information transmission, and the increasing cost of information storage, being able to process the information a machine receives without storing it is very helpful. A deterministic one-way transducer, as it reads its input once from left-to-right does not need to keep all its input in memory in order to look back at it later: it computes its result in real-time. However a two-way transducer needs to be able to access the totality of the input at any time and does not have this practical streaming property. This is why it is often more interesting in practice to use large one-way transducers than concise two-way ones. We managed to obtain an elementary procedure for deciding one-way definability which provides an equivalent one-way transducer. Moreover the size of that transducer is not too big, and even optimal for a subclass of two-way transducers, namely *sweeping* transducers. We used for this result an approach similar to the one of Shepherdson [78] (recall that the equivalence between one-way and two-way automata was proven independently by Shepherdson, and Rabin and Scott). Using a similar construction we can consider the whole run of the transducer and provide a condition on its shape.

This question about one-way definability led us to review the notion of memory cost for two-way machines. As the inputs of machines are of unbounded length, the issue of storing and accessing it is in practice at least as much important as the usual notions of space such as the number of states. Thus we naturally studied the number of reversals of two-way transducers (or number of passes in the case of sweeping transducers), how it was linked to other notions of space in different models, and if it was possible to minimize it. Finally, thanks to an elegant equivalence between some restrictions of Alur's streaming transducers and sweeping transducers, we managed to minimize the number of variables in some class of streaming transducers, which is a very natural and interesting open problem.

Overview. Chapter 2 introduces the computational models that are used in this thesis: finite-state transducers, *MSO* transductions on words, and streaming string transducers. We focused on developing the technical notions necessary to understand the questions of definability and equivalence that are tackled in this thesis, and are summarized in Figure 2.18. The reader will see that for each model the issues are often the same (determinization, one-way definability, resource minimization) and most of the time, equivalences between machines are helpful to derive new results for others models. Some of our new contributions can be found in this chapter, often with short or simple proofs, as they fit well in the presentation of the relations between models.

In Chapter 3 we tackle the main question of this thesis, that is one-way definability for finite-state transducers. This chapter contains two parts: the first one solves this problem for the restricted class of sweeping transducers, and the second one shows how to extend this result to the general case of two-way transducers. We believe that this presentation is simpler, as the main ideas are contained in the proof of the sweeping case, and the difficulties of the general cases lie in the combinatorics details. The new ideas in that case lie mainly in a good understanding of loops of two-way transducers. Another reason to follow this presentation is that we provided the proof of the sweeping case two years before the general case and it has lead us to look a bit more carefully at this class and ask new questions about it.

Those questions are dealt with in Chapter 4. First we solve a generalization of one-way definability, namely k-pass definability (that is, is the transducer equivalent to a sweeping transducer that performs at most k passes?). Then, by providing a bound on the number of passes of an equivalent sweeping transducer when it exists, we obtain a procedure for deciding sweeping definability and the minimization of the number of passes of sweeping transducers. Finally we present an equivalence between sweeping transducers and a restriction of streaming string transducers. This enhances the interest of the class of sweeping transducer as it allows us to solve a variable minimization problem for a large class of streaming transducers which is an important question, still open in the general case.

Author's publications. The proof of one-way definability for sweeping transducers in the first half of Chapter 3 was published in [8] along some results of Chapter 2 such as Proposition 2.2.8. In [9] we used this result to solve k-pass definability and the minimization questions of Chapter 4. This article also contains some insights about sweeping transducers, such as the logical characterization of Theorem 2.3.5, and one result that we did not present in this thesis: the equivalence between sweeping transducers and two-way transducers with a bounded number of reversals. This work was written in [9] for sweeping transducers, but as we managed later to solve one-way definability in the general case, we adapted for this thesis those results to two-way transducers. Finally we concluded this work with the publication of the general case of one-way definability in [10] which is presented in the second half of Chapter 3.

Chapter 2

Models for regular transductions

If in formal language theory regular languages are well-studied, the analysis of what we call transductions, that is functions or relations over words instead of sets, is less advanced. In this chapter, we describe the formal models of transductions that we will use and present how they relate together, justifying the notion of regular transductions.

We focus on three models:

- finite state machines with output: **2**-way (**N**on) **D**eterministic **F**inite state **T**ransducers (*2DFT*, *2NFT*)
- logical formulas defining transductions: (**N**on) deterministic **M**onadic **S**econd **O**rder **T**ransducers (*MSOT*, *NMSOT*)
- automata with write-only registers: (**N**on) **D**eterministic **S**treaming **S**tring **T**ransducers (*DSST*, *NSST*)

The deterministic models yield to partial functions instead of relations. Hence, determinization questions can only be asked in the functional setting where transductions defined are partial functions. As a consequence, the functionality of a transduction defined by a transducer will be studied in each of those models.

First we will lay in Section 2.1 some common grounds on the basics of formal language theory, that is automata and regular languages. It is well-known that non-deterministic and deterministic automata are equivalent [71], as well as two-way and one-way automata [66, 78]. We define in detail crossing sequences (heavily used in [78]) as they will be at the core of some transducer transformations presented in this Thesis.

Adding an output component to the transition function is the natural way to turn automata into machines computing functions, and finite state transducers were the first model introduced [44, 63, 76]. Yet, it appeared that they are more challenging than automata for which many questions were quickly solved [55, 61, 66, 78]. In particular non-deterministic and deterministic transducers are not equivalent (even in the functional case), and neither two-way and one-way ones. After some formal definitions we develop the new questions raised by transducers in Section 2.2.

One early and famous result in automata theory is the equivalence between regular languages and word languages described by Monadic Second Order logic (*MSO*) [18, 34, 83]. An analoguous model for transductions using a logical formalism (denoted *MSOT* and *NMSOT*) has been provided within the more general framework of graph transductions introduced by Courcelle [25]. The effective equivalence between *MSOT* and *2DFT* is shown in [36], but this equality is not true in the relational case, which is one of the reasons to consider only functional models (equivalent to *2DFT*). We define in Section 2.3 a syntactic property on *MSO* formulas that characterizes one-way (resp. *sweeping*) transducers [9, 15, 37]. If first-order logic corresponds to aperiodic word languages [60, 77], the situation is more complex in the transduction case that we will review at the end of the section.

Finally, Alur et al. introduced recently streaming string transducers (*DSST*, *NSST*), automata with write-only registers, that are equivalent to *MSO* transductions respectively in the deterministic and non-deterministic setting with effective translations [2, 4]. We introduce the determinization and minimization questions on *SST* in Section 2.4.

We conclude this chapter by giving an overview of the relations between these models and their restrictions.

2.1 Words, languages, and automata

Automata and the study of formal languages on finite alphabets is historically one of the first topics in theory of computation. Many results have been produced over the years, and we present in detail only those useful to this thesis. We will conclude by exhibiting some long-lasting open questions.

2.1.1 General definitions

Words over a finite alphabet are a natural way of representing linear information: numbers, sequence of bits, texts, encodings, etc... Moreover, the structure of such words is well-described in an algebraic framework.

Definition 2.1.1. *A word w over a finite alphabet Σ is a finite sequence of letters, that is an element of Σ^n for some $n \in \mathbb{N}$. The integer n is the length of the word w and is denoted $|w|$. If w is the sequence $a_1, a_2, \ldots, a_n$ we write $w = a_1 \cdots a_n$ and $w(i) = a_i$, that is the letter at the i-th position of w.*

The set $\Sigma^* = \bigcup_{k \in \mathbb{N}} \Sigma^k$ of all words is a monoid whose composition rule is the concatenation: $(a_0 \cdots a_n) \cdot (a'_0 \cdots a'_k) = a_0 \cdots a_n a'_0 \cdots a'_k$, and the neutral element is the empty word ϵ.

A language is a set (possibly infinite) of words over Σ, that is, a subset of Σ^*. As we will see, regular languages are sets that may be infinite but have a finite representation by mean of a machine, an expression or a logical formula. As we focus here on the machine aspects of the theory of rational languages, we will introduce this subject by defining recognizable languages as those that are accepted by some kind of automaton. The reader can refer to the usual textbooks in the domain [51, 71, 81] for more details on rational languages and rational expressions.

Definition 2.1.2. *A* two-way non-deterministic automaton *(*2NFA*) is a tuple $\mathcal{A} = (Q, \Sigma, \vdash, \dashv, \Delta, I, F)$, where*

- *Q is a finite set of states,*
- *Σ is a finite alphabet such that $\{\vdash, \dashv\} \cap \Sigma = \varnothing$,*
- *$\Delta \subseteq Q \times (\Sigma \cup \{\vdash, \dashv\}) \times Q \times \{\mathsf{left}, \mathsf{right}\}$ is a transition relation,*
- *$I, F \subseteq Q$ are sets of initial and final states, respectively.*

Runs and recognized language. Below we fix some vocabulary for two-way automata (or later, transducers). We assume that every input word can be written as $w = \vdash a_1 \cdots a_n \dashv$ with two special delimiting symbols $\vdash$ and $\dashv$ that do not belong to Σ. By convention the automaton cannot move to the left when reading $\vdash$; on the other hand it can move to the right when reading $\dashv$ but as we will see, necessarily as the last transition of some successful run. A *configuration* of $\mathcal{A}$ is of the form $u\,q\,v$, where $q \in Q$ and

$w = uv \in \{\vdash\} \cdot \Sigma^* \cdot \{\dashv\}$ is the input word. A configuration $u\,q\,v$ represents the situation where the current state of $\mathcal{A}$ is q and its head reads the first symbol of v (on input w). If $(q, a, q', \mathsf{right}) \in \Delta$, then there is a transition from any configuration of the form $u\,q\,av$ to the configuration $ua\,q'\,v$; we denote such a transition by $u\,q\,av \xrightarrow{a,\mathsf{right}} ua\,q'\,v$. Symmetrically, if $(q, a, q', \mathsf{left}) \in \Delta$, then there is a transition from any configuration of the form $ub\,q\,av$ to the configuration $u\,q'\,bav$, denoted as $ub\,q\,av \xrightarrow{a,\mathsf{left}} u\,q'\,bav$.

A *run* is a sequence of transitions such that each target configuration is the source configuration of the next transition. It is *successful* on an input word w if it starts in an initial configuration $q\,w$, with $q \in I$, and ends in a final configuration $w\,q'$, with $q' \in F$ — note that this latter configuration does not allow additional transitions.

A *reversal* is a pair of consecutive transitions going in opposite directions.

Definition 2.1.3. *The* language *recognized by $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of words $w \in \Sigma^*$ such that there exists a successful run of $\mathcal{A}$ on $\vdash w \dashv$.*

2.1.2 Equivalence properties

We will see that the class of languages that we defined can be characterized in various other ways.

Automata. It is natural to restrict the non-determinism or the two-wayness of *2NFA* and ask whether they define different classes; and if not how difficult are the translations from one class to another. The following definition describes natural restrictions:

Definition 2.1.4. *An automaton is said to be:*

- one-way *(*NFA*) if it has only* right *transition rules.*
- sweeping *if it can perform reversals only at the borders of the input word.*
- unambiguous *if for each input word, there is at most one accepting run.*
- deterministic *(*2DFA*, or* DFA *if it is also one-way) if the transition relation is a function.*

It is well-known that *NFA* and *DFA* define the same class of languages [51, 71]: the subset construction of Rabin and Scott transforms any *NFA* into an equivalent *DFA* at an exponential cost on the number of states [66]. It is also possible to turn any *2NFA* into an equivalent *NFA* with an exponential blow-up [66, 78], using for instance the notion of crossing sequence as in Shepherdson's proof [78]. All models thus define the same class of languages.

If the cost of most translations is known and tight, Sakoda and Sipser asked in 1978 the question of the cost of the transformation of a non-deterministic (one-way or two-way) automaton into a *2DFA* [74]. Despite the study of sweeping automata by Sipser [80], some recent results [64] and in particular a hierarchy of intermediate classes, this is still an open problem.

Logic. It is known since the early days of automata theory that word languages recognized by automata can be also described by logics. First, we need to define **M**onadic **S**econd **O**rder formulas on words over the alphabet Σ.

Definition 2.1.5. *The formulas of* M*onadic* S*econd-*O*rder logic over the alphabet* Σ *(hereafter denoted* $\mathrm{MSO}(\Sigma, <)$*) use the classical syntax of* $\mathrm{MSO}(<)$ *and a monadic predicate* P_a *for each* $a \in \Sigma$*. They are defined using the following grammar:*

$$\Phi ::= x \in X \mid P_a(x) \mid x < y \mid x = y \mid \exists x\ \Phi(x) \mid \exists X\ \Phi(X) \mid \Phi_1 \wedge \Phi_2 \mid \neg\Phi$$

We can use the notation $x \to y$ (y is the successor of x) for the formula $x < y \wedge \forall z\ x < z \Rightarrow y \leqslant z$.

A non-empty word $w = a_1 \cdots a_n$ of Σ^* defines a structure $(\{1, \dots n\}, <, \ell)$ where $<$ is the total order on the positions of w, and $\ell : \{1, \dots n\} \to \Sigma$ is the labelling by letters. The semantics of an $MSO(\Sigma, <)$ formula are given by $w, i \models P_a(x)$ iff the $w(i) = a$, and the usual interpretation of *MSO* operators.

The set of all words satisfying Φ is the language defined by Φ, denoted $L(\Phi)$. A language L is *MSO*-definable if there is a formula Φ of $MSO(\Sigma, <)$ such that $L = L(\Phi)$. The following theorem is due to Büchi, Elgot, and Trakhtenbrot. This theorem was originally stated for *NFA* but the inclusion of *2NFA* in *MSO* is easy to obtain using the tools developed in next section.

Theorem 2.1.6 ([18, 34, 83])**.** *A language* $L \subseteq \Sigma^*$ *is recognized by a* 2NFA *if and only if it is* MSO*-definable. The transformations in both directions are effective.*

We will provide in Section 2.3 a way to use $MSO(\Sigma, <)$ in order to define transductions.

A last well-known result of formal language theory is the equivalence between rational languages and languages defined by rational expressions –Kleene's theorem–. Even if there is a notion of "regular relations" for describing transductions [35] we will not present them and refer the reader to the literature [51, 70] or recent works on rational relations [22].

2.1.3 Representation of a run

As we focus in Chapter 3 on the question of the translation from two-way to one-way transducers, we will use several times the construction from *2NFA* to *NFA* and in particular, the crossing sequence construction of [78]. This is why we introduce in this section the definitions used to describe a run, and more precisely the global state of the transducer while reading the input.

Positions and locations. We follow the convenient presentation from [51], which appeals to a graphical representation of runs of a two-way transducer, where each configuration is seen as a point (location) in a two-dimensional space.

A *position* is an integer representing "cuts" between letters of a word $w = \vdash a_1 \cdots a_n \dashv$: 0 is before $\vdash$, 1 between $\vdash$ and a_1, etc... Note that we are not in the logical framework and thus that this notion of positions differs from positions as elements of a word structure. Each configuration of a run is associated with a position, but this position depends on the direction of the last transition. If the transducer arrives in the configuration $u\,q\,a_i v$ with a right transition, the associated position is i. If the same happens with a left transition, the associated position is $i + 1$. The main reason for this distinction if that left and right transitions between the positions i and $i + 1$ both read the letter a_{i+1}, that is the one between i and $i + 1$.

Definition 2.1.7. *A* location *of a run ρ is a pair (x, y) where x is a position and $y \geqslant 0$ is called the* level *of the location. The location (x, y) represents the $(y + 1)$-th configuration associated with the position x that is visited by ρ.*

As shown in Fig. 2.1, any two-way run can be represented as an annotated path between locations. Note that in a successful run ρ every right (resp. left) transition reaches a location with even (resp. odd) level. We can identify four types of transitions between locations, depending on the parities of the levels:

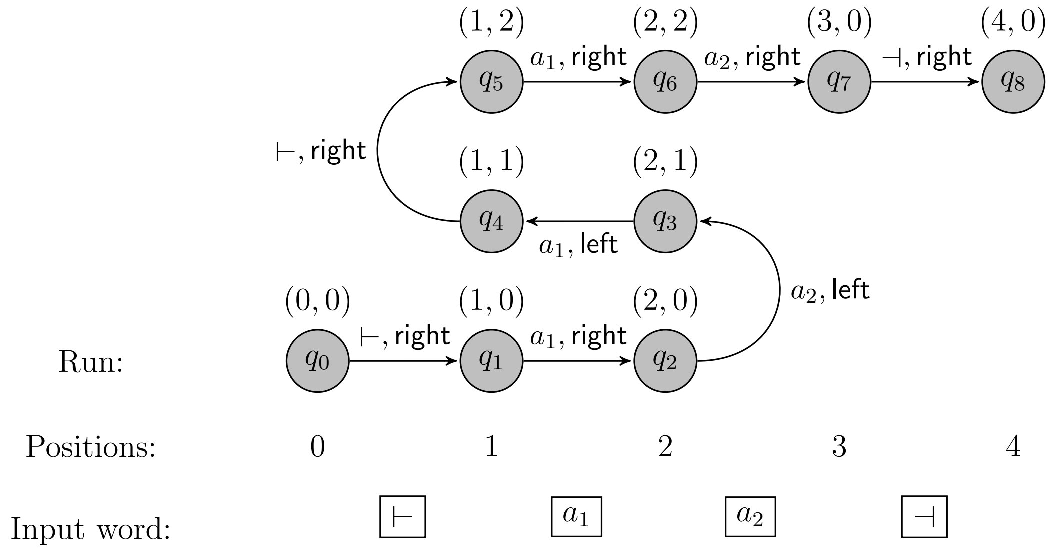


Figure 2.1: Graphical presentation of a run of an automaton.

$$(x, 2y) \xrightarrow{a_x,\,\mathsf{right}} (x{+}1, 2y')$$

$$\begin{tikzcd}(x, 2y{+}1) \\ (x, 2y) \arrow[u, bend right, "{a_x,\,\mathsf{left}}"']\end{tikzcd}$$

$$(x-1, 2y{+}1) \xleftarrow{a_{x-1},\,\mathsf{left}} (x, 2y'{+}1)$$

$$\begin{tikzcd}(x, 2y{+}2) \\ (x, 2y{+}1) \arrow[u, bend left, "{a_{x-1},\,\mathsf{right}}"]\end{tikzcd}$$

Hereafter, we will identify runs with the corresponding annotated paths between locations. It is also convenient to define a total order $\trianglelefteq$ on the locations of a run ρ by letting $\ell_1 \trianglelefteq \ell_2$ if ℓ_2 is reachable from ℓ_1 by following the path described by ρ — the order $\trianglelefteq$ on locations is called *run order*.

Given two locations $\ell_1 \trianglelefteq \ell_2$ of a run ρ, we write $\rho[\ell_1, \ell_2]$ for the factor of the run that starts in ℓ_1 and ends in ℓ_2. Note that the latter is also a run. Two runs ρ_1, ρ_2 can be concatenated, provided that ρ_1 ends in location (x, y), ρ_2 starts in location (x, y'), $y' = y \pmod 2$ and (x, y) and (x, y') are labelled by the same state. We denote by $\rho_1\rho_2$ the run resulting from concatenating ρ_1 with ρ_2. Clearly, we have $\rho[\ell_1, \ell_2]\ \rho[\ell_2, \ell_3] = \rho[\ell_1, \ell_3]$ for all locations $\ell_1 \trianglelefteq \ell_2 \trianglelefteq \ell_3$.

Crossing sequences. One of the main notion that we use through this thesis is that of crossing sequence. Let $w = \vdash a_1 \cdots a_n \dashv$ be an input word and let ρ be a run of a two-way automaton (or transducer) on w. Each location is associated to a state. Formally, we say that q is the *state at location* $\ell = (x, y)$ in ρ, and we denote this by writing $\rho(\ell) = q$, if the $(y+1)$-th configuration associated with position x can be written as $ua_i\, q\, a_{i+1}v$. In other words, the transducer is in the state $q = \rho(x, y)$ when visiting position x for the $(y + 1)$-th time.

Definition 2.1.8. *The* crossing sequence *at position x of ρ is the tuple $\rho|x = (q_0, \ldots, q_h)$, where the q_y's are all the states at locations of the form (x, y).*

Note that in a successful run ρ every crossing sequence has odd length: as one can see in Figure 2.2, the last transition on each position is left-to-right.

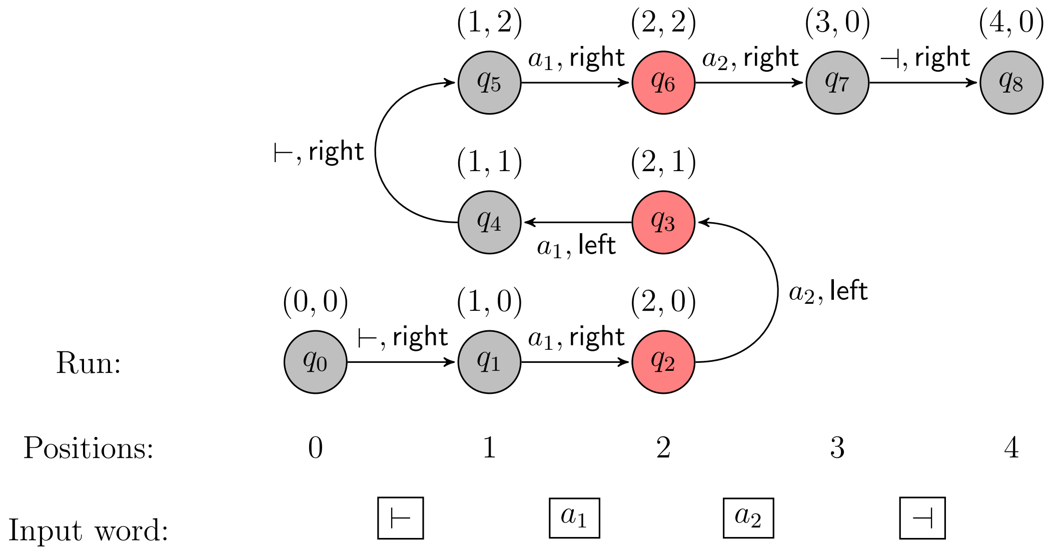


Figure 2.2: The crossing sequence at position 2 is (q_2, q_3, q_6)

The *crossing number* of a two-way automaton is the maximal length of a crossing sequence.

Normalization. A run of $\mathcal{A}$ is *normalized* if it never visits two locations with the same position, the same state, and both either at even or at odd level. It is easy to see that for any run, there is an equivalent normalized run. Indeed, if a successful run ρ is not normalized and visits two locations $\ell_1 = (x, y)$ and $\ell_2 = (x, y')$ with the same state $\rho(\ell_1) = \rho(\ell_2)$ and with

$y = y' \mod 2$, then we can delete the factor $\rho[\ell_1, \ell_2]$, thus obtaining a new run equivalent to ρ. By repeating this operation, we obtain a normalized run. This allows us to consider only successful, normalized runs.

We can notice that in every normalized successful run of an automaton $\mathcal{A}$, the crossing sequences have length at most $h_{\mathcal{A}} = 2|Q| - 1$, and thus that the crossing number of normalized automata is bounded. This implies that the number of crossing sequences is exponential in $|Q|$. The idea of the construction in [78] is to let a one-way automaton $\mathcal{A}'$ guess those crossing sequences, and check locally that the transitions between consecutive crossing sequences are compatible. In Figure 2.2 we highlighted the crossing sequence at position 2.

2.2 Finite state transducers

2.2.1 Definition

Transductions are relations over Σ^* and Γ^* (that is subsets of $\Sigma^* \times \Gamma^*$). A particular case of transductions occurs when, for any $u \in \Sigma^*$ there is at most one word $v \in \Gamma^*$ such that (u, v) belongs to the transduction. The transduction is then a *partial function.* As the important property here is being functional, and as they define total functions over their domain of definition, we will often refer to functional transductions simply as *functions. Two-way transducers* have been introduced in the early days of automata theory [63, 66, 76, 78] and they are our main model for describing transductions. Compared to automata, they add outputs to transitions:

Definition 2.2.1. *A two-way non-deterministic transducer (*2NFT*) is a two-way automaton with an additional output alphabet Γ and such that the transition relation Δ is a finite subset of $Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \Gamma^* \times Q \times \{\mathsf{left}, \mathsf{right}\}$.*

For a two-way transducer $\mathcal{T} = (Q, \Sigma, \vdash, \dashv, \Gamma, \Delta, I, F)$, we have a transition of the form $ub\,q\,av \xrightarrow{a,d|w} u'\,q'\,v'$, outputting w, whenever $(q, a, w, q', d) \in \Delta$ and either $u' = uba \;\wedge\; v' = v$ or $u' = u \;\wedge\; v' = bav$, depending on whether $d = \mathsf{right}$ or $d = \mathsf{left}$. The transducers have no transition reading ϵ (it is often called *real-time* in the literature). The *output* associated with a run $\rho = u_1\,q_1\,v_1 \xrightarrow{a_1,d_1|w_1} \ldots \xrightarrow{a_n,d_n|w_n} u_{n+1}\,q_{n+1}\,v_{n+1}$ of $\mathcal{T}$ is the word $\mathsf{out}(\rho) = w_1 \cdots w_n$.

The *domain* of $\mathcal{T}$, denoted $\mathsf{dom}(\mathcal{T})$, is the set of input words w such that $\vdash w \dashv$ has a successful run. A transducer $\mathcal{T}$ defines a relation $\mathscr{R}(\mathcal{T})$

consisting of all pairs (w, w') such that $w' = \mathsf{out}(\rho)$, for some successful run ρ on $\vdash w \dashv$.

For some transducers $\mathcal{T}, \mathcal{T}'$, we write $\mathcal{T}' \subseteq \mathcal{T}$ to mean that $\mathsf{dom}(\mathcal{T}') \subseteq \mathsf{dom}(\mathcal{T})$ and the transductions computed by $\mathcal{T}, \mathcal{T}'$ coincide on $\mathsf{dom}(\mathcal{T}')$. Given a transducer $\mathcal{T}$ and a regular language $D \subseteq \mathsf{dom}(\mathcal{T})$ the restriction of $\mathcal{T}$ to D, denoted $\mathcal{T}_{|D}$ is the transducer that simulates $\mathcal{T}$ but only accepts inputs in D. The transduction realized by $\mathcal{T}_{|D}$ is $\{(x, y) \in \mathscr{R}(\mathcal{T}) \mid x \in D\}$. We say that two transducers $\mathcal{T}, \mathcal{T}'$ are *equivalent* if they define the same transduction. One can notice that $\mathcal{T}' \subseteq \mathcal{T}$ and $\mathcal{T} \subseteq \mathcal{T}'$ imply that $\mathcal{T}$ and $\mathcal{T}'$ are equivalent.

Example 1. If $w = a_1 \cdots a_n$ is a word, then $\overline{w} = a_n a_{n-1} \cdots a_1$ is called mirror of w. Let f be the function on $\{a, b\}^*$ such that for all $w \in \{a, b\}^*$, we have $f(w) = w\overline{w}w$. In Figure 2.3, we describe a two-way transducer computing the function f (in this example and the followings we write c to represent a generic letter of the input alphabet Σ). It reads three times the input word, one for each state q_i: on q_1 and q_3 it reads the word from left to right and outputs w, and on q_2 it outputs $\overline{w}$ as it reads w from right to left.

Reversal and change of state are done when the transducer reads one of the endmarker symbols. This transducer is thus sweeping, as one can see on Figure 2.4.

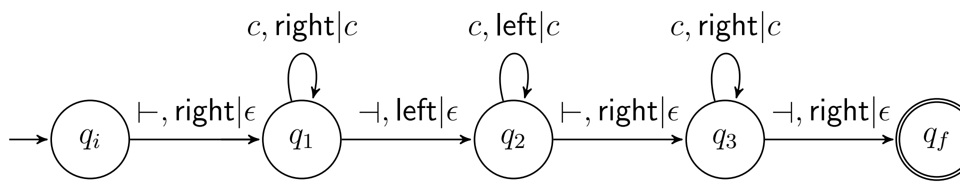

$c, \mathsf{right}|c$
$c, \mathsf{left}|c$
$c, \mathsf{right}|c$
$\vdash, \mathsf{right}|\epsilon$
$\dashv, \mathsf{left}|\epsilon$
$\vdash, \mathsf{right}|\epsilon$
$\dashv, \mathsf{right}|\epsilon$
q_i
q_1
q_2
q_3
q_f

Figure 2.3: A transducer computing the function $w \mapsto w\overline{w}w$

Transducers are extensions of automata: if we consider a transducer $\mathcal{T}$ and we remove the output component of the transition relation we obtain an automaton. It is called the *underlying automaton* of $\mathcal{T}$.

Definition 2.2.2. *A transducer $\mathcal{T}$ is one-way, sweeping, unambiguous if the underlying automaton of $\mathcal{T}$ is. It is deterministic*[1] *if the transition relation is a function from $Q \times (\Sigma \cup \{\vdash, \dashv\})$ to $\Gamma^* \times Q \times \{\mathsf{left}, \mathsf{right}\}$. We write:*

[1]Sometimes called *sequential* or *subsequential* in the literature.

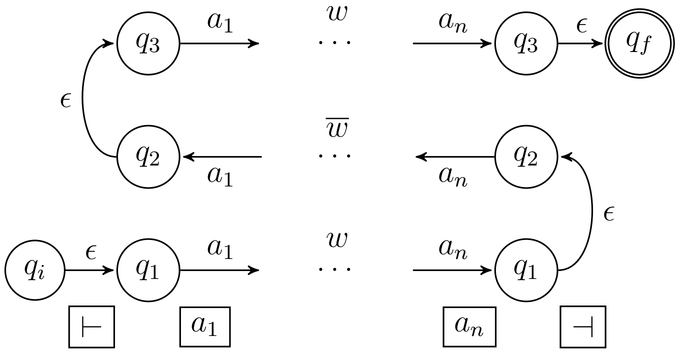


Figure 2.4: A run producing $a_1 \cdots a_n a_n \cdots a_1 a_1 \cdots a_n$ on the input $\vdash a_1 \cdots a_n \dashv$.

- 1DFT *for* **1***-way* ***D****eterministic* ***F****inite state* ***T****ransducers.*
- 2DFT *for* **2***-way* ***D****eterministic* ***F****inite state* ***T****ransducers.*
- 1NFT*,* 2NFT *for their non-deterministic counterparts.*

Moreover, we say that a transducer is functional[2] *if the defined transduction is a partial function. We write* 2fNFT *and* 1fNFT *for functional transducers, respectively two-way and one-way.*

One can notice that deterministic and unambiguous transducers are always functional.

2.2.2 Separating examples

There are two kind of questions to ask about two-way transductions: definability questions and semantic question. The first one considers a transduction (represented by a transducer, but the object of the question is the transduction) and asks whether it can be computed by a transducer satisfying some properties (for instance, deterministic). The latter considers the properties of a transduction computed by a specific transducer (for instance, its functionality). There are two main aspects of transducers that can be considered by those questions: one-wayness (or its generalization, sweepingness) and determinism (or its intermediate problems, ambiguousity and functionality).

[2]Also called *one-valued* or *single-valued* in the literature.

One-way vs two-way. We first show that the definitions of sweeping and one-way transducers are meaningful in the sense that they describe different classes of relations than two-way transducers. This holds true even when we restrict ourselves to the subclass of functional transducers.

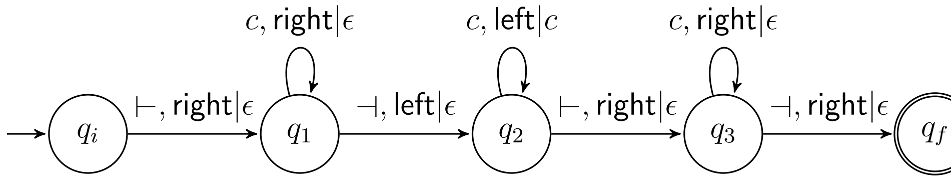
Example 2. Let g be the mirror transduction, that is the function on $\{a,b\}^*$ such that for any $w = a_1 \cdots a_n \in \{a,b\}^*$ we have $g(w) = \overline{w} = a_n \cdots a_1$. This function is trivially definable by a sweeping transducer (see Figures 2.5 and 2.6).

Unlike the sweeping transducer, a one-way transducer would have to store the entire input to be able to produce the correct letters after it has seen the first output a_n, that is at the end of the word (remember it cannot go back).

Claim. *The function g is definable by no* 1NFT*.*

Proof. Indeed, assume that there is a *1NFT* $\mathcal{T}$ computing g and let $w_i = a^i b^i$ be an input word for any $i \in \mathbb{N}$. Let $\rho_i = \rho_i^{(1)} \tau_i \rho_i^{(2)}$ be an accepting run of $\mathcal{T}$ on w_i such that τ_i is the first transition reading b. As the number of transitions is finite, there exist $i < j$ such that $\tau_i = \tau_j = \tau$. This implies that τ_i and τ_j arrive at the same configuration and that $\rho_i^{(1)} \tau \rho_j^{(2)}$ is a valid run of $\mathcal{T}$ on the input $a^i b^j$. Thus, the output of $\rho_i^{(1)}$ is a prefix of $b^i a^i$ and of $b^j a^i$, that is b^k for some $k \leqslant i$. As $\rho_i^{(1)} \tau \rho_j^{(2)}$ is an accepting run of $\mathcal{T}$ the output of $\tau \rho_j^{(2)}$ is $b^{j-k} a^i$. Since $j - k > 0$ this word is not a suffix of $b^j a^j$ which is in contradiction with the fact that $\rho_j = \rho_j^{(1)} \tau \rho_j^{(2)}$ is a run producing the output $b^j a^j$.

□



c, right|ϵ
c, left|c
c, right|ϵ
$\vdash$, right|ϵ
$\dashv$, left|ϵ
$\vdash$, right|ϵ
$\dashv$, right|ϵ
q_i
q_1
q_2
q_3
q_f

Figure 2.5: A transducer $\mathcal{G}$ computing the mirror transduction g

The mirror transduction is the natural example that comes to mind when it comes to exploiting the right-to-left passes. Note that it is an example

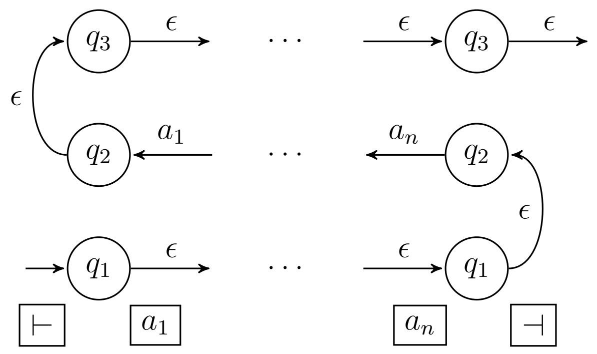


Figure 2.6: A run of $\mathcal{G}$ on $\vdash a_1 \cdots a_n \dashv$

of a *deterministic* sweeping transducer that is not definable by a one-way transducer, even a *non-deterministic* one. The next example illustrates the ability of two-way transducers to perform reversals at any location of the run and any number of times:

Example 3. Let g' be a total function on $\{a, b, \#\}^*$ such that $g'(w_1\#w_2 \cdots \#w_n) = \overline{w_1}\#\overline{w_2} \cdots \#\overline{w_n}$ when $w_i \in \{a, b, \#\}^*$.

The two-way transducer in Figure 2.7 computes the function g'. It performs a back-and-forth pass between two consecutive # symbols to produce the letters of $\overline{w_i}$ in the i-th right-to-left pass, as one can see in Figure 2.8. With a sweeping transducer, one would have to store the integer i (which is bounded by no constant) to know where to continue the run after a reversal. Thus this function can be described by no sweeping (and in particular, no one-way) transducer. This can be formally proved using the techniques of the previous example where we use identical pairs of crossing sequences instead of τ_i and τ_j to connect the different parts of the runs.

Note that in the two examples we reverse words over an alphabet of two letters (in Example 3 the reversed words do not contain the symbol #). If the alphabet had only one letter, then g and g' would simply be the identity function (because $\overline{a^n} = a^n$) which can be computed by a one-way transducer. As a matter of fact, when the output alphabet is unary, two-way and one-way transducers define the same class of transductions. The crucial point is that when using the crossing sequence construction for the underlying automaton we just need to count the number of letters produced on each transition of the crossing sequence. Indeed the one-way transducer can produce the same number of letters, as the order of how we produce them does not matter.

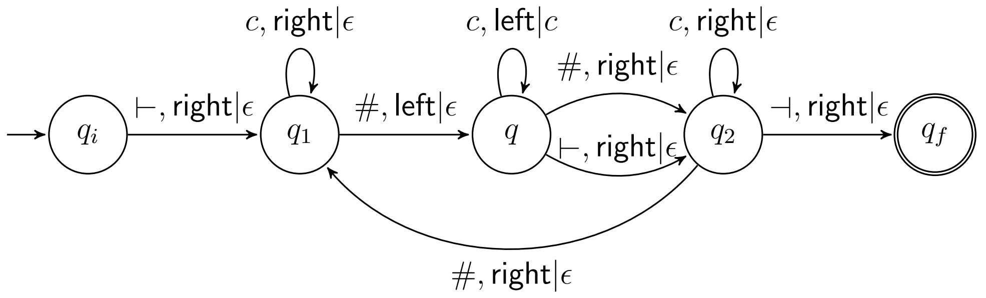


Figure 2.7: A transducer $\mathcal{G}'$ representing the function g'.

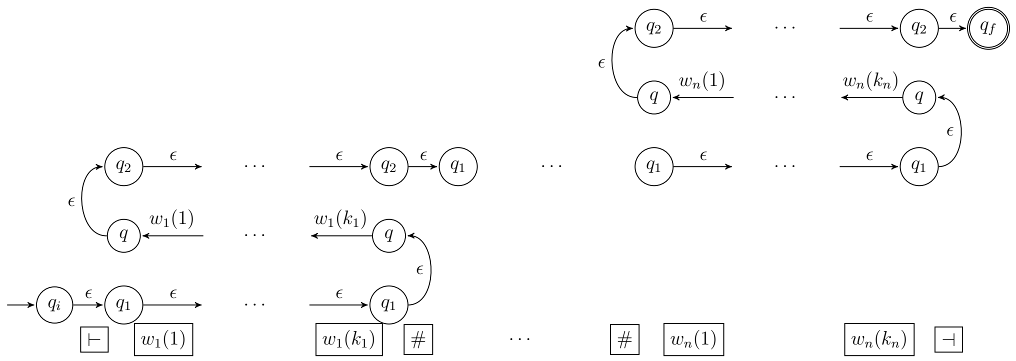


Figure 2.8: A run of $\mathcal{G}'$ on $\vdash w_1\# \cdots \#w_n \dashv$.

This raises the following questions, that we call *One-way definability* and *Sweeping definability*:

ONE-WAY DEFINABILITY	
Input:	A *2NFT* $\mathcal{T}$.
Question:	Is $\mathcal{T}$ equivalent to some *1NFT*?

SWEEPING DEFINABILITY	
Input:	A *2NFT* $\mathcal{T}$.
Question:	Is $\mathcal{T}$ equivalent to some sweeping transducer?

One of the starting points of this thesis is a recent proof of the decidability of the One-way definability problem in the functional case [39], which unfortunately only gives a non-elementary complexity. We will see in Chapter 3 that the general problem (without the restriction to functional transducers) is undecidable. The main result in Chapter 3 is a doubly exponential space decision procedure for the One-way definability problem in the functional case. When the given transducer is sweeping we gain one exponential and the bound on the size of an equivalent one-way transducer is tight.

***1NFT* vs *1DFT*.** As non-deterministic transducers define relations and deterministic ones define functions, transducers are not determinizable in the general case. This remains to some extent true even when the transduction is a function. Indeed, the classical subset construction (see for instance [71]) is not applicable to transducers: one would have to remember the outputs from the whole run already done as part of the states, which is not possible with a finite state machine.

The following example exploits this idea to show that there is sometimes no equivalent *1DFT* to a given *1fNFT*:

Example 4. Let h be the function over Σ^* such that $h(\epsilon) = \epsilon$ and for any $w = w'a$ where $a \in \Sigma$, we have $h(w) = a^{|w|}$. The function h is clearly definable by a one-way non-deterministic transducer described in Figure 2.9. It simply has to guess the last letter, output it on each following transition, and check its guess at the end of the word.

Such a transduction cannot be realized by a one-way deterministic transducer: the size of the input is unbounded, and cannot be guessed or stored in the states of the *1DFT* before reading the last letter a.

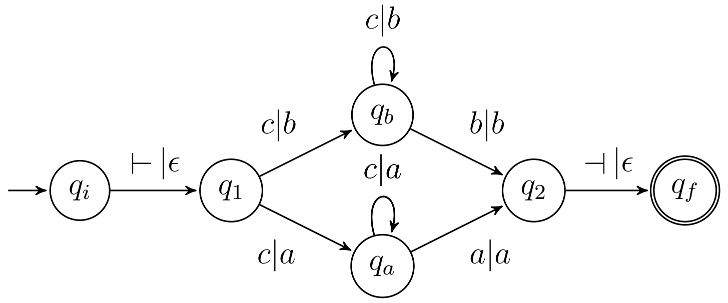


Figure 2.9: A one-way non deterministic transducer $\mathcal{H}$ computing h.

This example leads to the following question, which was historically called subsequentiality [20]:

DETERMINIZATION OF ONE-WAY TRANSDUCERS
Input: A *1fNFT* $\mathcal{T}$.
Question: Is $\mathcal{T}$ equivalent to some *1DFT*?

Choffrut proved in 1977 that the determinization of *1fNFT* is equivalent to a decidable property, called the twinning property [20]:

Definition 2.2.3. *Two states q_1 and q_2 of a* 1fNFT $\mathcal{T}$ *are* twinned *if for all pairs of runs of the form:*

- $i \xrightarrow{u|u_1} q_1 \xrightarrow{v|v_1} q_1$
- $i' \xrightarrow{u|u_2} q_2 \xrightarrow{v|v_2} q_2$

where i, i' are initial states and u, v, u_1, v_1, u_2, v_2 are words such that $|u_1| \geqslant |u_2|$, we have one of the following properties:

- $v_1 = v_2 = \epsilon$, *or*
- $u_1 = u_2 w$ *and* $wv_2 = v_1 w$ *for some* $w \in \Sigma^*$.

We say that $\mathcal{T}$ has the twinning property if any pair of reachable and co-reachable states is twinned (reachable states can be reached by the initial state by some run, co-reachable states q are such that a final state is reachable by some run starting in q).

Moreover, the algorithm for deciding the twinning property is polynomial [12, 20] and when possible we can obtain an equivalent deterministic transducer of exponential size [85].

Non-determinism for sweeping transducers Sweeping transducers behave like one-way transducers toward determinization: there exist functional sweeping transducers that are not determinizable (by a sweeping transducer). The following example is even stronger: it shows that there exist *1fNFT*'s that can be simulated by no deterministic sweeping transducer. As deterministic sweeping can compute the mirror transduction and *1fNFT*'s can not, we can conclude that these two subclasses of non-deterministic sweeping transducers are incomparable. As we provided a *mirror-by-block* transduction to separate sweeping and two-way, the following example is a "block" version of the previous Example 4.

Example 5. Let h' be the total function on $\{a,b,\#\}^*$ such that $h'(w_1\#w_2\#\cdots\#w_n) = h(w_1)\#h(w_2)\#\cdots\#h(w_n)$ where $w_i \in \{a,b\}^*$ for any i. The function h' is easily computed by a *1fNFT*: we simply have to modify $\mathcal{H}$ so that it comes back to q_1 after each time it has read the symbol # (see Figure 2.10). However h' can be computed by no deterministic sweeping transducer: indeed such a transducer would have to go to the end of the word to read the last letter of w_i in order to produce the right letter, and then come back to w_i in order to produce the right number of letters. But the transducer would need to store the number i in memory, which is unbounded as the number of blocks n is not fixed. As we have already seen, this is impossible with finite memory.

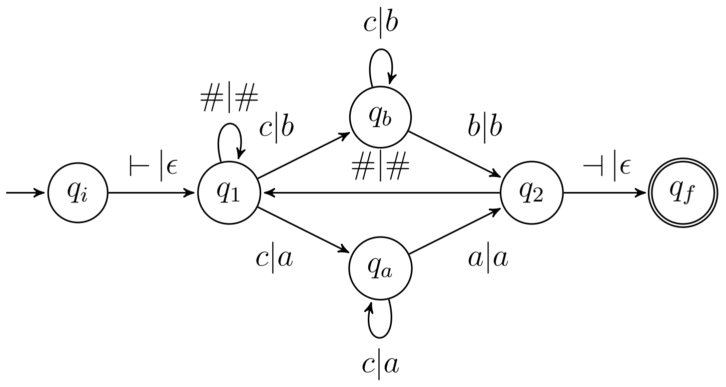


Figure 2.10: A one-way non-deterministic transducer $\mathcal{H}'$ computing h'.

Thus we have the following open problem:

Open problem: sweeping-determinization	
Input:	A functional sweeping transducer $\mathcal{T}$.
Question:	Is $\mathcal{T}$ equivalent to some deterministic sweeping transducer?

This problem is open as the generalization of the arguments of the one-way case is non-trivial: the pumping involved in the twinning property yield to a more complicated output in the sweeping case. However, one intermediate construction that will be very useful for the constructions of the next chapters is to turn a functional transducer into an unambiguous one.

Proposition 2.2.4. *Let $\mathcal{T}$ be a functional sweeping transducer. It is equivalent to some unambiguous sweeping transducer of exponential size in $\mathcal{T}$.*

Proof. As usual, this case can be dealt with by a form of subset construction: we consider crossing sequences of subsets of states. The key observation is that all successful runs of $\mathcal{T}$ can be followed simultaneously because they perform the same reversals at the same positions, namely, at the extremities of the input word. In other words, one can determinize the underlying sweeping automaton in simple exponential time.

□

$$\begin{tikzcd}
 & \mathit{1fNFT} \arrow[dr, "\text{Example 2}"] & & \\
\mathit{1DFT} \arrow[ur, "\text{Example 4}"] \arrow[dr, "\text{Example 2}"'] & & \text{functional Sweeping} \arrow[r, "\text{Example 3}"] & \mathit{2DFT} \\
 & \text{Deterministic Sweeping} \arrow[ur, "\text{Example 5}"'] & &
\end{tikzcd}$$

Figure 2.11: A synthesis of separation examples.

We summarize in Figure 2.11 the separation results given by the examples of this section. Remark that arrows are not inclusions yet: we have not shown that functional sweeping transducers are included in *2DFT*'s. Moreover we have seen that non-deterministic models are stronger in the one-way and the sweeping case, but we have not mentioned the two-way case yet: indeed these two questions are the subject of the next Section and we will see that such separations do not hold in the two-way case.

2.2.3 Functionality and determinization

Example. We come back now to Example 4: we showed that function h can be realized by no deterministic one-way transducer, but it could be done by a deterministic *two-way* transducer. Indeed, the sweeping transducer of Figure 2.12 computes h. As one can see in Figure 2.13, the transducer $\mathcal{H}$ goes to the last position of the input word, stores the letter in memory, and keeps producing it while going backward. Then it performs one last pass without producing anything to reach the final configuration. This suggests that determinization of functional transducers may be possible in the two-way case.

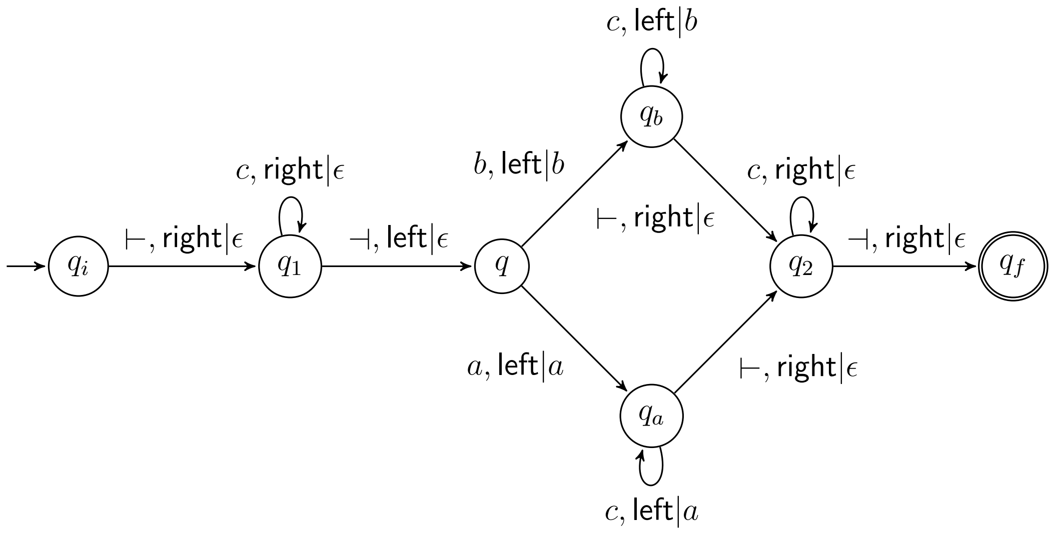

c, left$|b$
q_b
b, left$|b$
c, right$|\epsilon$
$\vdash$, right$|\epsilon$
c, right$|\epsilon$
$\vdash$, right$|\epsilon$
$\dashv$, left$|\epsilon$
$\dashv$, right$|\epsilon$
q_i
q_1
q
q_2
q_f
a, left$|a$
$\vdash$, right$|\epsilon$
q_a
c, left$|a$

Figure 2.12: A two-way deterministic transducer $\mathcal{H}$ computing function h.

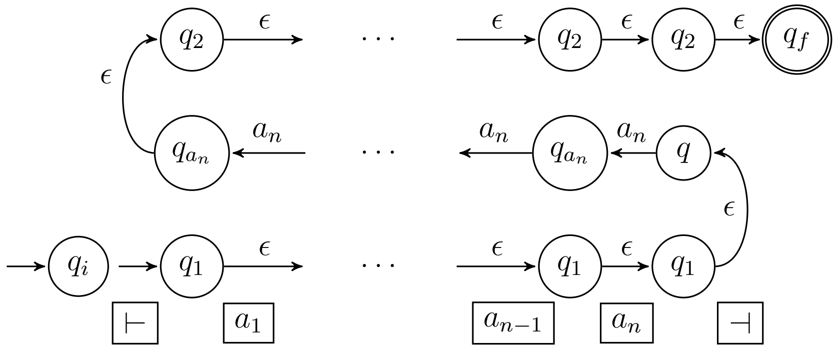


Figure 2.13: A run of $\mathcal{H}$ on the input $\vdash a_1 \cdots a_n \dashv$.

Determinization. We recall that every deterministic transducer is functional, while the opposite implication fails in general. The following results state that the idea behind Example 4 can be generalized to any functional two-way transducer. Even more, we can co-determinize it at the same time.

Theorem 2.2.5 ([28, 31])**.** *Given a* 2NFT $\mathcal{T}$ *with* n *states, one can effectively construct a deterministic and co-deterministic two-way transducer* $\mathcal{T}'$ *of size exponential in* n *such that* $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ *and* $\mathcal{R}(\mathcal{T}') \subseteq \mathcal{R}(\mathcal{T})$*.*

This theorem is what is called a *uniformization* theorem: we turn a relation into a function by choosing images of elements of the domain. When we apply this theorem to a functional transducer, we obtain an equivalent deterministic (and co-deterministic) transducer:

Corollary 2.2.6. *Every* 2fNFT $\mathcal{T}$ *is equivalent to some (co-deterministic)* 2DFT *of size exponential in* $|\mathcal{T}|$*.*

Functionality. The previous result motivates us to consider functional transducers, and in particular, the problem of functionality for two-way transducers. We show here that this problem is decidable and in PSpace. The proof is similar to the decidability proof for the equivalence problem of two-way deterministic transducers [48], as it reduces the functionality problem to the reachability problem of a 1-counter automaton of exponential size. A matching PSpace lower bound follows by a reduction of the emptiness problem for the intersection of finite-state automata [57].

Before presenting this result we need to remark that functional transducers can be normalized just like automata. It suffices to notice that when we have two locations ℓ_1, ℓ_2 with the same state and the same parity, the output produced between those locations is empty. Indeed one can repeat the run between those locations, and obtain new accepting runs on the same input, with repetitions of $\mathsf{out}(\rho[\ell_1, \ell_2])$ in the output. If $\mathsf{out}(\rho[\ell_1, \ell_2])$ is non-empty, then we produce different outputs on the same input which is in contradiction with functionality. The fact that such outputs are empty allows us to use the normalization procedure as in the automata case.

Proposition 2.2.7. *In every normalized successful run of a functional transducer, the crossing sequences have length at most* $2|Q| - 1$*.*

This allows us to assume that transducers are normalized in the following proposition:

Proposition 2.2.8 ([8])**.** *Functionality of two-way transducers can be decided in polynomial space. This problem is* PSpace*-hard already for sweeping transducers.*

Proof. Showing that the problem is PSpace-hard for sweeping transducers is easy: we do a reduction from the emptiness problem of the intersection of *NFA*'s.

Given n NFAs $\mathcal{A}_1, \dots, \mathcal{A}_n$ we build a sweeping transducer $\mathcal{S}$ which simulates the automaton $\mathcal{A}_i$ on its i-th pass. At the end of the i-th pass, if the simulation of $\mathcal{A}_i$ is in an accepting state we non-deterministically choose to produce either ϵ or i and then continue the computation (that is verifying if the input is in $L(\mathcal{A}_{i+1})$). If $\mathcal{S}$ ends the simulation in a non-accepting state $\mathcal{S}$ rejects the input. That way, a word u is in the domain of the transduction defined by $\mathcal{S}$ if and only if $u \in \mathcal{A}_i$ for any i. Moreover, if the domain of the transduction is not empty $\mathcal{S}$ is not functional as there are many possible outputs. This yields exactly to the desired property of $\mathcal{S}$: it is functional if and only if $\bigcap_{i \leqslant n} L(\mathcal{A}_i) = \varnothing$.

To show that the functionality problem is in PSpace for two-way transducers we reduce it to the emptiness problem for 1-counter machines. These are one-way, non-deterministic automata with a counter that can be incremented, decremented or tested for zero. The machine accepts if it reaches a final state. This problem belongs to Nlogspace and we will obtain an exponential-sized 1-counter machine, that can be simulated on-the-fly. Altogether this will give PSpace complexity. We recall that $N = 2|Q| - 1$ is the maximal crossing number of a normalized run in a functional transducer. We first show the following claim:

Claim. *If $\mathcal{S}$ is not functional, then either there exists* (1) *a successful run with crossing number at most* $2N$ *and non-empty output on some repetition in a crossing sequence, or* (2) *two normalized runs on the same input, each with crossing number at most* N*, and with different outputs.*

Proof. Let us consider two successful runs ρ_1 and ρ_2 producing two distinct outputs $u \neq v$ on the same input w. We show that if ρ_1 or ρ_2 is not normalized then (1) holds, and that otherwise (2) holds. Assume without loss of generality that ρ_1 is equivalent to no normalized run. That is, the crossing sequence contains two locations with the same state and both on even/odd level, such that the output on the factor run delimited by these locations is non-empty. We can first remove from this run all repetitions that produce empty outputs. Then, we can also remove the repetitions with non-empty outputs, obtaining other successful runs, until there remains only one non-empty repetition. The run produced is an instance of case (1).

On the other hand, if ρ_1 and ρ_2 are equivalent to some normalized runs ρ_1' and ρ_2', then ρ_1' and ρ_2' are an instance of (2).

□

To determine non-functionality, the 1-counter machine $\mathcal{M}$ will guess between the two cases of the lemma: in (1) it will guess a run of $\mathcal{S}$ of crossing number smaller than $2N$, and in (2) it will guess two runs of $\mathcal{S}$ on the same input of crossing number smaller than N. In the second case, we define u and v to be the two outputs of the runs, and $\mathcal{S}$ decides if it will check $|u| \neq |v|$ or the existence of a position i such that $u_i \neq v_i$.

In (1) and (2), $\mathcal{M}$ guesses two locations in the runs and marks locations on the crossing sequences that will help to identify certain factors of runs.

In (1) $\mathcal{M}$ guesses two locations in the run at the same crossing sequence, that have the same state, and the same movement of the input head, and $\mathcal{M}$ marks all locations between those two locations. It checks then that the output produced by the factor containing marked locations is non-empty.

In (2), if the transducer has chosen to verify that $|u| < |v|$ (the case $|v| < |u|$ is symmetric), it guesses one location in the second run, and marks the locations before it in the corresponding crossing sequences. The counter is incremented by the number of letters produced in the first run, and decremented by those produced by marked locations of the second. The guessed location represents the moment in the second run where it has produced an output longer than $|u|$.

If it has chosen to check the existence of a position such that $u_i \neq v_i$, it guesses one location in each run, and marks the locations before those in the corresponding crossing sequences. The counter is incremented (resp. decremented) by the number of letters produced by marked locations of the first (resp. second) run. The machine also checks that the letters produced at the guessed locations are different, and the counter's value is 0 at the end of the run. This guarantees that the letters were produced at the same place. $\square$

We can now give in Figure 2.14 all the relations between finite state transducers classes and the corresponding decision problems.

$$\begin{tikzcd}
\boxed{\mathit{1DFT}} \arrow[r, phantom, "\subsetneq"] & \boxed{\mathit{1fNFT}} \arrow[l, bend left, "\mathit{PTIME}"] \arrow[r, phantom, "\subsetneq"] & \boxed{\text{functional Sweeping}} \arrow[l, bend left, "?"] \arrow[r, phantom, "\subsetneq"] & \boxed{\mathit{2DFT} = \mathit{2fNFT}} \arrow[l, bend left, "?"] \arrow[ll, bend left=50, "?"] \arrow[r, phantom, "\subsetneq"] & \boxed{\mathit{2NFT}} \arrow[l, bend left, "\mathit{PSPACE}"]
\end{tikzcd}$$

Figure 2.14: All finite state transducers classes and their relations.

The decision problem of one-way definability, that is the arrows arriving to *1fNFT*'s, is dealt with in Chapter 3, and the decidability of sweeping-definability is obtained in Chapter 4.

2.3 Logic and transductions

2.3.1 MSO transductions

We present here a model for transductions based on *MSO* logic, that was originally developpped by Courcelle [25, 26] in the more general setting of graph transformations. Words are special instances of labelled graphs. One of our goals in this section will be to characterize in this framework the questions of one-way definability and sweepingness.

Definition 2.3.1. *An* MSO *transduction (*MSOT *for short) from* $\vdash \Sigma^* \dashv$ *to* Γ^* *is given by a finite set of copies* C *and the following* MSO *formulas over* $(\Sigma, <)$*:*

- *A domain formula* Φ_{dom}
- *Node formulas* $\Phi_{\gamma,c}(x)$ *for* $\gamma \in \Gamma$ *and* $c \in C$
- *Edge formulas* $\Phi_{c_1,c_2}(x,y)$ *for* $c_1, c_2 \in C$

For any word w *satisfying* Φ_{dom} *, the transduction defines an output graph as follows: for every position* x *of* w*, and every* $c \in C$ *there is a node* $x^{(c)}$ *if and only if* $\bigvee_{\gamma\in\Gamma} \Phi_{\gamma,c}(x)$ *is true over* w*. In addition,* $x^{(c)}$ *is labelled by the set of* γ *such that* $\Phi_{\gamma,c}(x)$ *is true over* x*. Moreover* $y^{(c_2)}$ *is the successor of* $x^{(c_1)}$ *if and only if* $\Phi_{c_1,c_2}(x,y)$ *is true over* w*. The formulas can ensure that the output graph represents a word* w' *(that is, the edge formulas induce a linear order, and each node is labelled by a single letter). This word* w' *is the image of the transduction on the word* $w \in \Sigma^*$*. Moreover, we defined* MSOT *on words of* $\vdash \Sigma^* \dashv$ *in order to be able to define an output on the empty word.*

Example 6. We exhibit an *MSO* transduction computing our running example $f(w) = w\overline{w}w$. We have $C = \{1,2,3\}$ and $\Phi_{\mathsf{dom}} = \top$. The transduction is given by the following formulas (recall that $\rightarrow$ represents the successor relation):

- $\Phi_{\gamma,c}(x) := P_\gamma(x)$ for any $c \in C$, $\gamma \in \Gamma$: the copies of the nodes are labeled by the input letter.

- $\Phi_{c,c}(x, y) := x \to y$ when $c = 1$ or $c = 3$: on the first and last copies the transduction computes w.
- $\Phi_{2,2}(x, y) := y \to x$: on the second copies the transduction computes $\overline{w}$.
- $\Phi_{1,2}(x, y) := (x = y) \wedge \forall z\ z \leqslant x$: we connect the first and second copies at the end of the input word.
- $\Phi_{2,3}(x, y) := (x = y) \wedge \forall z\ x \leqslant z$: we connect the second and third copies at the beginning of the input word.

One can see in Figure 2.15 the output graph of the transduction on $w = a_1 \cdots a_n$.

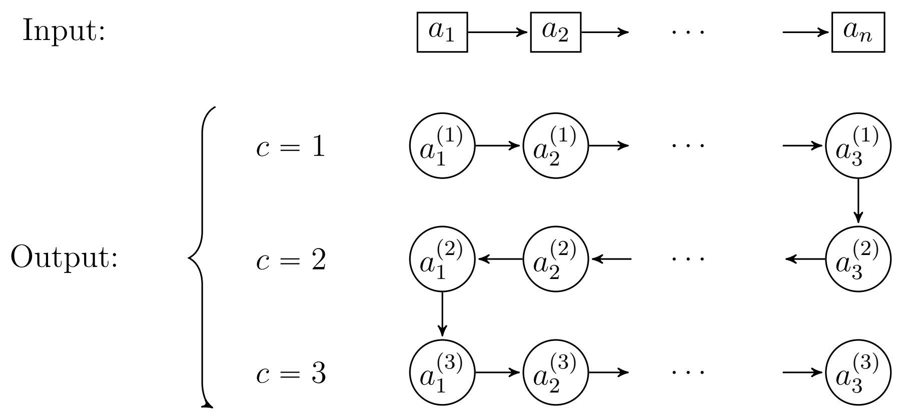


Figure 2.15: The output graph of the *MSO* transduction on $a_1 \cdots a_n$.

A non-deterministic variant of *MSOT* can be obtained by allowing the transducer to guess a finite number of sets of nodes, on the form of free set variables. The transduction produces an output for each valuation satisfying Φ_{dom}. By choosing canonically one particular set of variables using an *MSO* formula, and bounding the variables with an existential quantifier we can then perform a uniformization of the transducer and provide a function contained in the transduction (as we have done in Theorem 2.2.5 for *2NFT*).

Proposition 2.3.2. *For any* NMSOT $\mathcal{T}$ *there exists an* MSOT $\mathcal{T}'$ *such that* $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ *and* $\mathcal{R}(\mathcal{T}') \subseteq \mathcal{R}(\mathcal{T})$.

Proof. Let $\mathcal{T}$ be a transduction definable in *NMSOT* with k free variables $\overline{X} = X_1, \ldots, X_k$. We begin by choosing some total order $\leqslant$ on k-tuples of finite subsets of $\mathbb{N}$. We use the standard word-based representation of tuples: each tuple $\overline{X}$ is identified with a word of length $\max_{i \leqslant k, x \in X_i}(x)$ over the alphabet $\Sigma' = \{0,1\}^k$. The letter $(b_1, \ldots, b_k)$ at position i is such that $b_j = 1$ iff $i \in X_j$, for every $j \leqslant k$. For instance $\overline{X} = (\{2\}, \{2,3\}, \{1\})$ is identified with the word $(0,0,1) \cdot (1,1,0) \cdot (0,1,0)$. Our alphabet Σ' is finite and its size depends only on the transducer $\mathcal{T}$; we can thus fix an order on the letters and express it in $MSO(\Sigma')$. We also let $\leqslant$ be the induced lexicographic order on words over the alphabet Σ', and observe that this order can be expressed in $MSO(\Sigma, <)$.

Now, we replace each formula $\Phi(\overline{X}, \overline{y})$ of $\mathcal{T}$ (e.g. $\Phi_{c_1,c_2}(\overline{X}, y_1, y_2)$) by the formula $\Phi_{\min}(\overline{y}) = \exists \overline{X}\ \Phi(\overline{X}, \overline{y}) \wedge \forall\ \overline{X}'\ \Phi_{\mathsf{dom}}(\overline{X}') \Rightarrow \overline{X} \leqslant \overline{X}'$. Note that the latter formula is equivalent to the former when $\overline{X}$ is interpreted as the least tuple among those that satisfy Φ_{dom}. In this way we obtain an *MSO*-transducer with the same domain as $\mathcal{T}$, which chooses canonically a valuation of $\overline{X}$ to associate at most one output with each input. $\square$

When we apply the last theorem to a function of *NMSOT* we obtain the following corollary, similar to Corollary 2.2.6:

Corollary 2.3.3. *If a function is* NMSOT-*definable, then it is* MSOT-*definable.*

This result can also be derived from the fact that non-deterministic *MSO* transductions are relabellings of *MSO* transductions [36].

2.3.2 Sweeping vs MSO

Definition 2.3.4. *Let $\mathcal{T}$ be an MSO transduction with m copies. We say that $\mathcal{T}$ is*

- order-preserving *if each formula $\Phi_{c_1,c_2}(x,y)$ entails $x \leqslant y$;*
- order-inversing *if each formula $\Phi_{c_1,c_2}(x,y)$ entails $x \geqslant y$;*
- k-phase *if there is a partition $C_0, C_1, \ldots, C_{k-1}$ of the copy set C such that:*
    1. *for all i even (resp. odd) and all $c_1, c_2 \in C_i$, the formula $\Phi_{c_1,c_2}(x,y)$ entails $x \leqslant y$ (resp. $y \leqslant x$),*

2. *for all $i < j$ and $c_1 \in C_i$, $c_2 \in C_j$, the formula $\Phi_{c_1,c_2}(x,y)$ entails $x \leqslant y$.*

We know from [15, 37] that order-preserving MSO transductions capture precisely the functional *1NFT*'s. Symmetrically, order-inversing MSO transductions capture the transductions definable by one-way transducers that read the input from right to left. So it is not surprising that k-phase MSO transductions correspond to k-pass sweeping transducers, actually the proof that we give is similar to the one in [37] for order-preserving transductions:

Theorem 2.3.5 ([9])**.** *k-phase MSO transductions have the same expressive power as functional, k-pass sweeping transducers.*

Proof. We first show how to translate a k-pass sweeping, functional transducer $\mathcal{T}$ into an equivalent k-phase *MSO* transduction. The technique is a variant of the classical translation from two-way transducers to MSO transductions (see for instance [37]). First, using the classical correspondence between automata and MSO of Theorem 2.1.6 we can build an *MSO* sentence Φ_{dom} that tells whether a word u belongs to the domain of the transduction. Recall also from Proposition 2.2.4 that we can assume that $\mathcal{T}$ is unambiguous.

Recall that $\mathcal{T}$ performs at most k passes, and let $c_{\mathcal{T}}$ be the maximal number of characters output by a single transition of $\mathcal{T}$. To define the output of the *MSO* transduction, we will take $k \cdot c_{\mathcal{T}}$ copies of the input. Each copy of the input is thus indexed by a pair (h, i), where $0 \leqslant h < k$ and $1 \leqslant i \leqslant c_{\mathcal{T}}$. Intuitively, for a given position x, its copy indexed by (h, i) represents the i-th letter of the output produced by the transition leaving location (x, h). For each index $0 \leqslant h < k$ and each transition rule τ of $\mathcal{T}$, we can build an *MSO* formula $\Phi_{h,\tau}(x)$ such that, for all $u \in \mathsf{dom}(\mathcal{T})$, $u \models \Phi_{h,\tau}(x)$ iff the transition rule τ is applied at the location $\ell = (x, h)$ of the unique successful run of $\mathcal{T}$ induced by u. We complete the definition of the *MSO* transduction equivalent to $\mathcal{T}$ as follows:

- For the formulas defining the output elements and their labels, we let $\Phi_{\gamma,(h,i)}(x)$ be the disjunction of the formulas $\Phi_{h,\tau}(x)$, for all $\tau = (p, \gamma, v, q)$ such that $|v| \geqslant i$ and $v(i) = \gamma$. Note that, because only one transition rule can be used at each location of the successful run, for each $1 \leqslant x \leqslant |u|$ and each $0 \leqslant h < k$, there can exist at most one letter γ such that $u \models \Phi_{\gamma,(h,i)}(x)$.

- For the formulas representing the edges of the output elements, we define $\Phi_{(h,i),(h',i')}(x,y)$ by a case distinction:

$$\begin{cases} x = y & \text{if } h = h' \text{ and } i' = i+1 \\ x \to y \ \wedge \ \big(\bigwedge_{\gamma\in\Gamma} \neg\Phi_{\gamma,(h,i+1)}(x)\big) & \text{if } h = h' \text{ is even and } i' = 1 \\ y \to x \ \wedge \ \big(\bigwedge_{\gamma\in\Gamma} \neg\Phi_{\gamma,(h,i'+1)}(y)\big) & \text{if } h = h' \text{ is odd and } i = 1 \\ x = y \ \wedge \ \forall z \ z \leqslant x \ \wedge \ \big(\bigwedge_{\gamma\in\Gamma} \neg\Phi_{\gamma,(h,i+1)}(x)\big) & \text{if } h \text{ is even, } h' = h+1, \text{ and } i' = 1 \\ x = y \ \wedge \ \forall z \ x \leqslant z \ \wedge \ \big(\bigwedge_{\gamma\in\Gamma} \neg\Phi_{\gamma,(h,i+1)}(x)\big) & \text{if } h \text{ is odd, } h' = h+1, \text{ and } i' = 1. \end{cases}$$

  It is easy to see that the formulas $\Phi_{(h,i),(h',i')}(x,y)$ define a total order on the output structure.

It remains to show that the above MSO transduction is k-phase. For this, we order lexicographically the index set $\{(h,i) \ : \ 0 \leqslant h < k, \ 1 \leqslant i \leqslant c_S\}$, and partition it into k subsets $C_0 < C_1 < \ldots, C_{k-1}$, where $C_h = \{(h,i) \ : \ 1 \leqslant i \leqslant c_S\}$ for all $h = 0,1,\ldots,k-1$. We then observe that the defined partition satisfies Definition 2.3.4: indeed, for all pairs $(h,i),(h,j) \in C_h$, $\Phi_{(h,i)(h,j)}(x,y)$ entails either $x \leqslant y$ or $x \geqslant y$, depending on whether h is even or odd.

We now prove the converse translation, from a k-phase MSO transduction to an equivalent k-pass sweeping transducer. Let Φ_{dom}, $\Phi_{\gamma,i}(x)$, and $\Phi_{i,j}(x,y)$ be the formulas defining an arbitrary k-phase MSO transduction, where i,j range over some finite set C. Further let $C_0 < C_1 < \ldots, < C_{k-1}$ be a partition of C satisfying the third item of Definition 2.3.4. Note that, for a fixed $0 \leqslant h < k$, the family of formulas $\Phi_{\gamma,i}(x)$ and $\Phi_{i,j}(x,y)$ where i,j range over C_h define an MSO transduction that is either order-preserving or order-inversing, depending on whether h is even or odd. For the sake of brevity, we denote by $\mathcal{T}$ the original k-pass MSO transduction, and by $\mathcal{T}_0, \mathcal{T}_1, \ldots, \mathcal{T}_{k-1}$ the corresponding order-preserving/order-inversing MSO transductions.

If each $\mathcal{T}_h$ maps an input word u to an output v_h, then we know that $\mathcal{T}$ maps u to the juxtaposition $v_0 \cdot v_1 \cdot \ldots \cdot v_{k-1}$. Moreover, we know from [37] that we can translate the order-preserving transductions $\mathcal{T}_0, \mathcal{T}_2, \ldots, \mathcal{T}_k$ to equivalent one-way transducers $\mathcal{T}'_0, \mathcal{T}'_2, \ldots, \mathcal{T}'_k$. Similarly, we can translate the order-inversing transductions $\mathcal{T}_1, \mathcal{T}_3, \ldots, \mathcal{T}_{k-1}$ to equivalent transducers performing one single pass from right to left. $\mathcal{T}'_1, \mathcal{T}'_3, \ldots, \mathcal{T}'_{k-1}$. Thus, we can obtain the desired k-pass sweeping transducer $\mathcal{T}'$ by simply concatenating the transducers $\mathcal{T}'_0, \mathcal{T}'_1, \mathcal{T}'_2, \mathcal{T}'_3, \ldots, \mathcal{T}'_n$. □

2.3.3 FO-definability

It is well-known [60, 77] that languages defined by first-order formulas on words are equivalent to aperiodic languages, or equivalently, to languages recognized by counter-free automata. Such automata have the property that their transition monoid is aperiodic (for an overview of algrebraic theory of finite automata see [65]). We focus here on aperiodic transducers [27].

Definition 2.3.6. *We say that a transducer is* aperiodic *if its underlying automaton has an aperiodic transition monoid.*

The notion of transition monoid is standard only when the automaton is one-way. For a generalization to two-way machines one can use the definition of [19]. In that article, Carton and Dartois extend the classical result on automata to transducers:

Theorem 2.3.7 ([19])**.** *A* 2fNFT *is* FO*-definable iff it is aperiodic.*

However, unlike as in the automata case, this does not allow to decide *FO*-definability. Indeed, the fact that a transducer is not aperiodic does not mean that all equivalent transducers are not either. What we lack here is a notion of "canonical (or minimal) transducer".

When restricting to one-way transducers and order-preserving transductions, decidability of the *FO*-definability is recovered:

Theorem 2.3.8 ([38])**.** *A transduction is definable by an aperiodic* 1fNFT *iff it is definable by an order-preserving* FO*-transducer. Moreover, the latter property is decidable.*

One last open question is the *FO*-definability (not necessarily order-preserving) of a *1fNFT*. This question could be immediately solved using Theorem 2.3.8 if *FO*-definable *1fNFT* were equivalent to first-order transductions that are order-preserving. We conjecture that this is indeed the case and we note that the conjecture is implied by another conjecture, saying that any aperiodic *2NFT* that is one-way-definable can be transformed into an aperiodic *1NFT*. Indeed, assume that the latter property were true. Since by Theorem 2.3.7 any *FO*-definable *1fNFT* is equivalent to an aperiodic *2NFT* that is one-way definable, it suffices to turn this aperiodic *2NFT* into an aperiodic *1NFT* and the claim follows.

2.4 Streaming string transducers

Streaming transducers can implement the same functional transductions as *MSO* transducers [2, 4], but they do so using a single left-to-right pass and a fixed set of registers that can store words over an output alphabet.

Definition 2.4.1. *A* non-deterministic streaming transducer *is a tuple $\mathcal{T} = (Q, \Sigma, \Gamma, R, U, I, E, F)$, where Q is a finite set of states, Σ (resp. Γ) is a finite input (resp. output) alphabet, R is a finite set of registers disjoint from Γ, U is a finite set of* updates *for the registers, namely, functions from R to $(R \uplus \Gamma)^*$, I is a subset of Q representing the initial states, $E \subseteq Q \times \Sigma \times U \times Q$ is a finite set of transition rules, describing, for each state and input symbol, the possible updates and target states, and $F : Q \rightharpoonup (R \uplus \Gamma)^*$ is a partial output function.*

A well-behaved class of streaming transducers [2] is obtained by restricting the allowed types of updates and partial output functions to be copyless. A streaming transducer $\mathcal{T} = (Q, \Sigma, \Gamma, R, U, I, E, F)$ is *copyless* if (1) for every update $f \in U$, every register $z \in R$ appears at most once in $f(z_1) \cdots f(z_k)$, where $R = \{z_1, \ldots, z_k\}$, and (2) for every state $q \in Q$, every register $z \in R$ appears at most once in $F(q)$. Hereafter we assume that all transducers belong to the class of copyless non-deterministic streaming transducers, denoted *NSST*.

Semantic. In order to define the semantic of a streaming transducer $\mathcal{T} = (Q, \Sigma, \Gamma, R, U, I, E, F)$, we introduce *valuations* of registers in R. These are functions of the form $g : R \to \Gamma^\star$. Valuations can be homomorphically extended to words over $R \cup \Gamma$ and to updates, as follows. For every valuation $g : R \to \Gamma^\star$ and every word $w \in (R \cup \Gamma)^\star$, we let $g(w)$ be the word over Γ obtained from w by replacing every occurrence of a register z with its valuation $g(z)$. Similarly, for every valuation $g : R \to \Gamma^\star$ and every update $f : R \to (R \cup \Gamma)^\star$, we denote by $g \circ f$ the valuation that maps each register z to the word $g(f(z))$.

A *configuration* of $\mathcal{T}$ is a pair state-valuation (q, g). This configuration is said to be initial if $q \in I$ and $g(z) = \varepsilon$ for all registers $z \in R$. When reading a symbol a, the transducer can move from a configuration (q, g) to a configuration (q', g') if there exists a transition rule $(q, a, f, q') \in E$ such that $g' = g \circ f$. We denote this by $(q, g) \xrightarrow{a} (q', g')$.

A *run* of $\mathcal{T}$ on $u = a_1 \ldots a_n$ is a sequence of configurations and transitions of the form $\sigma = (q_0, g_0) \xrightarrow{a_1} (q_1, g_1) \xrightarrow{a_2} \ldots \xrightarrow{a_n} (q_n, g_n)$. The run ρ is *successful* if

(q_0, g_0) is an initial configuration and the partial output function F is defined on the last state q_n. In this case, the *output* of $\mathcal{T}$ on u is $g_n(F(q_n))$.

Deterministic, *functional* and *unambiguous* streaming transducers are defined as in the two-way case. In particular, in a deterministic *SST* there is only one possible register update for each transition of the automaton. The class of copyless deterministic streaming transducer is denoted *DSST*. A streaming transducer is called *k-register* if it uses at most k registers.

Example 7. Let f be our running example, that is the function on $\{a, b\}^*$ such that for all $w \in \{a, b\}^*$, we have $f(w) = w\overline{w}w$. The streaming transducer described in Figure 2.16 uses three registers: two where we store the word w (remember that registers can only appear once in the output function $F(q)$ so we need two registers for that) by adding the input letters at the right of the register, and one where we store $\overline{w}$ by adding the input letters at the left of the register (see 2.17).

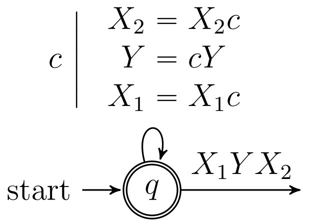


Figure 2.16: A *SST* where $F(q) = X_1YX_2$ computing the function f.

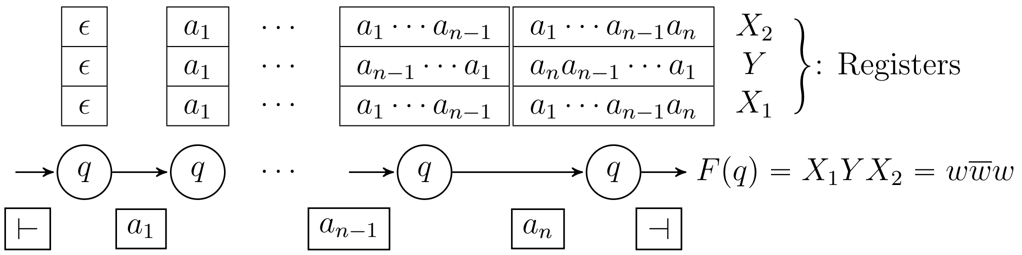


Figure 2.17: A run of a streaming transducer computing f on the word $w = a_1 \cdots a_n$.

2.4.1 Functionality and determinization

As always the deterministic model can only compute functions. The first step towards determinization is thus turning a functional transducer into an unambiguous one.

Proposition 2.4.2 ([9])**.** *For every k-register functional* NSST*, there exists an equivalent k-register unambiguous* NSST *with exponentially many more states.*

Proof. The transformation from a k-register functional streaming transducer $\mathcal{T}$ into an equivalent, unambiguous k-register streaming transducer $\mathcal{T}'$ can be performed in exponential time by using standard techniques. More precisely, one performs a subset construction on $\mathcal{A}$ the underlying finite state automaton of $\mathcal{T}$. This allows one to simulate deterministically all successful runs of $\mathcal{A}$ on the input word. Then one exploits non-determinism to canonically guess a single run among the successful ones —this can be the least run in some lexicographic ordering. Finally, one simulates the register updates of $\mathcal{T}$ along the guessed successful run. The resulting transducer $\mathcal{T}'$ has the same number of registers as $\mathcal{T}$, but exponentially many more states. In particular, $\mathcal{T}'$ can be produced in exponential time from $\mathcal{T}$. □

The next question to ask is whether we can determinize functional *NSST*'s. We will see with Proposition 2.5.4 that the link between *MSO* transductions and streaming transducers will give a positive answer to this question.

As in the case of finite state transducers we have a relatively efficient procedure to decide functionality. The proof of this result uses the same techniques as Theorem 2.2.8.

Theorem 2.4.3 ([4])**.** *The functionality of a* NSST *is decidable in* PSPACE*.*

2.4.2 Number of registers

We can notice that in the last example X_2 always follows Y in the output, and letters are only added at the left of Y and at the right of X_2. This means that we could have used only one register Y with the update $Y = cYc$ when reading the input letter c. We will look at a simpler version of this example, to show that 2-register *NSST*'s are strictly more expressive than 1-register *NSST*'s.

Proposition 2.4.4. *Let f' be the function on $\{a,b\}^*$ such that for all $w \in \{a,b\}^*$, we have $f'(w) = ww$. The transduction f' is clearly definable by a* 2*-register* NSST *but cannot be computed by a* 1*-register* NSST*.*

Proof. Suppose that $\mathcal{T}$ is a 1-register *NSST* computing the function f'. As f' is a function we can assume by Proposition 2.4.2 that $\mathcal{T}$ is unambiguous. Let $\mathcal{T}_1$ be a right-to-left one-way transducer that reads the input word w from right-to-left and guesses backwards the unique accepting run of $\mathcal{T}$ on w. For each transition labeled with a register update $X := uXv$, $\mathcal{T}_1$ outputs u until the first register update $X := u$. After having produced u the transducer continues to read the input to verify that the run it has guessed is correct, but produces no more output.

Analogously, we use an *1NFT* $\mathcal{T}_2$ that simulates the computation of the underlying automaton of $\mathcal{T}$ and guesses the last occurrence of a register update $X := u$. After this, it produces the output v for each transition labeled by $X := uXv$.

In this way, we have by construction that $\mathsf{dom}(\mathcal{T}) = \mathsf{dom}(\mathcal{T}_1) = \mathsf{dom}(\mathcal{T}_2)$ and $\mathcal{T}(w) = \mathcal{T}_1(w)\mathcal{T}_2(w)$ for any word $w \in \mathsf{dom}(\mathcal{T})$. Thus $\mathcal{T}_1$ computes a prefix of ww and $\mathcal{T}_2$ a suffix of ww. However, $\mathcal{T}_1$ as a right-to-left one-way transducer can only produce a prefix of w bounded by $|\mathcal{T}_1|$ (it is the argument of Example 2 in reverse). At the same time $\mathcal{T}_2$ can only produce a suffix $w'w$ of ww where the length of w' is bounded by $|\mathcal{T}_2|$. Altogether, $\mathcal{T}_1$ and $\mathcal{T}_2$ cannot produce ww if w is too long, hence the contradiction. ☐

One can use a similar proof to show that the k-duplication $u \mapsto u^k$ needs at least k registers. It is natural to ask the following open (and difficult) question:

Number of registers	
Input:	An *NSST* or a *DSST* $\mathcal{T}$ and a number k
Question:	Is $\mathcal{T}$ equivalent to an *NSST* or a *DSST* $\mathcal{T}'$ with k registers.

As the determinization of functional *NSST*'s may increase the number of registers, there are in fact three distinct equivalence questions.

In some way, we simulated in the proof of Proposition 2.4.4 the 1-register *DSST* by two passes of a finite state transducer. This idea will be used in Chapter 4 to prove a correspondence between sweeping and streaming transducers with a particular property. Moreover, as Chapter 3 will set the basis for a decision procedure to minimize the number of passes of a sweeping transducer, we will solve this question for a subclass of *NSST* that corresponds to sweeping transducers in Chapter 4.

2.5 Overview of the relations between models

2.5.1 Logic and finite state transducers

Engelfriet and Hoogeboom showed that two-way transducers and *MSO* transductions define effectively the same class of transductions in the deterministic case:

Theorem 2.5.1 ([36]). MSOT = 2DFT

This result, along Corollary 2.2.6, justifies the following definition, that we adopt following the point of view of [36]:

Definition 2.5.2. *We call* regular transductions *the class of transductions defined by a* 2DFT*.*

However the non-deterministic power of those transductions works differently, and this theorem is not true in the non-deterministic setting. The following separating examples can be found in [36] and show that *NMSOT* and *2NFT* are incomparable:

Example 8. Let $\mathcal{R}$ be a relation on $\Sigma = \{a, b\}^*$ such that for any $k \in \mathbb{N}, u \in \Sigma^*$, we have $(u, u^k) \in \mathcal{R}$. Clearly $\mathcal{R}$ is defined by a two-way transducer that non-deterministically chooses to stop or do another back-and-forth pass each time it has produced an occurrence of u.

However, this relation is *infinitary* that is one input word is associated with infinitely many outputs. It is not possible to compute such a transduction with *NMSOT* as the number of guessed valuations is finite. Let us mention that this infinitary property is crucial, as finitary *2NFT*'s are in *NMSOT* [36].

Example 9. Let $\mathcal{R}'$ be a transduction from $\{a\}^*$ to $\{a, b, \#\}^*$ such that for any $n \in \mathbb{N}$ and $w \in \{a, b\}^n$, we have $(a^n, w\#w) \in \mathcal{R}'$. This can easily be realized in *NMSOT*. We copy twice the input and label one copy with a and one with b. Then we guess two sets of positions X_a and X_b representing the positions for which we should consider the copy labelled by a or b.

However, $\mathcal{R}'$ is not definable by a *2NFT*. Let $n \in \mathbb{N}$ such that 2^n is larger than the number of configurations $|Q|(n+2)$. As there are 2^n possible outputs on a^n, we can find two runs with different outputs $u_1\#u_1$ and $u_2\#u_2$ with the same configuration when producing the # symbol. If we connect them we produce the output $u_1\#u_2$ which is impossible by definition of $\mathcal{R}'$.

2.5.2 Streaming transducers

One of the reasons to consider copyless streaming transducers is that they correspond to *MSO* transductions:

Theorem 2.5.3 ([2], [4])**.** *We have the following equivalence between classes:*

- DSST = MSOT
- NSST = NMSOT

As these equivalence are effective and in view of Corollary 2.3.3, we can answer positively to the question of determinization of *SST*'s. The construction involved in the following proposition does not preserve the number of registers.

Proposition 2.5.4 ([4])**.** *If a function is computed by a* NSST *then it can also be computed by a* DSST*.*

The proof of the first item of Theorem 2.5.3 in [2] uses Theorem 2.5.1 and only shows the following inclusions: *2DFT* $\subseteq$ *DSST* $\subseteq$ *MSOT*. A direct construction from *MSOT* to *DSST* is found in [5]. Moreover *2DFT* $\subseteq$ *DSST* in [2] requires an intermediate model: a direct construction, as well as one from *DSST* to *2DFT*, can be found in [58]. More precisely we have the following theorem:

Theorem 2.5.5 ([29])**.**
- *If $\mathcal{T}$ is a* DSST *with n states and m registers, it is effectively equivalent to a* 2DFT *with $\mathcal{O}(mn^n)$ states.*
- *If $\mathcal{T}$ is a* 2DFT *with n states, it is effectively equivalent to a* DSST *with $\mathcal{O}((2n)^{2n})$ states and $2n-1$ registers.*

We will do a very similar construction in Chapter 4 between sweeping transducers and a subclass of *NSST*.

An interesting consequence of this equivalence between *SST*'s and *MSO* transductions is an efficient procedure for deciding *FO*-definability. The definition of the following notion of *aperiodic SST*'s can be found in [40]:

Theorem 2.5.6 ([40])**.** *A* DSST $\mathcal{T}$ *is aperiodic if and only if the transduction it computes is* FO*-definable. Moreover it is* PSPACE*-complete to decide the aperiodicity of a* DSST*.*

If we add the other equivalence results mentionned in this chapter, we obtain an overview of regular transductions depicted in Figure 2.18. The notation *o.p. MSOT* stands for *order-preserving MSO* transductions of Section 2.3.2, and *r.a. fNSST*'s are *right-appending* streaming transducers: the updates are always of the form $X := Xu$. With one counter, those are syntaxically equivalent to one-way finite state transducers. Finally, *concatenation-free NSST*'s are introduced in Section 4.2.1.

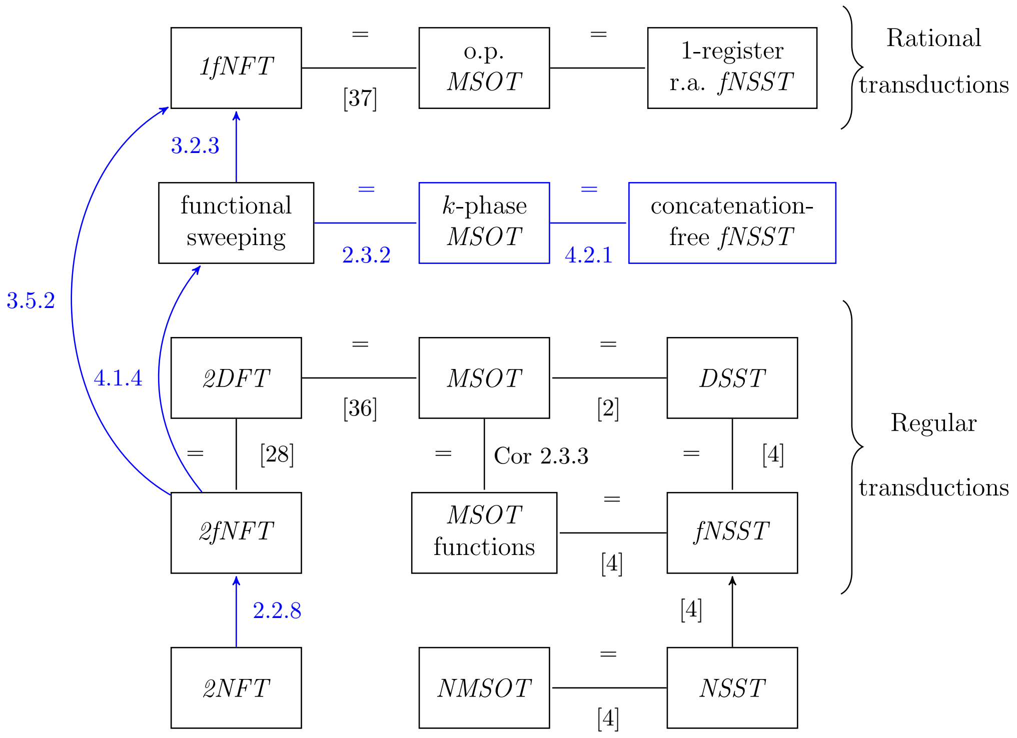


Figure 2.18: An overview of regular transductions. We highlighted in blue our contributions.

2.5.3 Other models

As one can notice by looking at the examples we gave, the case where the alphabet is unary is always particular. It has been for instance studied recently by Guillon [22, 47].

We focus in this thesis on functional transducers. But it is possible that several results presented for functional transducers hold for *finite-valued* transducers. One possible approach is to try to decompose a k-valued transducer into k functional ones. Such a decomposition has been obtained for one-way transducers [73, 84] and for SST's with one register [42] but the problem is open in the general case.

Aperiodic transducers have been studied both for algrebraic reasons and their connection with FO [19, 27, 29, 38]. In the next chapter we present a decision procedure for the one-way definability of a $2NFT$. To this regard it would be interesting to know if a transformation into a one-way transducer (when it is possible of course) preserves aperiodicity.

It is worth mentioning an algebraic characterization of regular transductions similar to the definition of regular languages using rational expressions [6].

In Chapter 4 we describe how to minimize the number of registers for a subclass of $NSST$. Other minimization results have been obtained for a slightly different model, *Cost-Register Automata*, by using a generalization of the twinning property described in Section 2.2 [30]. In the setting of SST's those automata correspond to *right-appending* $DSST$'s , that is transducers that have updates only at the right of registers.

Chapter 3

One-way definability

We have seen in the previous chapter that one-way transducers are less expressive than two-way ones. In [39] the question of *one-way definability* is shown decidable, but with non-elementary complexity. We provide here an elementary procedure based on a proper understanding of the pumping of two-way loops. This chapter is dedicated to the proof of the following theorem:

Theorem 3.0.1. *There is an algorithm that takes as input a functional two-way transducer $\mathcal{T}$ and outputs in* 3ExpTime *a* one-way *transducer $\mathcal{T}'$ satisfying the following properties:*

1. $\mathcal{T}' \subseteq \mathcal{T}$,

2. $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ *if and only if $\mathcal{T}$ is one-way definable.*

3. $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ *can be checked in* 2ExpSpace.

Moreover, if $\mathcal{T}$ is a sweeping transducer, then $\mathcal{T}'$ can be constructed in 2ExpTime *and $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ is decidable in* ExpSpace.

Remark 3.0.2.

- *The transducer $\mathcal{T}'$ constructed in the above theorem is in a certain sense maximal. We will make this more precise at the end of Section 3.5.*

- *There is a tight lower bound on the size of any one-way transducer equivalent to some sweeping transducer. We will show this at the end of Section 3.6*

Let us start with an example illustrating the main idea behind this result.

Example 10. We consider two-way transducers that accept any input u from a given regular language R and produce as output the word $u\,u$. We will argue how, depending on R, these transducers may or may not be one-way definable.

1. If $R = (a+b)^*$, then there is no equivalent one-way transducer, as the output language is not regular. If R is finite, however, then the transduction mapping $u \in R$ to $u\,u$ can be implemented by a one-way transducer that stores the input u (this requires at least as many states as the cardinality of R), and outputs $u\,u$ at the end of the computation.

2. Consider now the periodic language $R = (abc)^*$. The function that maps $u \in R$ to $u\,u$ can be easily implemented by a one-way transducer: it suffices to output alternatively ab, ca, bc for each input letter, while checking that the input is in R.

The main idea to prove Theorem 3.0.1 is to decompose a run of the two-way transducer $\mathcal{T}$ into factors that can be easily simulated in a one-way manner. We defer the formal definition of such a decomposition to Section 3.2, while here we refer to it simply as a "$\boldsymbol{B}$-decomposition", where $\boldsymbol{B}$ is a suitable number computed from $\mathcal{T}$. Intuitively, each factor of the $\boldsymbol{B}$-decomposition either looks like a computation of a one-way transducer, or it produces a periodic output, where the period is bounded by $\boldsymbol{B}$ (we have seen in Example 10 how this is helpful). The key notion that allows to identify factors with periodic outputs is that of an "inversion", and relies on combinatorial results shown in Section 3.1.

In order to provide a roadmap of our proof, we state below the equivalence between the key propositions and defer the technical definitions to the corresponding sections. Here, the number $\boldsymbol{B}$ is doubly exponential in the size of $\mathcal{T}$ in the general case, and simply exponential when $\mathcal{T}$ is sweeping.

Theorem 3.0.3. *Given a functional two-way transducer $\mathcal{T}$, an integer $\boldsymbol{B}$ can be computed such that the following are equivalent:*

P1) *$\mathcal{T}$ is one-way definable,*

P2) *for every successful run of $\mathcal{T}$ and every inversion in it, the output produced between the inversion has period at most $\boldsymbol{B}$,*

P3) *every input has a successful run of $\mathcal{T}$ that admits a $\boldsymbol{B}$-decomposition.*

Overview.

As the notions of inversion and $\boldsymbol{B}$-decomposition are simpler for sweeping transducers, we will first show the above implications for sweeping transducers, and focus later on unrestricted two-way transducers. Specifically, in Section 3.1 we introduce the basic combinatorics on words and the key notion of inversion for a run of a sweeping transducer, and we prove the implication **P1** $\Rightarrow$ **P2**. In Section 3.2 we define $\boldsymbol{B}$-decompositions of runs of sweeping transducers, prove the implication **P2** $\Rightarrow$ **P3**, and sketch a proof of **P3** $\Rightarrow$ **P1** (as a matter of fact, this latter implication can be proved in a way that is independent of whether $\mathcal{T}$ is sweeping or not, which explains why we only sketch the proof in the sweeping case). Section 3.3 lays down the appropriate definitions concerning loops in a run of a two-way transducer, and analyzes in detail the effect of pumping special idempotent loops. In Section 3.4 we further develop the combinatorial arguments that are used to prove the implication **P1** $\Rightarrow$ **P2** in the two-way case. In Section 3.5 we prove the implications **P2** $\Rightarrow$ **P3** $\Rightarrow$ **P1** in the two-way setting, and show how to decide the condition $\mathsf{dom}(\mathcal{T}') = \mathsf{dom}(\mathcal{T})$ of Theorem 3.0.1. Finally, we analyse the complexity in Section 3.6 and show that this problem is undecidable for relations, which justify to consider only functional transducers.

3.1 Basic combinatorics in the sweeping case

In this section, we intend to pump some factors of the input, in order to generate new accepting runs and use functionality to obtain words equations. Then combinatorial arguments due to Fine and Wilf [41] and Saarela [69] allow us to obtain the periodicity of some factors. We obtain equations whose shape is similar to the ones obtain via a pumping lemma of Tim Smith [82] but unlike this result we focus on pairs of input and output, and not only the generated languages. We fix for the rest of the section a functional *sweeping transducer* $\mathcal{T}$, an input word u, and a (normalized) successful run ρ of $\mathcal{T}$ on u.

3.1.1 Pumping loops.

For simplicity, we will denote by ω the maximal position of the input word.

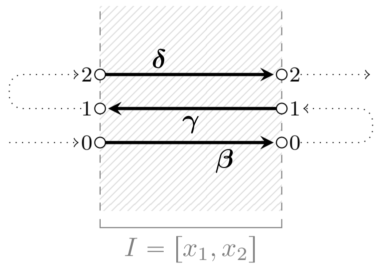


Figure 3.1: Intercepted factors.

We will consider *intervals of positions* of the form $I = [x_1, x_2]$, with $0 \leqslant x_1 < x_2 \leqslant \omega$. The *containment* relation $\subseteq$ on intervals is defined expected, as $[x_3, x_4] \subseteq [x_1, x_2]$ if $x_1 \leqslant x_3 < x_4 \leqslant x_2$.

A *factor* of a run ρ is a contiguous subsequence of ρ. A factor of ρ *intercepted* by an interval $I = [x_1, x_2]$ is a maximal factor that visits only positions $x \in I$, and never uses a left transition from position x_1 or a right transition from position x_2. Figure 3.1 on the right shows the factors $\alpha, \beta, \gamma, \delta, \zeta$ intercepted by an interval I. The numbers that annotate the endpoints of the factors represent their levels.

Loops are a basic concept needed for characterizing one-way definability. Formally, a *loop* of ρ is an interval $L = [x_1, x_2]$ such that $\rho|x_1 = \rho|x_2$, namely, with the same crossing sequences at the extremities. The run ρ can be pumped at any loop $L = [x_1, x_2]$, and this gives rise to new runs with iterated factors. Below we study precisely the shape of these pumped runs.

Definition 3.1.1 (anchor point, trace)**.** *Given a loop L and a location ℓ of ρ, we say that ℓ is an* anchor point *in L if ℓ is the first location of some factor of ρ that is intercepted by L; this factor is then called a* trace *and denoted*[1] *as* $\mathsf{tr}(\ell)$.

Observe that a loop can have at most $\boldsymbol{H} = 2|Q| - 1$ anchor points, since we consider only normalized runs.

Given a loop L of ρ and a number $n \in \mathbb{N}$, we can replicate n times the factor $u[x_1, x_2]$ of the input, obtaining a new input of the form

$$\mathsf{pump}_L^{n+1}(u) \;=\; u[1, x_1] \cdot (u[x_1 + 1, x_2])^{n+1} \cdot u[x_2 + 1, |w|]. \tag{3.1}$$

[1]This is a slight abuse of notation, since the factor $\mathsf{tr}(\ell)$ is not determined by ℓ alone, but requires also the knowledge of the loop L, which is usually clear from the context.

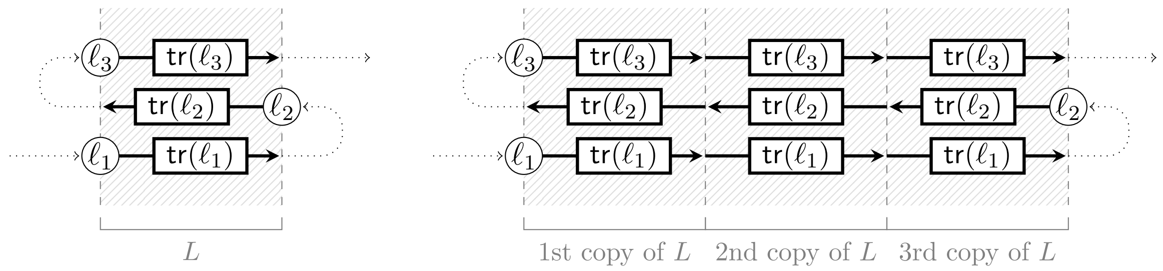


Figure 3.2: A loop L with 3 anchor points, and the result of pumping.

Similarly, we can replicate n times the intercepted factors $\mathsf{tr}(\ell)$ of ρ, for all anchor points ℓ of L. In this way we obtain a successful run on $\mathsf{pump}_L^{n+1}(u)$ that is of the form

$$\mathsf{pump}_L^{n+1}(\rho) \;=\; \rho_0 \; \mathsf{tr}(\ell_1)^n \; \rho_1 \; \dots \; \rho_{k-1} \; \mathsf{tr}(\ell_k)^n \; \rho_k \tag{3.2}$$

where $\ell_1 \trianglelefteq \cdots \trianglelefteq \ell_k$ are all the anchor points in L (listed according to the run order $\trianglelefteq$), ρ_0 is the prefix of ρ ending at ℓ_1, ρ_k is the suffix of ρ starting at ℓ_k, and for all $i = 1, \dots, k-1$, ρ_i is the factor of ρ between ℓ_i and ℓ_{i+1}. Note that $\mathsf{pump}_L^1(\rho)$ coincides with the original run ρ. As a matter of fact, one could define in a similar way the run $\mathsf{pump}_L^0(\rho)$ obtained from removing the loop L from ρ. However, we do not need this, and we will always parametrize the operation pump_L by a positive number $n+1$.

An example of a pumped run $\mathsf{pump}_{L_1}^3(\rho)$ is given in Figure 3.2, together with the indication of the anchor points ℓ_i and the intercepted factors $\mathsf{tr}(\ell_i)$.

3.1.2 Output minimality.

We are interested into factors of the run ρ that lie on a single level and that contribute to the final output, but in a minimal way, in the sense that is formalized by the following definition:

Definition 3.1.2. *Consider a factor $\alpha = \rho[\ell, \ell']$ of ρ. We say that α is* output-minimal *if $\ell = (x, y)$ and $\ell' = (x', y)$, and all loops $L \subsetneq [x, x']$ produce empty output at level y.*

In this section, we set the constant $\boldsymbol{B} = c_{\mathsf{max}} |Q|^{\boldsymbol{H}} + 1$, where c_{max} is the capacity of the transducer, that is, the maximal length of an output produced on a single transition (recall that $|Q|^{\boldsymbol{H}}$ is the maximal number of crossing sequences). As shown below, $\boldsymbol{B}$ bounds the length of the output produced by an output-minimal factor:

Lemma 3.1.3. *For all output-minimal factors α, $|\mathsf{out}(\alpha)| \leqslant \boldsymbol{B}$.*

Proof. Suppose by contradiction that $|\mathsf{out}(\alpha)| > \boldsymbol{B}$, with $\alpha = \rho[\ell, \ell']$, $\ell = (x, y)$ and $\ell = (x', y)$.

Let X be the set of all positions x'', with $\min(x, x') < x'' < \max(x, x')$, that are sources of transitions of α that produce non-empty output. Clearly, the total number of letters produced by the transitions that depart from locations in $X \times \{y\}$ is strictly larger than $\boldsymbol{B} - 1$. Moreover, since each transition emits at most c_{max} symbols, we have $|X| > \frac{\boldsymbol{B}-1}{c_{\mathsf{max}}} = |Q|^{\boldsymbol{H}}$. Now, recall that crossing sequences are sequences of states of length at most $\boldsymbol{H}$. Since $|X|$ is larger than the number of crossing sequences, X contains two positions $x_1 < x_2$ such that $\rho|x_1 = \rho|x_2$. In particular, $L = [x_1, x_2]$ is a loop strictly between x, x' with non-empty output on level y. This shows that $\rho[\ell, \ell']$ is not output-minimal. □

Inversions and periodicity.

Next, we define the crucial notion of inversion. Intuitively, an inversion in a run identifies a part of the run that is potentially difficult to simulate in a one-way manner because the order of generating the output is reversed w.r.t. the input. Inversions arise naturally in transducers that reverse arbitrarily long portions of the input, as well as in transducers that produce copies of arbitrarily long portions of the input.

Definition 3.1.4. *An* inversion *of the run ρ is a tuple $(L_1, \ell_1, L_2, \ell_2)$ such that*

1. *L_1, L_2 are loops of ρ,*

2. *$\ell_1 = (x_1, y_1)$ and $\ell_2 = (x_2, y_2)$ are anchor points of L_1 and L_2, respectively,*

3. *$\ell_1 \lhd \ell_2$ and $x_1 > x_2$*

   (namely, ℓ_2 follows ℓ_1 in the run, but the position of ℓ_2 precedes the position of ℓ_1),

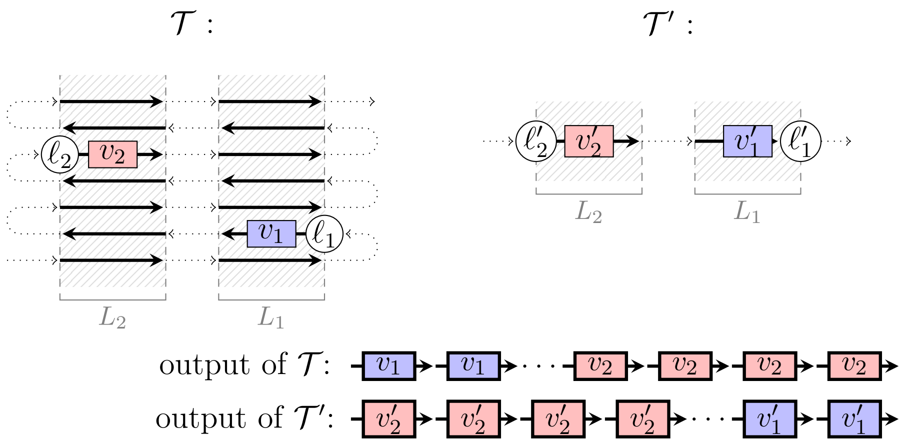


Figure 3.3: An inversion, and the effect of pumping in $\mathcal{T}$ and an equivalent one-way transducer $\mathcal{T}'$.

4. *for both $i = 1$ and $i = 2$, $\mathsf{out}(\mathsf{tr}(\ell_i)) \neq \varepsilon$ and $\mathsf{tr}(\ell_i)$ is output-minimal.*

The left hand-side of Figure 3.3 gives an example of an inversion, assuming that the outputs $v_1 = \mathsf{tr}(\ell_1)$ and $v_2 = \mathsf{tr}(\ell_2)$ are non-empty and the intercepted factors are output-minimal.

The rest of the section is devoted to prove the implication **P1** $\Rightarrow$ **P2** of Theorem 3.0.3. We recall that a word $w = a_1 \cdots a_n$ has *period* p if for every $1 \leqslant i \leqslant |w| - p$, we have $a_i = a_{i+p}$. For example, the word $abc\,abc\,ab$ has period 3.

We remark that, thanks to Lemma 3.1.3, for every inversion $(L_1, \ell_1, L_2, \ell_2)$, the outputs $\mathsf{out}(\mathsf{tr}(\ell_1))$ and $\mathsf{out}(\mathsf{tr}(\ell_2))$ have length at most $\boldsymbol{B}$. By pairing this with the assumption that the transducer $\mathcal{T}$ is one-way definable, and by using some classical word combinatorics, we show that the output produced between the anchor points of every inversion has period that divides the lengths of $\mathsf{out}(\mathsf{tr}(\ell_1))$ and $\mathsf{out}(\mathsf{tr}(\ell_2))$. In particular, this period is at most $\boldsymbol{B}$. The proposition below shows a slightly stronger periodicity property, which refers to the output produced between the anchor points ℓ_1, ℓ_2 of an inversion, but extended on both sides with the words $\mathsf{out}(\mathsf{tr}(\ell_1))$ and $\mathsf{out}(\mathsf{tr}(\ell_2))$. We will exploit this stronger periodicity property later, when dealing with overlapping portions of the run delimited by different inversions (cf. Lemma 3.2.5).

Proposition 3.1.5. *If $\mathcal{T}$ is one-way definable, then the following property* **P2** *holds:*

For all inversions $(L_1, \ell_1, L_2, \ell_2)$ of ρ, the period p of the word

$$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1, \ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$$

divides both $|\mathsf{out}(\mathsf{tr}(\ell_1))|$ and $|\mathsf{out}(\mathsf{tr}(\ell_2))|$. Moreover, $p \leqslant \boldsymbol{B}$.

The above proposition thus formalizes the implication **P1** $\Rightarrow$ **P2** of Theorem 3.0.3. Its proof relies on a few combinatorial results. The first one is Fine and Wilf's theorem [59]. In short, this theorem says that, whenever two periodic words w_1, w_2 share a sufficiently long factor, then they have the same period. Below, we state a slightly stronger variant of Fine and Wilf's theorem, which additionally shows how to align a common factor of the two words w_1, w_2 so as to form a third word containing a prefix of w_1 and a suffix of w_2. The additional claim will be exploited in the proof of Lemma 3.2.5.

Theorem 3.1.6 (Fine-Wilf's theorem)**.** *If $w_1 = w_1'\, w\, w_1''$ has period p_1, $w_2 = w_2'\, w\, w_2''$ has period p_2, and the common factor w has length at least $p_1 + p_2 - \gcd(p_1, p_2)$, then w_1, w_2, and $w_3 = w_1'\, w\, w_2''$ have period $\gcd(p_1, p_2)$.*

A second combinatorial result required in our proof was shown by Kortelainen [56], and later improved by Saarela [69]. The result is related to word equations with iterated factors, like those that arise from considering outputs of pumped runs. To improve readability, we highlight below the important iterations of factors inside the considered equation.

Theorem 3.1.7 (Theorem 4.3 in [69])**.** *Consider a word equation*

$$v_0\, \boldsymbol{v_1^n}\, v_2\, \cdots\, v_{k-1}\, \boldsymbol{v_k^n}\, v_{k+1} \;=\; w_0\, \boldsymbol{w_1^n}\, w_2\, \cdots\, w_{k'-1}\, \boldsymbol{w_{k'}^n}\, w_{k'+1}$$

where n is the unknown and v_i, w_j are words. Then the set of solutions of the equation is either finite or $\mathbb{N}$.

We will use the above result mainly to reason about periodicities of words with iterated factors, as it is done by the following corollary:

Corollary 3.1.8. *If $v_0\, \boldsymbol{v_1^n}\, v_2\, \cdots\, v_{k-1}\, \boldsymbol{v_k^n}\, v_{k+1}$ has period p for infinitely many n, then it has period p for all n.*

Proof. The fact that $v_0\, \boldsymbol{v_1^n}\, v_2\, \cdots\, v_{k-1}\, \boldsymbol{v_k^n}\, v_{k+1}$ has period p, for some given n, can be expressed by the equation $v_0\, \boldsymbol{v_1^n}\, v_2\, \cdots\, v_{k-1}\, \boldsymbol{v_k^n}\, v_{k+1} = w_n^{kn}\, w_n'$, where $|w_n| = p$, w_n' is a prefix of w_n of length smaller than p, and k does not depend on n. Moreover, since the latter equation holds for infinitely many n, we can assume, without loss of generality, that $w_n = w$ and $w'_n = w'$ for some words w, w' (and for infinitely many n). This means that $v_0\,\boldsymbol{v_1^n}\,v_2 \cdots v_{k-1}\,\boldsymbol{v_k^n}\,v_{k+1} = \boldsymbol{w^{kn}}\,w'$ holds for infinitely many n, and for fixed words $v_0, v_1, \ldots, v_{k+1}, w, w'$. By Theorem 3.1.7, we know that the previous equation holds for all n, and thus $v_0\,\boldsymbol{v_1^n}\,v_2 \cdots v_{k-1}\,\boldsymbol{v_k^n}\,v_{k+1}$ has period p for all n. □

Recall that our goal is to show that the output produced between every inversion has period bounded by $\boldsymbol{B}$. The general idea is to pump the loops of the inversion and compare the outputs of the two-way transducer $\mathcal{T}$ with those of an equivalent one-way transducer $\mathcal{T}'$. The comparison leads to an equation between words with iterated factors, where the iterations are parametrized by two unknowns n_1, n_2 that occur in opposite order in the left, respectively right hand-side of the equation. Our third and last combinatorial result considers a word equation of this precise form, and derives from it a periodicity property. For the sake of brevity, we use the notation $v^{(n_1,n_2)}$ to represent words with factors iterated n_1 or n_2 times, namely, words of the form $v_0\,v_1^{n_{i_1}}\,v_2 \cdots v_{k-1}\,v_k^{n_{i_k}}\,v_{k+1}$, where the $v_0, v_1, v_2, \ldots, v_{k-1}, v_k, v_{k+1}$ are fixed words (possibly empty) and each index among $i_1, \ldots, i_k$ is either 1 or 2.

Lemma 3.1.9. *Consider a word equation of the form*

$$v_0^{(n_1,n_2)}\,\boldsymbol{v_1^{n_1}}\,v_2^{(n_1,n_2)}\,\boldsymbol{v_3^{n_2}}\,v_4^{(n_1,n_2)} \quad = \quad w_0\,\boldsymbol{w_1^{n_2}}\,w_2\,\boldsymbol{w_3^{n_1}}\,w_4$$

where n_1, n_2 are the unknowns and v_1, v_3 are non-empty words. If the above equation holds for all $n_1, n_2 \in \mathbb{N}$, then

$$\boldsymbol{v_1}\;\boldsymbol{v_1^{n_1}}\;v_2^{(n_1,n_2)}\;\boldsymbol{v_3^{n_2}}\;\boldsymbol{v_3}$$

has period $\gcd(|v_1|, |v_3|)$ *for all* $n_1, n_2 \in \mathbb{N}$.

Proof. The idea of the proof is to let the parameters n_1, n_2 of the equation grow independently, and exploit Fine and Wilf's theorem (Theorem 3.1.6) a certain number of times to establish periodicities in overlapping factors of the considered words.

We begin by fixing n_1 large enough so that the factor $\boldsymbol{v_1^{n_1}}$ of the left hand-side of the equation becomes longer than $|w_0| + |w_1|$ (this is possible because v_1 is non-empty). Now, if we let n_2 grow arbitrarily large, we see that the length of the periodic word $\boldsymbol{w_1^{n_2}}$ is almost equal to the length of the left hand-side term $v_0^{(n_1,n_2)}\,\boldsymbol{v_1^{n_1}}\,v_2^{(n_1,n_2)}\,\boldsymbol{v_3^{n_2}}\,v_4^{(n_1,n_2)}$: indeed, the difference in length is given by the constant $|w_0| + |w_2| + n_1 \cdot |w_3| + |w_4|$. In particular, this implies that $\boldsymbol{w_1^{n_2}}$ covers arbitrarily long prefixes of $\boldsymbol{v_1}\ v_2^{(n_1,n_2)}\ \boldsymbol{v_3^{n_2}}\ \boldsymbol{v_3}$, (here we add an occurrence of $\boldsymbol{v_3}$ so that the output is long enough if $n_2 = 0$) which in its turn contains long repetitions of the word v_3. Hence, by Theorem 3.1.6, the word $\boldsymbol{v_1}\ v_2^{(n_1,n_2)}\ \boldsymbol{v_3^{n_2+1}}$ has period $|v_3|$.

We remark that the periodicity shown so far holds for a large enough n_1 and for all but finitely many n_2, where the threshold for n_2 depends on n_1: once n_1 is fixed, n_2 needs to be larger than $f(n_1)$, for a suitable function f. In fact, by using Corollary 3.1.8, with n_1 fixed and $n = n_2$, we deduce that the periodicity holds for large enough $n_1 \in \mathbb{N}$ and for all $n_2 \in \mathbb{N}$.

We could also apply a symmetric reasoning: we choose n_2 large enough and let n_1 grow arbitrarily large. Doing so, we prove that for a large enough n_2 and for all but finitely many n_1, the word $\boldsymbol{v_1}\ \boldsymbol{v_1^{n_1}}\ v_2^{(n_1,n_2)}\ \boldsymbol{v_3}$ is periodic with period $|v_1|$. As before, with the help of Corollary 3.1.8, this can be strengthened to hold for large enough $n_2 \in \mathbb{N}$ and for all $n_1 \in \mathbb{N}$.

Putting together the results proven so far, we get that for all but finitely many n_1, n_2,

$$\underbrace{\boldsymbol{v_1^{n_1}} \cdot \overbrace{\boldsymbol{v_1} \ \cdot\ v_2^{(n_1,n_2)} \ \cdot\ \boldsymbol{v_3}}_{\text{period } |v_1|}^{\text{period } |v_3|}}\ \cdot \boldsymbol{v_3^{n_2}}\ .$$

Finally, we observe that the prefix $\boldsymbol{v_1^{n_1+1}} \cdot v_2^{(n_1,n_2)} \cdot \boldsymbol{v_3}$ and the suffix $\boldsymbol{v_1} \cdot v_2^{(n_1,n_2)} \cdot \boldsymbol{v_3^{n_2+1}}$ share a common factor of length at least $|v_1| + |v_3|$. By Theorem 3.1.6, we derive that $\boldsymbol{v_1^{n_1+1}} \cdot v_2^{(n_1,n_2)} \cdot \boldsymbol{v_3^{n_2+1}}$ has period $\gcd(|v_1|, |v_3|)$ for all but finitely many n_1, n_2. Finally, by exploiting again Corollary 3.1.8, we conclude that the periodicity holds for all $n_1, n_2 \in \mathbb{N}$. ☐

We are now ready to prove the implication **P1** ⇒ **P2**:

Proof of Proposition 3.1.5. Let $\mathcal{T}'$ be a one-way transducer equivalent to $\mathcal{T}$, and consider an inversion $(L_1, \ell_1, L_2, \ell_2)$ of the successful run ρ of $\mathcal{T}$ on input u. The reader may refer to Figure 3.3 to get basic intuition about the proof technique. For simplicity, we assume that the loops L_1 and L_2 are disjoint, as shown in the figure. If this were not the case, we would have at least $\max(L_1) > \min(L_2)$, since the anchor point ℓ_1 is strictly to the right of the anchor point ℓ_2. We could then consider the pumped run $\mathsf{pump}_{L_1}^k(\rho)$ for a large enough $k > 1$ in such a way that the rightmost copy of L_1 turns out to be disjoint from and strictly to the right of L_2. We could thus reason as we do below, by replacing everywhere (except in the final part of the proof, cf. *Transferring periodicity to the original run*) the run ρ with the pumped run $\mathsf{pump}^k_{L_1}(\rho)$, and the formal parameter m_1 with $m_1 + k$.

Inducing loops in T'. We begin by pumping the run ρ and the underlying input u, on the loops L_1 and L_2, in order to induce new loops L'_1 and L'_2 that are also loops in a successful run of $\mathcal{T}'$. Assuming that L_1 is strictly to the right of L_2, we define for all numbers $m_1, m_2 \in \mathbb{N}$:

$$\begin{aligned} u^{(m_1,m_2)} &= \mathsf{pump}^{m_1+1}_{L_1}(\mathsf{pump}^{m_2+1}_{L_2}(u)) \\ \rho^{(m_1,m_2)} &= \mathsf{pump}^{m_1+1}_{L_1}(\mathsf{pump}^{m_2+1}_{L_2}(\rho)). \end{aligned}$$

In the pumped run $\rho^{(m_1,m_2)}$, we identify the positions that mark the endpoints of the occurrences of L_1, L_2. More precisely, if $L_1 = [x_1, x_2]$ and $L_2 = [x_3, x_4]$, with $x_1 > x_4$, then the sets of these positions are

$$\begin{aligned} X_2^{(m_1,m_2)} &= \big\{x_3 + i(x_4 - x_3) \;:\; 0 \leqslant i \leqslant m_2 + 1\big\} \\ X_1^{(m_1,m_2)} &= \big\{x_1 + j(x_2 - x_1) + m_2(x_4 - x_3) \;:\; 0 \leqslant j \leqslant m_1 + 1\big\}. \end{aligned}$$

Periodicity of outputs of pumped runs. We now exploit the fact that $\mathcal{T}'$ is a one-way transducer equivalent to $\mathcal{T}$. Let $\lambda^{(m_1,m_2)}$ be a successful run of $\mathcal{T}'$ on the input $u^{(m_1,m_2)}$. Since $\mathcal{T}'$ has finitely many states, we can find a large enough number k_0 and two positions $x'_1 < x'_2$, both in $X_1^{(k_0,k_0)}$, such that $L'_1 = [x'_1, x'_2]$ is a loop of $\lambda^{(k_0,k_0)}$. Similarly, we can find two positions $x'_3 < x'_4$, both in $X_2^{(k_0,k_0)}$, such that $L'_2 = [x'_3, x'_4]$ is a loop of $\lambda^{(k_0,k_0)}$. By construction L'_1 (resp. L'_2) consists of $k_1 \leqslant k_0$ (resp. $k_2 \leqslant k_0$) copies of L_1 (resp. L_2), and hence L'_1, L'_2 are also loops of $\rho^{(k_0,k_0)}$. In particular, this implies that for all $n_1, n_2 \in \mathbb{N}$:

$$\begin{aligned} \mathsf{pump}^{n_1+1}_{L'_1}(\mathsf{pump}^{n_2+1}_{L'_2}(u^{(k_0,k_0)})) &= u^{(f(n_1),g(n_2))} \\ \mathsf{pump}^{n_1+1}_{L'_1}(\mathsf{pump}^{n_2+1}_{L'_2}(\rho^{(k_0,k_0)})) &= \rho^{(f(n_1),g(m_2))} \\ \mathsf{pump}^{n_1+1}_{L'_1}(\mathsf{pump}^{n_2+1}_{L'_2}(\lambda^{(k_0,k_0)})) &= \lambda^{(f(n_1),g(n_2))}. \end{aligned}$$

where $f(n_1) = k_1 n_1 + k_0$ and $g(n_2) = k_2 n_2 + k_0$.

Now recall that $\rho^{(f(n_1),g(n_2))}$ and $\lambda^{(f(n_1),g(n_2))}$ are runs of $\mathcal{T}$ and $\mathcal{T}'$ on the same word $u^{(f(n_1),g(n_2))}$, and they produce the same output. Let us denote this output by $w^{(f(n_1),g(n_2))}$. Below, we show two possible factorizations of $w^{(f(n_1),g(n_2))}$ based on the shapes of the pumped runs $\lambda^{(f(n_1),g(n_2))}$ and

$\rho^{(f(n_1),g(n_2))}$. For the first factorization, we recall that L'_2 precedes L'_1, according to the ordering of positions, and that the run $\lambda^{(f(n_1),g(n_2))}$ is one-way. We thus obtain

$$w^{(f(n_1),g(n_2))} \;=\; w_0\ \boldsymbol{w_1^{n_2}}\ w_2\ \boldsymbol{w_3^{n_1}}\ w_4 \tag{3.3}$$

where

- $\boldsymbol{w_1}$ is the output produced by the (unique) factor of $\lambda^{(k_0,k_0)}$ intercepted by L'_2,
- $\boldsymbol{w_3}$ is the output produced by the (unique) factor of $\lambda^{(k_0,k_0)}$ intercepted by L'_1,
- w_0 is the output produced by the prefix of $\lambda^{(k_0,k_0)}$ up to the left border of L'_2,
- w_2 is the output produced by the factor of $\lambda^{(k_0,k_0)}$ between the right border of L'_2 and the left border of L'_1,
- w_4 is the output produced by the suffix of $\lambda^{(k_0,k_0)}$ after the right border of L'_1.

For the second factorization, we consider L'_1 and L'_2 as loops of $\rho^{(k_0,k_0)}$. We recall that ℓ_1, ℓ_2 are anchor points of the loops L_1, L_2 of ρ, and that there are corresponding copies of these anchor points in the pumped run $\rho^{(f(n_1),g(n_2))}$. We define ℓ'_1 (resp. ℓ'_2) to be the first (resp. last) location in $\rho^{(f(n_1),g(n_2))}$ that corresponds to ℓ_1 (resp. ℓ_2) and that is an anchor point of a copy of L'_1 (resp. L'_2). For example, if $\ell_1 = (x_1, y_1)$, with y_1 even, then $\ell'_1 = \big(x_1 + f(n_2)(x_4 - x_3), y_1\big)$. Thanks to Equation 3.2 we know that the output produced by $\rho^{(f(n_1),g(n_2))}$ is of the form

$$w^{(f(n_1),g(n_2))} \;=\; v_0^{(n_1,n_2)}\ \boldsymbol{v_1^{n_1}}\ v_2^{(n_1,n_2)}\ \boldsymbol{v_3^{n_2}}\ v_4^{(n_1,n_2)} \tag{3.4}$$

where

- $\boldsymbol{v_1} = \mathsf{out}(\mathsf{tr}(\ell'_1))$, where ℓ'_1 is seen as an anchor point in a copy of L'_1,
- $\boldsymbol{v_3} = \mathsf{out}(\mathsf{tr}(\ell'_2))$, where ℓ'_2 is seen as an anchor point in a copy of L'_2 (note that the words v_1, v_3 depend on k_0, but not on n_1, n_2),

- $v_0^{(n_1,n_2)}$ is the output produced by the prefix of $\rho^{(f(n_1),g(n_2))}$ that ends before the first repetition of $\mathsf{tr}(\ell'_1)$ (this word may depend on the parameters n_1, n_2 since the loops L'_1, L'_2 may be traversed several times before reaching the first occurrence of $\mathsf{tr}(\ell'_1)$),
- $v_2^{(n_1,n_2)}$ is the output produced by the factor of $\rho^{(f(n_1),g(n_2))}$ between the first repetition of $\mathsf{tr}(\ell'_1)$ and the last repetition of $\mathsf{tr}(\ell'_2)$,
- $v_4^{(n_1,n_2)}$ is the output produced by the suffix of $\rho^{(f(n_1),g(n_2))}$ after the last repetition of $\mathsf{tr}(\ell'_2)$.

Putting together Equations (3.3) and (3.4), we get

$$v_0^{(n_1,n_2)} \; \boldsymbol{v_1^{n_1}} \; v_2^{(n_1,n_2)} \; \boldsymbol{v_3^{n_2}} \; v_4^{(n_1,n_2)} \quad = \quad w_0 \; \boldsymbol{w_1^{n_2}} \; w_2 \; \boldsymbol{w_3^{n_1}} \; w_4. \tag{3.5}$$

Recall that the definition of inversions (Definition 3.1.4) states that the words v_1, v_3 are non-empty. This allows us to apply Lemma 3.1.9, which shows that the word $\boldsymbol{v_1} \; \boldsymbol{v_1^{n_1}} \; v_2^{(n_1,n_2)} \; \boldsymbol{v_3^{n_2}} \; \boldsymbol{v_3}$ has period $p = \gcd(|v_1|, |v_3|)$, for all $n_1, n_2 \in \mathbb{N}$.

Note that the latter period p still depends on $\mathcal{T}'$, since the words v_1, v_3 were obtained from loops L'_1, L'_2 on the run $\lambda^{(k_0,k_0)}$ of $\mathcal{T}'$. However, because each loop L'_i consists of k_i copies of the original loop L_i, we also know that $v_1 = (\mathsf{out}(\mathsf{tr}(\ell_1)))^{k_1}$ and $v_3 = (\mathsf{out}(\mathsf{tr}(\ell_2)))^{k_2}$. By Theorem 3.1.6, this implies that for all $n_1, n_2 \in \mathbb{N}$, the word

$$\big(\mathsf{out}(\mathsf{tr}(\ell_1))\big) \; \big(\mathsf{out}(\mathsf{tr}(\ell_1))\big)^{k_1 n_1} \; v_2^{(n_1,n_2)} \; \big(\mathsf{out}(\mathsf{tr}(\ell_2))\big)^{k_2 n_2} \; \big(\mathsf{out}(\mathsf{tr}(\ell_2))\big)$$

has a period that divides $|\mathsf{out}(\mathsf{tr}(\ell_1))|$ and $|\mathsf{out}(\mathsf{tr}(\ell_2))|$.

Transferring periodicity to the original run. The last part of the proof amounts at showing a similar periodicity property for the output produced by the original run ρ. By construction, the iterated factors inside $v_2^{(n_1,n_2)}$ in the previous word are all of the form $v^{k_1 n_1 + k_0}$ or $v^{k_2 n_2 + k_0}$, for some words v. By taking out the constant factors v^{k_0} from the latter repetitions, we can write $v_2^{(n_1,n_2)}$ as a word with iterated factors of the form $v^{k_1 n_1}$ or $v^{k_2 n_2}$, namely, as ${v'_2}^{(k_1 n_1, k_2 n_2)}$. So the word

$$\big(\mathsf{out}(\mathsf{tr}(\ell_1))\big) \; \big(\mathsf{out}(\mathsf{tr}(\ell_1))\big)^{k'_1} \; {v'_2}^{(k'_1,k'_2)} \; \big(\mathsf{out}(\mathsf{tr}(\ell_2))\big)^{k'_2} \; \big(\mathsf{out}(\mathsf{tr}(\ell_2))\big)$$

is periodic, with period that divides $|\mathsf{out}(\mathsf{tr}(\ell_1))|$ and $|\mathsf{out}(\mathsf{tr}(\ell_2))|$, for all $k'_1 \in \{k_1 n \;:\; n \in \mathbb{N}\}$ and all $k'_2 \in \{k_2 n \;:\; n \in \mathbb{N}\}$. We now apply Corollary

3.1.8, once with $n = k'_1$ and once with $n = k'_2$, to conclude that the latter periodicity property holds also for $k'_1 = 1$ and $k'_2 = 1$. This shows that the word when we take $k'_1 = k'_2 = 1$, which is:

$$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1, \ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$$

is periodic, with period that divides $|\mathsf{out}(\mathsf{tr}(\ell_1))|$ and $|\mathsf{out}(\mathsf{tr}(\ell_2))|$. □

3.2 One-way definability in the sweeping case

In the previous section we have shown the implication **P1** ⇒ **P2** for a functional sweeping transducer $\mathcal{T}$. Here we close the cycle by proving the implications **P2** ⇒ **P3** and **P3** ⇒ **P1**. In particular, we show how to derive the existence of successful runs admitting a $\boldsymbol{B}$-decomposition and construct a one-way transducer $\mathcal{T}'$ that simulates $\mathcal{T}$ on those runs. This will basically prove Theorem 3.0.3 in the sweeping case.

3.2.1 Run decomposition.

We begin by giving the definition of $\boldsymbol{B}$-decomposition for a run ρ of $\mathcal{T}$. Intuitively, a $\boldsymbol{B}$-decomposition of ρ identifies factors of ρ that can be easily simulated in a one-way manner. The definition below describes precisely the shape of these factors.

First we need to recall the notion of almost periodicity: a word w is *almost periodic with bound* p if $w = w_0\ w_1\ w_2$ for some words w_0, w_2 of length at most p and some word w_1 of period at most p.

We need to work with subsequences of the run ρ that are induced by particular sets of locations, not necessarily consecutive. Recall that $\rho[\ell, \ell']$ denotes the factor of ρ delimited by two locations $\ell \trianglelefteq \ell'$. Similarly, given any set Z of locations, we denote by $\rho|Z$ the subsequence of ρ induced by Z. Note that, even though $\rho|Z$ might not be a valid run of the transducer, we can still refer to the number of transitions in it and to the size of the produced output $\mathsf{out}(\rho|Z)$. Formally, a transition in $\rho|Z$ is a transition from some ℓ to ℓ', where both ℓ, ℓ' belong to Z. The output $\mathsf{out}(\rho|Z)$ is the concatenation of the outputs of the transitions of $\rho|Z$ (according to the order given by ρ).

Definition 3.2.1. *Consider a factor $\rho[\ell, \ell']$ of a run ρ of $\mathcal{T}$, where $\ell = (x, y)$, $\ell' = (x', y')$ are two locations with $x \leqslant x'$. We call $\rho[\ell, \ell']$*

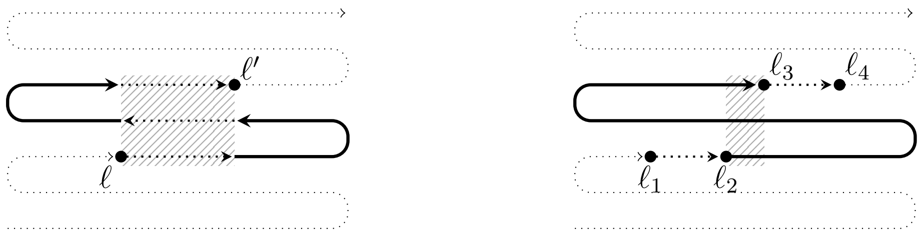


Figure 3.4: A block $\rho[\ell, \ell']$ and a diagonal $\rho[\ell_1, \ell_4]$. The bounded outputs are in bold font.

1. *a* floor *if $y = y'$ is even, i.e. $\rho[\ell, \ell']$ lies on the same level and is rightward oriented;*

2. *a $\boldsymbol{B}$-*diagonal *if there is a sequence $\ell = \ell_0 \trianglelefteq \ell_1 \trianglelefteq \cdots \trianglelefteq \ell_{2n+1} = \ell'$, where each $\rho[\ell_{2i+1}, \ell_{2i+2}]$ is a floor, each $\rho[\ell_{2i}, \ell_{2i+1}]$ produces an output of length at most $2\boldsymbol{HB}$, and the position of each ℓ_i is to the left of the position of ℓ_{i+1};*

3. *a $\boldsymbol{B}$-*block *if the output produced by $\rho[\ell, \ell']$ is almost periodic with bound $2\boldsymbol{B}$, and the output produced by the subsequence $\rho|Z$, where $Z = [\ell, \ell'] \setminus \big([x, x'] \times [y, y']\big)$, has length at most $2\boldsymbol{HB}$.*

Before continuing we give some intuition on the notions defined above. The simplest concept is that of floor, a rightwards oriented factor of a run. Diagonals are sequences of consecutive floors interleaved by factors that produce bounded outputs. Blocks may appear between two consecutive diagonals. An important constraint in the definition of a block is that the output produced by $\rho[\ell, \ell']$ must be almost periodic. In addition, the block must satisfy a constraint on the length of the output produced by the subsequence $\rho|Z$, where $Z = [\ell, \ell'] \setminus \big([x, x'] \times [y, y']\big)$ is the set of locations of $\rho[\ell, \ell']$ outside the block defined by ℓ, ℓ'. Figure 3.4 represents the block of ℓ, ℓ' by a hatched rectangle, and the subsequence $\rho|Z$ by some thick arrows.

Definition 3.2.2. *A $\boldsymbol{B}$-*decomposition *of a run ρ of $\mathcal{T}$ is a factorization $\prod_i \rho[\ell_i, \ell_{i+1}]$ of ρ into $\boldsymbol{B}$-diagonals and $\boldsymbol{B}$-blocks.*

Figure 3.5 gives an example of a decomposition. Each factor is either a diagonal D_i or a block B_i, and each diagonal is built up of one or more floors. The hatched rectangles represent the blocks. We observe that the floors and the blocks of the decomposition are arranged along a diagonal, i.e. following the natural order of positions and levels. Intuitively, this means that the output produced inside these floors and blocks can be simulated in a one-way manner. Moreover, most of the output of ρ is produced inside floors and blocks: indeed, thanks to Definition 3.2.1, at most $2\boldsymbol{H}^2\boldsymbol{B}$ symbols are produced outside floors and blocks.

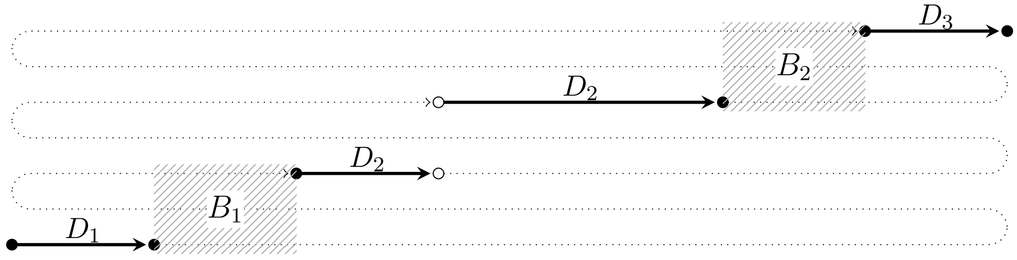


Figure 3.5: Decomposition of a run into diagonals and blocks.

3.2.2 From periodicity of inversions to existence of decompositions.

Now that we set up the definition of $\boldsymbol{B}$-decomposition, we turn towards proving the implication **P2** $\Rightarrow$ **P3** of Theorem 3.0.3. In fact, we will prove a slightly stronger result than **P2** $\Rightarrow$ **P3**, which is stated further below. Formally, when we say that a run ρ *satisfies* **P2** we mean that for every inversion $(L_1, \ell_1, L_2, \ell_2)$ of ρ, the word $\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1, \ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$ has period $\gcd(|\mathsf{out}(\mathsf{tr}(\ell_1))|, |\mathsf{out}(\mathsf{tr}(\ell_2))|) \leqslant \boldsymbol{B}$. We aim at proving that every run that satisfies **P2** enjoys a decomposition, independently of whether other runs do or do not satisfy **P2**:

Proposition 3.2.3. *If ρ is a run of $\mathcal{T}$ that satisfies* **P2***, then ρ admits a $\boldsymbol{B}$-decomposition.*

Let us fix a run ρ of $\mathcal{T}$ and assume that it satisfies **P2**. To identify the elements of a $\boldsymbol{B}$-decomposition of ρ (i.e. the diagonals and the blocks), we introduce a suitable equivalence relation between locations:

Definition 3.2.4. *Let S be the relation that pairs every two locations ℓ, ℓ' of ρ whenever there is an inversion $(L_1, \ell_1, L_2, \ell_2)$ of ρ such that $\ell_1 \trianglelefteq \ell, \ell' \trianglelefteq \ell_2$, namely, whenever ℓ and ℓ' occur within the same inversion. Let S^* be the reflexive and transitive closure of S.*

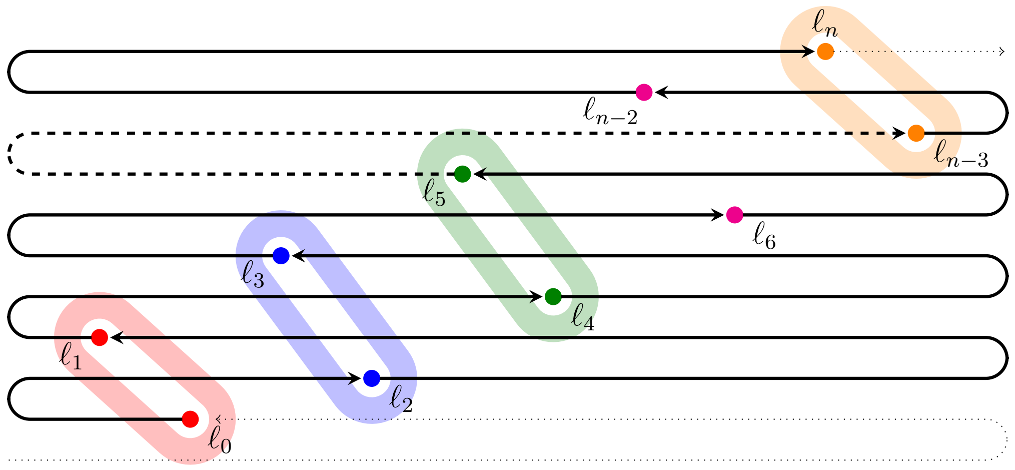


Figure 3.6: A non-singleton S^*-equivalence class seen as a series of overlapping inversions.

It is easy to see that every equivalence class of S^* is a convex subset with respect to the run order on locations of ρ. Moreover, every *non-singleton* equivalence class of S^* is a union of a series of inversions that are two-by-two overlapping. One can refer to Figure 3.6 for an intuitive account of what we mean by two-by-two overlapping: the thick arrows represent factors of the run that lie entirely inside an S^*-equivalence class, each inversion is identified by a pair of consecutive anchor points (given with the same color in the Figure). According to the run order, between every pair of anchor points with the same color, there is at least one anchor point of another inversion: this is what we mean when we say that the two inversions considered are overlapping.

Formally, we say that an inversion $(L_1, \ell_1, L_2, \ell_2)$ *covers* a location ℓ when $\ell_1 \trianglelefteq \ell \trianglelefteq \ell_2$. We say that two inversions $(L_1, \ell_1, L_2, \ell_2)$ and $(L_3, \ell_3, L_4, \ell_4)$ are *overlapping* if $(L_1, \ell_1, L_2, \ell_2)$ covers ℓ_3 and $(L_3, \ell_3, L_4, \ell_4)$ covers ℓ_2 (or the other way around).

The next lemma exploits the fact that ρ satisfies **P2** to deduce that the output produced inside every S^*-equivalence class has period at most $\boldsymbol{B}$. Note that the proof below does not exploit the fact that the transducer is sweeping.

Lemma 3.2.5. *If ρ satisfies* **P2** *and $\ell \trianglelefteq \ell'$ are two locations of ρ such that $\ell \ \mathsf{S}^* \ \ell'$, then the output $\mathsf{out}(\rho[\ell, \ell'])$ produced between these locations has* *period at most $\boldsymbol{B}$.*

Proof. The claim for $\ell = \ell'$ holds trivially, so we assume that $\ell \lhd \ell'$. Since the S^*-equivalence class that contains ℓ, ℓ' is non-singleton, we know that there is a series of inversions

$$(L_0, \ell_0, L_1, \ell_1) \quad (L_2, \ell_2, L_3, \ell_3) \quad \ldots\ldots \quad (L_{2k}, \ell_{2k}, L_{2k+1}, \ell_{2k+1})$$

that are two-by-two overlapping and such that $\ell_0 \trianglelefteq \ell \lhd \ell' \trianglelefteq \ell_{2k+1}$. Without loss of generality, we can assume that every inversion $(L_{2i}, \ell_{2i}, L_{2i+1}, \ell_{2i+1})$ is *maximal* in the following sense: there is no other inversion $(\tilde{L}, \tilde{\ell}, \tilde{L}', \tilde{\ell}') \neq (L_{2i}, \ell_{2i}, L_{2i+1}, \ell_{2i+1})$ such that $\tilde{\ell} \trianglelefteq \ell_{2i} \trianglelefteq \ell_{2i+1} \trianglelefteq \tilde{\ell}'$.

For the sake of brevity, let $v_i = \mathsf{out}(\mathsf{tr}(\ell_i))$ and $p_i = |v_i|$. Since ρ satisfies **P2** (recall Proposition 3.1.5), we know that, for all $i = 0, \ldots, n$, the word

$$v_{2i}\ \mathsf{out}(\rho[\ell_{2i}, \ell_{2i+1}])\ v_{2i+1}$$

has period that divides both p_{2i} and p_{2i+1} and is at most $\boldsymbol{B}$. In order to show that the period of $\mathsf{out}(\rho[\ell, \ell'])$ is also bounded by $\boldsymbol{B}$, it suffices to prove the following claim by induction on i:

Claim. *For all $i = 0, \ldots, k$, the word $\mathsf{out}\big(\rho[\ell_0, \ell_{2i+1}]\big)\ v_{2i+1}$ has period at most $\boldsymbol{B}$ that divides p_{2i+1}.*

The base case $i = 0$ follows immediately from our hypothesis, since $(L_0, \ell_0, L_1, \ell_1)$ is an inversion. For the inductive step, we assume that the claim holds for $i < k$, and we prove it for $i + 1$. First of all, we factorize our word as follows:

$$\mathsf{out}\big(\rho[\ell_0, \ell_{2i+3}]\big)\ v_{2i+3} \;=\; \underbrace{\mathsf{out}\big(\rho[\ell_0, \ell_{2i+2}]\big)\ \overbrace{\mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+1}]\big)}_{\text{period } p_{2i+1}}\ \mathsf{out}\big(\rho[\ell_{2i+1}, \ell_{2i+3}]\big)\ v_{2i+3}}^{\text{periods } p_{2i+2} \text{ and } p_{2i+3}}\,.$$

By the inductive hypothesis, the output produced between ℓ_0 and ℓ_{2i+1}, extended to the right with v_{2i+1}, has period that divides p_{2i+1}. Moreover, because ρ satisfies **P2** and $(L_{2i+2}, \ell_{2i+2}, L_{2i+3}, \ell_{2i+3})$ is an inversion, the output produced between the locations ℓ_{2i+2} and ℓ_{2i+3}, extended to the left with v_{2i+2} and to the right with v_{2i+3}, has period that divides both p_{2i+2} and p_{2i+3}. Note that this is not yet sufficient for applying Fine-Wilf's theorem, since the common factor $\mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+1}]\big)$ might be too short (possibly just equal to v_{2i+2}). The key argument here is to prove that the interval $[\ell_{2i+2}, \ell_{2i+1}]$ is covered by an inversion which is different from those that we considered above, namely, i.e. $(L_{2i+2}, \ell_{2i+2}, L_{2i+1}, \ell_{2i+1})$. For example, $[\ell_2, \ell_1]$ in Figure 3.6 is covered by the inversion $(L_2, \ell_2, L_1, \ell_1)$.

For this, we have to prove that the anchor points ℓ_{2i+2} and ℓ_{2i+1} are correctly ordered w.r.t. $\trianglelefteq$ and the ordering of positions (recall Definition 3.1.4). First, we observe that $\ell_{2i+2} \trianglelefteq \ell_{2i+1}$, since $(L_{2i}, \ell_{2i}, L_{2i+1}, \ell_{2i+1})$ and $(L_{2i+2}, \ell_{2i+2}, L_{2i+3}, \ell_{2i+3})$ are overlapping inversions. Next, we prove that the position of ℓ_{2i+1} is strictly to the left of the position of ℓ_{2i+2}. By way of contradiction, suppose that this is not the case, namely, $\ell_{2i+1} = (x_{2i+1}, y_{2i+1})$, $\ell_{2i+2} = (x_{2i+2}, y_{2i+2})$, and $x_{2i+1} \geqslant x_{2i+2}$. Because $(L_{2i}, \ell_{2i}, L_{2i+1}, \ell_{2i+1})$ and $(L_{2i+2}, \ell_{2i+2}, L_{2i+3}, \ell_{2i+3})$ are inversions, we know that ℓ_{2i+3} is strictly to the left of ℓ_{2i+2} and that ℓ_{2i+1} is strictly to the left of ℓ_{2i}. This implies that ℓ_{2i+3} is strictly to the left of ℓ_{2i}, and hence $(L_{2i}, \ell_{2i}, L_{2i+3}, \ell_{2i+3})$ is also an inversion. Moreover, recall that $\ell_{2i} \trianglelefteq \ell_{2i+2} \trianglelefteq \ell_{2i+1} \trianglelefteq \ell_{2i+3}$. This contradicts the maximality of $(L_{2i}, \ell_{2i}, L_{2i+1}, \ell_{2i+1})$, which we assumed at the beginning of the proof. Therefore, we must conclude that ℓ_{2i+1} is strictly to the left of ℓ_{2i+2}.

Now that we know that $\ell_{2i+2} \trianglelefteq \ell_{2i+1}$ and that ℓ_{2i+1} is to the left of ℓ_{2i+2}, we derive the existence of the inversion $(L_{2i+2}, \ell_{2i+2}, L_{2i+1}, \ell_{2i+1})$. Again, because ρ satisfies **P2**, we know that the word $v_{2i+2}\ \mathsf{out}(\rho[\ell_{2i+2}, \ell_{2i+1}])\ v_{2i+1}$ has period at most $\boldsymbol{B}$ that divides p_{2i+2} and p_{2i+1}. Summing up, we have:

1. $w_1 \ = \ \mathsf{out}\big(\rho[\ell_0, \ell_{2i+1}]\big)\ v_{2i+1}$ has period p_{2i+1},

2. $w_2 \ = \ v_{2i+2}\ \mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+1}]\big)\ v_{2i+1}$ has period $p = \gcd(p_{2i+2}, p_{2i+1})$,

3. $w_3 \ = \ v_{2i+2}\ \mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+3}]\big)\ v_{2i+3}$ has period $p' = \gcd(p_{2i+2}, p_{2i+3})$.

We are now ready to exploit our stronger variant of Fine-Wilf's theorem, that is, Theorem 3.1.6.

We begin with (1) and (2) above. Let $w = \mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+1}]\big)\ v_{2i+1}$ be the common suffix of w_1 and w_2. First note that since p divides p_{2i+2}, the word w is also a prefix of w_2, thus we can write $w_2 = w\, w_2'$. Second, note that the length of w is at least $|v_{2i+1}| = p_{2i+1} = p_{2i+1} + p - \gcd(p_{2i+1}, p)$. We can apply now Theorem 3.1.6 to $w_1 = w_1'\, w$ and $w_2 = w\, w_2'$ and obtain:

4. $w_4 \ = \ w_1'\, w\, w_2' \ = \ \mathsf{out}\big(\rho[\ell_0, \ell_{2i+2}]\big)\ v_{2i+2}\ \mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+1}]\big)\ v_{2i+1}$ has period p.

We apply next Theorem 3.1.6 to (2) and (3), namely, to the words w_2 and w_3 with v_{2i+2} as common factor. It is not difficult to check that $|v_{2i+2}| =$

$p_{2i+2} \geqslant p + p' - p''$ with $p'' = \gcd(p, p')$, using the definitions of p and p': we can write $p_{2i+2} = p''rq = p''r'q'$ with $p = p''r$ and $p' = p''r'$. It suffices to check that $p''rq + p''r'q' \geqslant 2(p''r + p''r' - p'')$, hence that $rq + r'q' \geqslant 2r + 2r' - 2$. This is clear if $\min(q, q') > 1$. Otherwise the inequality $p_{2i+2} \geqslant p + p' - p''$ follows easily because $p = p''$ or $p' = p''$ holds. Hence we obtain that w_2 and w_3 have both period p''.

Applying once more Theorem 3.1.6 to w_3 and w_4 with v_{2i+2} as common factor, yields period p'' for the word

$$w_5 \;=\; \mathsf{out}\big(\rho[\ell_0, \ell_{2i+2}]\big)\; v_{2i+2}\; \mathsf{out}\big(\rho[\ell_{2i+2}, \ell_{2i+3}]\big)\; v_{2i+3}$$

Finally, the periodicity is not affected when we remove factors of length multiple than the period. In particular, by removing the factor v_{2i+2} from w_5, we obtain the desired word $\mathsf{out}\big(\rho[\ell_0, \ell_{2i+3}]\big)\, v_{2i+3}$, whose period p'' divides p_{2i+3}. This proves the claim for the inductive step, and completes the proof of the proposition. □

The S^*-classes considered so far cannot be directly used to define the blocks in the desired decomposition of ρ, since the x-coordinates of their endpoints might not be in the appropriate order. The next definition takes care of this, by enlarging the S^*-classes according to x-coordinates of the anchor points in the equivalence class.

Definition 3.2.6. *Consider a non-singleton S^*-equivalence class $K = [\ell, \ell']$. Let $\mathsf{an}(K)$ be the restriction of K to the anchor points occurring in some inversion, and $X_{\mathsf{an}(K)} = \{x \,:\, \exists y\ (x, y) \in \mathsf{an}(K)\}$ be the projection of $\mathsf{an}(K)$ on positions. We define $\mathsf{block}(K) = [\tilde{\ell}, \tilde{\ell}']$, where*

- *$\tilde{\ell}$ is the latest location $(\tilde{x}, \tilde{y}) \trianglelefteq \ell$ such that $\tilde{x} = \min(X_{\mathsf{an}(K)})$,*
- *$\tilde{\ell}'$ is the earliest location $(\tilde{x}, \tilde{y}) \trianglerighteq \ell'$ such that $\tilde{x} = \max(X_{\mathsf{an}(K)})$*

(note that the locations $\tilde{\ell}, \tilde{\ell}'$ exist since ℓ, ℓ' are both anchor points in some inversion).

Lemma 3.2.7. *If K is a non-singleton S^*-equivalence class, then $\rho|\mathsf{block}(K)$ is a $\boldsymbol{B}$-block.*

Proof. Consider a non-singleton S^*-class $K = [\ell, \ell']$ and let $\mathsf{an}(K)$, $X_{\mathsf{an}(K)}$, and $\mathsf{block}(K) = [\tilde{\ell}, \tilde{\ell}']$ be as in Definition 3.2.6. We need to verify that $\rho[\tilde{\ell}, \tilde{\ell}']$ is a $\boldsymbol{B}$-block (cf. Definition 3.2.1), namely, that:

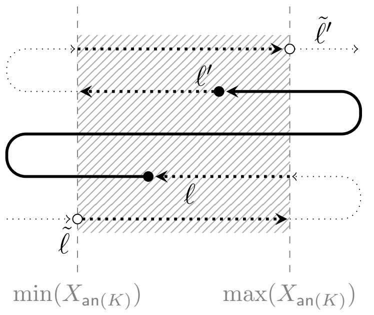


Figure 3.7: Block construction.

- $\tilde{\ell} = (\tilde{x}, \tilde{y})$, $\tilde{\ell}' = (\tilde{x}', \tilde{y}')$, with $\tilde{x} \leqslant \tilde{x}'$,
- the output produced by $\rho[\tilde{\ell}, \tilde{\ell}']$ is almost periodic with bound $2\boldsymbol{B}$,
- the output produced by the subsequence $\rho|Z$, where $Z = [\tilde{\ell}, \tilde{\ell}'] \setminus \big([\tilde{x}, \tilde{x}'] \times [\tilde{y}, \tilde{y}']\big)$, has length at most $2\boldsymbol{HB}$.

The first condition $\tilde{x} \leqslant \tilde{x}'$ follows immediately from the definition of $\tilde{x}$ and $\tilde{x}'$ as $\min(X_{\mathsf{an}(K)})$ and $\max(X_{\mathsf{an}(K)})$, respectively.

Next, we prove that the output produced by the factor $\rho[\tilde{\ell}, \tilde{\ell}']$ is almost periodic with bound $2\boldsymbol{B}$. By Definition 3.2.6, we have $\tilde{\ell} \trianglelefteq \ell \lhd \ell' \trianglelefteq \tilde{\ell}'$, and by Lemma 3.2.5 we know that $\mathsf{out}(\rho[\ell, \ell'])$ is periodic with period at most $\boldsymbol{B}$ ($\leqslant 2\boldsymbol{B}$). So it suffices to show that the lengths of the words $\mathsf{out}(\rho[\tilde{\ell}, \ell])$ and $\mathsf{out}(\rho[\ell', \tilde{\ell}'])$ are at most $2\boldsymbol{B}$. We shall focus on the former word, as the arguments for the latter are similar.

First, we note that the factor $\rho[\tilde{\ell}, \ell]$ lies entirely to the right of position $\tilde{x}$, and in particular, it starts at an even level $\tilde{y}$. This follows from the definition of $\tilde{\ell}$, and whether ℓ itself is at an odd/even level. In particular, the location ℓ is either at the same level as $\tilde{\ell}$, or just one level above.

Now, suppose, by way of contradiction, that $|\mathsf{out}(\rho[\tilde{\ell}, \ell])| > 2\boldsymbol{B}$. We head towards a contradiction by finding a location $\ell'' \lhd \ell$ that is S^*-equivalent to the first location ℓ of the S^*-equivalence class K. Since the location ℓ is either at the same level as $\tilde{\ell}$, or just above it, the factor $\rho[\tilde{\ell}, \ell]$ is of the form $\alpha\,\beta$, where α is rightward factor lying on the same level as $\tilde{\ell}$ and β is either empty or a leftward factor on the next level. Moreover, since $|\mathsf{out}(\rho[\tilde{\ell}, \ell])| > 2\boldsymbol{B}$, we know that either $|\mathsf{out}(\alpha)| > \boldsymbol{B}$ or $|\mathsf{out}(\beta)| > \boldsymbol{B}$. Thus, Lemma 3.1.3 says that one of the two factors α, β is not output-minimal. In particular, there is a loop L_1, strictly to the right of $\tilde{x}$, that intercepts a subfactor γ of $\rho[\tilde{\ell},\ell]$, with $\mathsf{out}(\gamma)$ non-empty and output-minimal.

Let ℓ'' be the first location of the factor γ. Clearly, ℓ'' is an anchor point of L and $\mathsf{out}(\mathsf{tr}(\ell'')) \neq \varepsilon$. Further recall that $\tilde{x} = \min(X_{\mathsf{an}(K)})$ is the leftmost position of locations in the class $K = [\ell, \ell']$ that are also anchor points of inversions. In particular, there is a loop L_2 with some anchor point $\ell''_2 = (\tilde{x}, y''_2) \in \mathsf{an}(K)$, and such that $\mathsf{tr}(\ell'')$ is non-empty and output-minimal. Since $\ell'' \lhd \ell \unlhd \ell''_2$ and the position of ℓ'' is to the right of the position of ℓ''_2, we know that $(L_1, \ell'', L_2, \ell''_2)$ is also an inversion, and hence $\ell''\ \mathsf{S}^*\ \ell''_2\ \mathsf{S}^*\ \ell$. But since $\ell'' \lhd \ell$, we get a contradiction with the assumption that ℓ is the first location of a S^*-class. In this way we have shown that $|\mathsf{out}(\rho[\ell_1,\ell])| \leqslant 2\boldsymbol{B}$.

It remains to show that the output produced by the subsequence $\rho|Z$, where $Z = [\tilde{\ell},\tilde{\ell}'] \setminus \big([\tilde{x},\tilde{x}'] \times [\tilde{y},\tilde{y}']\big)$, has length at most $2\boldsymbol{HB}$. For this it suffices to prove that $|\mathsf{out}(\alpha)| \leqslant \boldsymbol{B}$ for every factor α of $\rho[\tilde{\ell},\tilde{\ell}']$ that lies at a single level and either to the left of $\tilde{x}$ or to the right of $\tilde{x}'$. By symmetry, we consider only one of the two types of factors. Suppose, by way of contradiction, that there is a factor α at level y'', to the left of $\tilde{x}$, and such that $|\mathsf{out}(\alpha)| > \boldsymbol{B}$. By Lemma 3.1.3 we know that α is not output-minimal, so there is some loop L_2 strictly to the left of $\tilde{x}$ that intercepts an output-minimal subfactor β of α with non-empty output. Let ℓ'' be the first location of β. We know that $\tilde{\ell} \lhd \ell'' \unlhd \tilde{\ell}'$. Since the level $\tilde{y}$ is even, this means that the level of ℓ'' is strictly greater than $\tilde{y}$. Since we also know that ℓ is an anchor point of some inversion, we can take a suitable loop L_1 with anchor point ℓ and obtain that (L_1, ℓ, L_2, ℓ'') is an inversion, so $\ell''\ \mathsf{S}^*\ \ell$. But this contradicts the fact that $\tilde{x}$ is the leftmost position of $\mathsf{an}(K)$. We thus conclude that $|\mathsf{out}(\alpha)| \leqslant \boldsymbol{B}$, and this completes the proof that $\rho|\mathsf{block}(K)$ is a $\boldsymbol{B}$-block. □

The next lemma shows that blocks do not overlap along the input axis:

Lemma 3.2.8. *Suppose that K_1 and K_2 are two different non-singleton S^*-classes such that $\ell \lhd \ell'$ for all $\ell \in K_1$ and $\ell' \in K_2$. Let $\mathsf{block}(K_1) = [\ell_1, \ell_2]$ and $\mathsf{block}(K_2) = [\ell_3, \ell_4]$, with $\ell_2 = (x_2, y_2)$ and $\ell_3 = (x_3, y_3)$. Then $x_2 < x_3$.*

Proof. Suppose by contradiction that K_1 and K_2 are as in the statement, but $x_2 \geqslant x_3$. By Definition 3.2.6, $x_2 = \max(X_{\mathsf{an}(K_1)})$ and $x_3 = \min(X_{\mathsf{an}(K_2)})$. This implies the existence of some inversions (L, ℓ, L', ℓ') and $(L'', \ell'', L''', \ell''')$ such that $\ell = (x_2, y)$ and $\ell''' = (x_3, y''')$. Moreover, since $\ell \unlhd \ell'''$ and $x_2 \geqslant x_3$, we know that (L, ℓ, L''', ℓ''') is also an inversion, thus implying that $K_1 = K_2$. $\square$

For the sake of brevity, we call S^**-block* any factor of the form $\rho|\mathsf{block}(K)$ that is obtained by applying Definition 3.2.6 to a non-singleton S^*-class K. The results obtained so far imply that every location covered by an inversion is also covered by an S^*-block (Lemma 3.2.7), and that the order of occurrence of S^*-blocks is the same as the order of positions (Lemma 3.2.8). So the S^*-blocks can be used as factors for the $\boldsymbol{B}$-decomposition of ρ we are looking for. Below, we show that the remaining factors of ρ, which do not overlap the S^*-blocks, are $\boldsymbol{B}$-diagonals. This will complete the construction of a $\boldsymbol{B}$-decomposition of ρ.

Formally, we say that a factor $\rho[\ell, \ell']$ *overlaps* another factor $\rho[\ell'', \ell''']$ if $[\ell, \ell'] \cap [\ell'', \ell'''] \neq \emptyset$, $\ell' \neq \ell''$, and $\ell \neq \ell'''$.

Lemma 3.2.9. *Let $\rho[\ell, \ell']$ be a factor of ρ, with $\ell = (x, y)$, $\ell' = (x', y')$, and $x \leqslant x'$, that does not overlap any S^*-block. Then $\rho[\ell, \ell']$ is a $\boldsymbol{B}$-diagonal.*

Proof. Consider a factor $\rho[\ell, \ell']$, with $\ell = (x, y)$, $\ell' = (x', y')$, and $x \leqslant x'$, that does not overlap any S^*-block. We will focus on locations ℓ'' with $\ell \trianglelefteq \ell'' \trianglelefteq \ell'$ that are anchor points of some loop with $\mathsf{out}(\mathsf{tr}(\ell'')) \neq \varepsilon$. We denote by A the set of all such locations.

First, we show that the locations in A are monotonic w.r.t. the position order. Formally, we prove that for all $\ell_1, \ell_2 \in A$, if $\ell_1 = (x_1, y_1) \trianglelefteq \ell_2 = (x_2, y_2)$, then $x_1 \leqslant x_2$. Suppose that this were not the case, namely, that A contained two anchor points $\ell_1 = (x_1, y_1)$ and $\ell_2 = (x_2, y_2)$ with $\ell_1 \triangleleft \ell_2$ and $x_1 > x_2$. Let L_1, L_2 be the loops of ℓ_1, ℓ_2, respectively, and recall that $\mathsf{out}(\mathsf{tr}(\ell_1)), \mathsf{out}(\mathsf{tr}(\ell_2)) \neq \varepsilon$. This means that $(L_1, \ell_1, L_2, \ell_2)$ is an inversion, and hence $\ell_1 \; \mathsf{S}^* \; \ell_2$. But this contradicts the hypothesis that $\rho[\ell, \ell']$ does not overlap any S^*-block.

Next, we identify the floors of our diagonal. Let $y_0, y_1, \ldots, y_{n-1}$ be all the *even* levels that have locations in A. For each $i = 0, \ldots, n-1$, let ℓ_{2i+1} (resp. ℓ_{2i+2}) be the first (resp. last) anchor point of A at level y_i. Further let $\ell_0 = \ell$ and $\ell_{2n+1} = \ell'$. Clearly, each factor $\rho[\ell_{2i+1}, \ell_{2i+2}]$ is a floor. Moreover, thanks to the previous arguments, each location ℓ_{2i} is to the left of the location ℓ_{2i+1}.

It remains to prove that each factor $\rho[\ell_{2i}, \ell_{2i+1}]$ produces an output of length at most $2\boldsymbol{HB}$. By construction, A contains no anchor point at an even level and strictly between ℓ_{2i} and ℓ_{2i+1}. By Lemma 3.1.3 this means that the outputs produced by subfactors of $\rho[\ell_{2i}, \ell_{2i+1}]$ that lie entirely at an *even* level have length at most $\boldsymbol{B}$. Let us now consider the subfactors α of $\rho[\ell_{2i}, \ell_{2i+1}]$ that lie entirely at an *odd* level, and let us prove that they produce outputs of length at most $2\boldsymbol{B}$. Suppose that this is not the case, namely, that $|\mathsf{out}(\alpha)| > 2\boldsymbol{B}$. In this case we show that an inversion would exist at this level. Formally, we can find two locations $\ell'' \lhd \ell'''$ in α such that the prefix of α that ends at location ℓ'' and the suffix of α that starts at location ℓ''' produce outputs of length greater than $\boldsymbol{B}$. By Lemma 3.1.3, those two factors would not be output-minimal, and hence α would contain disjoint loops L_1, L_2 with anchor points ℓ''_1, ℓ''_2 forming an inversion $(L_1, \ell''_1, L_2, \ell''_2)$. But this would imply that ℓ''_1, ℓ''_2 belong to the same non-singleton S^*-equivalence class, which contradicts the hypothesis that $\rho[\ell, \ell']$ does not overlap any S^*-block. We must conclude that the subfactors of $\rho[\ell_{2i}, \ell_{2i+1}]$ produce outputs of length at most $2\boldsymbol{B}$.

Overall, this shows that the output produced by each factor $\rho[\ell_{2i}, \ell_{2i+1}]$ has length at most $2\boldsymbol{HB}$. □

We have just shown how to construct a $\boldsymbol{B}$-decomposition of the run ρ that satisfies **P2**. In particular, this proves Proposition 3.2.3, as well as the implication **P2** ⇒ **P3** of Theorem 3.0.3.

3.2.3 From existence of decompositions to an equivalent one-way transducer.

We now focus on the last implication **P3** ⇒ **P1** of Theorem 3.0.3. More precisely, we show how to construct a one-way transducer $\mathcal{T}'$ that simulates the outputs produced by the successful runs of $\mathcal{T}$ that admit $\boldsymbol{B}$-decompositions. In particular, $\mathcal{T}'$ turns out to be equivalent to $\mathcal{T}$ when $\mathcal{T}$ is one-way definable. Here we will only give a proof sketch of this construction assuming that $\mathcal{T}$ is a sweeping transducer; a fully detailed construction of $\mathcal{T}'$ from an arbitrary two-way transducer $\mathcal{T}$ will be given in Section 3.5 (Proposition 3.5.6), together with a procedure for deciding one-way definability of $\mathcal{T}$ (Proposition 3.6.2).

Proposition 3.2.10. *Given a functional sweeping transducer $\mathcal{T}$ a one-way transducer $\mathcal{T}'$ can be constructed in* 2EXPTIME *such that the following hold:*

1. $\mathcal{T}' \subseteq \mathcal{T}$, *2.* $\mathsf{dom}(\mathcal{T}')$ *contains all words that induce successful runs of* $\mathcal{T}$ *admitting* $\boldsymbol{B}$*-decompositions.*

In particular, $\mathcal{T}'$ *is equivalent to* $\mathcal{T}$ *iff* $\mathcal{T}$ *is one-way definable.*

Proof sketch. Given an input word u, the one-way transducer $\mathcal{T}'$ needs to guess a successful run ρ of $\mathcal{T}$ on u that admits a $\boldsymbol{B}$-decomposition. This can be done by guessing the crossing sequences of ρ at each position, together with a sequence of locations ℓ_i that identify the factors of a $\boldsymbol{B}$-decomposition of ρ. To check the correctness of the decomposition, $\mathcal{T}'$ also needs to guess a bounded amount of information (words of bounded length) to reconstruct the outputs produced by the $\boldsymbol{B}$-diagonals and the $\boldsymbol{B}$-blocks. For example, while scanning a factor of the input underlying a diagonal, $\mathcal{T}'$ can easily reproduce the outputs of the floors and the guessed outputs of factors between them. In a similar way, while scanning a factor of the input underlying a block, $\mathcal{T}'$ can simulate the almost periodic output by guessing its repeating pattern and the bounded prefix and suffix of it, and by emitting the correct amount of letters, as it is done in the second item of Example 10. In particular, one can verify that the capacity of $\mathcal{T}'$ is linear in $\boldsymbol{HB}$. Moreover, because the guessed objects are of size linear in $\boldsymbol{HB}$ and $\boldsymbol{HB}$ is a simple exponential in the size of $\mathcal{T}$, the one-way transducer $\mathcal{T}'$ has doubly exponential size in that of $\mathcal{T}$. □

We can now focus on the two-way case, as we have proven Theorem 3.0.3 for sweeping transducers.

3.3 The structure of two-way loops

Whereas the pumping of the loops of a sweeping transducer is rather simple, we need a much deeper understanding when it comes to pumping loops of runs of two-way transducers. This section is precisely devoted to untangling the structure of two-way loops. We will focus on specific types of loops, called idempotent loops, that when pumped generate repetitions with a "nice shape", very similar to the pumped runs of a sweeping transducer.

We fix throughout this section a functional two-way transducer $\mathcal{T}$, an input word u, and a (normalized) successful run ρ of $\mathcal{T}$ on u. As usual, $\boldsymbol{H} = 2|Q| - 1$ is the maximal length of a crossing sequence of ρ, and c_{max} is the maximal number of letters output by a single transition.

3.3.1 Flows and effects.

We start by analyzing the shape of factors of ρ intercepted by an interval $I = [x_1, x_2]$. We identify four types of factors α intercepted by I depending on the first location (x, y) and the last location (x', y'):

- α is an LL-factor if $x = x' = x_1$,
- α is an RR-factor if $x = x' = x_2$,
- α is an LR-factor if $x = x_1$ and $x' = x_2$,
- α is an RL-factor if $x = x_2$ and $x' = x_1$.

In Figure 3.1 we see that α is an LL-factor, β, δ are LR-factors, ζ is an RR-factor, and γ is an RL-factor.

Definition 3.3.1. *Let $I = [x_1, x_2]$ be an interval of ρ and h_i the length of the crossing sequence $\rho|x_i$, for both $i = 1$ and $i = 2$.*

The flow *F_I of I is the directed graph with set of nodes $\{0, \ldots, \max(h_1, h_2) - 1\}$ and set of edges consisting of all (y, y') such that there is a factor of ρ intercepted by I that starts at location (x_i, y) and ends at location (x_j, y'), for $i, j \in \{1, 2\}$.*

The effect *E_I of I is the triple (F_I, c_1, c_2), where $c_i = \rho|x_i$ is the crossing sequence at x_i.*

For example, the interval I of Figure 3.1 has the flow graph $0 \mapsto 1 \mapsto 3 \mapsto 4 \mapsto 2 \mapsto 0$.

It is easy to see that every node of a flow F_I has at most one incoming and at most one outgoing edge. More precisely, if $y < h_1$ is even, then it has one outgoing edge (corresponding to an LR- or LL-factor intercepted by I), and if it is odd it has one incoming edge (corresponding to an RL- or LL-factor intercepted by I). Similarly, if $y < h_2$ is even, then it has one incoming edge (corresponding to an LR- or RR-factor), and if it is odd it has one outgoing edge (corresponding to an RL- or RR-factor).

In the following we consider effects that are not necessarily associated with intervals of specific runs. The definition of such effects should be clear: they are triples consisting of a graph (called flow) and two crossing sequences of lengths $h_1, h_2 \leqslant \boldsymbol{H}$, with sets of nodes of the form $\{0, \ldots, \max(h_1, h_2) - 1\}$, that satisfy the in/out-degree properties stated above. It is convenient to distinguish the edges in a flow based on the parity of the source and target nodes. Formally, we partition any flow F into the following subgraphs:

- F_{LR} consists of all edges of F between pairs of even nodes,
- F_{RL} consists of all edges of F between pairs of odd nodes,
- F_{LL} consists of all edges of F from an even node to an odd node,
- F_{RR} consists of all edges of F from an odd node to an even node.

We denote by $\mathcal{F}$ (resp. $\mathcal{E}$) the set of all flows (resp. effects) augmented with a dummy element $\perp$. We equip both sets $\mathcal{F}$ and $\mathcal{E}$ with a semigroup structure, where the corresponding products $\circ$ and $\odot$ are defined below (similar definitions appear in [14]). Later we will use the semigroup structure to identify the *idempotent loops*, that play a crucial role in our characterization of one-way definability.

Definition 3.3.2. *For two graphs G, G', we denote by $G \cdot G'$ the graph with edges of the form (y, y'') such that (y, y') is an edge of G and (y', y'') is an edge of G', for some node y' that belongs to both G and G'. Similarly, we denote by G^* the graph with edges (y, y') such that there exists a (possibly empty) path in G from y to y'.*

The product of two flows F, F' is the unique flow $F \circ F'$ (if it exists) such that:

- $(F \circ F')_{\mathsf{LR}} = F_{\mathsf{LR}} \cdot (F'_{\mathsf{LL}} \cdot F_{\mathsf{RR}})^* \cdot F'_{\mathsf{LR}}$,
- $(F \circ F')_{\mathsf{RL}} = F'_{\mathsf{RL}} \cdot (F_{\mathsf{RR}} \cdot F'_{\mathsf{LL}})^* \cdot F_{\mathsf{RL}}$,
- $(F \circ F')_{\mathsf{LL}} = F_{\mathsf{LL}} \ \cup \ F_{\mathsf{LR}} \cdot (F'_{\mathsf{LL}} \cdot F_{\mathsf{RR}})^* \cdot F'_{\mathsf{LL}} \cdot F_{\mathsf{RL}}$,
- $(F \circ F')_{\mathsf{RR}} = F'_{\mathsf{RR}} \ \cup \ F'_{\mathsf{RL}} \cdot (F_{\mathsf{RR}} \cdot F'_{\mathsf{LL}})^* \cdot F_{\mathsf{RR}} \cdot F'_{\mathsf{LR}}$.

If no flow $F \circ F'$ exists with the above properties, then we let $F \circ F' = \perp$.

The product of two effects $E = (F, c_1, c_2)$ and $E' = (F', c'_1, c'_2)$ is either the effect $E \odot E' = (F \circ F', c_1, c'_2)$ or the dummy element $\perp$, depending on whether $F \circ F' \neq \perp$ and $c_2 = c'_1$.

For example, let F be the flow of interval I in Figure 3.1. Then $(F \circ F)_{\mathsf{LL}} = \{0 \mapsto 1, 2 \mapsto 3\}$, $(F \circ F)_{\mathsf{RR}} = \{1 \mapsto 2, 3 \mapsto 4\}$, and $(F \circ F)_{\mathsf{LR}} = \{4 \mapsto 0\}$ — one can quickly verify this with the help of Figure 3.8.

It is also easy to see that $(\mathcal{F}, \circ)$ and $(\mathcal{E}, \odot)$ are finite semigroups, and that for every run ρ and every pair of consecutive intervals $I = [x_1, x_2]$ and

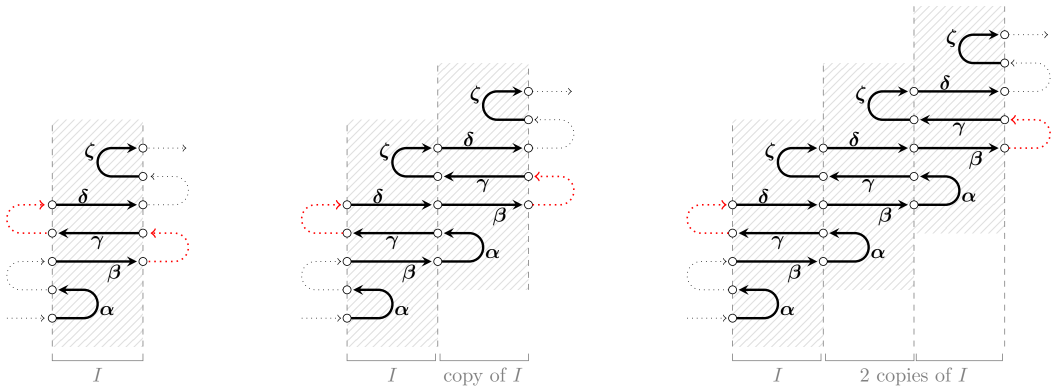


Figure 3.8: Pumping a loop in a two-way run.

$J = [x_2, x_3]$ of ρ, $F_{I \cup J} = F_I \circ F_J$ and $E_{I \cup J} = E_I \odot E_J$. In particular, the function E that associates each interval I of ρ with the corresponding effect E_I can be seen as a semigroup homomorphism.

3.3.2 Loops and components.

Recall that a loop is an interval $L = [x_1, x_2]$ with the same crossing sequences at x_1 and x_2. We will follow techniques similar to those presented in Section 3.1 to show that the outputs generated in non left-to-right manner are essentially periodic. However, differently from the sweeping case, we will consider only special types of loops:

Definition 3.3.3. *A loop L is* idempotent *if $E_L = E_L \odot E_L$ and $E_L \neq \perp$.*

For example, the interval I of Figure 3.1 is a loop, if one assumes that the crossing sequences at the borders of I are the same. By comparing with Figure 3.8, it is easy to see that I is not idempotent. On the other hand, the loop consisting of 2 copies of I is idempotent.

As usual, given a loop $L = [x_1, x_2]$ and a number $n \in \mathbb{N}$, we can introduce n new copies of L and connect the intercepted factors in the obvious way. This results in a new run $\mathsf{pump}_L^{n+1}(\rho)$ on the word $\mathsf{pump}_L^{n+1}(u)$. Figure 3.8 shows how to do this for $n = 1$ and $n = 2$. Below, we analyze in detail the shape of the pumped run $\mathsf{pump}_L^{n+1}(\rho)$ (and the produced output as well) when L is an *idempotent* loop. We will focus on idempotent loops because pumping non-idempotent loops may induce permutations of factors that are difficult to handle. For example, if we consider again the non-idempotent loop I to the left of Figure 3.8, the factor of the run between β and γ (to the right of I, highlighted in red) precedes the factor between γ and δ (to the left of I, again in red), but this ordering is reversed when a new copy of I is added.

When pumping a loop L, subsets of factors intercepted by L are glued together to form factors intercepted by the replication of L. The notion of component introduced below identifies groups of factors that are glued together.

Definition 3.3.4. *A* component *of a loop L is any strongly connected component of its flow F_L (note that this is also a cycle, since every node in it has in/out-degree 1).*

Given a component C, we denote by $\min(C)$ (resp. $\max(C)$) the minimum (resp. maximum) node in C. We say that C is left-to-right *(resp.* right-to-left*) if $\min(C)$ is even (resp., odd).*

An (L, C)-factor *is a factor of the run that is intercepted by L and that corresponds to an edge of C.*

We will usually list the (L, C)-factors based on their order of occurrence in the run. For example, the loop I of Figure 3.8 contains a single component $C = 0 \mapsto 1 \mapsto 3 \mapsto 4 \mapsto 2 \mapsto 0$ which is left-to-right. Another example is given in Figure 3.9, where the loop L has three components C_1, C_2, C_3 (colored in blue, red, and green, respectively): $\alpha_1, \alpha_2, \alpha_3$ are the (L, C_1)-factors, $\beta_1, \beta_2, \beta_3$ are the (L, C_2)-factors, and γ_1 is the unique (L, C_3)-factor.

Below, we show that the levels of each component of a loop (not necessarily idempotent) form an interval.

Lemma 3.3.5. *Let C be a component of a loop $L = [x_1, x_2]$. The nodes of C are precisely the levels in the interval $[\min(C), \max(C)]$. Moreover, if C is left-to-right (resp. right-to-left), then $\max(C)$ is the smallest level $\geqslant \min(C)$ such that between $(x_1, \min(C))$ and $(x_2, \max(C))$ (resp. $(x_2, \min(C))$ and $(x_1, \max(C))$) there are equally many LL-factors and RR-factors intercepted by L.*

Proof. To ease the understanding the reader may refer to Figure 3.10, that shows some factors intercepted by L and the corresponding edges in the flow.

We begin the proof by partitioning the set of levels of the flow into suitable intervals as follows. We observe that every loop $L = [x_1, x_2]$ intercepts

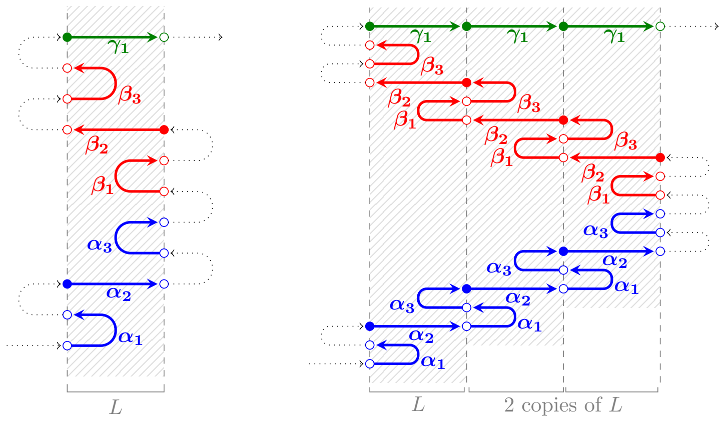


Figure 3.9: Pumping an idempotent loop with three components.

equally many LL-factors and RR-factors. This is so because the crossing sequences at x_1, x_2 have the same length h. We also observe that the sources of the factors intercepted by L are either of the form (x_1, y), with y even, or (x_2, y), with y odd. For any location $\ell \in \{x_1, x_2\} \times \mathbb{N}$ that is the source of an intercepted factor, we define d_ℓ to be the difference between the number of LL-factors and the number of RR-factors intercepted by L that *end* at a location *strictly before* ℓ. Intuitively, $d_\ell = 0$ when the prefix of the run up to

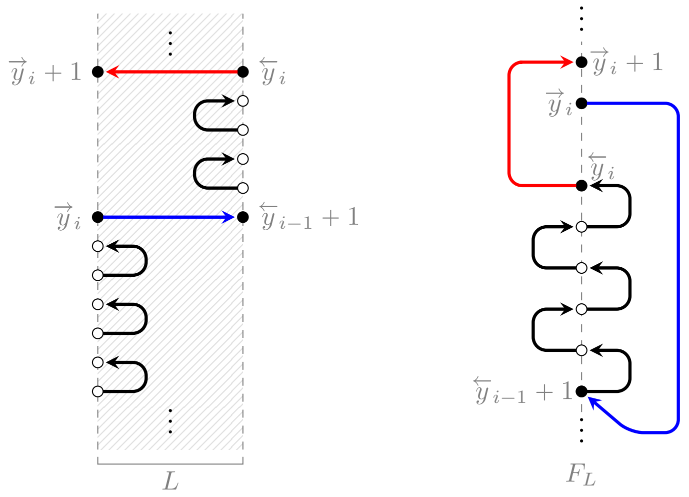

$\overrightarrow{y}_i + 1$
$\overleftarrow{y}_i$
$\overrightarrow{y}_i$
$\overleftarrow{y}_{i-1} + 1$
L
$\overrightarrow{y}_i + 1$
$\overrightarrow{y}_i$
$\overleftarrow{y}_i$
$\overleftarrow{y}_{i-1} + 1$
F_L

Figure 3.10: Some factors intercepted by L and the corresponding edges in the flow.

location ℓ has visited equally many times the position x_1 and the position x_2. For the sake of brevity, we let $d_y = d_{(x_1,y)}$ for an even level y, and $d_y = d_{(x_2,y)}$ for an odd level y. Note that $d_0 = 0$. We also let $d_{h+1} = 0$, by convention.

We now consider the numbers z's, with $0 \leqslant z \leqslant h+1$, such that $d_z = 0$, that is: $0 = z_0 < z_1 < \cdots < z_k = h+1$. Using a simple induction, we prove that for all $i \leqslant k$, the parity of z_i is the same as the parity of its index i. The base case $i = 0$ is trivial, since $z_0 = 0$. For the inductive case, suppose that z_i is even (the case of z_i odd is similar). We prove that z_{i+1} is odd by a case distinction based on the type of factor intercepted by L that starts at level z_i. If this factor is an LR-factor, then it ends at the same level z_i, and hence $d_{z_i+1} = d_{z_i} = 0$, which implies that $z_{i+1} = z_i + 1$ is odd. Otherwise, if the factor is an LL-factor, then for all levels z strictly between z_i and z_{i+1}, we have $d_z > 0$, and since $d_{z_{i+1}} = 0$, the last factor before z_{i+1} must decrease d_z, that is, must be an RR-factor. This implies that (x_2, z_{i+1}) is the source of an intercepted factor, and thus z_{i+1} is odd.

The levels $0 = z_0 < z_1 < \cdots < z_k = h+1$ induce a partition of the set of nodes of the flow into intervals of the form $Z_i = [z_i, z_{i+1} - 1]$. To prove the lemma, it is suffices to show that the subgraph of the flow induced by each interval Z_i is connected. Indeed, because the union of the previous intervals covers all the nodes of the flow, and because each node has one incoming and one outgoing edge, this will imply that the intervals coincide with the components of the flow.

Now, let us fix an interval of the partition, which we denote by Z to avoid clumsy notation. Hereafter, we will focus on the edges of subgraph of the flow induced by Z (we call it *subgraph of* Z for short). We prove a few basic properties of these edges. For the sake of brevity, we call LL-edges the edges of the subgraph of Z that correspond to the LL-factors intercepted by L, and similarly for the RR-edges, LR-edges, and RL-edges.

We make a series of assumption to simplify our reasoning. First, we assume that the edges are ordered based on the occurrences of the corresponding factors in the run. For instance, we may say the first, second, etc. LR-edge (of the subgraph of Z) — from now on, we tacitly assume that the edges are inside the subgraph of Z. Second, we assume that the first edge of the subgraph of Z starts at an even node, namely, it is an LL-edge or an LR-edge (if this were not the case, one could apply symmetric arguments to prove the lemma). From this it follows that the subgraph contains n LR-edges interleaved by $n-1$ RL-edges, for some $n > 0$. Third, we assume that

$\min(Z) = 0$, in order to avoid clumsy notations (otherwise, we need to add $\min(Z)$ to all the levels considered hereafter).

Now, we observe that, by definition of Z, there are equally many LL-edges and RR-edges: indeed, the difference between the number of LL-edges and the number of RR-edges at the beginning and at the end of Z is the same, namely, $d_z = 0$ for both $z = \min(Z)$ and $z = \max(Z)$. It is also easy to see that the LL-edges and the RR-edges are all of the form $y \to y + 1$, for some level y. We call these edges *incremental edges*.

For the other edges, we denote by $\overrightarrow{y}_i$ (resp. $\overleftarrow{y}_i$) the source level of the i-th LR-edge (resp. the i-th RL-edge). Clearly, each $\overrightarrow{y}_i$ is even, and each $\overleftarrow{y}_i$ is odd, and $i \leqslant j$ implies $\overrightarrow{y}_i < \overrightarrow{y}_j$ and $\overleftarrow{y}_i < \overleftarrow{y}_j$. Consider the location $(x_1, \overrightarrow{y}_i)$, which is the source of the i-th LR-edge (e.g. the edge in blue in the figure). The latest location at position x_2 that precedes $(x_1, \overrightarrow{y}_i)$ must be of the form $(x_2, \overleftarrow{y}_{i-1})$, provided that $i > 1$. This implies that, for all $1 < i \leqslant n$, the i-th LR-edge is of the form $\overrightarrow{y}_i \to \overleftarrow{y}_{i-1} + 1$. For $i = 1$, we recall that $\min(Z) = 0$ and observe that the first location at position x_2 that occurs after the location $(x_1, 0)$ is $(x_2, 0)$, and thus the first LR-edge has a similar form: $\overrightarrow{y}_1 \to \overleftarrow{y}_0 + 1$, where $\overleftarrow{y}_0 = -1$ by convention.

Using symmetric arguments, we see that the i-th RL-edge (e.g. the one in red in the figure) is of the form $\overleftarrow{y}_i \to \overrightarrow{y}_i + 1$. In particular, the last LR-edge starts at the level $\overrightarrow{y}_n = \max(Z)$.

Summing up, we have just seen that the edges of the subgraph of Z are of the following forms:

- $y \to y + 1$ (incremental edges),
- $\overrightarrow{y}_i \to \overleftarrow{y}_{i-1} + 1$ (i-th LR-edge, for $i = 1, \dots, n$),
- $\overleftarrow{y}_i \to \overrightarrow{y}_i + 1$ (i-th RL-edge, for $i = 1, \dots, n - 1$).

In addition, we have $\overrightarrow{y}_i + 1 = \overleftarrow{y}_i + 2d_{\overleftarrow{y}_i}$. Since $d_z > 0$ for all $\min(Z) < z < \max(Z)$, this implies that $\overrightarrow{y}_i > \overleftarrow{y}_i$.

The goal is to prove that the subgraph of Z is strongly connected, namely, it contains a cycle that visits all its nodes. As a matter of fact, because components are also strongly connected subgraphs, and because every node in the flow has in-/out-degree 1, this will imply that the considered subgraph coincides with a component C, thus implying that the nodes in C form an interval. Towards this goal, we will prove a series of claims that aim at identifying suitable sets of nodes that are covered by paths in the subgraph of Z. Formally, we say that a path *covers* a set Y if it visits all the nodes in Y, and possibly other nodes. As usual, when we talk of edges or paths, we tacitly understand that they occur inside the subgraph of Z. On the other hand, we do not need to assume $Y \subseteq Z$, since this would follow from the fact that Y is covered by a path inside Z. For example, the right hand-side of Figure 3.10 shows a path from $\overrightarrow{y}_i$ to $\overrightarrow{y}_i + 1$ that covers the set $Y = \{\overrightarrow{y}_i, \overrightarrow{y}_i + 1\} \cup [\overleftarrow{y}_{i-1} + 1, \overleftarrow{y}_i]$.

The covered sets will be intervals of the form

$$Y_i \;=\; [\overleftarrow{y}_{i-1} + 1, \overleftarrow{y}_i].$$

Note that the sets Y_i are well-defined for all $i = 1, \ldots, n-1$, but not for $i = n$ since $\overleftarrow{y}_n$ is not defined either (the subgraph of Z contains only $n-1$ RL-edges).

Claim. *For all $i = 1, \ldots, n-1$, there is a path from $\overrightarrow{y}_i$ to $\overrightarrow{y}_i + 1$ that covers Y_i (for short, we call it an* incremental path*).*

Proof. We prove the claim by induction on i. The base case $i = 1$ is rather easy. Indeed, we recall the convention that $\overleftarrow{y}_0 + 1 = \min(Z) = 0$. In particular, the node $\overleftarrow{y}_0 + 1$ is the target of the first LR-edge of the subgraph of Z. Before this edge, according to the order induced by the run, we can only have LL-edges of the form $y \to y+1$, with $y = 0, 2, \ldots, \overrightarrow{y}_1 - 2$. Similarly, after the LR-edge we have RR-edges of the form $y \to y+1$, with $y = 1, 3, \ldots, \overleftarrow{y}_1 - 2$. Those incremental edges can be connected to form the path $\overleftarrow{y}_{i-1} + 1 \to^* \overleftarrow{y}_1$ that covers the interval $[\overleftarrow{y}_0 + 1, \overleftarrow{y}_1]$. By prepending to this path the LR-edge $\overrightarrow{y}_1 \to \overleftarrow{y}_0 + 1$, and by appending the RL-edge $\overleftarrow{y}_1 \to \overrightarrow{y}_1 + 1$, we get a path from $\overrightarrow{y}_1$ to $\overrightarrow{y}_1 + 1$ that covers the interval $[\overleftarrow{y}_0 + 1, \overleftarrow{y}_1]$. The latter interval is precisely the set Y_1.

For the inductive step, we fix $1 < i < n$ and we construct the desired path from $\overrightarrow{y}_i$ to $\overrightarrow{y}_i + 1$. The initial edge of this path is defined to be the LR-edge $\overrightarrow{y}_i \to \overleftarrow{y}_{i-1} + 1$. Similarly, the final edge of the path will be the RL-edge $\overleftarrow{y}_i \to \overrightarrow{y}_i + 1$, which exists since $i < n$. It remains to connect $\overleftarrow{y}_{i-1} + 1$ to $\overleftarrow{y}_i$. For this, we consider the edges that depart from nodes strictly between $\overleftarrow{y}_{i-1}$ and $\overleftarrow{y}_i$.

Let y be an arbitrary node in $[\overleftarrow{y}_{i-1} + 1, \overleftarrow{y}_i - 1]$. Clearly, y cannot be of the form $\overleftarrow{y}_j$, for some j, because it is strictly between $\overleftarrow{y}_{i-1}$ and $\overleftarrow{y}_i$. So y cannot be the source of an RL-edge. Moreover, recall that the LL-edges and the RR-edges are the of the form $y \to y + 1$. As these incremental edges do not pose particular problems for the construction of the path, we focus mainly on the LR-edges that depart from nodes inside $[\overleftarrow{y}_{i-1}+1, \overleftarrow{y}_i - 1]$.

Let $\overrightarrow{y}_j \rightarrow \overleftarrow{y}_{j-1}+1$ be such an LR-edge, for some j such that $\overrightarrow{y}_j \in [\overleftarrow{y}_{i-1}+1, \overleftarrow{y}_i - 1]$. If we had $j \geqslant i$, then we would have $\overrightarrow{y}_j \geqslant \overrightarrow{y}_i > \overleftarrow{y}_i$, but this would contradict the assumption that $\overrightarrow{y}_j \in [\overleftarrow{y}_{i-1}+1, \overleftarrow{y}_i - 1]$. So we know that $j < i$. This enables the use of the inductive hypothesis, which implies the existence of an incremental path from $\overrightarrow{y}_j$ to $\overrightarrow{y}_j + 1$ that covers the interval Y_j.

Finally, by connecting the above paths using the incremental edges, and by adding the initial and final edges $\overrightarrow{y}_i \rightarrow \overleftarrow{y}_{i-1}+1$ and $\overleftarrow{y}_i \rightarrow \overrightarrow{y}_i + 1$, we obtain a path from $\overrightarrow{y}_i$ to $\overrightarrow{y}_i + 1$. It is easy to see that this path covers the interval Y_i. □

Next, we define

$$Y \;=\; [\overleftarrow{y}_{n-1}+1, \overrightarrow{y}_n] \;\cup \bigcup_{1 \leqslant i < n} Y_i.$$

We prove a claim similar to the previous one, but now aiming to cover Y with a *cycle*. Towards the end of the proof we will argue that the set Y coincides with the full interval Z, thus showing that there is a component C whose set of notes is precisely Z.

Claim. *There is a cycle that covers Y.*

Proof. It is convenient to construct our cycle starting from the last LR-edge, that is, $\overrightarrow{y}_n \rightarrow \overleftarrow{y}_{n-1}+1$, since this will cover the upper node $\overrightarrow{y}_n = \max(Z)$. From there we continue to add edges and incremental paths, following an approach similar to the proof of the previous claim, until we reach the node $\overrightarrow{y}_n$ again. More precisely, we consider the edges that depart from nodes strictly between $\overleftarrow{y}_{n-1}$ and $\overrightarrow{y}_n$. As there are only $n-1$ RL-edges, we know that every node in the interval $[\overleftarrow{y}_{n-1}+1, \overrightarrow{y}_n - 1]$ must be source of an LL-edge, an RR-edge, or an LR-edge. As usual, incremental edges do not pose particular problems for the construction of the cycle, so we focus on the LR-edges. Let $\overrightarrow{y}_i \rightarrow \overleftarrow{y}_{i-1}+1$ be such an LR-edge, with $\overrightarrow{y}_i \in [\overleftarrow{y}_{n-1}+1, \overrightarrow{y}_n - 1]$. Since $i < n$, we know from the previous claim that there is a path from $\overrightarrow{y}_i$ to $\overrightarrow{y}_i + 1$ that covers Y_i. We can thus build a cycle π by connecting the above paths using the incremental edges and the LR-edge $\overrightarrow{y}_n \rightarrow \overleftarrow{y}_{n-1}+1$.

By construction, the cycle π covers the interval $[\overleftarrow{y}_{n-1}+1, \overrightarrow{y}_n]$, and for every $i < n$, if π visits $\overrightarrow{y}_i$, then π covers Y_i. So to complete the proof —

namely, to show that π covers the entire set Y — it suffices to prove that π visits each node $\overrightarrow{y}_i$, with $i < n$.

Suppose, by way of contradiction, that $\overrightarrow{y}_i$ is the node with the highest index $i < n$ that is not visited by π. Recall that $\overrightarrow{y}_i > \overleftarrow{y}_i$. This shows that

$$\overrightarrow{y}_i \in [\overleftarrow{y}_i + 1, \overrightarrow{y}_n] = \bigcup_{i \leqslant j < n-1} [\overleftarrow{y}_j + 1, \overleftarrow{y}_{j+1}] \cup [\overleftarrow{y}_{n-1} + 1, \overrightarrow{y}_n].$$

As we already proved that π covers the interval $[\overleftarrow{y}_{n-1} + 1, \overrightarrow{y}_n]$, we know that $\overrightarrow{y}_i \in [\overleftarrow{y}_j + 1, \overleftarrow{y}_{j+1}]$ for some j with $i \leqslant j < n-1$. Now recall that $\overrightarrow{y}_i$ is the highest node that is not visited by π. This means that $\overrightarrow{y}_{j+1}$ is visited by π. Moreover, since $j+1 < n$, we know that π uses the incremental path from $\overrightarrow{y}_{j+1}$ to $\overrightarrow{y}_{j+1} + 1$, which covers $Y_{j+1} = [\overleftarrow{y}_j + 1, \overleftarrow{y}_{j+1}]$. But this contradicts the fact that $\overrightarrow{y}_i$ is not visited by π, since $\overrightarrow{y}_i \in [\overleftarrow{y}_j + 1, \overleftarrow{y}_{j+1}]$. □

We know that the set Y is covered by a cycle of the subgraph of Z, and that Z is an interval whose endpoints are consecutive levels $z < z'$, with $d_z = d_{z'} = 0$. For the homestretch, we prove that $Y = Z$. This will imply that the nodes of the cycle are precisely the nodes of the interval Z. Moreover, because the cycle must coincide with a component C of the flow (recall that all the nodes have in-/out-degree 1), this will show that the nodes of C are precisely those of Z.

To prove $Y = Z$ it suffices to recall its definition as the union of the interval $[\overleftarrow{y}_{n-1} + 1, \overrightarrow{y}_n]$ with the sets Y_i, for all $i = 1, \dots, n-1$. Clearly, we have that $Y \subseteq Z$. For the converse inclusion, we also recall that $\overleftarrow{y}_0 + 1 = 0 = \min(Z)$ and $\overrightarrow{y}_n = \max(Z)$. Consider an arbitrary level $z \in Z$. Clearly, we have either $z \leqslant \overleftarrow{y}_i$, for some $1 \leqslant i < n$, or $z > \overleftarrow{y}_n$. In the former case, by choosing the smallest index i such that $z \leqslant \overleftarrow{y}_i$, we get $z \in [\overleftarrow{y}_{i-1} + 1, \overleftarrow{y}_i]$, whence $z \in Y_i \subseteq Y$. In the latter case, we immediately have $z \in Y$, by construction. □

The next lemma describes the precise shape and order of the intercepted factors when the loop L is idempotent.

Lemma 3.3.6. *If C is a left-to-right (resp. right-to-left) component of an idempotent loop L, then the (L, C)-factors are in the following order: k* LL*-factors (resp.* RR*-factors), followed by one* LR*-factor (resp.* RL*-factor), followed by k* RR*-factors (resp.* LL*-factors), for some $k \geqslant 0$.*

Proof. Suppose that C is a left-to-right component of L. We show by way of contradiction that C has only one LR-factor and no RL-factor. By Lemma 3.3.5 this will yield the claimed shape. Figure 3.11 can be used as a reference example for the arguments that follow.

We begin by listing the (L, C)-factors. As usual, we order them based on their occurrences in the run ρ. Let γ be the first (L, C)-factor that is not an LL-factor, and let $\beta_1, \ldots, \beta_k$ be the (L, C)-factors that precede γ (these are all LL-factors). Because γ starts at an even level, it must be an LR-factor. Suppose that there is another (L, C)-factor, say ζ, that comes after γ and it is neither an RR-factor nor an LL-factor. Because ζ starts at an odd level, it must be an RL-factor. Further let $\delta_1, \ldots, \delta_{k'}$ be the intercepted RR-factors that occur between γ and ζ. We claim that $k' < k$, namely, that the number of RR-factors between γ and ζ is strictly less than the number of LL-factors before γ. Indeed, if this were not the case, then, by Lemma 3.3.5, the level where ζ starts would not belong to the component C.

Now, consider the pumped run $\rho' = \mathsf{pump}^2_L(\rho)$, obtained by adding a new copy of L. Let L' be the loop of ρ' obtained from the union of L and its copy. Since L is idempotent, the components of L are isomorphic to the components of L'. In particular, we can denote by C' the component of L' that is isomorphic to C. Let us consider the (L', C')-factors of ρ'. The first k factors are isomorphic to the k LL-factors $\beta_1, \ldots, \beta_k$ from ρ. However, the $(k + 1)$-th element has a different shape: it is isomorphic to $\gamma\ \beta_1\ \delta_1\ \beta_2\ \cdots\ \delta_{k'}\ \beta_{k'+1}\ \zeta$, and in particular it is an LL-factor. This implies that the $(k+1)$-th edge of C' is of the form $(y, y+1)$, while the $(k+1)$-th edge of C is of the form $(y, y-2k)$. This contradiction comes from having assumed the existence of the RL-factor ζ, and is illustrated in Figure 3.11. $\square$

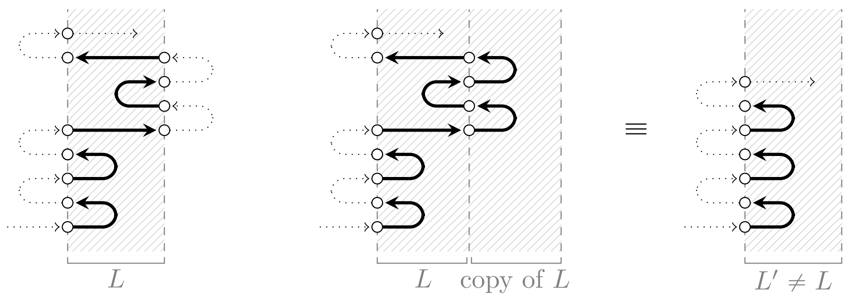


Figure 3.11: Pumping a loop L with a wrong shape and showing it is not idempotent.

Remark 3.3.7. *Note that every loop in the sweeping case is idempotent. Moreover, the (L,C)-factors are precisely the factors intercepted by the loop L.*

3.3.3 Pumping idempotent loops.

To describe in a formal way the run obtained by pumping an idempotent loop, we need to generalize the notion of anchor point in the two-way case (the reader may compare this with the analogous definitions in Section 3.1 for the sweeping case). Intuitively, the anchor point of a component C of an idempotent loop L is the source location of the unique LR- or RL-factor intercepted by L that corresponds to an edge of C (recall Lemma 3.3.6):

Definition 3.3.8. *Let C be a component of an idempotent loop $L = [x_1, x_2]$. The* anchor point *of C inside L, denoted*[2] $\mathsf{an}(C)$, *is either the location $(x_1, \max(C))$ or the location $(x_2, \max(C))$, depending on whether C is left-to-right or right-to-left.*

We will usually depict anchor points by black circles (like, for instance, in Figure 3.9).

It is also convenient to redefine the notation $\mathsf{tr}(\ell)$ for representing an appropriate sequence of transitions associated with each anchor point ℓ of an idempotent loop:

Definition 3.3.9. *Let C be a component of some idempotent loop L, let $\ell = \mathsf{an}(C)$ be the anchor point of C inside L, and let $i_0 \mapsto i_1 \mapsto i_2 \mapsto \cdots \mapsto i_k \mapsto i_{k+1}$ be a cycle of C, where $i_0 = i_{k+1} = \max(C)$. For every $j = 0, \ldots, k$, further let β_j be the factor intercepted by L that corresponds to the edge $i_j \mapsto i_{j+1}$ of C.*

The trace *of ℓ inside L is the run $\mathsf{tr}(\ell) = \beta_0\ \beta_1\ \cdots\ \beta_k$.*

Note that $\mathsf{tr}(\ell)$ is not necessarily a factor of the original run ρ. However, $\mathsf{tr}(\ell)$ is indeed a run, since L is a loop and the factors β_i are concatenated according to the flow. As we will see below, $\mathsf{tr}(\ell)$ will appear as (iterated) factor of the pumped version of ρ, where the loop L is iterated.

[2]In denoting the anchor point — and similarly the trace — of a component C inside a loop L, we omit the annotation specifying L, since this is often understood from the context.

As an example, by referring again to the components C_1, C_2, C_3 of Figure 3.9, we have the following traces: $\mathsf{tr}(\mathsf{an}(C_1)) = \alpha_2\,\alpha_1\,\alpha_3$, $\mathsf{tr}(\mathsf{an}(C_2)) = \beta_2\,\beta_1\,\beta_3$, and $\mathsf{tr}(\mathsf{an}(C_3)) = \gamma_1$.

The next proposition shows the effect of pumping idempotent loops. The reader can note the similarity with the sweeping case.

Proposition 3.3.10. *Let L be an idempotent loop of ρ with components $C_1, \dots, C_k$, listed according to the order of their anchor points: $\ell_1 = \mathsf{an}(C_1) \lhd \cdots \lhd \ell_k = \mathsf{an}(C_k)$. For all $n \in \mathbb{N}$, we have*

$$\mathsf{pump}_L^{n+1}(\rho) \;=\; \rho_0\ \mathsf{tr}(\ell_1)^n\ \rho_1\ \cdots\ \rho_{k-1}\ \mathsf{tr}(\ell_k)^n\ \rho_k$$

where

- *ρ_0 is the prefix of ρ that ends at the first anchor point ℓ_1,*
- *ρ_k is the suffix of ρ that starts at the last anchor point ℓ_k,*
- *ρ_i is the factor $\rho[\ell_i, \ell_{i+1}]$, for all $1 \leqslant i < k$.*

Proof. Along the proof we sometimes refer to Figure 3.9 to ease the intuition of some definitions and arguments. For example, in the left hand-side of Figure 3.9, the run ρ_0 goes until the first location marked by a black circle; the runs ρ_1 and ρ_2, resp., are between the first and the second black dot, and the second and third black dot; finally, ρ_3 is the suffix starting at the last black dot. The pumped run $\mathsf{pump}_L^{n+1}(\rho)$ for $n = 2$ is depicted to the right of the figure.

Let $L = [x_1, x_2]$ be an idempotent loop and, for all $i = 0, \dots, n$, let $L'_i = [x'_i, x'_{i+1}]$ be the i-th copy of the loop L in the pumped run $\rho' = \mathsf{pump}_L^{n+1}(\rho)$, where $x'_i = x_1 + i \cdot (x_2 - x_1)$ (the "0-th copy of L" is the loop L itself). Further let $L' = L'_0 \cup \cdots \cup L'_n = [x'_0, x'_{n+1}]$, that is, L' is the loop of ρ' that spans across the $n+1$ occurrences of L. As L is idempotent, the loops $L'_0, \dots, L'_n$ and L' have all the same effect as L. In particular, the components of $L'_0, \dots, L'_n$, and L' are isomorphic to and in same order as those of L. We denote these components by $C_1, \dots, C_k$.

We let $\ell_j = \mathsf{an}(C_j)$ be the anchor point of each component C_j inside the loop L of ρ (these locations are marked by black dots in the left hand-side of Figure 3.9). Similarly, we let $\ell'_{i,j}$ (resp. ℓ'_j) be the anchor point of C_j inside the loop L'_i (resp. L'). From Definition 3.3.8, we have that either $\ell'_j = \ell'_{1,j}$ or

$\ell'_j = \ell'_{n,j}$, depending on whether C_j is left-to-right or right-to-left (or, equally, on whether j is odd or even).

Now, let us consider the factorization of the pumped run ρ' induced by the locations $\ell'_{i,j}$, for all $i = 0, \ldots, n$ and for $j = 1, \ldots, k$ (these locations are marked by black dots in the right hand-side of the figure). By construction, the prefix of ρ' that ends at location $\ell'_{0,1}$ coincides with the prefix of ρ that ends at ℓ_1, i.e. ρ_0 in the statement of the proposition. Similarly, the suffix of ρ' that starts at location $\ell'_{n,k}$ is isomorphic to the suffix of ρ that starts at ℓ_k, i.e. ρ_k in the statement. Moreover, for all odd (resp. even) indices j, the factor $\rho'[\ell'_{n,j}, \ell'_{n,j+1}]$ (resp. $\rho'[\ell_{0,j}, \ell_{0,j+1}]$) is isomorphic to $\rho[\ell_j, \ell_{j+1}]$, i.e. the ρ_j of the statement.

The remaining factors of ρ' are those delimited by the pairs of locations $\ell'_{i,j}$ and $\ell'_{i+1,j}$, for all $i = 0, \ldots, n-1$ and all $j = 1, \ldots, k$. Consider one such factor $\rho'[\ell'_{i,j}, \ell'_{i+1,j}]$, and assume that the index j is odd (the case of an even j is similar). This factor can be seen as a concatenation of factors intercepted by L that correspond to edges of C_j inside L'_i. More precisely, $\rho'[\ell'_{i,j}, \ell'_{i+1,j}]$ is obtained by concatenating the unique LR-factor of C_j — recall that by Lemma 3.3.6 there is exactly one such factor — with an interleaving of the LL-factors and the RR-factors of C_j. As the components are the same for all L'_i's and for L, this corresponds precisely to the trace $\mathsf{tr}(\ell_j)$ (cf. Definition 3.3.9). Now that we know that $\rho'[\ell'_{i,j}, \ell'_{i+1,j}]$ is isomorphic to $\mathsf{tr}(\ell_j)$, we can conclude that $\rho'[\ell'_{0,j}, \ell'_{n,j}] = \rho'[\ell'_{0,j}, \ell'_{1,j}] \ \ldots \ \rho'[\ell'_{n-1,j}, \ell'_{n,j}]$ is isomorphic to $\mathsf{tr}(\ell_j)^n$. □

3.4 Combinatorics in the two-way case

In this section we develop the main combinatorial techniques underlying the characterization of one-way definability for two-way transducers. In particular, we will show how to derive the existence of idempotent loops with bounded outputs using Ramsey-based arguments, and we will use this to derive periodicity properties for the outputs produced between inversions.

As usual, ρ is a fixed successful run of $\mathcal{T}$ on some input word u.

3.4.1 Ramsey-type arguments.

We start with a technique used for bounding the lengths of the outputs of certain factors, or subsequences of a two-way run. This technique is a

Ramsey-type argument, more precisely it relies on Simon's "factorization forest" theorem [24, 79], which is recalled below. The classical version of Ramsey theorem would yield a similar result, but without the tight bounds that we get here.

Let X be a set of positions of ρ. A *factorization forest* for X is an unranked tree, where the nodes are intervals I with endpoints in X and labeled with the corresponding effect E_I, the ancestor relation is given by the containment order on intervals, the leaves are the minimal intervals $[x_1, x_2]$, with x_2 successor of x_1 in X, and for every internal node I with children $J_1, \ldots, J_k$, we have:

- $I = J_1 \cup \cdots \cup J_k$,
- $E_I = E_{J_1} \odot \cdots \odot E_{J_k}$,
- if $k > 2$, then $E_I = E_{J_1} = \cdots = E_{J_k}$ is an idempotent of the semigroup $(\mathcal{E}, \odot)$.

Recall that in a normalized run there are at most $|Q|^{\boldsymbol{H}}$ distinct crossing sequences. Moreover, a flow contains at most $\boldsymbol{H}$ edges, and each edge has one of the 4 possible types $\mathsf{LL}, \mathsf{LR}, \mathsf{RL}, \mathsf{RR}$. So the effect semigroup $(\mathcal{E}, \odot)$ has size at most $\boldsymbol{E} = (2|Q|)^{2\boldsymbol{H}}$. Further recall that c_{max} is the maximum number of letters output by a single transition of $\mathcal{T}$. Like we did in the sweeping case, we define the constant $\boldsymbol{B} = \boldsymbol{c}_{\mathsf{max}} \cdot \boldsymbol{H} \cdot (\boldsymbol{2^{3E}} + \boldsymbol{4}) + \boldsymbol{4c}_{\mathsf{max}}$ that will be used to bound the lengths of some outputs of $\mathcal{T}$.

Theorem 3.4.1 (Factorization forest theorem [24, 79])**.** *For every set X of positions of ρ, there is a factorization forest for X of height at most $3\boldsymbol{E}$.*

The above theorem can be used to show that if ρ produces an output longer than $\boldsymbol{B}$, then it contains an idempotent loop and a trace with non-empty output. Below, we present a result in the same spirit, but refined in a way that it can be used to find anchor points inside specific intervals. To formally state the result, we consider subsequences of ρ induced by sets of locations that are not necessarily contiguous. Recall the notation $\rho|Z$ introduced at page 72: $\rho|Z$ is the subsequence of ρ induced by the location set Z. For example, Figure 3.12 depicts a set $Z = [\ell_1, \ell_2] \cap (I \times \mathbb{N})$ by a hatched area, together with the induced subrun $\rho|Z$, represented by thick arrows.

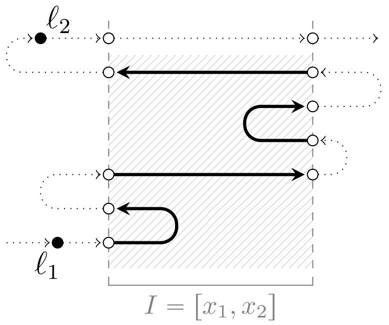


Figure 3.12: A sub-run $\rho|Z$.

Theorem 3.4.2. *Let $I = [x_1, x_2]$ be an interval of positions, $K = [\ell_1, \ell_2]$ an interval of locations, and $Z = K \cap (I \times \mathbb{N})$. If $|\mathsf{out}(\rho|Z)| > \boldsymbol{B}$, then there exist an idempotent loop L and an anchor point ℓ of L such that*

1. $x_1 < \min(L) < \max(L) < x_2$ *(in particular, $L \subsetneq I$),*

2. $\ell_1 \lhd \ell \lhd \ell_2$ *(in particular, $\ell \in K$),*

3. $\mathsf{out}(\mathsf{tr}(\ell)) \neq \varepsilon$.

Proof. Let I, K, Z be as in the statement, and suppose that $\big|\mathsf{out}(\rho|Z)\big| > \boldsymbol{B}$. We define $Z' = Z \setminus (\{\ell_1, \ell_2\} \cup (\{x_1, x_2\}) \times \mathbb{N}))$ and we observe that there are at most $2\boldsymbol{H} + 2$ that are missing from Z'. This means that $\rho|Z'$ contains all but $4\boldsymbol{H} + 4$ transitions of $\rho|Z$, and because each transition outputs at most c_{max} letters, we have $|\mathsf{out}(\rho|Z')| > \boldsymbol{B} - 4c_{\mathsf{max}} \cdot \boldsymbol{H} - 4c_{\mathsf{max}} = c_{\mathsf{max}} \cdot \boldsymbol{H} \cdot 2^{3\boldsymbol{E}}$.

For every level y, let X_y be the set of positions x such that (x, y) is the source location of some transition of $\rho|Z'$ that produces non-empty output.

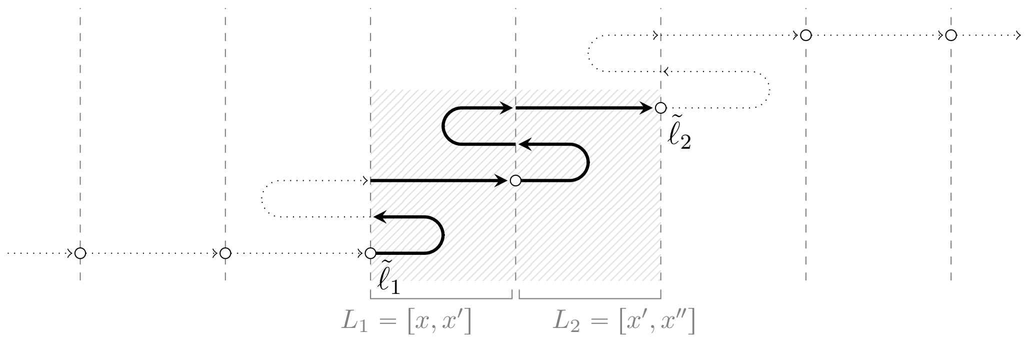


Figure 3.13: Two consecutive idempotent loops with the same effect.

For example, if we refer to Figure 3.13, the vertical dashed lines represent the positions of X_y for a particular level y; accordingly, the circles in the figure represent the locations of the form (x, y), for all $x \in X_y$. Since each transition outputs at most c_{max} letters, we have $\sum_y |X_y| > \boldsymbol{H} \cdot 2^{3\boldsymbol{E}}$. Moreover, since there are at most $\boldsymbol{H}$ levels, there is a level y (which we fix hereafter) such that $|X_y| > 2^{3\boldsymbol{E}}$.

We now prove the following:

Claim. *There are two consecutive loops $L_1 = [x, x']$ and $L_2 = [x', x'']$ with endpoints $x, x', x'' \in X_y$ and such that $E_{L_1} = E_{L_2} = E_{L_1 \cup L_2}$.*

Proof. By Theorem 3.4.1, there is a factorization forest for X_y of height at most $3\boldsymbol{E}$. Since ρ is a valid run, the dummy element $\perp$ of the effect semigroup does not appear in this factorization forest. Moreover, since $|X_y| > 2^{3\boldsymbol{E}}$, we know that the factorization forest contains an internal node $L' = [x'_1, x'_{k+1}]$ with $k > 2$ children, say $L_1 = [x'_1, x'_2], \ldots, L_k = [x'_k, x'_{k+1}]$. By definition of factorization forest, the effects $E_{L'}$, E_{L_1}, ..., E_{L_k} are all equal and idempotent. In particular, the effect $E_{L'} = E_{L_1} = \cdots = E_{L_k}$ is a triple of the form $(F_{L'}, c_1, c_2)$, where $c_i = \rho|x_i$ is the crossing sequence at x'_i. Finally, since $E_{L'}$ is idempotent, we have that $c_1 = c_2$ and this is equal to the crossing sequences of ρ at the positions $x'_1, \ldots, x'_{k+1}$. This shows that L_1, L_2 are idempotent loops. □

Turning back to the proof of the theorem, we know from the above claim that there are two consecutive idempotent loops $L_1 = [x, x']$ and $L_2 = [x', x'']$ with the same effect and with endpoints $x, x', x'' \in X_y \subseteq I \backslash \{x_1, x_2\}$ (see again Figure 3.13).

Let $\tilde{\ell}_1 = (x, y)$ and $\tilde{\ell}_2 = (x'', y)$, and observe that $\tilde{\ell}_1, \tilde{\ell}_2 \in Z'$. In particular, $\tilde{\ell}_1$ and $\tilde{\ell}_2$ are strictly between ℓ_1 and ℓ_2. Suppose by symmetry that $\tilde{\ell}_1 \trianglelefteq \tilde{\ell}_2$. Further let C be the component of $L_1 \cup L_2$ (or, equally, of L_1 or L_2) that contains the node y. Below, we focus on the factors of $\rho[\tilde{\ell}_1, \tilde{\ell}_2]$ that are intercepted by $L_1 \cup L_2$: these are represented in Figure 3.13 by the thick arrows. By Lemma 3.3.6 all these factors correspond to edges of the same component C, namely, they are $(L_1 \cup L_2, C)$-factors.

Let us fix an arbitrary factor α of $\rho[\tilde{\ell}_1, \tilde{\ell}_2]$ that is intercepted by $L_1 \cup L_2$, and assume that $\alpha = \beta_1 \cdots \beta_k$, where $\beta_1, \ldots, \beta_k$ are the factors intercepted by either L_1 or L_2.

Claim. *If β, β' are two factors intercepted by $L_1 = [x, x']$ and $L_2 = [x', x'']$, with $E_{L_1} = E_{L_2} = E_{L_1 \cup L_2}$, and β, β' are adjacent in the run ρ (namely, they* *share an endpoint at position x'), then β, β' correspond to edges in the same component of L_1 (or, equally, L_2).*

Proof. Let C be the component of L_1 and $y_1 \mapsto y_2$ the edge of C that corresponds to the factor β intercepted by L_1. Similarly, let C' be the component of L_2 and $y_3 \mapsto y_4$ the edge of C' that corresponds to the factor β' intercepted by L_2. Since β and β' share an endpoint at position x', we know that $y_2 = y_3$. This shows that $C \cap C' \neq \varnothing$, and hence $C = C'$. □

The above claim shows that any two adjacent factors β_i, β_{i+1} correspond to edges in the same component of L_1 and L_2, respectively. Thus, by transitivity, all factors $\beta_1, \dots, \beta_k$ correspond to edges in the same component, say C'. We claim that $C' = C$. Indeed, if β_1 is intercepted by L_1, then $C' = C$ because α and β_1 start from the same location and hence they correspond to edges of the flow that depart from the same node. The other case is where β_1 is intercepted by L_2, for which a symmetric argument can be applied.

So far we have shown that every factor of $\rho[\tilde{\ell}_1, \tilde{\ell}_2]$ intercepted by $L_1 \cup L_2$ can be factorized into some (L_1, C)-factors and some (L_2, C)-factors. We conclude the proof with the following observations:

- By construction, both loops L_1, L_2 are contained in the interval of positions $I = [x_1, x_2]$, and have endpoints different from x_1, x_2.

- Both anchor points of C inside L_1, L_2 belong to the interval of locations $K \setminus \{\ell_1, \ell_2\}$. This holds because $\rho[\tilde{\ell}_1, \tilde{\ell}_2]$ contains a factor α that is intercepted by $L_1 \cup L_2$ and spans across all the positions from x to x'', namely, an LR-factor. This factor starts at the anchor point of C inside L_1 and visits the anchor point of C inside L_2. Moreover, by construction, α is also a factor of the subsequence $\rho|Z'$. This shows that the anchor points of C inside L_1 and L_2 belong to Z', and in particular to $K \setminus \{\ell_1, \ell_2\}$.

- The first factor of $\rho[\tilde{\ell}_1, \tilde{\ell}_2]$ that is intercepted by $L_1 \cup L_2$ starts at $\tilde{\ell}_1 = (x, y)$, which by construction is the source location of some transition producing non-empty output. By the previous arguments, this factor is a concatenation of (L_1, C)-factors and (L_2, C)-factors. This implies that the trace of the anchor point of C inside L_1, or the trace of C inside L_2 produces non-empty output. □

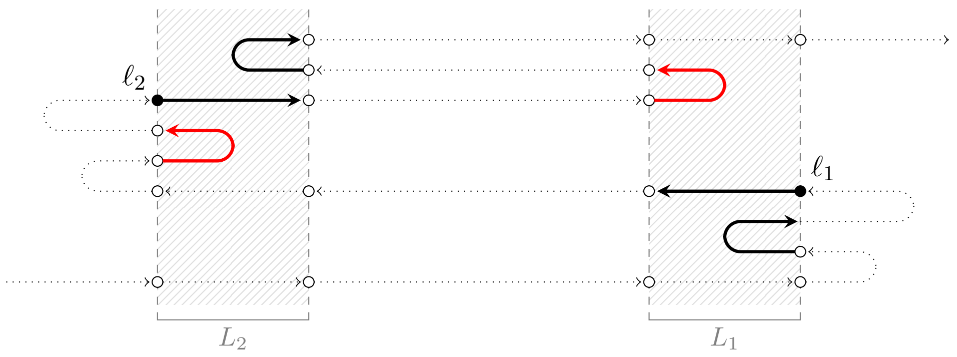


Figure 3.14: An example of an inversion $(L_1, \ell_1, L_2, \ell_2)$ of a two-way run.

3.4.2 Inversions and periodicity.

The first important notion that is used to characterize one-way definability is that of inversion. It turns out that the definition of inversion in the sweeping case (see page 64) can be reused almost verbatim in the two-way setting. The only difference is that here we require the loops to be idempotent and we do not enforce output-minimality (we will discuss this latter choice further below, with a formal definition of output-minimality at hand).

Definition 3.4.3. *An* inversion *of the run ρ is a tuple $(L_1, \ell_1, L_2, \ell_2)$ such that*

1. *L_1, L_2 are idempotent loops,*
2. *$\ell_1 = (x_1, y_1)$ and $\ell_2 = (x_2, y_2)$ are anchor points inside L_1 and L_2, respectively,*
3. *$\ell_1 \lhd \ell_2$ and $x_1 > x_2$,*
4. *for both $i = 1$ and $i = 2$, $\mathsf{out}(\mathsf{tr}(\ell_i)) \neq \varepsilon$.*

Figure 3.14 gives an example of an inversion involving the idempotent loop L_1 with anchor point ℓ_1, and the idempotent loop L_2 with anchor point ℓ_2. The intercepted factors that form the corresponding traces are represented by thick arrows; the one highlighted in red are those that produce non-empty output.

The implication **P1** $\Rightarrow$ **P2** of Theorem 3.0.3 in the two-way case is formalized below exactly as in Proposition 3.1.5, and the proof is very similar to the sweeping case. More precisely, it can be checked that the proof of the first claim in Proposition 3.1.5 is independent of the sweeping assumption — one just needs to replace the use of Equation 3.2 with Proposition 3.3.10. The sweeping assumption was used only for deriving the notion of *output-minimal* factor, which was crucial to conclude that the period p is bounded by the specific constant $\boldsymbol{B}$. In this respect, the proof of Proposition 3.4.4 requires a different argument for showing that $p \leqslant \boldsymbol{B}$:

Proposition 3.4.4. *If $\mathcal{T}$ is one-way definable, then the following property* **P2** *holds:*

For all inversions $(L_1, \ell_1, L_2, \ell_2)$ of ρ, the period p of the word

$$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1,\ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$$

divides both $|\mathsf{out}(\mathsf{tr}(\ell_1))|$ and $|\mathsf{out}(\mathsf{tr}(\ell_2))|$. Moreover, $p \leqslant \boldsymbol{B}$.

We only need to show here that $p \leqslant \boldsymbol{B}$. Recall that in the sweeping case we relied on the assumption that the factors $\mathsf{tr}(\ell_1)$ and $\mathsf{tr}(\ell_2)$ of an inversion are output-minimal, and on Lemma 3.1.3. In the general case we need to replace output-minimality by the following notion:

Definition 3.4.5. *Consider pairs (L, C) consisting of an idempotent loop L and a component C of L.*

1. *On such pairs, define the relation $\sqsubset$ by $(L', C') \sqsubset (L, C)$ if $L' \subsetneq L$ and at least one (L', C')-factor is contained in some (L, C)-factor.*
2. *A pair (L, C) is* output-minimal *if $(L', C') \sqsubset (L, C)$ implies $\mathsf{out}(\mathsf{tr}(\mathsf{an}(C'))) = \varepsilon$.*

Note that the relation $\sqsubset$ is not a partial order in general (it is however antisymmetric). Moreover, it is easy to see that the notion of output-minimal pair (L, C) generalizes that of output-minimal factor introduced in the sweeping case: indeed, if ℓ is the anchor point of a loop L of a sweeping transducer and $\mathsf{tr}(\ell)$ satisfies Definition 3.1.2, then the pair (L, C) is output-minimal, where C is the unique component whose edge corresponds to $\mathsf{tr}(\ell)$.

The following lemma bounds the length of the output trace $\mathsf{out}(\mathsf{tr}(\mathsf{an}(C)))$ for an output-minimal pair (L, C):

Lemma 3.4.6. *For every output-minimal pair (L, C), $|\mathsf{out}(\mathsf{tr}(\mathsf{an}(C)))| \leqslant \boldsymbol{B}$.*

Proof. Consider a pair (L, C) consisting of an idempotent loop $L = [x_1, x_2]$ and a component C of L. Suppose by contradiction that $|\mathsf{out}(\mathsf{tr}(\mathsf{an}(C)))| > \boldsymbol{B}$. We will show that (L, C) is not output-minimal.

Recall that $\mathsf{tr}(\mathsf{an}(C))$ is a concatenation of (L, C)-factors, say, $\mathsf{tr}(\mathsf{an}(C)) = \beta_1 \cdots \beta_k$. Let ℓ_1 (resp. ℓ_2) be the first (resp. last) location that is visited by these factors. Further let $K = [\ell_1, \ell_2]$ and $Z = K \cap (L \times \mathbb{N})$. By construction, the subsequence $\rho|Z$ can be seen as a concatenation of the factors $\beta_1, \ldots, \beta_k$, possibly in a different order than that of $\mathsf{tr}(\mathsf{an}(C))$. This implies that $|\mathsf{out}(\rho|Z)| > \boldsymbol{B}$.

By Theorem 3.4.2, we know that there exist an idempotent loop $L' \subsetneq L$ and a component C' of L' such that $\mathsf{an}(C') \in K$ and $\mathsf{out}(\mathsf{tr}(\mathsf{an}(C'))) \neq \varepsilon$. Note that the (L', C')-factor that starts at the anchor point $\mathsf{an}(C')$ (an LR- or RL-factor) is entirely contained in some (L, C)-factor. This implies that $(L', C') \sqsubset (L, C)$, and thus (L, C) is not output-minimal. □

We remark that the above lemma cannot be used directly to bound the period of the output produced between an inversion. The reason is that we cannot restrict ourselves to inversions $(L_1, \ell_1, L_2, \ell_2)$ that induce output-minimal pairs (L_i, C_i) for $i = 1, 2$, where C_i is the unique component of the anchor point ℓ_i. An example is given in Figure 3.14, assuming that the factors depicted in red are the only ones that produce non-empty output, and the lengths of these outputs exceed $\boldsymbol{B}$. On the one hand $(L_1, \ell_1, L_2, \ell_2)$ is an inversion, but (L_1, C_1) is not output-minimal. On the other hand, it is possible that ρ contains no other inversion than $(L_1, \ell_1, L_2, \ell_2)$: any loop strictly contained in the red factor in L_1 will have the anchor point *after* ℓ_2.

We are now ready to show the second claim of Proposition 3.4.4.

Proof of Proposition 3.4.4. The proof of the second claim requires a refinement of the arguments that involve pumping the run ρ simultaneously on three different loops. As usual, we assume that the loops L_1, L_2 of the inversion are disjoint (otherwise, we preliminarily pump one of the two loops a few times).

Recall that the word

$$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1, \ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$$

has period $p = \gcd\big(|\mathsf{out}(\mathsf{tr}(\ell_1))|, |\mathsf{out}(\mathsf{tr}(\ell_2))|\big)$, but that we cannot bound p by assuming that $(L_1, \ell_1, L_2, \ell_2)$ is output-minimal. However, in the pumped run $\rho^{(2,1)}$ we do find inversions with output-minimal pairs. For example, as depicted in the right part of Figure 3.15, we can consider the left and right copy of L_1 in $\rho^{(2,1)}$, denoted by $\overleftarrow{L}_1$ and $\overrightarrow{L}_1$, respectively. Accordingly, we denote by $\overleftarrow{\ell}_1$ and $\overrightarrow{\ell}_1$ the left and right copy of ℓ_1 in $\rho^{(2,1)}$. Now, let (L_0, C_0) be any *output-minimal* pair such that L_0 is an idempotent loop, $\mathsf{out}(\mathsf{tr}(\mathsf{an}(C_0))) \neq \varepsilon$, and either $(L_0, C_0) = (\overleftarrow{L}_1, C_1)$ or $(L_0, C_0) \sqsubset (\overleftarrow{L}_1, C_1)$. Such a loop L_0 is represented in Figure 3.15 by the red vertical stripe. Further let $\ell_0 = \mathsf{an}(C_0)$.

We claim that either $(L_0, \ell_0, L_2, \ell_2)$ or $(\overrightarrow{L}_1, \overrightarrow{\ell}_1, L_0, \ell_0)$ is an inversion of the run $\rho^{(2,1)}$, depending on whether ℓ_0 occurs before or after ℓ_2. First, note that all the loops L_0, L_2, $\overrightarrow{L}_1$ are idempotent and non-overlapping; more precisely, we have $\max(L_2) \leqslant \min(L_0)$ and $\max(L_0) \leqslant \min(\overrightarrow{L}_1)$. Moreover, the outputs of the traces $\mathsf{tr}(\ell_0)$, $\mathsf{tr}(\overrightarrow{\ell}_1)$, and $\mathsf{tr}(\ell_2)$ are all non-empty. So it remains to distinguish two cases based on the ordering of the anchor points ℓ_0, $\overrightarrow{\ell}_1$, ℓ_2. If $\ell_0 \lhd \ell_2$, then $(L_0, \ell_0, L_2, \ell_2)$ is an inversion. Otherwise, because $(\overrightarrow{L}_1, \overrightarrow{\ell}_1, L_2, \ell_2)$ is an inversion, we know that $\overrightarrow{\ell}_1 \lhd \ell_2 \unlhd \ell_0$, and hence $(\overrightarrow{L}_1, \overrightarrow{\ell}_1, L_0, C_0)$ is an inversion.

Now, we know that $\rho^{(2,1)}$ contains the inversion $(\overrightarrow{L}_1, \ell_1, L_2, \ell_2)$, but also an inversion involving the output-minimal pair (L_0, C_0), with L_0 strictly between $\overrightarrow{L}_1$ and L_2. For all m_0, m_1, m_2, we define $\rho^{(m_0,m_1,m_2)}$ as the run obtained from $\rho^{(2,1)}$ by pumping m_0, m_1, m_2 times the loops $L_0, \overrightarrow{L}_1, L_2$, respectively. By reasoning as we did in the proof of Proposition 3.1.5 (cf. *Periodicity*

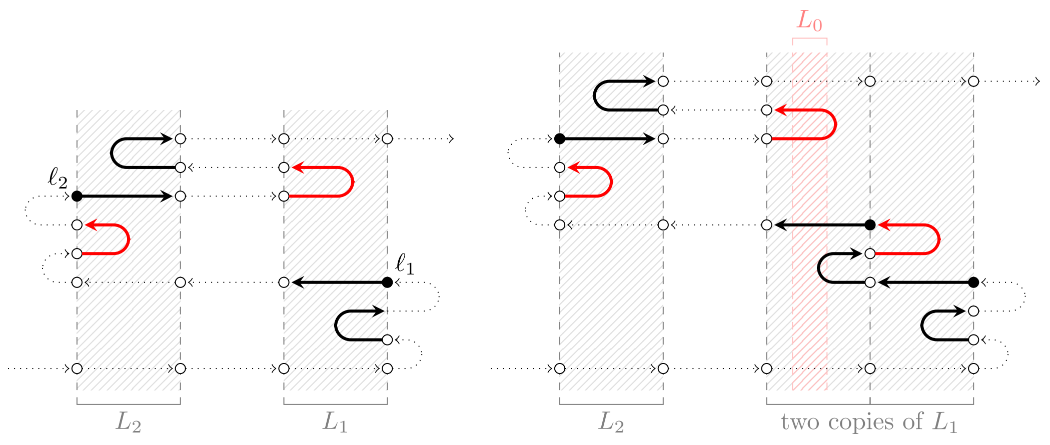


Figure 3.15: An inversion $(L_1, \ell_1, L_2, \ell_2)$ that induces non output-minimal pairs (L_i, C_i).

of outputs of pumped runs), one can show that there are arbitrarily large output factors of $\rho^{(m_0,m_1,m_2)}$ that are produced within the inversion on ℓ_0 (i.e. either $(L_0, \ell_0, L_2, \ell_2)$ or $(\overrightarrow{L}_1, \overrightarrow{\ell}_1, L_0, \ell_0)$) and that are periodic with period p' that divides $|\mathsf{out}(\mathsf{tr}(\ell_0))|$. In particular, by Lemma 3.4.6, we know that $p' \leqslant \boldsymbol{B}$. Moreover, large portions of these factors are also produced within the inversion $(\overrightarrow{L}_1, \overrightarrow{\ell}_1, L_2, \ell_2)$, and hence by Theorem 3.1.6 they have period $\gcd(p, p')$.

To conclude the proof we need to transfer the periodicity property from the pumped runs $\rho^{(m_0,m_1,m_2)}$ to the original run ρ. This is done exactly like in Proposition 3.1.5 by relying on Corollary 3.1.8: one observe that the periodicity property holds for infinitely many parameters m_0, m_1, m_2, and hence also for $m_0 = m_1 = m_2 = 1$. This shows that the word

$$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1, \ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$$

has period $\gcd(p, p') \leqslant \boldsymbol{B}$. □

So far we have shown that the output produced between every inversion of a run of a one-way definable two-way transducer is periodic, with period bounded by $\boldsymbol{B}$ and dividing the lengths of the trace outputs of the inversion. This basically proves the implication **P1** ⇒ **P2** of Theorem 3.0.3. In the next section we will follow a line of arguments similar to that of Section 3.2 to prove the remaining implications **P2** ⇒ **P3** ⇒ **P1**.

3.5 The characterization in the two-way case

In this section we generalize the characterization of one-way definability of sweeping transducers to the general two-way case. As usual, we fix through the rest of the section a successful run ρ of $\mathcal{T}$ on some input word u.

3.5.1 From periodicity of inversions to existence of decompositions.

We continue by proving the second implication **P2** ⇒ **P3** of Theorem 3.0.3 in the two-way case. This requires showing the existence of a suitable decomposition of a run ρ that *satisfies* property **P2**. Recall that **P2** says that for every inversion $(L_1, \ell_1, L_2, \ell_2)$, the period of the word

$\mathsf{out}(\mathsf{tr}(\ell_1))\ \mathsf{out}(\rho[\ell_1,\ell_2])\ \mathsf{out}(\mathsf{tr}(\ell_2))$ divides $\gcd(|\mathsf{out}(\mathsf{tr}(\ell_1))|, |\mathsf{out}(\mathsf{tr}(\ell_2))|) \leqslant \boldsymbol{B}$. The definitions underlying the decomposition of ρ are similar to those given in the sweeping case:

Definition 3.5.1. *Let $\rho[\ell,\ell']$ be a factor of a run ρ of $\mathcal{T}$, where $\ell = (x,y)$, $\ell' = (x',y')$, and $x \leqslant x'$. We call $\rho[\ell,\ell']$*

- *a $\boldsymbol{B}$-*diagonal *if for all $z \in [x,x']$, there is a location ℓ_z at position z such that $\ell \trianglelefteq \ell_z \trianglelefteq \ell'$ and the words $\mathsf{out}(\rho|Z^{\nwarrow}_{\ell_z})$ and $\mathsf{out}(\rho|Z^{\searrow}_{\ell_z})$ have length at most $\boldsymbol{B}$, where $Z^{\nwarrow}_{\ell_z} = [\ell_z,\ell'] \cap \big([0,z] \times \mathbb{N}\big)$ and $Z^{\searrow}_{\ell_z} = [\ell,\ell_z] \cap \big([z,\omega] \times \mathbb{N}\big)$;*

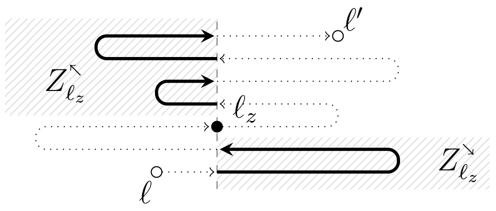


Figure 3.16: Outputs bounded in a diagonal.

- *a $\boldsymbol{B}$-*block *if the word $\mathsf{out}(\rho[\ell,\ell'])$ is almost periodic with bound $\boldsymbol{B}$, and $\mathsf{out}(\rho|Z^{\leftarrow})$ and $\mathsf{out}(\rho|Z^{\rightarrow})$ have length at most $\boldsymbol{B}$, where $Z^{\leftarrow} = [\ell,\ell'] \cap \big([0,x] \times \mathbb{N}\big)$ and $Z^{\rightarrow} = [\ell,\ell'] \cap \big([x',\omega] \times \mathbb{N}\big)$.*

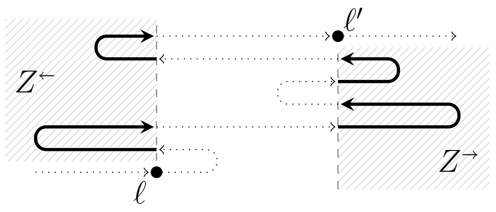


Figure 3.17: Outputs bounded in a block.

The definition of $\boldsymbol{B}$-decomposition is copied verbatim from the sweeping case, but uses the new notions of $\boldsymbol{B}$-diagonal and $\boldsymbol{B}$-block:

Definition 3.5.2. *A $\boldsymbol{B}$-*decomposition *of a run ρ of $\mathcal{T}$ is a factorization $\prod_i \rho[\ell_i,\ell_{i+1}]$ of ρ into $\boldsymbol{B}$-diagonals and $\boldsymbol{B}$-blocks.*

To provide further intuition, we consider the transduction of Example 10 and the two-way transducer $\mathcal{T}$ that implements it in the most natural way. Figure 3.18 shows an example of a run of $\mathcal{T}$ on an input of the form $u_1 \# u_2 \# u_3 \# u_4$, where $u_2, u_4 \in (abc)^*$, $u_1\, u_3 \notin (abc)^*$, and u_3 has even length. The factors of the run that produce long outputs are highlighted by the bold arrows. The first and third factors of the decomposition, i.e. $\rho[\ell_1, \ell_2]$ and $\rho[\ell_3, \ell_4]$, are diagonals (represented by the blue hatched areas); the second and fourth factors $\rho[\ell_2, \ell_3]$ and $\rho[\ell_4, \ell_5]$ are blocks (represented by the red hatched areas).

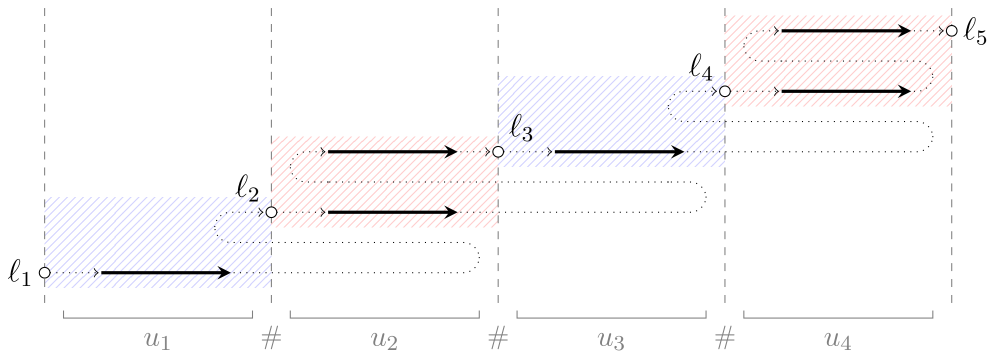


Figure 3.18: A decomposition of a run of a two-way transducer.

To identify the blocks of a possible decomposition of ρ, we reuse the equivalence relation S^* introduced in Definition 3.2.4. Recall that this is the reflexive and transitive closure of the relation S that groups any two locations ℓ, ℓ' that occur between ℓ_1, ℓ_2, for some inversion $(L_1, \ell_1, L_2, \ell_2)$.

The proof that the output produced inside each S^*-equivalence class is periodic, with period at most $\boldsymbol{B}$ (Lemma 3.2.5) carries over in the two-way case without modifications. Similarly, every S^*-equivalence class can be extended to the left and to the right by using Definition 3.2.6, which we report here for the sake of readability.

Definition 3.5.3. *Let $K = [\ell, \ell']$ be a non-singleton S^*-equivalence class. We let*

$$\mathsf{an}(K) = \{\ell \in K : \ell \in \{\ell_1, \ell_2\} \text{ for some inversion } (L_1, \ell_1, L_2, \ell_2)\}$$

be the restriction of K to anchor points occurring in some inversion, and $X_{\mathsf{an}(K)} = \{x : \exists y\, (x, y) \in \mathsf{an}(K)\}$ be the projection of $\mathsf{an}(K)$ on positions. We define $\mathsf{block}(K) = [\tilde{\ell}, \tilde{\ell}']$, where

- *$\tilde{\ell}$ is the latest location $(x, y) \trianglelefteq \ell$ such that $x = \min(X_{\mathsf{an}(K)})$,*
- *$\tilde{\ell}'$ is the earliest location $(x, y) \trianglerighteq \ell'$ such that $x = \max(X_{\mathsf{an}(K)})$*

(note that the locations $\tilde{\ell}, \tilde{\ell}'$ exist since ℓ, ℓ' are both anchor points in some inversion).

As usual, we call S^*-*block* any factor of ρ of the form $\rho|\mathsf{block}(K)$ that is obtained by applying the above definition to a non-singleton S^*-class K. Lemma 3.2.7, which shows that S^*-blocks can indeed be used as $\boldsymbol{B}$-blocks in a decomposition of ρ, generalizes easily to the two-way case:

Lemma 3.5.4. *If K is a non-singleton S^*-equivalence class, then $\rho|\mathsf{block}(K)$ is a $\boldsymbol{B}$-block.*

Proof. The proof is similar to that of Lemma 3.2.7. The main difference is that here we will bound the lengths of some outputs by using a Ramsey-type argument (Theorem 3.4.2), instead of output-minimality of factors (Lemma 3.1.3).

Let $K = [\ell, \ell']$, $\mathsf{an}(K)$, $X_{\mathsf{an}(K)}$, and $\mathsf{block}(K) = [\tilde{\ell}, \tilde{\ell}']$ be as in Definition 3.5.3, where $\tilde{\ell} = (\tilde{x}, \tilde{y})$, $\tilde{\ell}' = (\tilde{x}', \tilde{y}')$, $\tilde{x} = \min(X_{\mathsf{an}(K)})$, and $\tilde{x}' = \max(X_{\mathsf{an}(K)})$. We need to verify that $\rho|\mathsf{block}(K)$ is a $\boldsymbol{B}$-block, namely, that:

- $\tilde{x} \leqslant \tilde{x}'$,
- $\mathsf{out}(\rho[\ell, \ell'])$ is almost periodic with bound $\boldsymbol{B}$,
- $\mathsf{out}(\rho|Z^{\leftarrow})$ and $\mathsf{out}(\rho|Z^{\rightarrow})$ have length at most $\boldsymbol{B}$, where $Z^{\leftarrow} = [\ell, \ell'] \cap \big([0, x] \times \mathbb{N}\big)$ and $Z^{\rightarrow} = [\ell, \ell'] \cap \big([x', \omega] \times \mathbb{N}\big)$.

The first condition $\tilde{x} \leqslant \tilde{x}'$ follows immediately from $\tilde{x} = \min(X_{\mathsf{an}(K)})$ and $\tilde{x}' = \max(X_{\mathsf{an}(K)})$.

Next, we prove that the output produced by the factor $\rho[\tilde{\ell}, \tilde{\ell}']$ is almost periodic with bound $\boldsymbol{B}$. By Definition 3.5.3, we have $\tilde{\ell} \trianglelefteq \ell \lhd \ell' \trianglelefteq \tilde{\ell}'$, and by Lemma 3.2.5 we know that $\mathsf{out}(\rho[\ell, \ell'])$ is periodic with period at most $\boldsymbol{B}$. So it suffices to bound the length of the words $\mathsf{out}(\rho[\tilde{\ell}, \ell])$ and $\mathsf{out}(\rho[\ell', \tilde{\ell}'])$. We shall focus on the former word, as the arguments for the latter are similar.

First, we show that the factor $\rho[\tilde{\ell}, \ell]$ lies entirely to the right of position $\tilde{x}$ (in particular, it starts at an even level $\tilde{y}$). Indeed, if this were not the case, there would exist another location $\ell'' = (\tilde{x}, \tilde{y} + 1)$, on the same position as $\tilde{\ell}$, but at a higher level, such that $\tilde{\ell} \lhd \ell'' \trianglelefteq \ell$. But this would contradict Definition 3.5.3 ($\tilde{\ell}$ is the *latest* location $(x, y) \trianglelefteq \ell$ such that $x = \min(X_{\mathsf{an}(K)})$).

Suppose now that the length of $|\mathsf{out}(\rho[\tilde{\ell}, \ell])| > \boldsymbol{B}$. We head towards a contradiction by finding a location $\ell'' \lhd \ell$ that is S^*-equivalent to the first location ℓ of the S^*-equivalence class K. Since the factor $\rho[\tilde{\ell}, \ell]$ lies entirely to the right of position $\tilde{x}$, it is intercepted by the interval $I = [\tilde{x}, \omega]$. So $|\mathsf{out}(\rho[\tilde{\ell}, \ell])| > \boldsymbol{B}$ is equivalent to saying $|\mathsf{out}(\rho|Z)| > \boldsymbol{B}$, where $Z = [\tilde{\ell}, \ell] \cap \big([\tilde{x}, \omega] \times \mathbb{N}\big)$. Then, Theorem 3.4.2 implies the existence of an idempotent loop L and an anchor point ℓ'' of L such that

- $\min(L) > \tilde{x}$,
- $\tilde{\ell} \lhd \ell'' \lhd \ell$,
- $\mathsf{out}(\mathsf{tr}(\ell'')) \neq \varepsilon$.

Further recall that $\tilde{x} = \min(X_{\mathsf{an}(K)})$ is the leftmost position of locations in the class $K = [\ell, \ell']$ that are also anchor points of inversions. In particular, there is an inversion $(L_1, \ell''_1, L_2, \ell''_2)$, with $\ell''_2 = (\tilde{x}, y''_2) \in K$. Since $\ell'' \lhd \ell \trianglelefteq \ell''_2$ and the position of ℓ'' is to the right of the position of ℓ''_2, we know that $(L, \ell'', L_2, \ell''_2)$ is also an inversion, and hence $\ell'' \; \mathsf{S}^* \; \ell''_2 \; \mathsf{S}^* \; \ell$. But since $\ell'' \neq \ell$, we get a contradiction with the assumption that ℓ is the first location of the S^*-class K. In this way we have shown that $|\mathsf{out}(\rho[\tilde{\ell}, \ell])| \leqslant \boldsymbol{B}$.

It remains to bound the lengths of the outputs produced by the subsequences $\rho|Z^{\leftarrow}$ and $\rho|Z^{\rightarrow}$, where $Z^{\leftarrow} = [\tilde{\ell}, \tilde{\ell}'] \cap \big([0, \tilde{x}] \times \mathbb{N}\big)$ and $Z^{\rightarrow} = [\tilde{\ell}, \tilde{\ell}'] \cap \big([\tilde{x}', \omega] \times \mathbb{N}\big)$. As usual, we consider only one of the two symmetric cases. Suppose, by way of contradiction, that $|\mathsf{out}(\rho|Z^{\leftarrow})| > \boldsymbol{B}$. By Theorem 3.4.2, there exist an idempotent loop L and an anchor point ℓ'' of L such that

- $\max(L) < \tilde{x}$,
- $\tilde{\ell} \lhd \ell'' \lhd \tilde{\ell}'$,
- $\mathsf{out}(\mathsf{tr}(\ell'')) \neq \varepsilon$.

By following the same line of reasoning as before, we recall that ℓ is the first location of the non-singleton class K. From this we derive the existence an inversion $(L_1, \ell''_1, L_2, \ell''_2)$ where $\ell''_1 = \ell$. We claim that $\ell \trianglelefteq \ell''$. Indeed, if this were not the case, then, because ℓ'' is strictly to the left of $\tilde{x}$ and ℓ is to the right of $\tilde{x}$, there would exist a location ℓ''' between ℓ'' and ℓ that lies at position $\tilde{x}$. But $\tilde{\ell} \lhd \ell'' \trianglelefteq \ell''' \trianglelefteq \ell$ would contradict the fact that $\tilde{\ell}$ is the *latest* location before ℓ that lies at the position $\tilde{x}$. Now that we know that $\ell \trianglelefteq \ell''$ and that ℓ'' is to the left of $\tilde{x}$, we observe that $(L_1, \ell''_1, L, \ell'')$ is also an inversion, and hence $\ell'' \in \mathsf{an}(K)$. Since ℓ'' is strictly to the left of $\tilde{x}$, we get a contradiction with the definition of $\tilde{x}$ as leftmost position of the locations in $\mathsf{an}(K)$. So we conclude that $|\mathsf{out}(\rho|Z^{\leftarrow})| \leqslant \boldsymbol{B}$. $\square$

The proof of Lemma 3.2.8, which shows that S^*-blocks do not overlap along the input axis, carries over in the two-way case, again without modifications. Finally, we generalize Lemma 3.2.9 to the new definition of diagonal, which completes the construction of a $\boldsymbol{B}$-decomposition for the run ρ:

Lemma 3.5.5. *Let $\rho[\ell, \ell']$ be a factor of ρ, with $\ell = (x, y)$, $\ell' = (x', y')$, and $x \leqslant x'$, that does not overlap any S^*-block. Then $\rho[\ell, \ell']$ is a $\boldsymbol{B}$-diagonal.*

Proof. Suppose by way of contradiction that there is some $z \in [x, x']$ such that, for all locations ℓ'' at position z and between ℓ and ℓ', one of the two conditions holds:

1. $|\mathsf{out}(\rho|Z^{\nwarrow}_{\ell''})| > \boldsymbol{B}$, where $Z^{\nwarrow}_{\ell''} = [\ell'', \ell'] \cap \big([0, z] \times \mathbb{N}\big)$,

2. $|\mathsf{out}(\rho|Z^{\searrow}_{\ell''})| > \boldsymbol{B}$, where $Z^{\searrow}_{\ell''} = [\ell, \ell''] \cap \big([z, \omega] \times \mathbb{N}\big)$.

First, we claim that *each* of the two conditions above are satisfied at some locations $\ell'' \in [\ell, \ell']$ at position z. Consider the highest even level y'' such that $\ell'' = (z, y'') \in [\ell, \ell']$ (use Figure 3.16 as a reference). Since $z \leqslant x'$, the outgoing transition at ℓ'' is rightward oriented, and the set $Z^{\nwarrow}_{\ell''}$ is empty. This means that condition (1) is trivially violated at ℓ'', and hence condition (2) holds at ℓ'' by the initial assumption. Symmetrically, condition (1) holds at the location $\ell'' = (z, y'')$, where y'' is the lowest even level with $\ell'' \in [\ell, \ell']$.

Let us now compare the levels where the above conditions hold. Clearly, the lower the level of location ℓ'', the easier it is to satisfy condition (1), and symmetrically for condition (2). So, let $\ell^+ = (z, y^+)$ (resp. $\ell^- = (z, y^-)$) be the highest (resp. lowest) location in $[\ell, \ell']$ at position z that satisfies condition (1) (resp. condition (2)).

We claim that $y^+ \geqslant y^-$. For this, we first observe that $y^+ \geqslant y^- - 1$, since otherwise there would exist a location $\ell'' = (z, y'')$, with $y^+ < y'' < y^-$, that violates both conditions (1) and (2). Moreover, y^+ must be odd, otherwise the transition departing from $\ell^+ = (z, y^+)$ would be rightward oriented and the location $\ell'' = (z, y^+ + 1)$ would still satisfy condition (1), contradicting the definition of the highest location ℓ^+. For similar reasons, y^- must also be odd, otherwise there would be a location $\ell'' = (z, y^- - 1)$ below ℓ^- that satisfies condition (2). But since $y^+ \geqslant y^- - 1$ and both y^+ and y^- are odd, we need to have $y^+ \geqslant y^-$.

In fact, from the previous arguments we know that the location $\ell'' = (z, y^+)$ (or equally the location (x, y^-)) satisfies *both* conditions (1) and (2).

We can thus apply Theorem 3.4.2 to the sets $Z^{\searrow}_{\ell''}$ and $Z^{\nwarrow}_{\ell''}$, deriving the existence of two idempotent loops L_1, L_2 and two anchor points ℓ_1, ℓ_2 of L_1, L_2, respectively, such that

- $\max(L_2) < z < \min(L_1)$,
- $\ell \lhd \ell_1 \lhd \ell'' \lhd \ell_2 \lhd \ell'$,
- $\mathsf{out}(\mathsf{tr}(\ell_1)), \mathsf{out}(\mathsf{tr}(\ell_2)) \neq \varepsilon$.

In particular, since ℓ_1 is to the right of ℓ_2 w.r.t. the order of positions, we know that $(L_1, \ell_1, L_2, \ell_2)$ is an inversion, and hence $\ell_1 \; \mathsf{S}^* \; \ell_2$. But this contradicts the assumption that $\rho[\ell, \ell']$ does not overlap with any S^*-block. $\square$

3.5.2 From existence of decompositions to an equivalent one-way transducer.

It remains to prove the last implication **P3** $\Rightarrow$ **P1** of Theorem 3.0.3, which amounts to construct a one-way transducer $\mathcal{T}'$ equivalent to $\mathcal{T}$.

Hereafter, we denote by D the language of words $u \in \mathsf{dom}(\mathcal{T})$ such that *all* successful runs of $\mathcal{T}$ on u admit a $\boldsymbol{B}$-decomposition. So far, we know that if $\mathcal{T}$ is one-way definable (**P1**), then $D = \mathsf{dom}(\mathcal{T})$ (**P3**). As a matter of fact, this reduces the one-way definability problem for $\mathcal{T}$ to the containment problem $\mathsf{dom}(\mathcal{T}) \subseteq D$. We will see later how the latter problem can be decided in double exponential space by further reducing it to checking the emptiness of the intersection of the languages $\mathsf{dom}(\mathcal{T})$ and $D^{\complement}$, where $D^{\complement}$ is the complement of D.

Below, we show how to construct a one-way transducer $\mathcal{T}'$ of triple exponential size such that $\mathcal{T}' \subseteq \mathcal{T}$ and $\mathsf{dom}(\mathcal{T}')$ is the set of all input words that have *some* successful run admitting a $\boldsymbol{B}$-decomposition (hence $\mathsf{dom}(\mathcal{T}') \supseteq D$). In particular, we will have that

$$\mathcal{T}|_D \subseteq \mathcal{T}' \subseteq \mathcal{T}.$$

Note that this will prove **P3** to **P1**, as well as the second item of Theorem 3.0.1, since $D = \mathsf{dom}(\mathcal{T})$ if and only if $\mathcal{T}$ is one-way definable. A sketch of the proof of this construction when $\mathcal{T}$ is a sweeping transducer was given at the end of Section 3.2.

Proposition 3.5.6. *Given a functional two-way transducer $\mathcal{T}$, a one-way transducer $\mathcal{T}'$ satisfying*

$$\mathcal{T}' \subseteq \mathcal{T} \quad \text{and} \quad \mathsf{dom}(\mathcal{T}') \supseteq D$$

can be constructed in 3ExpTime. *Moreover, if $\mathcal{T}$ is sweeping, then $\mathcal{T}'$ can be constructed in* 2ExpTime.

Proof. Given an input word u, the transducer $\mathcal{T}'$ will guess (and check) a successful run ρ of $\mathcal{T}$ on u, together with a $\boldsymbol{B}$-decomposition $\prod_i \rho[\ell_i, \ell_{i+1}]$. The latter decomposition will be used by $\mathcal{T}'$ to simulate the output of ρ in left-to-right manner, thus proving that $\mathcal{T}' \subseteq \mathcal{T}$. Moreover, $u \in D$ implies the existence of a successful run that can be decomposed, thus proving that $\mathsf{dom}(\mathcal{T}') \supseteq D$. We now provide the details of the construction of $\mathcal{T}'$.

Guessing the run ρ is standard (see, for instance, [51, 78]): it amounts to guess the crossing sequences $\rho|x$ for each position x of the input. Recall that this is a bounded amount of information for each position x, since the run is normalized. As concerns the decomposition of ρ, it can be encoded by the endpoints ℓ_i of its factors, that is, by annotating the position of each ℓ_i as the level of ℓ_i. In a similar way $\mathcal{T}'$ guesses the information of whether each factor $\rho[\ell_i, \ell_{i+1}]$ is a $\boldsymbol{B}$-diagonal or a $\boldsymbol{B}$-block.

Thanks to the definition of decomposition (Definition 3.5.2), every two distinct factors span across non-overlapping intervals of positions. This means that each position x is covered by exactly one factor of the decomposition. We call this factor the *active factor at position x*. The mode of computation of the transducer will depend on the type of active factor: if the active factor is a diagonal (resp. a block), then we say that $\mathcal{T}'$ is in *diagonal mode* (resp. *block mode*). Below we describe the behaviour for these two modes of computation.

Diagonal mode. We recall the key condition satisfied by the diagonal $\rho[\ell, \ell']$ that is active at position x (cf. Definition 3.5.1 and Figure 3.16): there is a location $\ell_x = (x, y_x)$ between ℓ and ℓ' such that the words $\mathsf{out}(\rho|Z^{\nwarrow}_{\ell_x})$ and $\mathsf{out}(\rho|Z^{\searrow}_{\ell_x})$ have length at most $\boldsymbol{B}$, where $Z^{\nwarrow}_{\ell_x} = [\ell_x, \ell'] \cap \big([0, x] \times \mathbb{N}\big)$ and $Z^{\searrow}_{\ell_x} = [\ell, \ell_x] \cap \big([x, \omega] \times \mathbb{N}\big)$.

Besides the run ρ and the decomposition, the transducer $\mathcal{T}'$ will also guess the locations $\ell_x = (x, y_x)$, that is, will annotate each x with the corresponding y_x. Without loss of generality, we can assume that the function that associates each position x with the guessed location $\ell_x = (x, y_x)$ is monotone, namely, $x \leqslant x'$ implies $\ell_x \trianglelefteq \ell_{x'}$. While the transducer $\mathcal{T}'$ is in diagonal mode, the goal is to preserve the following invariant:

> *After reaching a position x covered by the active diagonal, $\mathcal{T}'$ must have produced the output of ρ up to location ℓ_x.*

To preserve the above invariant when moving from x to the next position $x+1$, the transducer should output the word $\mathsf{out}(\rho[\ell_x, \ell_{x+1}])$. This word consists of the following parts:

1. The words produced by the single transitions of $\rho[\ell_x, \ell_{x+1}]$ with endpoints in $\{x, x+1\} \times \mathbb{N}$. Note that there are at most $\boldsymbol{H}$ such words, each of them has length at most c_{max}, and they can all be determined using the crossing sequences at x and $x+1$ and the information about the levels of ℓ_x and ℓ_{x+1}. We can thus assume that this information is readily available to the transducer.

2. The words produced by the factors of $\rho[\ell_x, \ell_{x+1}]$ that are intercepted by the interval $[0, x]$. Thanks to the definition of diagonal, we know that the total length of these words is at most $\boldsymbol{B}$. These words cannot be determined from the information on $\rho|x$, $\rho|x+1$, ℓ_x, and ℓ_{x+1} alone, so they need to be constructed while scanning the input. For this, some additional information needs to be stored.

   More precisely, at each position x of the input, the transducer stores all the outputs produced by the factors of ρ that are intercepted by $[0, x]$ and that occur *after* a location of the form $\ell_{x'}$, for any $x' \geqslant x$ that is covered by a diagonal. This clearly includes the previous words when $x' = x$, but also other words that might be used later for processing other diagonals. Moreover, by exploiting the properties of diagonals, one can prove that those words have length at most $\boldsymbol{B}$, so they can be stored with triply exponentially many states. Using classical techniques, the stored information can be maintained while scanning the input u using the guessed crossing sequences of ρ.

3. The words produced by the factors of $\rho[\ell_x, \ell_{x+1}]$ that are intercepted by the interval $[x+1, \omega]$. These words must be guessed, since they depend on a portion of the input that has not been processed yet. Accordingly, the guesses need to be stored into memory, in such a way that they

can be checked later. For this, the transducer stores, for each position x, the guessed words that correspond to the outputs produced by the factors of ρ intercepted by $[x,\omega]$ and occurring *before* a location of the form $\ell_{x'}$, for any $x' \leqslant x$ that is covered by a diagonal.

Block mode. Suppose that the active factor $\rho[\ell,\ell']$ is a $\boldsymbol{B}$-block. Let $I = [x,x']$ be the set of positions covered by this factor. Moreover, for each position $z \in I$, let $Z^{\leftarrow}_z = [\ell,\ell'] \cap \big([0,z] \times \mathbb{N}\big)$ and $Z^{\rightarrow}_z = [\ell,\ell'] \cap \big([z,\omega] \times \mathbb{N}\big)$. We recall the key property of a block (cf. Definition 3.5.1 and Figure 3.4): the word $\mathsf{out}(\rho[\ell,\ell'])$ is almost periodic with bound $\boldsymbol{B}$, and the words $\mathsf{out}(\rho|Z^{\leftarrow}_x)$ and $\mathsf{out}(\rho|Z^{\rightarrow}_{x'})$ have length at most $\boldsymbol{B}$.

For the sake of brevity, suppose that $\mathsf{out}(\rho[\ell,\ell']) = w_1\, w_2, w_3$, where w_2 is periodic with period $\boldsymbol{B}$ and w_1, w_2 have length at most $\boldsymbol{B}$. Similarly, let $w_0 = \mathsf{out}(\rho|Z^{\leftarrow}_x)$ and $w_4 = \mathsf{out}(\rho|Z^{\rightarrow}_{x'})$. The invariant preserved by $\mathcal{T}'$ in block mode is the following:

After reaching a position z covered by the active block $\rho[\ell,\ell']$, $\mathcal{T}'$ must have produced the output of the prefix of ρ up to location ℓ, followed by a prefix of $\mathsf{out}(\rho[\ell,\ell']) = w_1\, w_2\, w_3$ of the same length as $\mathsf{out}(\rho|Z^{\leftarrow}_z)$.

The initialization of the invariant is done when reaching the left endpoint x. At this moment, it suffices that $\mathcal{T}'$ outputs a prefix of $w_1\, w_2\, w_3$ of the same length as $w_0 = \mathsf{out}(\rho|Z^{\leftarrow}_x)$, thus bounded by $\boldsymbol{B}$. Symmetrically, when reaching the right endpoint x', $\mathcal{T}'$ will have produced almost the entire word $\mathsf{out}(\rho[\ell,\ell'])\, w_1\, w_2\, w_3$, but without the suffix $w_4 = \mathsf{out}(\rho|Z^{\rightarrow}_{x'})$ of length at most $\boldsymbol{B}$. Thus, before moving to the next factor of the decomposition, the transducer will produce the remaining suffix, so as to complete the output of ρ up to location ℓ_{i_x+1}.

It remains to describe how the above invariant can be maintained when moving from a position z to the next position $z+1$ inside $I = [x,x']$. For this, it is convenient to succinctly represent the word w_2 by its repeating pattern, say v, of length at most $\boldsymbol{B}$. To determine the symbols that have to be output at each step, the transducer will maintain a pointer on either $w_1\, v$ or w_3. The pointer is increased in a deterministic way, and precisely by the amount $|\mathsf{out}(\rho|Z^{\leftarrow}_{z+1})| - |\mathsf{out}(\rho|Z^{\leftarrow}_z)|$. The only exception is when the pointer lies in $w_1\, v$, but its increase would go over $w_1\, v$: in this case the transducer has the choice to either bring the pointer back to the beginning of v (representing a periodic output inside w_2), or move it to w_3. Of course, this is a non-deterministic choice, but it can be validated when reaching the right endpoint of I. Concerning the number of symbols that need to be emitted at each step, this can be determined from the crossing sequences at z and $z+1$, and from the knowledge of the lowest and highest levels of locations that are at position z and between ℓ and ℓ'. We denote the latter levels by y_z^- and y_z^+, respectively.

Overall, this shows how to maintain the invariant of the block mode, assuming that the levels y_z^-, y_z^+ are known, as well as the words w_0, w_1, v, w_3, w_4 of bounded length. Like the mapping $z \mapsto \ell_z = (z, y_z)$ used in diagonal mode, the mapping $z \mapsto (y_z^-, y_z^+)$ can be guessed and checked using the crossing sequences. Similarly, the words w_1, v, w_3 can be guessed just before entering the active block, and can be checked along the process. As concerns the words w_0, w_4, these can be guessed and checked in a way similar to the words that we used in diagonal mode. More precisely, for each position z of the input, the transducer stores the following additional information:

1. the outputs produced by the factors of ρ that are intercepted by $[0, z]$ and that occur after the beginning ℓ'' of some block, with $\ell'' = (x'', y'')$ and $x'' \geqslant z$;

2. the outputs produced by the factors of ρ that are intercepted by $[z, \omega]$ and that occur before the ending ℓ''' of a block, where $\ell''' = (x''', y''')$ and $x''' \leqslant z$.

By the definition of blocks, the above words have length at most $\boldsymbol{B}$ and can be maintained while processing the input and the crossing sequences. Finally, we observe that the words, together with the information given by the lowest and highest levels y_z^-, y_z^+, for both $z = x$ and $z = x'$, are sufficient for determining the content of w_0 and w_4.

We have just shown how to construct a one-way transducer $\mathcal{T}' \subseteq \mathcal{T}$ such that $\mathsf{dom}(\mathcal{T}') \supseteq D$. From the above construction it is easy to see that the number of states and transitions of $\mathcal{T}'$, as well as the number of letters emitted by each transition, are at most exponential in $\boldsymbol{B}$. Since $\boldsymbol{B}$ is doubly exponential in the size of $\mathcal{T}$, this shows that $\mathcal{T}'$ can be constructed from $\mathcal{T}$ in 3ExpTime. Note that the triple exponential complexity comes from the lengths of the words that need to be guessed and stored in the control states, and these lengths are bounded by $\boldsymbol{B}$. However, if $\mathcal{T}$ is a sweeping transducer, then, according to the results proved in Section 3.2, the bound $\boldsymbol{B}$

is simply exponential. In particular, in the sweeping case we can construct the one-way transducer $\mathcal{T}'$ in 2EXPTIME. □

3.5.3 Generality of the construction.

We conclude the section with a discussion on the properties of the one-way transducer $\mathcal{T}'$ constructed from $\mathcal{T}$. Roughly speaking, we would like to show that, even when $\mathcal{T}$ is not one-way definable, $\mathcal{T}'$ is somehow the *best one-way under-approximation of* $\mathcal{T}$. However, strictly speaking, the latter terminology is meaningless: if $\mathcal{T}'$ is a one-way transducer strictly contained in $\mathcal{T}$, then one can always find a better one-way transducer $\mathcal{T}''$ that satisfies $\mathcal{T}' \subsetneq \mathcal{T}'' \subsetneq \mathcal{T}$, for instance by extending $\mathcal{T}'$ with a single input-output pair. Below, we formalize in an appropriate way the notion of "best one-way under-approximation".

We are interested in comparing the domains of transducers, but only up to a certain amount. In particular, we are interested in languages that are preserved under pumping loops of runs of $\mathcal{T}$. Formally, given a language $\mathcal{L}$, we say that $\mathcal{L}$ is $\mathcal{T}$*-pumpable* if $\mathcal{L} \subseteq \mathsf{dom}(\mathcal{T})$ and for all words $u \in \mathcal{L}$, all successful runs ρ of $\mathcal{T}$ on u, all loops L of ρ, and all positive numbers n, the word $\mathsf{pump}_L^n(u)$ also belongs to $\mathcal{L}$. Clearly, the domain $\mathsf{dom}(\mathcal{T})$ of a transducer $\mathcal{T}$ is a regular $\mathcal{T}$-pumpable language.

Another noticeable example of $\mathcal{T}$-pumpable regular language is the domain of the one-way transducer $\mathcal{T}'$, as defined in Proposition 3.5.6. Indeed, $\mathsf{dom}(\mathcal{T}')$ consists of words $u \in \mathsf{dom}(\mathcal{T})$ that induce successful runs with $\boldsymbol{B}$-decompositions, and the property of having a $\boldsymbol{B}$-decomposition is preserved under pumping.

The following result shows that $\mathcal{T}'$ is the best under-approximation of $\mathcal{T}$ within the class of one-way transducers with $\mathcal{T}$-pumpable domains:

Corollary 3.5.7. *Given a functional two-way transducer $\mathcal{T}$, one can construct a one-way transducer $\mathcal{T}'$ such that*

- $\mathcal{T}' \subseteq \mathcal{T}$ *and* $\mathsf{dom}(\mathcal{T}')$ *is* $\mathcal{T}$*-pumpable,*
- *for all one-way transducers* $\mathcal{T}''$*, if* $\mathcal{T}'' \subseteq \mathcal{T}$ *and* $\mathsf{dom}(\mathcal{T}'')$ *is* $\mathcal{T}$*-pumpable, then* $\mathcal{T}'' \subseteq \mathcal{T}'$*.*

Proof. The transducer $\mathcal{T}'$ is precisely the one defined in Proposition 3.5.6. As already explained, its domain $\mathsf{dom}(\mathcal{T}')$ is a $\mathcal{T}$-pumpable language. In particular, $\mathcal{T}'$ satisfies the conditions in the first item.

For the conditions in the second item, consider a one-way transducer $\mathcal{T}'' \subseteq \mathcal{T}$ with a $\mathcal{T}$-pumpable domain $L = \mathsf{dom}(\mathcal{T}'')$. Let $\tilde{\mathcal{T}}$ be the transducer obtained from $\mathcal{T}$ by restricting its domain to L. Clearly, $\tilde{\mathcal{T}}$ is one-way definable, and one could apply Proposition 3.4.4 to $\tilde{\mathcal{T}}$, using $\mathcal{T}''$ as a witness of one-way definability. In particular, when it comes to comparing the outputs of the pumped runs of $\tilde{\mathcal{T}}$ and $\mathcal{T}''$, one could exploit the fact that the domain $\mathcal{L}$ of $\mathcal{T}''$, and hence the domain of $\tilde{\mathcal{T}}$ as well, is $\mathcal{T}$-pumpable. This permits to derive periodicities of inversions with the same bound $\boldsymbol{B}$ as before, but only restricted to the successful runs of $\mathcal{T}$ on the input words that belong to $\mathcal{L}$. As a consequence, one can define $\boldsymbol{B}$-decompositions of successful runs of $\mathcal{T}$ on words in $\mathcal{L}$, thus showing that $\mathcal{L} \subseteq \mathsf{dom}(\mathcal{T}')$. This proves that $\mathcal{T}'' \subseteq \mathcal{T}'$. $\square$

3.6 Complexity of the one-way definability problem

3.6.1 Complexity analysis

In this section we analyze the complexity of the problem of deciding whether a transducer $\mathcal{T}$ is one-way definable. We begin with the case of a functional two-way transducer. In this case, thanks to the results presented in Section 3.5, we know that $\mathcal{T}$ is one-way definable if and only if $\mathsf{dom}(\mathcal{T}) \subseteq D$, where D is the language of words $u \in \mathsf{dom}(\mathcal{T})$ such that all successful runs of $\mathcal{T}$ on u admit a $\boldsymbol{B}$-decomposition. In particular, the one-way definability problem reduces to an emptiness problem for the intersection of two languages:

$$\mathcal{T} \text{ one-way definable} \qquad \text{if and only if} \qquad \mathsf{dom}(\mathcal{T}) \cap D^{\complement} = \varnothing.$$

The following lemma exploits the characterization of Theorem 3.0.3 to show that the language $D^{\complement}$ can be recognized by a non-deterministic finite automaton $\mathcal{D}$ of triply exponential size w.r.t. $\mathcal{T}$. In fact, this lemma shows that the automaton recognizing $D^{\complement}$ can be constructed using doubly exponential *workspace*. As before, we gain an exponent when restricting to sweeping transducers.

Lemma 3.6.1. *Given a functional two-way transducer $\mathcal{T}$, an NFA $\mathcal{D}$ recognizing $D^{\complement}$ can be constructed in* 2ExpSpace. *Moreover, when $\mathcal{T}$ is sweeping, the NFA $\mathcal{D}$ can be constructed in* ExpSpace.

Proof. Consider an input word u. By Theorem 3.0.3 we know that $u \in D^{\complement}$ iff there exist a successful run ρ of $\mathcal{T}$ on u and an inversion $\mathcal{I} = (L_1, \ell_1, L_2, \ell_2)$ of ρ such that no positive number $p \leqslant \boldsymbol{B}$ is a period of the word

$$w_{\rho,\mathcal{I}} \;=\; \mathsf{out}\big(\mathsf{tr}(\ell_1)\big)\ \mathsf{out}\big(\rho[\ell_1,\ell_2]\big)\ \mathsf{out}\big(\mathsf{tr}(\ell_2)\big).$$

The latter condition on $w_{\rho,\mathcal{I}}$ can be rephrased as follows: there is a function $f : \{1,\dots,\boldsymbol{B}\} \to \{1,\dots,|w_{\rho,\mathcal{I}}|\}$ such that $w_{\rho,\mathcal{I}}\big(f(p)\big) \neq w_{\rho,\mathcal{I}}\big(f(p)+p\big)$ for all positive numbers $p \leqslant \boldsymbol{B}$. In particular, each of the images of the latter function f, that is, $f(1),\dots,f(\boldsymbol{B})$, can be encoded by a suitable marking of the crossing sequences of ρ. This shows that the run ρ, the inversion $\mathcal{I}$, and the function f described above can all be guessed within space $\mathcal{O}(\boldsymbol{B})$: ρ is guessed on-the-fly, the inversion is guessed by marking the anchor points, and for f we only store two symbols and a counter $\leqslant \boldsymbol{B}$, for each $1 \leqslant i \leqslant \boldsymbol{B}$. That is, any state of $\mathcal{D}$ requires doubly exponential space, resp. simply exponential space, depending on whether $\mathcal{T}$ is arbitrary two-way or sweeping. □

As a consequence of the previous lemma, the emptiness problem for the language $\mathsf{dom}(\mathcal{T}) \cap D^{\complement}$, and thus the one-way definability problem for $\mathcal{T}$, can be decided in 2ExpSpace or ExpSpace, depending on whether $\mathcal{T}$ is two-way or sweeping:

Proposition 3.6.2. *The problem of deciding whether a functional two-way transducer $\mathcal{T}$ is one-way definable is in* 2ExpSpace. *When $\mathcal{T}$ is sweeping, the problem is in* ExpSpace.

3.6.2 Lower bound

We provide a two-exponential lower bound for the size of the equivalent transducer. As the lower bound is achieved by a sweeping transduction, this gives a tight lower bound on the size of any one-way transducer equivalent to some sweeping transducer.

Proposition 3.6.3. *There is a family $(f_n)_{n\in\mathbb{N}}$ of transductions such that*

1. *f_n can be implemented by a sweeping transducer of size $\mathcal{O}(n^2)$,*

2. *f_n can be implemented by a one-way transducer,*

3. *every one-way transducer that implements f_n has size $\Omega(2^{2^n})$.*

Proof. The family of transformations is precisely the one described in Example 10 (2), where f_n maps inputs of the form $u = a_0\, w_0 \cdots a_{2^n-1}\, w_{2^n-1}$ to outputs of the form $u\,u$, where $a_i \in \{a,b\}$ and $w_i \in \{0,1\}^n$ is the binary encoding of i. A sweeping transducer implementing f_n first checks that the binary encodings w_i, for $i = 0,\dots,2^n-1$, are correct. This can be done with n passes: the j-th pass uses $\mathcal{O}(n)$ states to check the correctness of the j-th bits of the binary encodings. Then, the sweeping transducer performs two additional passes to copy the input twice. Overall, the sweeping transducer has size $\mathcal{O}(n^2)$.

As already mentioned, every one-way transducer that implements f_n needs to remember input words u of exponential length in order to output $u\,u$, which roughly requires doubly exponentially many states. A more formal argument providing a lower bound to the size of a one-way transducer implementing f_n goes as follows.

First of all, one observes that given a one-way transducer $\mathcal{T}$, the language of its outputs, i.e., $L^{\text{out}}_{\mathcal{T}} = \{w \;:\; (u,w) \in \mathscr{L}(\mathcal{T}) \text{ for some } u\}$ is regular. More precisely, if $\mathcal{T}$ has size N, then the language $L^{\text{out}}_{\mathcal{T}}$ is recognized by an automaton of size linear in N. Indeed, while parsing w, the automaton can guess an input word u and a run on u, together with a factorization of w in which the i-th factor corresponds to the output of the transition on the i-th letter of u. Basically, this requires storing as control states the transition rules of $\mathcal{T}$ and the suffixes of outputs.

Now, suppose that the function f_n is implemented by a one-way transducer $\mathcal{T}$ of size N. The language $L^{\text{out}}_{\mathcal{T}} = \{u\,u \;:\; u \in \mathsf{dom}(f_n)\}$ is then recognized by an automaton of size $\mathcal{O}(N)$. Finally, we recall a result from [45], which shows that, given a sequence of pairs of words (u_i, v_i), for $i = 1,\dots,M$, every non-deterministic automaton that separates the language $\{u_i\, v_i \;:\; 1 \leqslant i \leqslant M\}$ from the language $\{u_i\, u_j \;:\; 1 \leqslant i \neq j \leqslant M\}$ must have at least M states. By applying this result to our language $L^{\text{out}}_{\mathcal{T}}$, where $u_i = v_i$ for all $i = 1,\dots,M = 2^{2^n}$, we get that N must be at least linear in M, and hence $N \in \Omega(2^{2^n})$. □

3.6.3 Undecidability of the general case

We conclude this chapter by showing that when the restriction to functional transducers is removed, the problem becomes undecidable.

Proposition 3.6.4. *The one-way definability problem for* non-functional *sweeping transducers is undecidable.*

Proof. The proof uses some ideas and variants of constructions provided in [52], concerning the proof of undecidability of the equivalence problem for one-way non-functional transducers.

We show a reduction from the Post Correspondence Problem (PCP). A *PCP instance* is described by two finite alphabets Σ and Δ and two morphisms $f, g : \Sigma^* \to \Delta^*$. A *solution* of such an instance is any non-empty word $w \in \Sigma^+$ such that $f(w) = g(w)$. We recall that the problem of testing whether a PCP instance has a solution is undecidable.

Below, we fix a tuple $\tau = (\Sigma, \Delta, f, g)$ describing a PCP instance and we show how to reduce the problem of testing the *non-existence of solutions* of τ to the problem of deciding *one-way definability* of a relation computed by a sweeping transducer. Roughly, the idea is to construct a relation B_τ between words over a suitable alphabet Γ that encodes all the *non-solutions* to the PCP instance τ (this is simpler than encoding solutions because the presence of errors can be easily checked). The goal is to have a relation B_τ that (i) can be computed by a sweeping transducer and (ii) coincides with a trivial one-way definable relation when τ has no solution.

We begin by describing the encodings for the solutions of the PCP instance. We assume that the two alphabets of the PCP instance, Σ and Δ, are disjoint and we use a fresh symbol $\# \notin \Sigma \cup \Delta$. We define the new alphabet $\Gamma = \Sigma \cup \Delta \cup \{\#\}$ that will serve both as input alphabet and as output alphabet for the transduction. We call *encoding* any pair of words over Γ of the form $(w \cdot u, w \cdot v)$, where $w \in \Sigma^+$, $u \in \Delta^*$, and $v \in \{\#\}^*$. We will write the encodings as vectors to improve readability, e.g., as

$$\begin{pmatrix} w \cdot u \\ w \cdot v \end{pmatrix} .$$

We denote by E_τ the set of all encodings and we observe that E_τ is computable by a one-way transducer (note that this transducer needs ε-transitions). We then restrict our attention to the pairs in E_τ that are encodings of valid solutions of the PCP instance. Formally, we call *good encodings* the pairs in E_τ of the form

$$\begin{pmatrix} w \cdot u \\ w \cdot \#^{|u|} \end{pmatrix} \qquad \text{where } u = f(w) = g(w) \ .$$

All the other pairs in E_τ are called *bad encodings*. Of course, the relation that contains the good encodings is not computable by a transducer. On the other hand, we can show that the complement of this relation w.r.t. E_τ is computable by a sweeping transducer. Let B_τ be the set of all bad encodings. Consider $(w{\cdot}u, w{\cdot}\#^m) \in E_\tau$, with $w \in \Sigma^+$, $u \in \Delta^*$, and $m \in \mathbb{N}$, and we observe that this pair belongs to B_τ if and only if one of the following conditions is satisfied:

1. $m < |u|$,
2. $m > |u|$,
3. $u \neq f(w)$,
4. $u \neq g(w)$.

We explain how to construct a sweeping transducer $\mathcal{S}_\tau$ that computes B_τ. Essentially, $\mathcal{S}_\tau$ guesses which of the above conditions holds and processes the input accordingly. More precisely, if $\mathcal{S}_\tau$ guesses that the first condition holds, then it performs a single left-to-right pass, first copying the prefix w to the output and then producing a block of occurrences of the symbol # that is shorter than the suffix u. This task can be easily performed while reading u: it suffices to emit at most one occurrence of # for each position in u, and at the same time guarantee that, for at least one such position, no occurrence of # is emitted. The second condition can be dealt with by a similar strategy: first copy the prefix w, then output a block of # that is longer than the suffix u. To deal with the third condition, the transducer $\mathcal{S}_\tau$ has to perform two left-to-right passes, interleaved by a backward pass that brings the head back to the initial position. During the first left-to-right pass, $\mathcal{S}_\tau$ copies the prefix w to the output. During the second left-to-right pass, it reads again the prefix w, but this time he guesses a factorization of it of the form $w_1\, a\, w_2$. On reading w_1, $\mathcal{S}_\tau$ will output $\#^{|f(w_1)|}$. After reading w_1, $\mathcal{S}_\tau$ will store the symbol a and move to the position where the suffix u begins. From there, it will guess a factorization of u of the form $u_1\, u_2$, check that u_2 does not begin with $f(a)$, and emit one occurrence of # for each position in u_2. The number of occurrences of # produced in the output is thus $m = |f(w_1)| + |u_2|$, and the fact that u_2 does not begin with $f(a)$ ensures that the factorizations of w and u do not match, i.e.

$$m \neq |f(w)|$$

Note that the described behaviour does not immediately guarantee that $u \neq f(w)$. Indeed, it may still happen that $u = f(w)$, but as a consequence $m \neq |u|$. This case is already covered by the first and second condition, so the computation is still correct in the sense that it produces only bad encodings. On the other hand, if m happens to be the same as $|u|$, then $|u| = m \neq |f(w)|$ and thus $u \neq f(w)$. A similar behaviour can be used to deal with the fourth condition.

We have just shown that there is a sweeping non-functional transducer $\mathcal{S}_\tau$ that computes the relation B_τ containing all the bad encodings. Note that, if the PCP instance τ admits no solution, then all encodings are bad, i.e., $B_\tau = E_\tau$, and hence B_τ is one-way definable. It remains to show that when τ has a solution, B_τ is not one-way definable. Suppose that τ has solution $w \in \Sigma^+$ and let $\big(w \cdot u,\ w \cdot \#^{|u|}\big)$ be the corresponding good encoding, where $u = f(w) = g(w)$. Note that every exact repetition of w is also a solution, and hence the pairs $\big(w^n \cdot u^n,\ w^n \cdot \#^{n \cdot |u|}\big)$ are also good encodings, for all $n \geqslant 1$.

Suppose, by way of contradiction, that there is a one-way transducer $\mathcal{T}$ that computes the relation B_τ. For every $n, m \in \mathbb{N}$, we define the encoding

$$\alpha_{n,m} \;=\; \begin{pmatrix} w^n \cdot u^m \\ w^n \cdot \#^{m \cdot |u|} \end{pmatrix}$$

and we observe that $\alpha_{n,m} \in B_\tau$ if and only if $n \neq m$ (recall that $w \neq \varepsilon$ is the solution of the PCP instance τ and $u = f(w) = g(w)$). Below, we consider bad encodings like the above ones, where the parameter n is supposed to be large enough. Formally, we define the set I of all pairs of indices $(n, m) \in \mathbb{N}^2$ such that (i) $n \neq m$ (this guarantees that $\alpha_{n,m} \in B_\tau$) and (ii) n is larger than the number $|Q|$ of states of $\mathcal{T}$.

We consider some pair $(n, m) \in I$ and we choose a successful run $\rho_{n,m}$ of $\mathcal{T}$ that witnesses the membership of $\alpha_{n,m}$ in B_τ, namely, that reads the input $w^n \cdot u^m$ and produces the output $w^n \cdot \#^{m \cdot |u|}$. We can split the run $\rho_{n,m}$ into a prefix $\overleftarrow{\rho}_{n,m}$ and a suffix $\overrightarrow{\rho}_{n,m}$ in such a way that $\overleftarrow{\rho}_{n,m}$ consumes the prefix w^n and $\overrightarrow{\rho}_{n,m}$ consumes the remaining suffix u^m. Since n is larger than the number of state of $\mathcal{T}$, we can find a factor $\hat{\rho}_{n,m}$ of $\overleftarrow{\rho}_{n,m}$ that starts and ends with the same state and consumes a non-empty exact repetition of w, say w^{n_1}, for some $1 \leqslant n_1 \leqslant |Q|$. We claim that the output produced by the factor $\hat{\rho}_{n,m}$ must coincide with the consumed part w^{n_1} of the input. Indeed, if this were not the case, then deleting the factor $\hat{\rho}_{n,m}$ from $\rho_{n,m}$ would result in a new successful run that reads $w^{n-n_1} \cdot u^m$ and produces $w^{n-n_2} \cdot \#^{m \cdot |u|}$ as output, for some $n_2 \neq n_1$. This however would contradict the fact that, by definition of encoding, the possible outputs produced by $\mathcal{T}$ on input $w^{n-n_1} \cdot u^m$ must agree on the prefix w^{n-n_1}. We also remark that, even if we do not annotate this explicitly, the number n_1 depends on the choice of the pair $(n, m) \in I$. This number, however, range over the fixed finite set $J = \big[1, |Q|\big]$.

We can now pump the factor $\hat{\rho}_{n,m}$ of the run $\rho_{n,m}$ any arbitrary number of times. In this way, we obtain new successful runs of $\mathcal{T}$ that consume inputs of the form $w^{n+k \cdot n_1} \cdot u^m$ and produce outputs of the form $w^{n+k \cdot n_1} \cdot \#^m$, for all $k \in \mathbb{N}$. In particular, we know that B_τ contains all pairs of the form $\alpha_{n+k \cdot n_1, m}$. Summing up, we can claim the following:

Claim. *There is a function $h : I \to J$ such that, for all pairs $(n, m) \in I$,*

$$\big\{(n + k \cdot h(n, m), m) \bigm| k \in \mathbb{N}\big\} \subseteq I \ .$$

We can now head towards a contradiction. Let $\tilde{n}$ be the maximum common multiple of the numbers $h(n, m)$, for all $(n, m) \in I$. Let $m = n + \tilde{n}$ and observe that $n \neq m$, whence $(n, m) \in I$. Since $\tilde{n}$ is a multiple of $h(n, m)$, we derive from the above claim that the pair $(n + \tilde{n}, m) = (m, m)$ also belongs to I. However, this contradicts the definition of I, since we observed earlier that $\alpha_{n,m}$ is a bad encoding if and only if $n \neq m$. We conclude that B_τ is not one-way definable when τ has a solution. $\square$

Chapter 4

Minimization of resources

Minimization problems have been studied for obvious reasons since the early days of automata theory. For example, several algorithms have been proposed for minimizing the number of states of (one-way) finite state automata [16, 62, 63]. State minimization algorithms have been also proposed for deterministic *one-way* finite state transducers [21]. However, for *two-way* machines, much less is known about state minimization, and this holds true even for automata.

Yet for two-way (and streaming) transducers, there are other kind of resources than state space that are amenable to minimization. For example, a parameter that should be kept low in order to "simplify" the evaluation process of the transduction is the number of passes performed by a sweeping transducer, namely, the number of times the machine has to read the input in order to produce the correct output. This parameter if particularly important when the input is generated by a remote process and accessed by the transducer, as minimizing the number of passes reduces the number of times the machine has to wait to receive the input information. For similar reasons one may want to minimize the number of registers in a streaming transducer: even though the content of registers may grow arbitrarily, updating a large number of registers simultaneously raises implementation issues.

The number of passes and the number of registers induce strict hierarchies of machines. For example, the k-duplication function $w \mapsto w^k$ is easily computed by a k-pass sweeping transducer and by a k-register $DSST$, but is not definable by a $(k-1)$-pass sweeping transducer, nor a $(k-1)$ register $DSST$. Formal proofs of the latter statements are similar to the proof of Proposition 2.4.4.

In this chapter we provide a generalization of the methods described in Chapter 3 to characterize k-pass definability:

k-PASS DEFINABILITY	
Input:	A *2fNFT* $\mathcal{T}$ and a number k
Question:	Is $\mathcal{T}$ equivalent to some k-pass non-deterministic sweeping transducer $\mathcal{T}'$?

In particular, an answer to the above question allows to minimize the number of passes of a given sweeping transducer. Moreover, we provide a bound on the number of passes for an equivalent sweeping transducer, when one exists. Along with the procedure for deciding k-pass definability, this allows us to characterize definability by some sweeping transducer (when the number of passes is not fixed).

Later, in Section 4.2 we will focus on *NSST* (recall that this stands for non-deterministic streaming string transducers), and consider the number of registers as a natural notion of complexity for such machines. We will aim at minimizing the number of registers for *NSST*'s:

REGISTER MINIMIZATION	
Input:	An *NSST* $\mathcal{T}$ and a number k
Question:	Is $\mathcal{T}$ equivalent to some *NSST* with at most k registers?

Similar problems concerning register minimization have been also considered in [7] and [30], but for rather different settings. In [7] the output alphabet is unary and register concatenation is disallowed, whereas in [30] the outputs lies in an unspecified monoid and the updates only allow multiplication from the right (while in the setting of the free monoid, our model allows adding words at the right and the left).

We also remark that the minimum number of registers required to implement a given transduction depends on whether we allow or not non-determinism. For example, the function $u \to a^{|u|}$, where a is the last letter of u, can be implemented by an *NSST* with just 1 register, but any equivalent *SST* will require as many registers as the size of the input alphabet. In particular, since our register minimization problem allows non-determinism, it is of rather different nature than those considered in [7, 30].

The register minimization problem for *NSST* is challenging, and we are not able to solve it in full generality. However, we do provide a solution for a subclass of *NSST*'s that is obtained by disallowing register concatenation. Intuitively, the solution exploits a suitable correspondence between the number of registers used by concatenation-free *NSST*'s and the number of passes performed by sweeping transducers. We will use this correspondence and the minimization of the number of passes of a sweeping transducer to derive a minimization procedure for the number of registers in the restricted class of *NSST*'s.

4.1 Passes of a sweeping transducer

In this section we show how to decide in 2ExpSpace the k-pass definability problem (recall that this is the problem of telling whether a given two-way transducer is equivalent to some k-pass sweeping transducer). As we will see, when $\mathcal{T}$ is k-pass definable, we also provide a k-pass sweeping transducer $\mathcal{T}'$ of triple exponential size w.r.t. $|\mathcal{T}|$ that is equivalent to $\mathcal{T}$. We reuse the notation $\boldsymbol{B}$ from the previous chapter: let us recall that $\boldsymbol{B}$ is doubly exponential in the size of $\mathcal{T}$ in the general case, and simply exponential when $\mathcal{T}$ is sweeping.

4.1.1 k-pass definability

We begin by defining the objects that need to be considered for characterizing *k-pass definability*. Let $\mathcal{T}$ be a functional two-way transducer. The idea is to identify factors of runs of $\mathcal{T}$ that can be simulated alternatively from left to right and from right to left. We begin by introducing a notion of co-inversion, which can be thought of just as an inversion, but when the input is read from right to left. More precisely, the definition of co-inversion is obtained from that of inversion (Definition 3.4.3) by reversing the order of the positions of the witnessing loops (we highlight in bold this difference):

Definition 4.1.1. *A* co-inversion *of a run ρ is a tuple $(L_1, \ell_1, L_2, \ell_2)$ such that:*

- *L_1, L_2 are idempotent loops,*
- *$\ell_1 = (x_1, y_1)$ and $\ell_2 = (x_2, y_2)$ are anchor points inside L_1 and L_2, respectively,*

- $\ell_1 \lhd \ell_2$ *and* $\boldsymbol{x_1} < \boldsymbol{x_2}$,
- $\mathsf{out}(\mathsf{tr}(\ell_i)) \neq \epsilon$.

We say that the above loops L_1 and L_2 are the witnessing loops *of the co-inversion $(L_1, \ell_1, L_2, \ell_2)$.*

We then combine inversions and co-inversions, as follows:

Definition 4.1.2. *A* k-inversion *of ρ is a sequence $\overline{\ell} = (L_1, \ell_1, L_2, \ell_2), \ldots, (L_{2k-1}, \ell_{2k-1}, L_{2k}, \ell_{2k})$ such that:*

- *$\ell_1 \lhd \ell_2 \lhd \ldots \lhd \ell_{2k-1} \lhd \ell_{2k}$ are distinct locations in ρ,*
- *for all even i such that $0 \leqslant i < k$, $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ is an inversion of ρ,*
- *for all odd i such that $0 \leqslant i < k$, $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ is a co-inversion of ρ.*

An example of a 3-inversion is depicted on Figure 4.1 (we did not show the loops to keep the figure readable).

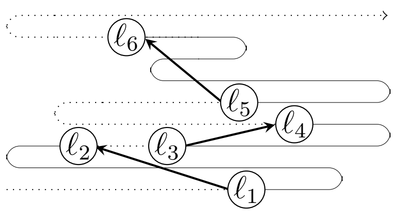


Figure 4.1: An example of a 3-inversion.

Definition 4.1.3. *We say that $\overline{\ell}$ is $\boldsymbol{B}$*-safe *if $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ has period at most $\boldsymbol{B}$, for some $i \in \{0, \ldots, k-1\}$. We denote by $L_{\mathcal{T}}^{(k)}$ the language of words $u \in \mathsf{dom}(\mathcal{T})$ such that all k-inversions of all successful runs of $\mathcal{T}$ on u are $\boldsymbol{B}$*-safe.

Example 11. Consider a 3-pass transducer $\mathcal{T}$ that receives an input $u\#v$, with $u, v \in \{a, b\}^*$, and outputs $(ab)^{|uvv|}(ba)^{|uuv|}$. The 2-inversion of $\mathcal{T}$ depicted on Figure 4.2 is $\boldsymbol{B}$-safe, as the output $\rho[\ell_3, \ell_4]$ has period 2. In fact we can see that every 2-inversion of $\mathcal{T}$ is $\boldsymbol{B}$-safe because either ℓ_2 belongs to

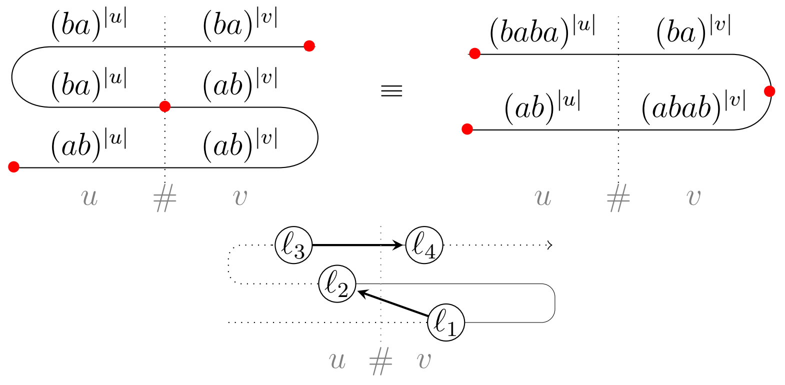


Figure 4.2: A run of $\mathcal{T}$ with 3 passes, its equivalent with 2 passes, and a $\boldsymbol{B}$-safe 2-inversion of $\mathcal{T}$.

the part producing ab or ℓ_3 belongs to the part producing ba (as in Figure 4.2). Thus the realized transduction can also be implemented in 2 passes: one needs to output $abab$ for each letter of v on the first pass and $baba$ for each letter of u on the second pass (as we can see in Figure 4.2) .

Note that the definition of 1-inversion is the same as Definition 3.4.3. In particular, by Theorem 3.0.3, we know that $\mathcal{T}$ is one-way definable iff $L^{(1)}_{\mathcal{T}} = \mathsf{dom}(\mathcal{T})$. The generalization of this result is provided in Theorem 4.1.4 below: k-pass definability is equivalent to asking that every k-inversion is $\boldsymbol{B}$-safe, in the same way as one-way definability is equivalent to asking that every inversion has output with period at most $\boldsymbol{B}$ (property **P2**, see Proposition 3.4.4).

Theorem 4.1.4. *A functional two-way transducer $\mathcal{T}$ is k-pass sweeping definable iff $L^{(k)}_{\mathcal{T}} = \mathsf{dom}(\mathcal{T})$, and this can be decided in* 2ExpSpace *w.r.t. $|\mathcal{T}|$ and in polynomial space w.r.t. k. Moreover, given a* 2fNFT *$\mathcal{T}$, one can construct in* 3ExpTime *an unambiguous k-pass sweeping transducer $\mathcal{T}'$ equivalent to $\mathcal{T}|_{L^{(k)}_{\mathcal{T}}}$.*

If the given transducer is already sweeping, the decision procedure is in ExpSpace *and the construction is in* 2ExpTime.

The proof of Theorem 4.1.4 is split into two parts. The first part deals with the construction of the k-pass sweeping transducer $\mathcal{T}'$ of the second claim in Section 4.1.2. Since $L^{(k)}_{\mathcal{T}} = \mathsf{dom}(\mathcal{T})$ implies that $\mathcal{T}'$ is equivalent to $\mathcal{T}$, this construction also proves the right-to-left direction of the first claim.

Moreover, as a side result, we prove that whether $L_{\mathcal{T}}^{(k)} = \mathsf{dom}(\mathcal{T})$ holds is decidable in 2EXPSPACE. We will show in Section 4.1.3 the other direction of the first claim. The additional difficulty regarding k-inversions is that in order to work out the combinatorics, we have to carefully separate the (co-)inversions into separated one-way factors. The remark about the case of sweeping transducers will be a consequence of the better complexity obtained for sweeping transducers in Chapter 3.

4.1.2 Soundness

We show how to construct from $\mathcal{T}$ a k-pass sweeping transducer $\mathcal{T}'$ equivalent to $\mathcal{T}|_{L_{\mathcal{T}}^{(k)}}$. The idea is to consider a successful run ρ of $\mathcal{T}$ on a word $u \in L_{\mathcal{T}}^{(k)}$, and divide it into k factors. We then simulate each factor of the run in a single pass, alternatively from left to right and from right to left, using Theorem 3.0.1.

First we need a notion of *k-$\boldsymbol{B}$-factorization* that can be used to separate a run into k parts, each of them implementable by a one-way transducer (either from left to right or from right to left).

Definition 4.1.5. *A k-$\boldsymbol{B}$-*factorization *of a successful run ρ of $\mathcal{T}$ is any sequence of locations $\bar{\ell} = \ell_0, \ell_1, \ldots, \ell_k$ of ρ such that:*

- *$\ell_0 \lhd \ell_1 \lhd \cdots \lhd \ell_k$, ℓ_0 is the first location of ρ, and ℓ_k is the last location of ρ,*
- *for all even indexes i, with $0 \leqslant i < k$, and all inversions (L, ℓ, L', ℓ') of ρ, with $\ell_i \lhd \ell \lhd \ell' \lhd \ell_{i+1}$, the word* $\mathsf{out}(\rho[\ell, \ell'])$ *has period at most $\boldsymbol{B}$,*
- *for all odd indexes i, with $1 \leqslant i < k$, and all co-inversions (L, ℓ, L', ℓ') of ρ, with $\ell_i \lhd \ell \lhd \ell' \lhd \ell_{i+1}$, the word* $\mathsf{out}(\rho[\ell, \ell'])$ *has period at most $\boldsymbol{B}$.*

Example 12. We consider the transducer of Example 11 and we depict a 2-$\boldsymbol{B}$-factorization of a run of it (gray nodes) in Figure 4.3. All the inversions between ℓ_0 and ℓ_1, and all the co-inversions between ℓ_1 and ℓ_2, have period bounded by 2. Note that for every $\boldsymbol{B}$ there exists some run that does not admit a 1-$\boldsymbol{B}$-factorization.

The following lemma shows that we can reason equally in terms of $\boldsymbol{B}$-safe k-inversions (Definition 4.1.2) and in terms of k-$\boldsymbol{B}$-factorizations.

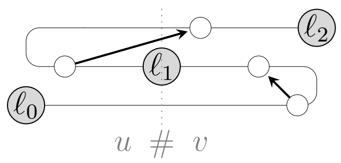


Figure 4.3: A 2-$\boldsymbol{B}$-factorization of $\mathcal{T}$

Lemma 4.1.6. *For every word $u \in \mathsf{dom}(\mathcal{T})$, we have that $u \in L_{\mathcal{T}}^{(k)}$ if and only if every successful run of $\mathcal{T}$ on u admits some k-$\boldsymbol{B}$-factorization.*

Proof. We prove the left-to-right direction. Let $u \in L_{\mathcal{T}}^{(k)}$ and let ρ be a successful run of $\mathcal{T}$ on u. We define the locations $\ell_0, \ell_1, \ldots, \ell_k$ forming a k-$\boldsymbol{B}$-factorization of ρ inductively. The location ℓ_0 is the first of the run. Let us now assume that we have defined the locations up to ℓ_i, with $i < k$. We distinguish two cases depending on the parity of i. If i is even, then we look at the inversions (L, ℓ, L', ℓ') of ρ such that $\ell_i \lhd \ell$ and the period of $\mathsf{out}(\rho[\ell, \ell'])$ is strictly larger than $\boldsymbol{B}$. For short, we call such inversions *bad inversions after ℓ_i*. If there are no bad inversions after ℓ_i, then we simply define ℓ_{i+1} to be the last location of the run. Otherwise, we take the first bad inversion after ℓ_i, following the lexicographic order on pairs of locations, and we denote it by $(\tilde{L}_i, \tilde{\ell}_i, \tilde{L}'_i, \tilde{\ell}'_i)$. Accordingly, we define ℓ_{i+1} to be the immediate predecessor of $\tilde{\ell}'_i$ in the run ρ. The case where i is odd is dealt with in a similar way, by considering co-inversions instead of inversions.

We verify that the constructed sequence $\ell_0, \ell_1, \ldots, \ell_k$ is indeed a k-$\boldsymbol{B}$-factorization. By definition, for all even (resp. odd) indexes $0 \leqslant i < k$ and all inversions (resp. co-inversions) (L, ℓ, L', ℓ'), with $\ell_i \lhd \ell \lhd \ell' \lhd \ell_{i+1}$, the period of $\mathsf{out}(\rho[\ell, \ell'])$ is at most $\boldsymbol{B}$ otherwise (L, ℓ, L', ℓ') would be a bad inversion after ℓ_i smaller (in lexicograpical order) than $(\tilde{L}_i, \tilde{\ell}_i, \tilde{L}'_i, \tilde{\ell}'_i)$. By construction, ℓ_0 is the first location of ρ, and $\ell_0 \lhd \ell_1 \lhd \cdots \lhd \ell_k$. It remains to prove that ℓ_k is the last location of ρ. Suppose that this is not the case. This can happen only if there is still some bad inversion or co-inversion after ℓ_k (depending on k's parity), so $(\tilde{L}_i, \tilde{\ell}_i, \tilde{L}'_i, \tilde{\ell}'_i)$ is defined for all $i = 1, \ldots, k$. The sequence $(\tilde{L}_1, \tilde{\ell}_1, \tilde{L}'_1, \tilde{\ell}'_1), \ldots, (\tilde{L}_k, \tilde{\ell}_k, \tilde{L}'_k, \tilde{\ell}'_k)$ is clearly a k-inversion. Moreover, it is not $\boldsymbol{B}$-safe, because the period of every word $\mathsf{out}(\rho[\tilde{\ell}_i, \tilde{\ell}'_i])$ exceeds $\boldsymbol{B}$. However, this contradicts the hypothesis that ρ is a successful run on $u \in L_{\mathcal{T}}^{(k)}$. We thus conclude that $\overline{\ell} = \ell_0, \ell_1, \ldots, \ell_k$ is a k-$\boldsymbol{B}$-factorization of ρ.

We now prove the converse direction. Fix a word u such that all successful runs on u admit k-$\boldsymbol{B}$-factorizations. Consider a successful run ρ on u and a k-inversion $\overline{\ell'} = (L'_1, \ell'_1, L'_2, \ell'_2), \dots, (L'_{2k-1}, \ell'_{2k-1}, L'_{2k}, \ell'_{2k})$ of ρ. The goal is to prove that the k-inversion is $\boldsymbol{B}$-safe. By assumption, ρ admits a k-$\boldsymbol{B}$-factorization, say $\overline{\ell} = \ell_0, \dots, \ell_k$. As the locations of the k-inversion are ordered, i.e. $\ell'_1 \leqslant \ell'_2 \leqslant \dots \leqslant \ell'_{2k-1} \leqslant \ell'_{2k}$, there is an index $i \in \{0, \dots, k-1\}$ such that $\ell_i \leqslant \ell'_{2i+1} \leqslant \ell'_{2i+2} \leqslant \ell_{i+1}$. By definition of k-$\boldsymbol{B}$-factorization, this means that the period of $\mathsf{out}(\rho[\ell'_{2i+1}, \ell'_{2i+2}])$ is at most $\boldsymbol{B}$, and hence the k-inversion $\overline{\ell'}$ is $\boldsymbol{B}$-safe. As we have chosen ρ and the k-inversion arbitrarily, we conclude that $u \in L^{(k)}_{\mathcal{T}}$. □

We can notice that, for $k = 1$, $u \in L^{(1)}_{\mathcal{T}}$ implies that $u \in \mathsf{dom}(\mathcal{T}')$, where $\mathcal{T}'$ is the transducer described in Chapter 3. For short, we let $L_{\mathcal{T}} = L^{(1)}_{\mathcal{T}}$, and define a symmetric version of this language: $R_{\mathcal{T}}$ is the set of inputs for which all the co-inversions have a period at most $\boldsymbol{B}$.

Encodings of the factors. Next we show that being a k-$\boldsymbol{B}$-factorization is a *regular* property. To formalize this, we need to explain how to encode runs and sequences of locations as annotations of the underlying input. Formally, given a word $u \in \mathsf{dom}(\mathcal{T})$, a successful run ρ of $\mathcal{T}$ on u, and a sequence of locations $\overline{\ell} = \ell_1 \lhd \ell_2 \dots \lhd \ell_m$ in ρ, we denote by $\langle u, \rho, \overline{\ell} \rangle$ the word obtained by annotating each position $1 \leqslant x < |u|$ with the crossing sequence $\rho|x$ and with the m-tuple $\overline{y} = (y_1(x), \dots, y_m(x))$, where each $y_i(x)$ is either the level of ℓ_i (which is bounded by the crossing number of the transducer) or $\perp$, depending on whether ℓ_i is at position x or not. Based on this encoding, we can define the language of factorizations, and define transducers that produce some parts of such factorizations:

Definition 4.1.7. *Let $\mathcal{T}$ be a functional two-way transducer.*

- *We denote by $F^{(k)}_{\mathcal{T}}$ the set of all words of the form $\langle u, \rho, \overline{\ell} \rangle$, where ρ is a successful run of $\mathcal{T}$ on u and $\overline{\ell} = \ell_0, \dots, \ell_k$ is a k-$\boldsymbol{B}$-factorization of ρ.*
- *We denote by $\mathcal{T}_i$ the two-way transducer that reads words of the form $\langle u, \rho, \overline{\ell} \rangle$ and produces* $\mathsf{out}(\rho[\ell_i, \ell_{i+1}])$.

Note that $\mathcal{T}_i$ does not check that $\overline{\ell}$ is a factorization, so its size is polynomial in $|\mathcal{T}|$.

Lemma 4.1.8 below proves that the language $F^{(k)}_{\mathcal{T}}$ is regular. In fact, in order to better handle the complexity of our characterization, the lemma shows that both $F^{(k)}_{\mathcal{T}}$ and its complement $F^{(k)^{\complement}}_{\mathcal{T}}$ are recognized by automata of triple-exponential size.

Lemma 4.1.8. *The language $F^{(k)}_{\mathcal{T}}$ and its complement $F^{(k)^{\complement}}_{\mathcal{T}}$ are recognized by non-deterministic finite state automata of size triply exponential w.r.t. $\mathcal{T}$. Moreover, both automata can be constructed on-the-fly in double-exponential space.*

Proof. To prove that $F^{(k)}_{\mathcal{T}}$ is recognized by an automaton of doubly exponential size, we rely again on k applications of Theorem 3.0.1. Recall that $L^{(1)}_{\mathcal{T}}$ is the language of words that induce successful runs such that, for all inversions (L, ℓ, L', ℓ'), the period of $\mathsf{out}(\rho[\ell, \ell'])$ is at most $\boldsymbol{B}$. Theorem 3.0.3 shows that $L^{(1)}_{\mathcal{T}}$ is the domain of a one-way transducer $\mathcal{T}'$ that can be constructed from $\mathcal{T}$ in triple exponential time, so $L^{(1)}_{\mathcal{T}}$ is recognized by an automaton of triple exponential size w.r.t. $\mathcal{T}$. We can apply this theorem to the transducer $\mathcal{T}_i$ for any i and obtain an automaton that recognizes the language of words $\langle u, \rho, \overline{\ell}\rangle$ for which the inversions (or co-inversions depending on the parity of i) between ℓ_i and ℓ_{i+1} have a period bounded by $\boldsymbol{B}$.

This means that to recognize $F^{(k)}_{\mathcal{T}}$, it is sufficient to construct an automaton that reads a word $\langle u, \rho, \overline{\ell}\rangle$ and checks that (i) ρ is a successful run of $\mathcal{T}$ on u, (ii) $\ell_0 \lhd \ldots \lhd \ell_k$ are locations that delimit factors of ρ, and (iii) for every $i \in \{0, \ldots, k-1\}$, $\langle u, \rho, \overline{\ell}\rangle$ belongs to either $L_{\mathcal{T}_i}$ or $R_{\mathcal{T}_i}$, depending on the parity of i. A close inspection to the above constructions shows that all the automata can be produced in triple exponential time w.r.t. $|\mathcal{T}|$.

We now show that the complement $F^{(k)^{\complement}}_{\mathcal{T}}$ can also be recognized by an automaton of triple exponential size.

Indeed, checking that a word $\langle u, \rho, \overline{\ell}\rangle$ does *not* belong to $F^{(k)}_{\mathcal{T}}$ boils down to verifying that one the following conditions holds (a similar technique was used in the proof of Lemma 3.6.1):

1. ρ is not a successful run on u. This can be checked by looking at the crossing sequences annotated on the positions of u, using an exponential number of states.

2. The locations in $\overline{\ell}$ do not define a factorization of ρ. This can be easily checked with polynomially many states.

3. There exist an index $i \in \{0, \ldots, k-1\}$ and an inversion (L, ℓ, L', ℓ') (or a co-inversion, depending on the parity of i), with $\ell_i \lhd \ell \lhd \ell' \leqslant \ell_{i+1}$,

such that, for all periods $0 \leqslant p \leqslant \boldsymbol{B}$, $\mathsf{out}(\rho[\ell,\ell'])[z] \neq \mathsf{out}(\rho[\ell,\ell'])[z+p]$ for some position z. Note that this is equivalent to saying that $\overline{\ell}$ is not a k-factorization. The latter condition can be checked by guessing i, (ℓ,ℓ'), and a function that maps numbers $p \in \{0,\ldots,\boldsymbol{B}\}$ to positions z_p in $\mathsf{out}(\rho[\ell,\ell'])$. This can be done with triply exponentially many states. □

We conclude this paragraph by showing how to decide in double exponential space if $L_{\mathcal{T}}^{(k)} = \mathsf{dom}(\mathcal{T})$. In fact, as we already know that $L_{\mathcal{T}}^{(k)} \subseteq \mathsf{dom}(\mathcal{T})$, it suffices to decide the containment $L_{\mathcal{T}}^{(k)} \supseteq \mathsf{dom}(\mathcal{T})$. We know from Lemma 4.1.6 that the language $L_{\mathcal{T}}^{(k)}$ coincides with the projection of $F_{\mathcal{T}}^{(k)}$ on the underlying words u. Thus, we have

$$L_{\mathcal{T}}^{(k)} \supseteq \mathsf{dom}(\mathcal{T}) \qquad \text{if and only if} \qquad {F_{\mathcal{T}}^{(k)}}^{\complement} \cap D = \varnothing$$

where $D = \{\langle u,\rho,\overline{\ell}\rangle \;:\; u \in \mathsf{dom}(\mathcal{T})\}$. If we simulate the automaton for ${F_{\mathcal{T}}^{(k)}}^{\complement}$ on-the-fly, we can check the emptiness of ${F_{\mathcal{T}}^{(k)}}^{\complement} \cap D$ in 2EXPSPACE.

Obtaining a canonical factorization. The main idea behind the construction of an equivalent k-pass sweeping transducer $\mathcal{T}'$ is to guess a factorization and use Theorem 3.0.1 between consecutives locations of the factorization of the run of $\mathcal{T}$. To do that, we need to guess the *same* run of $\mathcal{T}$ and the *same* k-$\boldsymbol{B}$-factorization of ρ on each pass of $\mathcal{T}'$.

First, we set an arbitrary order on the set of states of $\mathcal{T}$. It induces a lexicographic order on crossing sequences and this order induces another one on finite sequences of crossing sequences. As runs are matching sequences of crossing sequences we obtain an order on runs.

Proposition 4.1.9. *Given a* 2fNFT $\mathcal{T}$*, there exists a* NFA $\mathcal{A}$ *of size doubly exponential in* $\mathcal{T}$ *such that* $\mathcal{A}$ *recognizes the words* $\langle u,\rho,\overline{\ell}\rangle$ *for which* ρ *is the least accepting run on* u *according to the lexicographic order.*

Proof. The construction of $\mathcal{A}$ is a variant of the classical technique from [78] that uses crossing sequences (cf. end of Section 2.1).

First of all, the automaton $\mathcal{A}$ needs to check that the run ρ encoded in its input $\langle u,\rho,\overline{\ell}\rangle$ is a successful run of $\mathcal{T}$ on the word u. Formally, the automaton $\mathcal{A}$ reads two consecutive crossing sequences $c_x = (q_0,\ldots,q_m)$ and $c_{x+1} = (q'_0,\ldots,q'_{m'})$ around the the position x. To check that these crossing sequences correctly represent transitions of a valid run of $\mathcal{T}$ on the x-th letter

a_x of u, $\mathcal{A}$ verifies that there is a flow F such that, for every edge $i \to j$ of F, the following are valid transitions of $\mathcal{T}'$:

- $q_i \xrightarrow{a_x} q'_j$, if both i and j are even.
- $q'_i \xrightarrow{a_{x+1}} q_j$, if both i and j are odd.
- $q_i \xrightarrow{a_x} q_j$, if i is even and j is odd.
- $q'_i \xrightarrow{a_{x+1}} q'_j$, if i is odd and j is even.

In addition, $\mathcal{A}$ checks that the encode run ρ is successful (this can be easily done by inspecting the first and the last crossing sequences).

Finally, the automaton needs to check that ρ is the least among all successful runs of $\mathcal{T}$ on u. For this, $\mathcal{A}$ performs a sort of subset construction on the series of crossing sequences that encode successful runs potentially below ρ. More precisely, for each position x, $\mathcal{A}$ maintains two sets A_x, B_x of pairs of crossing sequences that encode valid transitions of $\mathcal{T}$ on the letter a_x. At each transition, $\mathcal{A}$ discards those pairs in A_x that are strictly above the pair of crossing sequences (c_x, c_{x+1}), and moves from A_x to B_x those pairs that are strictly below (c_x, c_{x+1}). Intuitively, a pair is in A_x if the run encoded up to that position coincides with a portion of ρ; similarly, it falls into B_x if the encoded run is strictly below ρ. All other pairs encode runs above ρ. At the end, the automaton accepts iff the only pairs that remain in the set B_x encode non-successful runs. $\square$

Now we can order the factorizations over the same run ρ, in order to be able to chose later a canonical one (e.g. the maximal). Given two factorizations $\overline{\ell}$ and $\overline{\ell'}$ we say that $\overline{\ell}$ is greater than $\overline{\ell'}$ if there exists i such that $\ell_j = \ell'_j$ for all $j < i$ and $\ell'_i \trianglelefteq \ell_i$. In other words, the maximal k-$\boldsymbol{B}$-factorization is obtained by iteratively defining ℓ_i as the latest location ℓ such that the inversions contained in ℓ_{i-1} and ℓ have a small period. The next proposition explains how one can efficiently guess such a factorization by characterizing it with a simpler form. In order to keep the notations as simple as possible, we will write $\ell + 1$ for the next location after ℓ in the run ρ when there is no ambiguity about the latter.

Proposition 4.1.10. *A k-$\boldsymbol{B}$-factorization $\overline{\ell}$ of a run ρ is maximal among all k-$\boldsymbol{B}$-factorizations of a run ρ if and only if for all $i < k-1$ there is a (co-)inversion*[1] *between ℓ_i and $\ell_{i+1} + 1$ that produces an output with period*

[1] Inversion for even i and co-inversion for odd i.

greater than $\boldsymbol{B}$.
In particular, a maximal k-$\boldsymbol{B}$-factorization can be guessed in double-exponential space.

Proof. If there exists $i < k-1$ such that all (co-) inversions between ℓ_i and $\ell_{i+1}+1$ have period at most $\boldsymbol{B}$, then $\ell_0, \cdots, \ell_i, \ell_{i+1}+1, \dots \ell_{k-1}$ would be a k-$\boldsymbol{B}$-factorization greater than $\overline{\ell}$, which contradicts the assumption. If $\overline{\ell}$ is not maximal, we let $\overline{\ell'}$ be a k-$\boldsymbol{B}$-factorization greater than $\overline{\ell}$, and i be the first index such that $\ell_i \lhd \ell'_i$. Then, $\rho[\ell_{i-1}, \ell_i+1]$ is contained in $\rho[\ell'_{i-1}, \ell'_i]$ and thus contains only (co-)inversions whose output has period bounded by $\boldsymbol{B}$, as $\overline{\ell'}$ is a k-$\boldsymbol{B}$-factorization.

When guessing a k-$\boldsymbol{B}$-factorization $\overline{\ell}$ we can check that it is maximal among all k-$\boldsymbol{B}$-factorizations by verifying that $\overline{\ell}$ is in $F^{(k)}_{\mathcal{T}}$ and that for all i, with $0 < i < k$ (it is important to notice here that k is a constant of the problem), $(\ell_0, \dots, \ell_{i-1}, \ell_i+1, \dots, \ell_{k-1})$ is in ${F^{(k)}_{\mathcal{T}}}^{\complement}$. Thanks to Lemma 4.1.8, this allows us to verify that a k-$\boldsymbol{B}$-factorization is maximal within double exponential space. □

Construction of $\mathcal{T}'$. Here we exploit the previous properties to construct a k-pass sweeping transducer $\mathcal{T}'$ equivalent to $\mathcal{T}|_{L^{(k)}_{\mathcal{T}}}$, as in the claim of Theorem 4.1.4.

Formally, we define $\mathcal{T}'$ from the transducer $\mathcal{T}^{\diamondsuit}$ that reads inputs of the form $\langle u, \rho, \overline{\ell}\rangle$, checks that ρ is a valid run of $\mathcal{T}$ on u and outputs $\mathsf{out}(\rho[u])$. That is $\mathcal{T}'$ is a k-pass transducer that guesses a canonical run ρ and a canonical k-$\boldsymbol{B}$-factorization $\overline{\ell}$, and simulates the transducer $\mathcal{T}^{\diamondsuit}$ on $\langle u, \rho, \overline{\ell}\rangle$, which by construction has the same output as $\mathcal{T}$ on u.

If $\overline{\ell}$ is indeed a k-$\boldsymbol{B}$-factorization then for all even (resp. odd) i, $\langle u, \rho, \overline{\ell}\rangle$ belongs to $L_{\mathcal{T}_i}$ (resp. $R_{\mathcal{T}_i}$). We let $\overrightarrow{\mathcal{T}_i}$ (resp. $\overleftarrow{\mathcal{T}_i}$) e the left-to-right (resp. right-to-left) one-way transducer obtained by applying Theorem 3.0.1 (resp. the mirror of Theorem 3.0.1) to $\mathcal{T}_i$ when i is even (resp. odd). We define the k-pass transducer $\mathcal{T}'$ as the concatenation

$$\overrightarrow{\mathcal{T}_0}(\langle u, \rho, \overline{\ell}\rangle) \cdot \overleftarrow{\mathcal{T}_1}(\langle u, \rho, \overline{\ell}\rangle) \cdot \ldots \cdot \overrightarrow{\mathcal{T}_{k-1}}(\langle u, \rho, \overline{\ell}\rangle).$$

There are two technical details here. The first one is that we want $\mathcal{T}'$ to read an input word u, not an encoding $\langle u, \rho, \overline{\ell}\rangle$ which is the form of the input of the transducers $\mathcal{T}_i$ and thus $\overrightarrow{\mathcal{T}_i}$ and $\overleftarrow{\mathcal{T}_i}$ too. The second one is that ρ and $\overline{\ell}$ must be the canonical run on u and the canonical factorization so that the different passes of $\mathcal{T}'$ are coherent.

We can overcome those problems by guessing canonical encodings $\langle u, \rho, \overline{\ell} \rangle$ in the language $F_{\mathcal{T}}^{(k)}$. This can be done by using Propositions 4.1.10 and 4.1.9, and checking those guesses in parallel to the one-way construction.

This yields to a transducer $\mathcal{T}'$ of triple-exponential size, that is the size of the equivalent transducer of $\mathcal{T}_i$ given by Theorem 3.0.1, and thus doubly-exponential when $\mathcal{T}$ is sweeping. By construction, the transducer $\mathcal{T}'$ that we just constructed is unambiguous.

4.1.3 Completeness

Here we prove the left-to-right direction of the first claim of Theorem 4.1.4: we suppose that $\mathcal{T}$ is a functional two-way transducer and $\mathcal{T}'$ is an equivalent k-pass sweeping transducer, and we derive from this $L_{\mathcal{T}}^{(k)} = \mathsf{dom}(\mathcal{T})$.

We fix, once and for all, a successful run ρ of $\mathcal{T}$ on u and a k-inversion $\overline{\ell} = (L_1, \ell_1, L_2, \ell_2), \ldots, (L_{2k-1}, \ell_{2k-1}, L_{2k}, \ell_{2k})$ of ρ. The goal is to prove that $\overline{\ell}$ is $\boldsymbol{B}$-safe, namely, that the factor of the output produced between the locations of some (co-)inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ of $\overline{\ell}$ is periodic, with period bounded by $\boldsymbol{B}$. The main idea is to try to find a factor $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ that is entirely covered by the output produced along a single pass of the equivalent transducer $\mathcal{T}'$, and apply a suitable generalization of Proposition 3.4.4. Informally, this will be done by pumping the output produced by $\rho[\ell_{2i+1}, \ell_{2i+2}]$ on the loops L_{2i+1} and L_{2i+2}, and similarly, the output produced by $\mathcal{T}'$ along the single pass. Then, by analyzing how the former outputs are covered by the latter outputs, we deduce the periodicity of $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$.

The main difficulty in formalizing the above idea lies in the fact that the k passes of the supposed transducer $\mathcal{T}'$ cannot be easily identified on the run ρ of $\mathcal{T}$. Therefore we need to reason in a proper way about *families* of factors associated with (co-)inversions inside pumped runs. Below, we introduce some terminology and notation to ease this task.

Pumping several loops at the same time.

Recall that $\overline{\ell} = (L_1, \ell_1, L_2, \ell_2), \ldots, (L_{2k-1}, \ell_{2k-1}, L_{2k}, \ell_{2k})$ is a k-inversion of the run ρ. For a given tuple of numbers $\overline{n} = (n_1, \ldots, n_{2k}) \in \mathbb{N}^{2k}$, we define $\rho^{\overline{n}} = \mathsf{pump}_{\overline{L}}^{\overline{n}}(\rho)$, where $\overline{L} = L_1, \ldots, L_{2k}$ and $\overline{n} = n_1, \ldots, n_{2k}$ (recall that this is the run obtained by pumping the loops $L_1, \ldots, L_{2k}$ respectively $n_1, \ldots, n_{2k}$ times, as described in Section 3.3.3). Similarly, we denote by $u^{\overline{n}}$ the input word parsed by the pumped run $\rho^{\overline{n}}$.

Next, we need to map the inversions and the co-inversions of $\overline{\ell}$ on the pumped runs $\rho^{\overline{n}}$. Consider an inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$, for some $i \in \{1, \dots, 2k\}$ (the case of a co-inversion is similar). Recall that when pumping loops in ρ, several copies of the original loops (and therefore the associated locations) may be introduced. In particular, among the possible copies of the inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ that appear in the pumped run $\rho^{\overline{n}}$, we are interested in the maximal one, which is identified by taking the *first* copy $(\tilde{L}_{2i+1}, \tilde{\ell}_{2i+1})$ of (L_{2i+1}, ℓ_{2i+1}) and the *last* copy $(\tilde{L}_{2i+2}, \tilde{\ell}_{2i+2})$ of (L_{2i+2}, ℓ_{2i+2}), following the natural order on positions of the input. For the sake of brevity, we say that $(\tilde{L}_{2i+1}, \tilde{\ell}_{2i+1}, \tilde{L}_{2i+2}, \tilde{\ell}_{2i+2})$ is the inversion of $\rho^{\overline{n}}$ that *corresponds* to $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$.

We can now define the key objects for our reasoning, that is, the factors of the output of a pumped run $\rho^{\overline{n}}$ that correspond in the original run ρ to the factors produced between the locations of the (co-)inversions of $\overline{\ell}$. Formally, for every $2k$-tuple $\overline{n}$ of natural numbers and every index $i = 0, \dots, k-1$, we define

$$v^{\overline{n}}_{\overline{\ell}}(i) \;=\; \mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2i+2}])$$

Note that the above factors depend on the k-inversion $\overline{\ell}$ and on the tuple $\overline{n}$, which represents the number of times we pump each witnessing loop of $\overline{\ell}$. For simplicity, since $\overline{\ell}$ is understood from the context, we will often drop the subscript from the notation, thus writing $v^{\overline{n}}(i)$.

Below we highlight the relevant factors inside the output $\mathcal{T}(u^{\overline{n}})$ produced by $\mathcal{T}$ on input $u^{\overline{n}}$:

$$\mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_0, \tilde{\ell}_1]) \cdot \boldsymbol{v^{\overline{n}}(0)} \cdot \mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_2, \tilde{\ell}_3]) \cdot \ldots \cdot \boldsymbol{v^{\overline{n}}(k-1)} \cdot \mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_{2k}, \tilde{\ell}_{2k+1}]) \quad (4.1)$$

where $\tilde{\ell}_0$ is the first location of $\rho^{\overline{n}}$, $\tilde{\ell}_{2k+1}$ is the last location of $\rho^{\overline{n}}$.

In a similar way, we can factorize the output produced by the k-pass sweeping transducer $\mathcal{T}'$ when reading the input $u^{\overline{n}}$. However, the focus here is on the factors of the output produced along each pass. Formally, given $\overline{n} \in \mathbb{N}^{2k}$, we let $\sigma^{\overline{n}}$ be some successful run of $\mathcal{T}'$ on $u^{\overline{n}}$. For every $j = 0, \dots, k-1$, we let ℓ'_j be the first location of $\sigma^{\overline{n}}$ at level j. We further let ℓ'_k be the last location of $\sigma^{\overline{n}}$, which is at level $k-1$. We then define

$$w^{\overline{n}}(j) \;=\; \mathsf{out}(\sigma^{\overline{n}}[\ell'_j, \ell'_{j+1}])$$

and factorize the output $\mathcal{T}'(u^{\overline{n}})$ of $\mathcal{T}'$ on $u^{\overline{n}}$ as follows:

$$\boldsymbol{w^{\overline{n}}(0)} \cdot \ldots \cdot \boldsymbol{w^{\overline{n}}(k-1)}. \quad (4.2)$$

Below, we show how to exploit the hypothesis that $\mathcal{T}$ and $\mathcal{T}'$ are equivalent and some combinatorial arguments to prove that the original k-inversion is $\boldsymbol{B}$-safe.

Combinatorics.

As $\mathcal{T}$ and $\mathcal{T}'$ are equivalent we know that Equations (4.1) and (4.2) represent the same word. From this we derive that, for every $\overline{n} \in \mathbb{N}^{2k}$, at least one of the words $v^{\overline{n}}(i)$ highlighted in Equation (4.1) is a factor of the word $w^{\overline{n}}(i)$ highlighted in Equation (4.2). However, the index i for which this coverability relation holds depends on the parameter $\overline{n}$. In order to enable a reasoning similar to that of Proposition 3.4.4, we need to find a single index i such that, for "sufficiently many" parameters $\overline{n}$, $v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$. The definition below, formalizes what we mean precisely by "sufficiently many" $\overline{n}$ — intuitively, we require that specific coordinates of $\overline{n}$ are unbounded, and the differences between these coordinates as well.

Definition 4.1.11. *Let $\mathcal{P}(\overline{n})$ denote an arbitrary property of tuples $\overline{n} \in \mathbb{N}^{2k}$. Further let h, h' be two distinct coordinates in $\{1, \ldots, 2k\}$. We say that $\mathcal{P}(\overline{n})$ holds* unboundedly on the coordinates h, h' of $\overline{n}$ *if, for all numbers $n_0 \in \mathbb{N}$, there exist $\overline{n}_1, \overline{n}_2 \in \mathbb{N}^{2k}$ such that:*

- *$\mathcal{P}(\overline{n}_1)$ and $\mathcal{P}(\overline{n}_2)$ hold,*
- *$\overline{n}_1[h] \geqslant n_0$ and $\overline{n}_1[h'] - \overline{n}_1[h] \geqslant n_0$,*
- *$\overline{n}_2[h'] \geqslant n_0$ and $\overline{n}_2[h] - \overline{n}_2[h'] \geqslant n_0$.*

We recall that each factor $v^{\overline{n}}(i)$ is associated with the (co-)inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$, and that the corresponding components $\overline{n}[2i+1]$ and $\overline{n}[2i+2]$ of the parameter $\overline{n}$ denote the number of times the witnessing loops L_{2i+1} and L_{2i+2} are pumped in $\rho^{\overline{n}}$. The specific properties we are interested in are the following ones, for $i = 0, \ldots, k-1$:

$$\mathcal{P}_i(\overline{n}) \;=\; \text{“}v^{\overline{n}}(i) \textit{ is a factor of } w^{\overline{n}}(i)\text{”}.$$

By the definitions of $\mathcal{T}(u^{\overline{n}})$ and $\mathcal{T}'(u^{\overline{n}})$ we know that for every tuple $\overline{n} \in \mathbb{N}^{2k}$, $\mathcal{P}_i(\overline{n})$ holds for some $i \in \{0, \ldots, k-1\}$. From this, using a suitable counting argument, we can prove the crucial lemma below.

Lemma 4.1.12. *There exists an index $i \in \{0, \ldots, k-1\}$ such that the property $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$" holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$.*

Proof. The proof is rather technical, as we need to reason on coverability between pairs of factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(j)$, for possibly distinct indexes $i, j \in \{0, \ldots, k-1\}$. We define the following sets of tuples:

- $C_{i,j}$ contains $\overline{n} \in \mathbb{N}^{2k}$ iff $v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(j)$.
- $D_{i,j}$ contains $\overline{n} \in \mathbb{N}^{2k}$ iff $\mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2k+1}])$ is a factor of $\mathsf{out}(\sigma^{\overline{n}}[\ell'_j, \ell'_k])$

Note that $\mathsf{out}(\rho^{\overline{n}}[\tilde{\ell}_{2i+1}, \tilde{\ell}_{2k+1}])$ is the suffix of $\mathcal{T}(u^{\overline{n}})$ that starts with $v^{\overline{n}}(i)$ and $\mathsf{out}(\sigma^{\overline{n}}[\ell'_j, \ell'_k])$ is the suffix of $\mathcal{T}'(u^{\overline{n}})$ that starts with $w^{\overline{n}}(j)$. Moreover, we have $D_{0,0} = \mathbb{N}^{2k}$, since $\mathcal{T}(u^{\overline{n}}) = \mathcal{T}'(u^{\overline{n}})$. As a convention, we let $C_{i,j} = D_{i,j} = \varnothing$ when $i = k$ or $j = k$.

Claim. *For all $i, j \in \{0, \ldots, k-1\}$, we have $D_{i,j} \subseteq C_{i,j} \cup D_{i+1,j+1}$.*

Proof. Consider a tuple $\overline{n}$ in $D_{i,j}$. By definition, we know that the suffix of $\mathcal{T}(u^{\overline{n}})$ that starts with $v^{\overline{n}}(i)$ is a factor of the suffix of $\mathcal{T}'(u^{\overline{n}})$ that starts with $w^{\overline{n}}(j)$. We distinguish some cases depending on whether $j = k-1$ or $j < k-1$, and whether $\overline{n} \in C_{i,j}$ or not. If $j = k-1$, we prove the claim by observing that $\overline{n}$ must belong to $C_{i,j}$: indeed, if this were not the case, then the last factor $w^{\overline{n}}(k-1)$ of $\mathcal{T}'(u^{\overline{n}})$ would end strictly before the end of the factor $v^{\overline{n}}(i)$, thus contradicting the hypothesis that $\mathcal{T}(u^{\overline{n}}) = \mathcal{T}'(u^{\overline{n}})$. If $j < k-1$ and $\overline{n} \in C_{i,j}$, then the claim follows trivially. Finally, suppose that $j < k-1$ and $\overline{n} \notin C_{i,j}$. Since $\overline{n} \in D_{i,j}$, we know that the factor $v^{\overline{n}}(i)$ begins after the beginning of $w^{\overline{n}}(j)$. However, because $v^{\overline{n}}(i)$ is not a factor of $w^{\overline{n}}(j)$, we also know that $v^{\overline{n}}(i)$ ends after the ending of $w^{\overline{n}}(j)$. This implies that $v^{\overline{n}}(i+1)$ begins after the beginning of $w^{\overline{n}}(j+1)$, whence $\overline{n} \in D_{i+1,j+1}$. □

Using the above claim we can derive the first important equation:

$$\mathbb{N}^{2k} \;=\; D_{0,0} \;\subseteq\; C_{0,0} \cup D_{1,1} \;\subseteq\; \ldots \;\subseteq\; \bigcup\nolimits_i C_{i,i} \;\cup\; D_{k,k} \;=\; \bigcup\nolimits_i C_{i,i}. \quad (4.3)$$

This basically means that it is sufficient to consider only the coverability between pairs of factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(i)$, having the same index i.

Recall that $C_{i,i} = \{\overline{n} \mid \mathcal{P}_i(\overline{n})\}$. Let us now prove the lemma by way of contradiction: we assume that for all indexes i the property $\mathcal{P}_i(\overline{n})$ does not hold unboundedly on $2i+1$ and $2i+2$. By Definition 4.1.11, this means that there exists a number $n_i \in \mathbb{N}$ that satisfies one of the two cases below:

a) for all $\overline{n} \in C_{i,i}$, $\overline{n}[2i+1] < n_i$ or $\overline{n}[2i+2] - \overline{n}[2i+1] < n_i$,

b) for all $\overline{n} \in C_{i,i}$, $\overline{n}[2i+2] < n_i$ or $\overline{n}[2i+1] - \overline{n}[2i+2] < n_i$.

We define a tuple $\overline{m} \in \mathbb{N}^{2k}$ such that, for all $i \leqslant k$:

$$\begin{cases} \overline{m}[2i+1] = n_i,\ \overline{m}[2i+2] = 2n_i & \text{if case a) holds} \\ \overline{m}[2i+2] = n_i,\ \overline{m}[2i+1] = 2n_i & \text{if case b) holds.} \end{cases}$$

By definition, $\overline{m}$ cannot belong to $C_{i,i}$ for any $i \leqslant 2k$. However, this is in contradiction with $\mathbb{N}^{2k} = \bigcup_i C_{i,i}$ as implied by Equation 4.3. This concludes the proof that for some i, $\mathcal{P}_i(\overline{n})$ holds unboundedly on $2i+1$ and $2i+2$. □

The last piece of the puzzle consists of generalizing the statement of Proposition 3.4.4. The idea is that we can replace the hypothesis that $\mathcal{T}$ is one-way definable by the weaker assumption of Lemma 4.1.12. That is, if $\mathcal{P}_i(\overline{n})$ holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$, we can still use the same arguments based on pumping and Fine-Wilf's Theorem as in Proposition 3.4.4, in order to deduce that the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ between the locations of the (co-)inversion is periodic:

Proposition 4.1.13. *If the property $\mathcal{P}_i(\overline{n})$ = "$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$" holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$, then the output $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ produced between the locations of the (co-)inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ is periodic, with period at most $\boldsymbol{B}$. In particular, the k-inversion $\overline{\ell} = (L_1, \ell_1, L_2, \ell_2), \ldots, (L_{2k-1}, \ell_{2k-1}, L_{2k}, \ell_{2k})$ is $\boldsymbol{B}$-safe.*

Proof. We begin by distinguishing two cases, depending on whether i is even or odd. If i is even, then $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ is an inversion and the factor $w^{\overline{n}}(i)$ is produced along a left-to-right pass of $\mathcal{T}'$, exactly like in a one-way transduction. Otherwise, if i is odd, then $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ is a co-inversion and the factor $w^{\overline{n}}(i)$ is produced along a right-to-left pass of $\mathcal{T}'$. As the two cases are symmetric, we can focus only on one of the two, say the case where i is even. The proof that $\mathsf{out}(\rho[\ell_{2i+1}, \ell_{2i+2}])$ is periodic is exactly the same as that of Proposition 3.4.4. We briefly recall the crucial arguments below.

In that proof we considered an inversion $(L_1, \ell_1, L_2, \ell_2)$ and the outputs produced by runs of the form $\mathsf{pump}_{L_2}^{n_2}(\mathsf{pump}_{L_1}^{n_1}(\rho))$ between the locations ℓ_1 and ℓ_2. The latter outputs were then compared with the outputs produced by an equivalent one-way transducer $\mathcal{T}'$. In particular, we observed that the former outputs are factors of the latter outputs. More precisely, by letting the parameters n_1 and n_2 grow independently, it was possible to exploit Fine-Wilf's Theorem and derive the periodicity of the former outputs.

The same argument can be applied here with the inversion $(L_{2i+1}, \ell_{2i+1}, L_{2i+2}, \ell_{2i+2})$ and the factors $v^{\overline{n}}(i)$ and $w^{\overline{n}}(i)$, produced respectively by $\mathcal{T}$ and $\mathcal{T}'$. Indeed, to apply Fine-Wilf's Theorem, it is sufficient that the coverability relationship holds for pairs of arbitrarily large numbers $\overline{n}[2i+1]$ and $\overline{n}[2i+2]$, and that these numbers can vary independently of each other. This is precisely what it means for the property $\mathcal{P}_i(\overline{n})$ to hold unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$. $\square$

To conclude, we assumed that the two-way transducer $\mathcal{T}$ is equivalent to a k-pass sweeping transducer $\mathcal{T}'$. We considered a successful run ρ of $\mathcal{T}$ and an arbitrary k-inversion $\overline{\ell}$ of it. Using Lemma 4.1.12, we derived the existence of an index $i \in \{0, \ldots, k-1\}$ such that the property $\mathcal{P}_i(\overline{n}) =$ "*$v^{\overline{n}}(i)$ is a factor of $w^{\overline{n}}(i)$* " holds unboundedly on the coordinates $2i+1$ and $2i+2$ of $\overline{n}$. From this, using Proposition 4.1.13, we derived that the k-inversion $\overline{\ell}$ is $\boldsymbol{B}$-safe. This proves the left-to-right direction of the first claim of Theorem 4.1.4. $\square$

4.1.4 Minimization and sweeping definability

Recall that, without loss of generality, we can focus on *normalized* runs of functional transducers, and from this we know that $2|Q|$ passes suffice to implement every transduction definable by a sweeping transducer. Then, Theorem 4.1.4 immediately gives a procedure for minimizing the number of passes of a given sweeping transducer $\mathcal{T}$: one simply needs to decide k-pass definability on $\mathcal{T}$, for every $k = 1, \ldots, 2|Q|$.

Corollary 4.1.14. *One can compute in* ExpSpace *the minimum number of passes needed to implement a transduction given as a functional sweeping transducer.*

A transducer with a minimal number of passes can be constructed in 2ExpTime.

Analogous results can be proven for arbitrary two-way (not necessarily sweeping) transducers. The idea relies on bounding the number of passes required by any sweeping transducer in order to compute a function defined by a two-way transducer. Of course, the argument is more complicated than the one based on normalized runs, since a two-way transducer can perform an arbitrary number of reversals on its inputs.

Bounding the number of passes.

Recall that $\boldsymbol{E} = (2|Q|)^{2\boldsymbol{H}}$ is the number of distinct effects. We prove the following result:

Theorem 4.1.15. *A functional two-way transducer $\mathcal{T}$ is sweeping definable if and only if it is k-pass sweeping definable for $k = 2\boldsymbol{H}(2^{3\boldsymbol{E}} + 1)$.*

As the k-pass definability complexity is polynomial in k, and $2\boldsymbol{H}(2^{3\boldsymbol{E}}+1)$ is a double exponential in $\mathcal{T}$ we immediately have the following corollary:

Corollary 4.1.16. *1. The problem of deciding sweeping definability of a functional two-way transducer is in* 2ExpSpace.

2. One can compute in 2ExpSpace *the minimum number of passes needed to implement a transduction given as a functional two-way transducer.*

A transducer with a minimal number of passes can be constructed in 3ExpTime.

Proof. Suppose that $\mathcal{T}$ is not k-pass sweeping definable for $k = 2\boldsymbol{H}\cdot(2^{3\boldsymbol{E}}+1)$. We aim at proving that $\mathcal{T}$ is not m-pass sweeping definable for all $m > 0$. By Theorem 4.1.4, we know that there exist a successful run ρ and a k-inversion $\overline{\mathcal{I}} = (\mathcal{I}_0, \ldots, \mathcal{I}_{k-1})$ of it, with $\mathcal{I}_i = (L_i, \ell_i, L'_i, \ell'_i)$, that is not $\boldsymbol{B}$-safe. We consider the locations of ρ that are visited between the beginning of an inversion $\mathcal{I}_i$ and the ending of the next co-inversion $\mathcal{I}_{i+1}$. Formally, for all even indices $i = 0, 2, \ldots, k-1$, we consider the interval of locations

$$K_i \;=\; [\ell_i, \ell'_{i+1}].$$

We then project each interval K_i on the x-coordinates:

$$X_i \;=\; \{x \;:\; \exists\, \ell = (x, y) \in K_i\}.$$

Since each K_i is an interval of locations and the transducer $\mathcal{T}$ can only move its head between consecutive positions, we know that each X_i is an interval of positions. Hereafter, we often use the term "interval" to denote a set of the form X_i, for some even index $i \in \{0, 2, \ldots, \kappa - 1\}$.

Below we prove that there is a large enough set of pairwise non-overlapping intervals X_i:

Lemma 4.1.17. *There is a set $\mathcal{X} = \{X_i\}_{i \in I}$ of n intervals, where $n = 2^{3\boldsymbol{E}}+1$ and $I \subseteq \{0, 2, \dots, k-1\}$ (recall that $k = 2\boldsymbol{H}\cdot(2^{3\boldsymbol{E}}+1)$), such that $X \cap X' = \varnothing$ for all $X \neq X' \in \mathcal{X}$.*

Proof. In this proof, we consider an ordering on the intervals X_i different from the one induced by the indices i. This is given by the lexicographic order on the endpoints, where the dominant element is the rightmost endpoint, namely, we let $X_i < X_j$ if either $\max(X_i) < \max(X_j)$, or $\max(X_i) = \max(X_j)$ and $\min(X_i) < \min(X_j)$.

We construct the set $\mathcal{X}$ inductively, by following the lexicographic ordering. Formally, for all $j = 0, \dots, n$, we construct:

- a set $\mathcal{X}_j$ of size j such that $X \cap X' = \varnothing$ for all $X \neq X' \in \mathcal{X}_j$
- a set $\mathcal{X}'_j$ of size at least $\boldsymbol{H} \cdot (2^{3\boldsymbol{E}} + 1 - j)$ such that, for all $X \in \mathcal{X}_j$ and all $X' \in \mathcal{X}'_j$, $\max(X) < \min(X')$ (namely, all intervals of $\mathcal{X}'_j$ are strictly to the right of the intervals of $\mathcal{X}_j$).

The base case $j = 0$ of the induction is immediate: we let $\mathcal{X}_0 = \varnothing$ and $\mathcal{X}'_0$ be the set of all intervals. It only suffices to observe that $\mathcal{X}'_0$ has cardinality $\frac{k}{2} = \boldsymbol{H} \cdot (2^{3\boldsymbol{E}} + 1)$.

For the inductive step, suppose that $j < n = 2^{3\boldsymbol{E}} + 1$ and that we constructed $\mathcal{X}_j$ and $\mathcal{X}'_j$ satisfying the inductive hypothesis. We let X be the least element in $\mathcal{X}'_j$ according to the lexicographic order (note that $\mathcal{X}'_j \neq \varnothing$ since $j < n$). Accordingly, we define $\mathcal{X}_{j+1} = \mathcal{X}_j \cup \{X\}$ and $\mathcal{X}'_{j+1}$ as the subset of $\mathcal{X}'_j$ that contains the intervals strictly to the right of X. It remains to verify that $\mathcal{X}'_{j+1}$ has cardinality at least $\boldsymbol{H} \cdot \big(2^{3\boldsymbol{E}} + 1 - (j+1)\big)$. For this we recall that the run ρ is normalized. This implies that there are at most $\boldsymbol{H}$ intervals in $\mathcal{X}'_j$ that cover the position $x = \max(X)$. All other intervals of $\mathcal{X}'_j$ are necessarily to the right of X: indeed, because X is minimal in the lexicographic ordering, we know that every interval of $\mathcal{X}'_j$ has the right endpoint to the right of x, and as they do not cover the position x, their left endpoint too. This shows that there are at most $\boldsymbol{H}$ intervals in $\mathcal{X}'_j \setminus \mathcal{X}'_{j+1}$, so $|\mathcal{X}'_{j+1}| \geqslant \boldsymbol{H} \cdot \big(2^{3\boldsymbol{E}} + 1 - (j+1)\big)$. □

Turning back to the proof of the theorem, we consider the left endpoints of the intervals in $\mathcal{X}$, say

$$\overleftarrow{X} \;=\; \{\min(X) \;:\; X \in \mathcal{X}\}.$$

Since $|\overleftarrow{X}| > 2^{3\boldsymbol{E}}$, we can use Theorem 3.4.1 to derive the existence of three distinct positions $x < x' < x'' \in \overleftarrow{X}$ such that $[x, x']$ and $[x', x'']$ are consecutive idempotent loops of ρ with the same effect. We let $L = [x, x'']$ be the union of those two loops, and we consider the intermediate position x'. We recall that x' is the left endpoint of an interval of $\mathcal{X}$, which we denote by X_i for simplicity. We also recall that X_i is the set of positions visited by a factor of the run ρ that goes from the first anchor ℓ_i of the inversion $\mathcal{I}_i = (L_i, \ell_i, L'_i, \ell'_i)$ to the second anchor ℓ'_{i+1} of the co-inversion $\mathcal{I}_{i+1} = (L_{i+1}, \ell_{i+1}, L'_{i+1}, \ell'_{i+1})$.

We claim that the inversion $\mathcal{I}_i$ and the co-inversion $\mathcal{I}_{i+1}$ occur in the same factor intercepted by L. Indeed, the factor $\rho[\ell_i, \ell'_{i+1}]$ visits only positions inside the interval X_i. Moreover, the endpoints of X_i are strictly between the endpoints of L, namely,

$$\min(L) \;=\; x \;<\; x' \;=\; \min(X_i) \;\leqslant\; \max(X_i) \;<\; x'' \;=\; \max(L).$$

This shows that the inversion $\mathcal{I}_i = (L_i, \ell_i, L'_i, \ell'_i)$ and the co-inversion $\mathcal{I}_{i+1} = (L_{i+1}, \ell_{i+1}, L'_{i+1}, \ell'_{i+1})$ occur in the same factor intercepted by L, which we denote by α.

Now, we can easily introduce new copies of the factor α, and hence new copies of the (co)-inversions $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$, by pumping the idempotent loop L. Formally, for all $m > 0$, we denote by $\mathcal{I}_i^{(1)}, \dots, \mathcal{I}_i^{(m)}$ (resp. $\mathcal{I}_{i+1}^{(1)}, \dots, \mathcal{I}_{i+1}^{(m)}$) the m copies of the inversion $\mathcal{I}_i$ (resp. the m copies of the co-inversion $\mathcal{I}_{i+1}$) that appear in the pumped run $\mathsf{pump}_L^m(\rho)$. For the sake of simplicity, we assume that those copies are listed according to their order of occurrence in the pumped run, namely,

$$\mathcal{I}_i^{(1)} \;\lhd\; \mathcal{I}_{i+1}^{(1)} \;\lhd\; \mathcal{I}_i^{(2)} \;\lhd\; \mathcal{I}_{i+1}^{(2)} \;\lhd\; \dots \;\lhd\; \mathcal{I}_i^{(m)} \;\lhd\; \mathcal{I}_{i+1}^{(m)}$$

(the order $\lhd$ is extended from locations to (co-)inversions in the natural way).

Towards a conclusion, we observe that $\big(\mathcal{I}_i^{(1)}, \mathcal{I}_{i+1}^{(1)}, \dots, \mathcal{I}_i^{(m)}, \mathcal{I}_{i+1}^{(m)}\big)$ is a $2m$-inversion of the successful run $\mathsf{pump}_L^m(\rho)$ of $\mathcal{T}$. Moreover, this $2m$-inversion is not $\boldsymbol{B}$-safe, since it consists of (co-)inversions that do not generate periodic outputs — more precisely, the period of the word $\mathsf{out}(\mathsf{tr}(\ell_i))\ \mathsf{out}(\rho[\ell_i, \ell'_i])\ \mathsf{out}(\mathsf{tr}(\ell'_i))$ (resp. $\mathsf{out}(\mathsf{tr}(\ell_{i+1}))\ \mathsf{out}(\rho[\ell_{i+1}, \ell'_{i+1}])\ \mathsf{out}(\mathsf{tr}(\ell'_{i+1}))$) is larger than $\boldsymbol{B}$ or does not divide $|\mathsf{out}(\mathsf{tr}(\ell_i))|$ and $|\mathsf{out}(\mathsf{tr}(\ell'_i))|$ (resp. $|\mathsf{out}(\mathsf{tr}(\ell_{i+1}))|$ and $|\mathsf{out}(\mathsf{tr}(\ell'_{i+1}))|$). By Theorem 4.1.4, this proves that $\mathcal{T}$ is not m-pass sweeping definable. Finally, since the above holds for all $m > 0$, we conclude that $\mathcal{T}$ is not sweeping definable. □

4.2 Registers of streaming string transducers

In this section we focus on the register minimization problem for the restricted class of *NSST*'s that is obtained by disallowing register concatenation in the updates:

Definition 4.2.1. *An* NSST $\mathcal{T} = (Q, \Sigma, \Gamma, R, U, I, E, F)$ *is* concatenation-free *if, for all registers* $z \in R$ *and all updates* $f \in U$*, we have* $f(z) \in \Gamma^\star \cdot (R \cup \{\varepsilon\}) \cdot \Gamma^\star$*.*

Intuitively, a concatenation-free *NSST* forbids register updates $f : R \to (R \uplus \Gamma)^*$ that have two or more registers inside the same right-hand side $f(z)$. Concatenations of registers can still appear in the output function.

If needed, this restriction can be relaxed slightly, without increasing the expressive power: for this, one can allow boundedly many updates with concatenations, and still the resulting transductions could be simulated by a concatenation-free *NSST* (just like one can allow a sweeping transducer to perform boundedly many reversals in the middle of the word, and remain sweeping definable [9]). One should note, however, that this comes at the cost of increasing the number of registers.

The register minimization problem is solved by exploiting a suitable correspondence between the number of registers used by concatenation-free *NSST*'s and the number of passes performed by sweeping transducers.

4.2.1 Translations between *SST* and *2DFT*

It is known (see Chapter 2) that *SST*'s are equivalent to *2DFT*'s, and that the latter are equivalent to *2fNFT*'s. Direct constructions can be found in [29], and can be generalized to some extent to concatenation-free *NSST* and sweeping transducers.

We have already seen in Chapter 2 that the function $u \mapsto \overline{u}u$ can be implemented by an *SST* with one register. The same function can also be implemented by a sweeping transducer that performs first a right-to-left pass to produce $\overline{u}$, and then a left-to-right pass to produce u. More generally, an arbitrary *SST* with one register can be simulated by a 2-pass sweeping transducer that starts its computation from the right endpoint. It is then natural to try to lift such a correspondence to *SST*'s with multiple registers, by considering sweeping transducers that start their computation at the right endpoint of their input — we call such transducers *R-sweeping*.

The following theorem shows that, indeed, there is a correspondence between the number of registers of concatenation-free *SST*'s (or even functional *NSST*'s) and the number of passes of R-sweeping transducers:

Theorem 4.2.2. *Every concatenation-free functional* NSST *with k registers can be transformed in* ExpTime *into an equivalent unambiguous $2k$-pass R-sweeping transducer. If the* NSST *is unambiguous, then the transformation is in* Ptime.

Conversely, every k-pass R-sweeping functional transducer can be transformed in 2ExpTime *into an equivalent concatenation-free unambiguous* NSST *with $\lceil \frac{k}{2} \rceil$ registers. If the R-sweeping transducer is unambiguous, then the transformation is in* ExpTime.

The two directions of the theorem will be proven in the next two sections. Here, we briefly explain how this correspondence, paired with the characterization of k-pass sweeping definability (Theorem 4.1.4), can be used to derive a procedure for minimizing the number of registers in a concatenation-free *NSST*. Given an *NSST*, one first constructs an equivalent R-sweeping transducer, using the first claim of Theorem 4.2.2. Then, one derives the analogous of Theorem 4.1.4 for characterizing k-pass sweeping definability: this is easily done by mirroring the input and by reversing the transition directions. Finally, one uses the characterization to compute the minimum k for which the given transduction is definable by a $2k$-pass R-sweeping transducers: in view of the second claim of Theorem 4.2.2, the resulting k is precisely the minimum number of registers needed by a concatenation-free *NSST* equivalent to the original one. We sum up the result in the following corollary:

Corollary 4.2.3. *One can compute in* 2ExpSpace *the minimum number of registers needed to implement a transduction given as a concatenation-free* NSST. *The complexity is* ExpSpace *if the given* NSST *is unambiguous.*

4.2.2 From concatenation-free *NSST* to sweeping transducers

Here we prove the first claim of Theorem 4.2.2.

Recall that Proposition 2.4.2 shows how to turn in ExpTime an *NSST* into an equivalent unambiguous *NSST* with the same number of registers. Thanks to this, we can focus on proving only the part of the claim that concerns the transformation of a given concatenation-free unambiguous *NSST*.

We fix for the rest of the section a concatenation-free unambiguous *NSST* $\mathcal{T} = (Q, \Sigma, \Gamma, R, U, I, E, F)$ with $k = |R|$ registers. We show how to transform $\mathcal{T}$ into an equivalent unambiguous $2k$-pass R-sweeping transducer $\mathcal{T}'$.

Let $u = a_1 \dots a_n$ be an arbitrary input for $\mathcal{T}$ and $\sigma = (q_0, g_0) \xrightarrow{a_1} (q_1, g_1) \xrightarrow{a_2} \dots \xrightarrow{a_n} (q_n, g_n)$ the unique successful run of $\mathcal{T}$ on u. We can write the corresponding output as

$$\mathcal{T}(u) \;=\; v_0 \cdot g_n(z_1) \cdot v_1 \cdot g_n(z_2) \cdot \ldots \cdot g_n(z_h) \cdot v_h$$

where g_n is the final register valuation and $F(q_n) = v_0 \cdot z_1 \cdot v_1 \cdot z_2 \cdot \ldots \cdot z_h \cdot v_h$, for some $h \leqslant k$, describes how the final output is produced from the registers (note that $F(q_n)$, and in particular the order of the registers in it, depends on the final state q_n). Further let $f_1, \dots, f_n$ be the sequence of register updates induced by the above transitions, in such a way that $g_x = g_{x-1} \circ f_x$ for all $x = 1, \dots, n$.

The idea underlying the construction of $\mathcal{T}'$ is to output each factor $v_{i-1} \cdot g_n(z_i)$ during two consecutive passes that start and end at the rightmost position. For this, we fix an index $i \in \{1, \dots, h\}$ and we consider how the factor $g_n(z_i)$ is produced along the run σ. Since $\mathcal{T}$ is concatenation-free and unambiguous, there exist a unique position $x_i \in \{0, \dots, n\}$, a unique sequence of registers $z_i = z_{i,n},\ z_{i,n-1},\ \dots,\ z_{i,x_i} \in R$, and a unique sequence of words $w_{i,n}, w'_{i,n},\ \dots,\ w_{i,x_i} \in \Gamma^*$ such that

$$\begin{cases} g_n(z_{i,n}) &= w_{i,n} \cdot g_{n-1}(z_{i,n-1}) \cdot w'_{i,n} \\ & \vdots \\ g_{x_i+1}(z_{i,x_i+1}) &= w_{i,x_i+1} \cdot g_{x_i}(z_{i,x_i}) \cdot w'_{i,x_i+1} \\ f_{x_i}(z_{i,x_i}) &= w_{i,x_i}. \end{cases}$$

By convention we set $w'_{i,x_i} = \epsilon$ and we can write $f_{x_i}(z_{i,x_i}) = w_{i,x_i} \cdot w'_{i,x_i}$. This basically means that the factor $g_n(z_i)$ is obtained from the empty word by repeatedly prepending and appending finite words $w_{i,x}$ and $w'_{i,x}$, where x goes from x_i to n. Moreover, each pair of words $w_{i,x}$ and $w'_{i,x}$ is determined by the x-th transition $(q_{x-1}, g_{x-1}) \xrightarrow{a_x} (q_x, g_x)$ of $\mathcal{T}$.

The R-sweeping transducer $\mathcal{T}'$ will behave as follows.

At the beginning of a right-to-left pass at level $2i-1$, $\mathcal{T}'$ guesses which z_i it has to produce, outputs the word v_{i-1} that precedes the factor $g_n(z_n)$ in $\mathcal{T}(u)$. Then, during that pass, it reads the input in backward direction and, while guessing the transitions $(q_{x-1}, g_{x-1}) \xrightarrow{a_x} (q_x, g_x)$, it outputs the corresponding words $w_{i,x}$. Once the position x_i is reached, $\mathcal{T}'$ continues to move leftward while simulating the transitions of $\mathcal{T}$, but this time without producing any output. This is needed to check that the state of $\mathcal{T}$ reached at the leftmost position is initial (and moreover to have a sweeping run). Then $\mathcal{T}'$ performs a reversal. Similarly, during the left-to-right pass at level $2i$, the transducer $\mathcal{T}'$ guesses the transitions $(q_{x-1}, g_{x-1}) \xrightarrow{a_x} (q_x, g_x)$ and, if $x \geqslant x_i$, it outputs the corresponding words $w'_{i,x}$. Once the rightmost position is reached again, it checks that the last guessed state is final and performs the next reversal. The last pass performed by $\mathcal{T}'$ is the left-to-right pass at level $2h$, where the last piece v_h of the output can be produced.

Clearly, $\mathcal{T}'$ can be implemented with approximately $2k$ copies of the state space of $\mathcal{T}$ (one copy for each pass), and it performs at most $2k$ passes on any input. We remark that for this construction it is crucial that the streaming transducer $\mathcal{T}$ is unambiguous, as otherwise the described R-sweeping transducer $\mathcal{T}'$ may guess different runs along two consecutive passes, eventually producing an output that differs from $\mathcal{T}(u)$.

4.2.3 From sweeping transducers to concatenation-free *NSST*

Let us now prove the second claim of Theorem 4.2.2. As before, in view of Proposition 2.2.4, we can focus only of the part of claim that concerns the transformation of a k-pass R-sweeping *unambiguous* transducer $\mathcal{T}$.

Let $\mathcal{A}$ be the R-sweeping unambiguous automaton underlying $\mathcal{T}$, which recognizes the language $\mathsf{dom}(\mathcal{T})$. By applying the classical construction based on crossing sequences [78], we transform $\mathcal{A}$ into an equivalent unambiguous one-way automaton $\mathcal{B}$ of exponential size. To obtain an *NSST* equivalent to $\mathcal{T}$, we equip the automaton $\mathcal{B}$ with $\lceil \frac{k}{2} \rceil$ registers, say $z_1, \dots, z_{\lceil \frac{k}{2} \rceil}$, and we extend the transitions with suitable register updates. The idea is that each register z_i stores the output produced along the passes of $\mathcal{T}$ at levels $2i-1$ and $2i$.

Consider an arbitrary input $u = a_1 \dots a_n$ for $\mathcal{T}$, where $a_1 = \rhd$ and $a_n = \lhd$, and recall that $\mathcal{B}$ admits a unique run on $a_2 \dots a_{n-1}$, say:

$$\sigma \;=\; s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \dots \xrightarrow{a_{n-1}} s_{n-1}.$$

Recall that the run σ determines a unique successful run ρ of $\mathcal{T}$ on u. In particular, each state s_x determines a crossing sequence $\rho|x$. If $\rho|x = (q_1, \dots, q_h)$

and $\rho|x+1 = (q'_1, \ldots, q'_h)$ are the crossing sequences of ρ at two adjacent positions x and $x+1$, for some $h \leqslant k$, then the corresponding update f_{x+1} for the registers of $\mathcal{B}$ must satisfy:

$$f_{x+1}(z_i) \;=\; w_{x+1} \cdot z_i \cdot w'_{x+1}$$

where w_{x+1} is the output produced by $\mathcal{T}$ with the right-to-left transition $q'_{2i-1} \overset{a_{x+1}}{\longrightarrow} q_{2i-1}$ and w'_{x+1} is the output produced by $\mathcal{T}$ with the left-to-right transition $q_{2i} \overset{a_{x+1}}{\longrightarrow} q'_{2i}$. Since f_{x+1} is uniquely determined by the control states s_x and s_{x+1} and by the input symbol a_{x+1}, the above equations can be easily turned into transition rules for an *NSST* $\mathcal{T}'$ having $\mathcal{B}$ as underlying automaton. In particular, the *NSST* $\mathcal{T}'$ is unambiguous, since $\mathcal{B}$ is so.

Finally, we specify the partial output function of $\mathcal{T}'$, which maps any state s of $\mathcal{B}$ to the juxtaposition $z_1 \cdot \ldots \cdot z_{\lceil \frac{h}{2} \rceil}$ of the first $\lceil \frac{h}{2} \rceil$ registers, where h is the length of the crossing sequence determined by s. The resulting *NSST* $\mathcal{T}'$ is unambiguous, uses at most $\lceil \frac{k}{2} \rceil$ registers, and is equivalent to $\mathcal{T}$.

Chapter 5

Conclusion

We conclude this thesis by summarizing our work and giving some ideas for the future.

One-way definability. The main subject of this thesis is the proof of the decidability of one-way definability for two-way transducers and sweeping transducers in elementary complexity. Our decision procedure requires to build the equivalent one-way transducer (or the best approximation of it when it is not possible) and by consequence is probably not optimal. One possible goal for further research would be to obtain a decision procedure without constructing this transducer but we believe it is less important in practice to have a fast but non-effective algorithm to decide one-way definability. A more interesting goal would be to understand the precise bounds for the effective characterization. We obtained a tight two-exponential construction for sweeping transducers but we do not know if a two-exponential one exists in the two-way case or if our three-exponential procedure is optimal in that case. However we believe our approach about analysing two-way loops with the concept of component is the right one, and that if a two-exponential construction exists, the improvement would come from finding idempotent loops in polynomial time. Personal communications with Ismaël Jecker lead us to believe this is possible.

Sweeping definability. An aspect of our work that was not planned is the understanding of the sweeping class, which is also the class of bounded-reversal two-way transducers. The fact that we were able to solve k-pass definability, and sweeping definability sets up sweeping transducers as one sound intermediate class of transductions between rational and regular. However, in the framework of finite state transducers, there is one definability question that remains open: given a non-deterministic sweeping (or two-way) transducer, is it equivalent to a deterministic sweeping one? One possible approach would be to try to extend modern proofs of the one-way procedure [11] to sweeping transducers.

Streaming transducers. Our work on streaming transducers in Section 4.2 is a first step for a minimization procedure for the registers of SST's. To our knowledge it is the first time this question was shown decidable for a large class of SST's. In 2013, it was solved in the case of a unary alphabet (which is always quite a particular case) [3]. More recently, a procedure was obtained for the minimization of a subclass of cost-register automata [30]. These automata have an output that belong to any kind of monoid, but if we restrict this class to the free monoid over the alphabet Σ, our model allows more kinds of updates. Anyway, the main difference with [30] lies in the determinism of cost-register automata.

Other models. A model whose popularity is increasing is the one of transducers with origin information (that is, we know for each letter of the output from which part of the input it derives), proposed by Bojańczyk [15]. In the model of origin semantics, the one-way definability problem is PSpace-complete and an equivalent one-way transducer has exponential size. One can ask whether a given set of "origin graphs" can be realized for instance by a sweeping transducer.

One can also look at aperiodic two-way transducers and try to provide a characterization of the definability in first-order logic. As we mentioned in this thesis, the fact that a transducer is not aperiodic does not mean that all equivalent transducers are not either [19], and is the main obstacle to such a characterization. One important result in that regard would be to know if the constructions we presented preserve aperiodicity.

Finally, one can consider tree transducers and visibly pushdown transducers, and ask similar definability questions over for such machines. However, our proofs being based on word combinatorics, one will have to find new techniques to lift the regular look-ahead condition that some tree transducer have [13].

Bibliography

[1] A. Aho, J. Hopcroft, and J. Ullman. A general theory of translation. *Math. Syst. Theory*, 3(3):193–221, 1969.

[2] R. Alur and P. Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[3] R. Alur, L. D'Antoni, J. Deshmukh, M. Raghothaman, and Y. Yuan. Regular functions and cost register automata. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 13–22, Washington, DC, USA, 2013. IEEE Computer Society.

[4] R. Alur and J. V. Deshmukh. Nondeterministic streaming string transducers. In L. Aceto, M. Henzinger, and J. Sgall, editors, *Automata, Languages and Programming: 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part II*, pages 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[5] R. Alur, A. Durand-Gasselin, and A. Trivedi. From monadic second-order definable string transformations to transducers. In *Proceedings of the 2013 28th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '13, pages 458–467, Washington, DC, USA, 2013. IEEE Computer Society.

[6] R. Alur, A. Freilich, and M. Raghothaman. Regular combinators for string transformations. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic*

in Computer Science (LICS), CSL-LICS '14, pages 9:1–9:10, New York, NY, USA, 2014. ACM.

[7] R. Alur and M. Raghothaman. Decision problems for additive regular functions. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 37–48, 2013.

[8] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. One-way definability of sweeping transducer. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 178–191. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[9] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. Minimizing resources of sweeping and streaming string transducers. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 114:1–114:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. Full version available at `https://hal.archives-ouvertes.fr/hal-01274992`.

[10] F. Baschenis, O. Gauwin, A. Muscholl, and G. Puppis. Untwisting two-way transducers in elementary time. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017.

[11] M. Béal, O. Carton, C. Prieur, and J. Sakarovitch. Squaring transducers: an efficient procedure for deciding functionality and sequentiality. *Theoretical Computer Science*, 292(1):45–63, 2003.

[12] M.-P. Béal and O. Carton. Determinization of transducers over finite and infinite words. *Theoretical Computer Science*, 289(1):225–251, 2002.

[13] M. Benedikt, J. Engelfriet, and S. Maneth. Determinacy and rewriting of functional top-down and MSO tree transformations. *J. Comput. Syst. Sci.*, 85:57–73, 2017.

[14] J. Birget. Two-way automaton computations. *RAIRO - Theoretical Informatics and Applications - Informatique Théorique et Applications*, 24(1):47–66, 1990.

[15] M. Bojańczyk. Transducers with origin information. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II*, pages 26–37, 2014.

[16] J. A. Brzozowski. Canonical regular expressions and minimal state graphs for definite events. In *Mathematical theory of Automata*, Volume 12 of MRI Symposia Series, pages 529–561. Polytechnic Press, Polytechnic Institute of Brooklyn, N.Y., 1962.

[17] J. Büchi. On a decision method in restricted second- order arithmetic. In *Froceed- ings of the 1960 Congress on Logic, Methdology and Philosoph y of Science*. Stanford University Press, 1962.

[18] J. R. Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960.

[19] O. Carton and L. Dartois. Aperiodic two-way transducers and FO-transductions. In *24th EACSL Annual Conference on Computer Science Logic (CSL)*, volume 41 of *LIPIcs*, pages 160–174. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

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

[21] C. Choffrut. Minimizing subsequential transducers: a survey. *Theor. Comput. Sci.*, 292(1):131–143, 2003.

[22] C. Choffrut and B. Guillon. An algebraic characterization of unary two-way transducers. In *Proceedings of the 15th Italian Conference on Theoretical Computer Science, Perugia, Italy, September 17-19, 2014.*, volume 1231 of *CEUR Workshop Proceedings*, pages 279–283. CEUR-WS.org, 2014.

[23] C. Choffrut and B. Guillon. An algebraic characterization of unary two-way transducers. In *Mathematical Foundations of Computer Science*

2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part I, volume 8634 of *Lecture Notes in Computer Science*, pages 196–207. Springer, 2014.

[24] T. Colcombet. Factorisation forests for infinite words. In *FCT*, volume 4639 of *LNCS*, pages 226–237. Springer, 2007.

[25] B. Courcelle. Monadic second-order definable graph transductions: A survey. *Theor. Comput. Sci.*, 126(1):53–75, 1994.

[26] B. Courcelle and J. Engelfriet. *Graph structure and monadic second-order logic. A language-theoretic approach.* Encyclopedia of Mathematics and its applications, Vol. 138. Cambridge University Press, June 2012. Collection Encyclopedia of Mathematics and Applications, Vol. 138.

[27] L. Dartois. *Méthodes algébriques pour la théorie des automates.* PhD thesis, Université Paris-Diderot, 2014.

[28] L. Dartois, P. Fournier, I. Jecker, and N. Lhote. On reversible transducers. In *44rd International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[29] L. Dartois, I. Jecker, and P.-A. Reynier. Aperiodic string transducers. In *Proc. 20th International Conference on Developments in Language Theory (DLT 2016)*, volume 9840 of *Lecture Notes in Computer Science*, pages 125–137, Montreal, Canada, 2016. Springer.

[30] L. Daviaud, P. Reynier, and J. Talbot. A generalised twinning property for minimisation of cost register automata. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 857–866. ACM, 2016.

[31] R. de Souza. Uniformisation of two-way transducers. In A.-H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Language and Automata Theory and Applications: 7th International Conference, LATA 2013, Bilbao, Spain, April 2-5, 2013. Proceedings*, pages 547–558. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[32] V. Diekert and P. Gastin. First-order definable languages. In *Logic and Automata: History and Perspectives, Texts in Logic and Games*, pages 261–306. Amsterdam University Press, 2008.

[33] S. Eilenberg. *Automata, Langages and Machines.* Academic Press, 1976.

[34] C. C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98(1):21–51, 1961.

[35] C. C. Elgot and J. E. Mezei. On relations defined by generalized finite automata. *IBM J. Res. Dev.*, 9(1):47–68, Jan. 1965.

[36] J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Logic*, 2(2):216–254, 2001.

[37] E. Filiot. Logic-automata connections for transformations. In M. Banerjee and S. N. Krishna, editors, *Logic and Its Applications: 6th Indian Conference, ICLA 2015, Mumbai, India, January 8-10, 2015. Proceedings*, pages 30–57. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[38] E. Filiot, O. Gauwin, and N. Lhote. First-order definability of rational transductions: An algebraic approach. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS'16, New York, NY, USA, July 5-8, 2016*, pages 387–396. ACM, 2016.

[39] E. Filiot, O. Gauwin, P. Reynier, and F. Servais. From two-way to one-way finite state transducers. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 468–477. IEEE Computer Society, 2013.

[40] E. Filiot, S. N. Krishna, and A. Trivedi. First-order definable string transformations. In *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 147–159. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[41] N. Fine and H. Wilf. Uniqueness theorems for periodic functions. *Proceedings of the American Mathematical Society*, 16:109–114, 1965.

[42] P. Gallot, A. Muscholl, G. Puppis, and S. Salvati. On the Decomposition of Finite-Valued Streaming String Transducers. In H. Vollmer and B. Vallee, editors, *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, volume 66 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 34:1–34:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[43] S. Ginsburg and G. F. Rose. Operations which preserve definability in languages. *J. ACM*, 10(2):175–195, Apr. 1963.

[44] S. Ginsburg and G. F. Rose. Preservation of languages by transducers. *Information and Control*, 9(2):153 – 176, 1966.

[45] I. Glaister and J. Shallit. A lower bound technique for the size of nondeterministic finite automata. *Information Processing Letters*, 59:75–77, 1996.

[46] B. Guillon. Sweeping weakens two-way transducers even with a unary output alphabet. In *Seventh Workshop on Non-Classical Models of Automata and Applications - NCMA 2015, Porto, Portugal, August 31 - September 1, 2015. Proceedings*, volume 318 of *books@ocg.at*, pages 91–108. Österreichische Computer Gesellschaft, 2015.

[47] B. Guillon. Input- or output-unary sweeping transducers are weaker than their 2-way counterparts. *RAIRO-Theor. Inf. Appl.*, 50(4):275–294, 2016.

[48] E. M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 83–85, 1980.

[49] E. M. Gurari and O. H. Ibarra. A note on finite-valued and finitely ambiguous transducers. *Mathematical systems theory*, 16(1):61–66, Dec 1983.

[50] J. Hopcroft and J. Ullman. An approach to a unified theory of automata. In *SWAT*, FOCS '67, pages 140–147. IEEE Computer Society, 1967.

[51] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[52] O. H. Ibarra. The unsolvability of the equivalence problem for epsilon-free NGSM's with unary input (output) alphabet and applications. *SIAM J. Comput.*, 7(4):524–532, 1978.

[53] C. A. Kapoutsis. Minicomplexity. *JALC*, 17(2–4):205–224, 2012.

[54] J. Karhumäki and C. Choffrut. *Combinatorics of words*. Springer-Verlag, 1997.

[55] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.

[56] J. Kortelainen. On the system of word equations $x_0u_1^ix_1u_2^ix_2\dots u_m^ix_m = y_0v_1^iy_1v_2^iy_2\dots v_m^iy_m$ $(i = 0, 1, 2, \dots)$ in a free monoid. *Journal of Automata, Languages and Combinatorics*, 3(1):43–57, 1998.

[57] D. Kozen. Lower bounds for natural proof systems. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS'77, pages 254–266. IEEE Computer Society, 1977.

[58] J. Ledent. Internship report: Streaming string transducers. Technical report, LaBRI, University of Bordeaux, 2013.

[59] M. Lothaire. *Combinatorics on words.* Cambridge University Press, 1997.

[60] R. McNaughton and S. A. Papert. *Counter-Free Automata (M.I.T. Research Monograph No. 65).* The MIT Press, 1971.

[61] R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, EC-9(1):39–47, March 1960.

[62] G. H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[63] E. F. Moore. Gedanken experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton U., 1956.

[64] G. Pighizzini. Two-way finite automata: Old and recent results. *Fundam. Inform.*, 126(2-3):225–246, 2013.

[65] J.-E. Pin. Syntactic semigroups. In *Handbook of formal languages*, pages 679–746. Springer Berlin Heidelberg, 1997.

[66] M. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Dev.*, 3(2):114–125, 1959.

[67] M. O. Rabin. Decidability of second-order theories and automata on infinite trees. *Bull. Amer. Math. Soc.*, 74(5):1025–1029, 09 1968.

[68] F. Ramsey. On a problem of formal logic. In *Proceedings of the London Mathematical Society*, volume 30, pages 264–286, 1929.

[69] A. Saarela. Systems of word equations, polynomials and linear algebra: a new approach. *European Journal of Combinatorics*, 47(5):1–14, 2015.

[70] J. Sakarovitch. Kleene's theorem revisited. In A. Kelemenová and J. Kelemen, editors, *Trends, Techniques, and Problems in Theoretical Computer Science: 4th International Meeting of Young Computer Scientists Smolenice, Czechoslovakia, October 13–17, 1986 Selected Contributions*, pages 39–50. Springer Berlin Heidelberg, Berlin, Heidelberg, 1987.

[71] J. Sakarovitch. *Elements of Automata Theory.* Cambridge University Press, New York, NY, USA, 2009.

[72] J. Sakarovitch and R. de Souza. On the decidability of bounded valuedness for transducers. In *33rd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, volume 5162 of *Lecture Notes in Computer Science*, pages 588–600. Springer, 2008.

[73] J. Sakarovitch and R. de Souza. On the decomposition of k-valued rational relations. *CoRR*, abs/0802.2823, 2008.

[74] W. J. Sakoda and M. Sipser. Nondeterminism and the size of two way finite automata. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, pages 275–286, New York, NY, USA, 1978. ACM.

[75] M. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2):245 – 270, 1961.

[76] M. Schützenberger. A remark on finite transducers. *Information and Control*, 4(2-3):185–196, 1961.

[77] M. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190 – 194, 1965.

[78] J. Shepherdson. The reduction of two-way automata to one-way automata. *IBM J. Res. Dev.*, 3(2):198–200, 1959.

[79] I. Simon. Factorization forests of finite height. *Theoretical Computer Science*, 72(1):65–94, 1990.

[80] M. Sipser. Lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 21(2):195 – 202, 1980.

[81] M. Sipser. *Introduction to the Theory of Computation.* International Thomson Publishing, 1st edition, 1996.

[82] T. Smith. A pumping lemma for two-way finite transducers. In *Proceedings of the 39th International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 523–534. Springer Berlin Heidelberg, 2014.

[83] B. A. Trakhtenbrot. Finite automata and logic of monadic predicates. *Doklady Akademii Nauk*, SSSR 140:326 – 329, 1962. In Russian.

[84] A. Weber. *Decomposing a k-valued transducer into k unambiguous ones*, pages 503–515. Springer Berlin Heidelberg, Berlin, Heidelberg, 1992.

[85] A. Weber and R. Klemm. Economy of description for single-valued transducers. *Information and Computation*, 118(2):327 – 340, 1995.