Chapitre 1

Transductions

Emmanuel Filiot
Pierre-Alain Reynier

La robuste théorie des langages rationnels est basée sur trois piliers fondamentaux : le calcul (automates), la logique, et l'algèbre. Dans ce chapitre, nous introduisons le lecteur aux extensions de ces trois piliers aux fonctions de mots finis vers mots finis, appelées transductions, en fournissant les résultats, anciens comme récents, les plus importants de ce domaine de recherche toujours actif. Nous considérons en particulier deux classes de transductions : les transductions rationnelles, définies par des automates finis étendus avec des mots de sorties (qu'on appelle transducteurs finis), et les transductions régulières, définies par des transducteurs à tête de lecture bidirectionnelle.

1.1 Introduction

Des liens importants entre le calcul, la logique et l'algèbre ont été établis pour les langages réguliers (ou rationnels) de mots finis. La classe des langages réguliers correspond à la classe des langages reconnus par automates finis, à la classe des langages définissables en logique monadique du second ordre (*monadic second-order logic – MSO –*) avec un successeur [13, 30, 53], ainsi qu'à la classe des langages dont la congruence syntaxique, une congruence canoniquement associée à tout langage, est d'indice fini (voir par exemple [52]). Alors que les automates sont bien adaptés à l'étude des propriétés algorithmiques des langages réguliers, la vision algébrique a permis de fournir des caractérisations décidables de

sous-classes de langages réguliers. En particulier, un des résultats importants est la décidabilité, étant donné un langage régulier (représenté par un automate), de sa définissabilité en logique du premier ordre : il suffit de tester si sa congruence syntaxique (qui est calculable), est apériodique, ce qui est décidable en PSpace. Voir [29] pour plus de détails sur les langages définissables en logique du premier ordre, et [52] pour d'autres variétés de monoïdes et fragments de la logique MSO. Dans ce chapitre, notre objectif est de donner les principaux résultats, certains anciens, d'autres plus récents, qui étendent ces trois principaux piliers de la théorie des langages aux fonctions de mots vers mots.

Au point de vue computationnel, les fonctions de mots sont définies par ce qu'on appelle les *transducteurs*, qui étendent les automates par des sorties sur leur transitions. Ils peuvent être vus comme des machines de Turing à deux bandes : une bande de lecture seule, contenant initialement le mot d'entrée, et une bande d'écriture seule (toujours de gauche à droite), qui contiendra, si le mot d'entrée est accepté, l'image, par la fonction, de ce mot d'entrée, qui est elle-même un mot.

Nous allons prendre en compte deux classes importantes de transducteurs : unidirectionnel et bidirectionnel. Pour la classe des transducteurs unidirectionnels, la tête de lecture est limitée aux déplacements vers la droite. Ces transducteurs définissent la classes des fonctions dites *rationnelles*, étudiées depuis longtemps [10]. Dans le cas bidirectionnel, il n'y a pas de restrictions sur la tête de lecture et la classe que ces transducteurs définissent est appelée la classe des fonctions régulières. Cette classe a reçu une attention particulière ces dernières années, et de nouveaux résultats intéressants, que nous allons décrire, ont été obtenus.

Au niveau logique, les *MSO transducteurs* ont été introduits par Courcelle, dans le contexte plus général des transformations de graphes [18]. Restreints aux structures logiques de mots, ils permettent de définir des fonctions de mots par des formules logiques. Un mot u sur un alphabet fini Σ peut en effet être représenté par une structure logique dont le domaine est l'ensemble des positions du mot, *i.e.* l'ensemble $\{1, \dots, n\}$ où n est la longueur de u, muni des prédicats $x < y$ et $\sigma(x)$, pour toute lettre $\sigma \in \Sigma$, respectivement interprétés par l'ordre entre les positions du mot (l'ordre naturel entre les entiers) et l'ensemble des positions étiquetées par la lettre σ.

La principale idée des MSO transducteurs est de définir les prédicats du mot de sortie (ordre et lettres) par des formules (à variables libres) interprétées sur le mot d'entrée. Nous présentons des résultats établissant des liens entre transducteurs, unidirectionnels et bidirectionnels, et des classes de transducteurs MSO.

Au niveau algébrique, nous allons présenter une notion de congruence syntaxique pour les fonctions, introduite par Choffrut, qui sera d'indice fini ssi la fonction est *séquentielle*, c'est-à-dire qu'elle peut être définie par un transducteur unidirectionnel déterministe sur l'entrée. Nous présentons également une généralisation de cette caractérisation algébrique à la classe des fonctions rationnelles, un résultat dû à Reutenauer et Schützenberger. Nous donnons une application de cette dernière caractérisation au problème de définissabilité en logique du premier ordre des fonctions rationnelles. Pour les fonctions régulières, rien n'est encore connu au niveau algébrique.

Le document s'articule naturellement autour des trois piliers : automates pour les fonctions de mots (transducteurs), logique et algèbre.

1.2 Transducteurs

Dans ce chapitre, Σ représente un alphabet fini. Nous notons Σ^* l'ensemble des mots finis sur Σ et ϵ le mot vide, c'est-à-dire de longueur nulle. Pour tout mot u de longueur n, on note $\mathrm{dom}(u) = \{1, \ldots, n\}$ l'ensemble des *positions* de u ($\mathrm{dom}(u) = \varnothing$ si $u = \epsilon$). Pour $i \in \mathrm{dom}(u)$, $u[i] \in \Sigma$ est la i-ème lettre de u, $u[:i]$ est le préfixe de u jusqu'à la position i (incluse) et $u[i:]$ le suffixe de u à partir de la position i. On étend ces notations à 0 et $n+1$, $u[:0] = u[n+1:] = \epsilon$. On notera $|u|$ la longueur de tout mot u. Lorsqu'un mot u est préfixe d'un mot v, on notera $u^{-1}v$ l'unique suffixe de v tel que $u(u^{-1}v) = v$. En particulier, $u^{-1}u = \epsilon$.

Une *transduction* f est une relation binaire sur Σ^*, *i.e.* $f \subseteq \Sigma^* \times \Sigma^*$. Dans la suite, nous serons particulièrement intéressés par les transductions dont le graphe est une fonction (partielle), que nous appellerons transductions fonctionnelles. Formellement, f est fonctionnelle si et seulement si pour tout $(u, v_1), (u, v_2) \in f$, on a $v_1 = v_2$. On notera alors $f(u) = v_1$ plutôt que $(u, v_1) \in f$, et $\mathrm{dom}(f)$ sera le domaine de définition de f.

Exemple 1.2.1. *Dans ce chapitre, nous allons utiliser trois exemples filés, les fonctions f_{eff}, f_{ech} et f_{mir}, sur l'alphabet $\Sigma = \{a, b\}$. La fonction f_{eff} efface les lettres a du mot d'entrée, par exemple $f_{\mathrm{eff}}(abba) = bb$. On a de plus $dom(f_{\mathrm{eff}}) = \Sigma^*$. La fonction f_{ech} place le symbole en dernière position en première position ('ech' pour 'échange'). Elle est définie par $f_{\mathrm{ech}}(\epsilon) = \epsilon$ et pour tout $u \in \Sigma^*$ et $\sigma \in \Sigma$, par $f_{\mathrm{ech}}(u\sigma) = \sigma u$. Enfin, la fonction f_{mir} associe à un mot u le mot obtenu en concaténant u avec son image miroir, notée $\overline{u}$, c'est-à-dire $f_{\mathrm{mir}}(u) = u\overline{u}$ pour tout $u \in \Sigma^*$. Par exemple, $f_{\mathrm{mir}}(ab) = abba$.*

1.2.1 Transducteurs unidirectionnels

Les transducteurs unidirectionnels sont des machines de Turing équipées de deux bandes sur lesquelles la machine ne peut se déplacer que de la gauche vers la droite La première bande est en lecture seule et contient le mot d'entrée. La seconde bande est en écriture seule, et est utilisée pour écrire le mot produit en sortie.

Par exemple, pour réaliser la fonction f_{eff} de l'exemple 1.2.1 à l'aide d'un transducteur unidirectionnel, la machine va simplement lire le mot d'entrée de gauche à droite, et pour chaque symbole d'entrée σ, écrire σ sur la bande de sortie si σ est différent du symbole a. Formellement, un transducteur unidirectionnel est un 5-uplet $T = (Q, I, F, \Delta, t)$ où Q est un ensemble fini d'états, $I \subseteq Q$ est l'ensemble des états initiaux, $F \subseteq Q$ est l'ensemble des états finaux, $\Delta \subseteq Q \times \Sigma \times \Sigma^* \times Q$ est un ensemble fini de transitions, et $t : F \to \Sigma^*$ est la fonction de sortie terminale. Une transition $\gamma = (p, a, w, q) \in \Delta$ de T est représentée par $p \xrightarrow{a|w} q$. Intuitivement, ceci signifie que si le transducteur se trouve dans l'état p, et que le prochain symbole d'entrée est le symbole a, alors il peut passer dans l'état q et écrire le mot w sur la bande de sortie. Sans perte de généralité, on suppose que pour tous $p, q \in Q$, $a \in \Sigma$, il existe au plus un mot $w \in \Sigma^*$ tel que $(p, a, w, q) \in \Delta$. La classe des transducteurs ainsi définie est notée NFT, abréviation de transducteurs finis non-déterministes.

Étant donné un mot $u = a_1 \dots a_n \in \Sigma^*$, une exécution de T de p à q sur le mot u est une séquence d'états $\rho = (q_i)_{i=0..n}$ telle que $q_0 = p$, $q_n = q$ et, pour tout $i \in \{1, \dots, n\}$, il existe une transition de la forme $q_{i-1} \xrightarrow{a_i|w_i} q_i$ dans Δ. On dira que l'exécution $\rho = (q_i)_{i=0..n}$ est acceptante si $q_0 \in I$ et $q_n \in F$. Le mot de sortie associé à une telle exécution acceptante ρ sur le mot u, que l'on note $\text{out}^u(\rho)$, est définie comme la concaténation des mots produits par les transitions qui composent l'exécution ρ, et par l'image de l'état q_n par la fonction de sortie terminale t, c'est-à-dire $\text{out}^u(\rho) = w_1 \dots w_n t(q_n) \in \Sigma^*$.

La transduction définie (ou réalisée) par T est la relation $[\![T]\!]$ composée des paires (u, w) telles qu'il existe une exécution acceptante ρ sur le mot u vérifiant $w = \text{out}^u(\rho)$.

Un NFT T est *fonctionnel* si la transduction qu'il réalise est une fonction. La classe des NFT fonctionnels est notée fNFT. La classe des fonctions réalisées par la classe fNFT est appelée la classe des *fonctions rationnelles* [1]. Un NFT est *séquentiel* si l'automate d'entrée sous-jacent, obtenu en ignorant

1. Elles sont appelées rationnelles car elles peuvent également être définies par la notion classique de partie rationnelle d'un monoïde, voir [10].

les mots produits en sortie, est déterministe. Observons que trivialement les transducteurs séquentiels réalisent des fonctions. La classe des transducteurs séquentiels est notée SFT. La classe des fonctions réalisées par la classe SFT est appelée la classe des *fonctions séquentielles*.

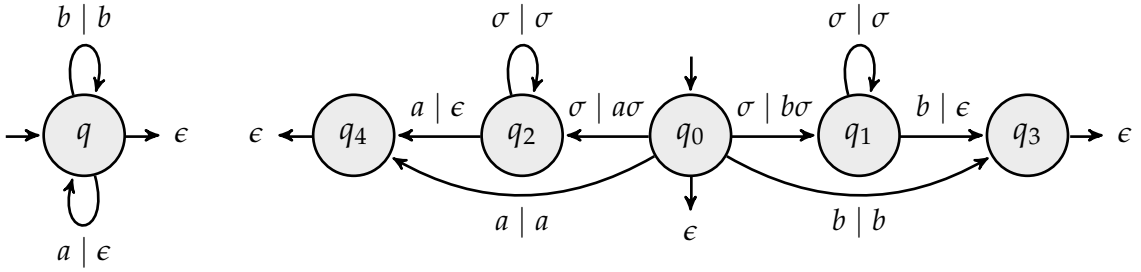


FIGURE 1.1 – *Deux transducteurs unidirectionnels T_0 et T_1 réalisant respectivement les fonctions f_{eff} et f_{ech}. Sur cette figure, $\sigma \in \{a, b\}$.*

Exemple 1.2.2. *Nous décrivons deux exemples sur l'alphabet $\Sigma = \{a, b\}$. Le transducteur T_0, représenté sur la gauche de la Figure 1.1, est fonctionnel, séquentiel, et réalise la fonction f_{eff}. Sa fonction de sortie terminale est constante et égale à ϵ. Elle est représentée par la transition sortante issue de l'état q, pointant vers le symbole ϵ.*

Le transducteur T_1, représenté sur la droite de la Figure 1.1, est fonctionnel mais n'est pas séquentiel. Il réalise la fonction f_{ech}. Intuitivement, le non déterminisme est nécessaire pour deviner la dernière lettre du mot d'entrée.

1.2.2 Transducteurs bidirectionnels

Les transducteurs bidirectionnels étendent les transducteurs unidirectionnels de la façon suivante : au lieu d'être uniquement parcourue de gauche à droite, la bande contenant le mot d'entrée est bidirectionnelle Par exemple, ceci va permettre au transducteur de se déplacer à la fin du mot d'entrée, puis de le recopier sur la bande de sortie en le lisant de la droite vers la gauche. Cette transformation va associer à un mot d'entrée son image miroir.

Formellement, nous considérons que les transducteurs bidirectionnels prennent en entrée un mot $u \in \Sigma^*$ entouré de marqueurs de début et de fin de mot, notés $\vdash$ et $\dashv$, dont on suppose qu'ils ne font pas partie de Σ. Ces marqueurs permettent au transducteur d'identifier le début et la fin du mot. Nous notons $\Sigma_{\vdash}$ l'alphabet $\Sigma \cup \{\vdash, \dashv\}$. Un *transducteur bidirectionnel* est un 4-uplet $T = (Q, I, F, \Delta)$ où Q, I et F sont comme pour les transducteurs unidirectionnels. Δ est un ensemble fini de transitions qui sont des éléments

de la forme $(p, a, w, q, m) \in Q \times \Sigma_{\vdash} \times \Sigma^* \times Q \times \{-1, +1\}$. En comparaison avec les transducteurs unidirectionnels, les transitions contiennent en plus une composante $m \in \{-1, +1\}$ qui indique la direction du déplacement (-1 pour la gauche, $+1$ pour la droite). Nous exigeons également que les transitions lisant le marqueur de début de mot se déplacent toujours vers la droite.

Notons aussi que la définition ne contient pas de fonction de sortie terminale. En effet, cette fonction n'est pas utile pour la définition des transducteurs bidirectionnels que nous avons adoptée ici car elle peut être simulée grâce au marqueur de fin de mot.

La classe de ces transducteurs est notée 2NFT, acronyme de transducteurs bidirectionnels non déterministes à états finis.

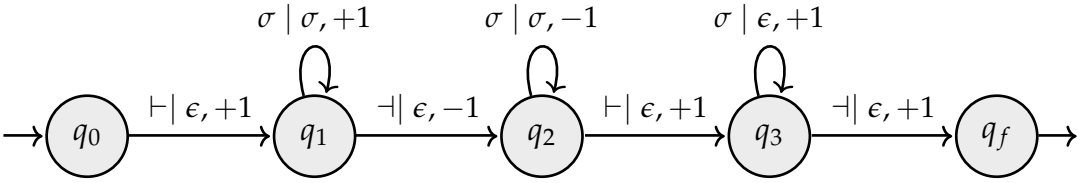


FIGURE 1.2 – *Un transducteur bidirectionnel T_2 réalisant la fonction f_{mir}.*

Une *configuration* d'un transducteur bidirectionnel T est une paire $(q, i) \in Q \times \mathbb{N}$ dans laquelle q est un état et i une position sur la bande d'entrée. On fixe à présent un mot d'entrée $u = a_1 \dots a_n \in \Sigma^*$. Une *exécution* de T sur l'entrée $u' = \vdash u \dashv$ est une séquence finie de configurations $\rho = (q_0, i_0) \dots (q_m, i_m)$ telle que $i_0 = 0$, $i_m = n + 2$, et pour tout $k \in \{0, \dots, m - 1\}$, $0 \leqslant i_k \leqslant n + 1$ et [2] $(q_k, u'[i_k], w_k, q_{k+1}, i_{k+1} - i_k) \in \Delta$, pour un certain mot w_k. Le mot de sortie associé à une telle exécution ρ sur $\vdash u \dashv$, que l'on note $\text{out}^u(\rho)$, est défini comme la concaténation des mots produits par les transitions qui composent ρ, c'est-à-dire le mot fini $w_0 \dots w_m \in \Sigma^*$.

Cette exécution est acceptante si $q_0 \in I$ et $q_m \in F$. Ceci signifie que pour toutes les configurations, sauf la dernière, la tête de lecture se situe en face d'un symbole de u'. Pour que l'exécution soit acceptante, la dernière configuration doit avoir la tête de lecture immédiatement après le marqueur de fin de mot ($i_m = n + 2$).

Comme pour les NFT, la transduction définie par T est la relation $[\![T]\!]$ composée des paires (u, w) telles qu'il existe une exécution acceptante ρ de T sur $\vdash u \dashv$ vérifiant $w = \text{out}^u(\rho)$.

Comme pour les transducteurs unidirectionnels, nous serons particu-

2. Nous utilisons la notation suivante : $u'[0]$ représente le symbole $\vdash$, $u'[i]$ pour $1 \leqslant i \leqslant n$ représente le symbole $u[i]$, et $u'[n + 1]$ représente le symbole $\dashv$.

lièrement intéressés par les sous-classes formées des transducteurs *fonctionnels* et *séquentiels*, notées respectivement f2NFT et 2SFT, et définies respectivement comme les transducteurs dont la sémantique est une fonction, et ceux dont l'automate d'entrée sous-jacent (bidirectionnel) est déterministe.

La classe des fonctions réalisées par f2NFT est appelée la classe des *fonctions régulières*. Cette terminologie provient de l'équivalence avec le formalisme logique des MSO transductions, qui va être présenté dans la prochaine section.

Exemple 1.2.3. *Considérons le transducteur T_2 réalisant la fonction f_{mir} qui est représenté sur la Figure 1.2. Il transforme tout mot d'entrée u en le mot de sortie $u\bar{u}$, où $\bar{u}$ représente l'image miroir du mot u. Les marqueurs de début et de fin de mot sont importants pour réaliser cette transformation avec un transducteur séquentiel.*

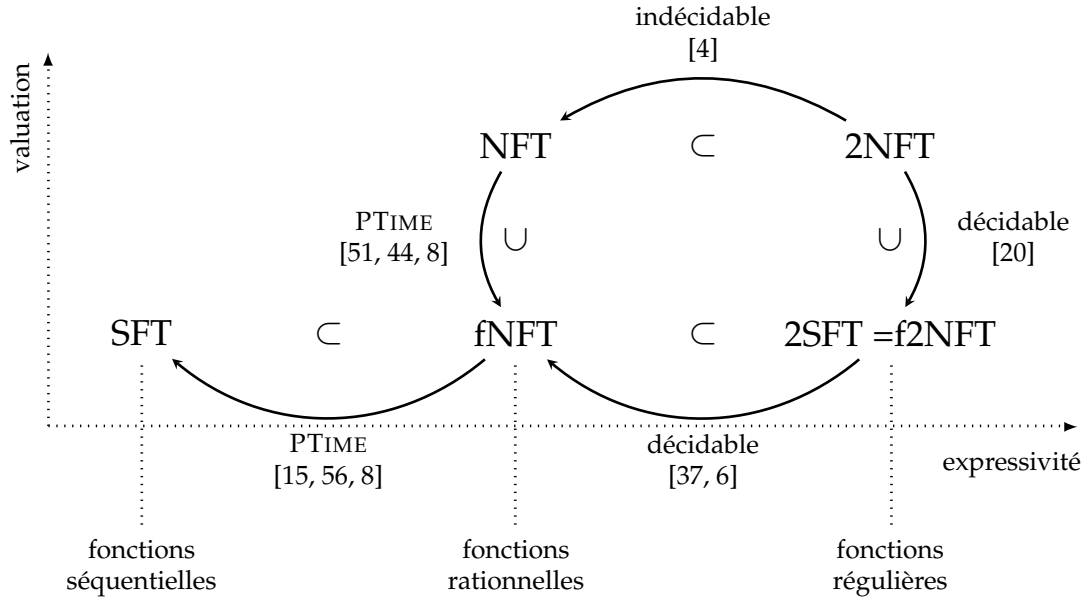


FIGURE *1.3 – Un panorama des classes de transducteurs considérées.*

1.2.3 Panorama des classes de transducteurs

La Figure 1.3 donne les relations d'inclusion existant entre les six classes de transducteurs que nous avons introduites précédemment. Nous détaillons d'abord ces inclusions. Les relations étendent naturellement les fonctions. Les fonctions rationnelles contiennent strictement les fonctions séquentielles, le caractère strict étant par exemple illustré par la fonction f_{ech}. Les fonctions régulières contiennent strictement les rationnelles,

comme le démontre la fonction f_{mir}. Enfin, contrairement au cas unidirectionnel, lorsque l'on considère des transducteurs bidirectionnels, le non-déterminisme fonctionnel n'augmente pas l'expressivité par rapport au déterminisme (égalité entre f2NFT et 2SFT), ce résultat a été démontré dans [31], et cette équivalence est effective.

Nous indiquons aussi sur cette figure le statut de différents problèmes de décision concernant l'appartenance à des sous-classes. Nous avons par exemple démontré dans [37] qu'il est décidable, étant donné un f2NFT, de déterminer s'il existe un transducteur unidirectionnel équivalent. Ce résultat de décidabilité est indiqué sur la flèche allant de f2NFT à fNFT. Concernant ce résultat, la complexité de notre procédure de décision est non-élémentaire, mais un résultat récent a démontré que ce problème peut être résolu par une procédure utilisant un espace triplement exponentiel [6]. Les autres résultats de décidabilité et d'indécidabilité sont représentés de façon similaire sur les autres flèches de la figure.

1.2.4 Problèmes d'équivalence

Transducteurs non déterministes

Tester si deux transducteurs sont équivalents, c'est-à-dire s'ils définissent la même transduction, est un problème naturel et très important. Il a par conséquent été étudié par de nombreux auteurs pour différentes classes de transducteurs. Malheureusement, ce problème est rapidement indécidable lorsque l'on considère des transducteurs non déterministes.

Le théorème suivant a été démontré dans [45] et la preuve a été simplifiée par Ismaël Jecker (Université libre de Bruxelles), lors d'une discussion pendant la préparation de ce chapitre.

Théorème 1.2.4. *Le problème suivant est indécidable : étant donnés deux* NFT T_1 *et* T_2 *sur un alphabet* Σ *quelconque contenant le symbole* $1 \in \Sigma$*, et définissant respectivement des relations* $R_1, R_2 \subseteq \Sigma^* \times \{1\}^*$*, décider si* $R_1 = R_2$*.*

Démonstration. Nous allons partir du problème de correspondance de Post (PCP), dont chaque instance est représentée par un ensemble $\{(u_1, v_1), \ldots, (u_n, v_n)\}$ de paires de mots sur l'alphabet $\{0, 1\}$, et dont la réponse est positive si et seulement si il existe une séquence d'indices $s = i_1, \ldots, i_k$ telle que $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$.

Avant de faire la réduction, introduisons tout d'abord quelques notations utiles. On prend $\Sigma = \{1, \ldots, n\}$ et pour toute séquence $s = i_1 \ldots i_k \in \Sigma^*$, on définit :

$$u_s =_{def} u_{i_1} \ldots u_{i_k}\# \qquad v_s =_{def} v_{i_1} \ldots v_{i_k}\#$$

.

On note SOL l'ensemble des solutions de l'instance de PCP.

Enfin, pour tout s, on définit la fonction $\alpha_s : \mathbb{N}^2 \to \mathbb{N}$ définie pour tout $(i,j) \in \mathrm{dom}(u_s) \times \mathrm{dom}(v_s)$ par

$$\alpha_s(i,j) =_{def} |u_s[i+1:]| + |v_s| + |v_s[:j]| \ .$$

Autrement dit, $\alpha_s(i,j)$ retourne la longueur du suffixe de u_s à partir de la position $i+1$, additionnée à la longueur de v_s et à la longueur du préfixe de v_s jusqu'à la position j (qui est donc compté deux fois).

—— **Observation 0 :** Pour toute séquence s, $\alpha_s(i,j) \leqslant 2(|u_s| + |v_s|)$.

—— **Observation 1 :** Pour tout séquence s, $\alpha_s(i,j) = |u_s| + |v_s|$ ssi $i = j$.

Par définition de α_s, on a $\alpha_s(i,j) = |u_s| + |v_s|$ ssi $|u_s[i+1:]| + |v_s| + |v_s[:j]| = |u_s| + |v_s|$, ssi $|u_s[i+1:]| + |v_s[:j]| = |u_s|$, ssi $|v_s[:j]| = |u_s[:i]|$ ssi $i = j$.

Enfin, on définit la relation

$$M = \{(s, 1^{\alpha_s(i,j)}) \mid (i,j) \in \mathrm{dom}(u_s) \times \mathrm{dom}(v_s) \ : \ u_s[i] \neq v_s[j]\} \ .$$

—— **Observation 2 :** pour toute séquence s, $(s, 1^{|u_s|+|v_s|}) \notin M$ ssi $s \in SOL$.

Démontrons cette observation. Supposons que $s \notin SOL$. Alors nécessairement, on a $u_s \neq v_s$. Puisque u_s et v_s se terminent par un symbole unique #, on déduit de $u_s \neq v_s$ qu'il existe une position $i \in \mathrm{dom}(u_s) \cap \mathrm{dom}(v_s)$ telle que $u_s[i] \neq v_s[i]$. De plus, par l'observation 1, $\alpha_s(i,i) = |u_s| + |v_s|$, donc $(s, 1^{|u_s|+|v_s|}) \in M$.

Réciproquement, si $(s, 1^{|u_s|+|v_s|}) \in M$, alors par définition de M, il existe i, j tels que $\alpha_s(i,j) = |u_s| + |v_s|$ et $u_s[i] \neq v_s[j]$ et par l'observation 1, on a nécessairement $i = j$, donc $u_s[i] \neq v_s[i]$. Autrement dit, $s \notin SOL$.

Basées sur ces observations, nous définissons deux transductions R_1, R_2 telles que $R_1 = R_2$ si et seulement si $SOL = \varnothing$:

$$\begin{array}{lcl} R_1 & = & \{(s, 1^x) \mid 0 \leqslant x \leqslant 2(|u_s| + |v_s|)\} \\ R_2 & = & \{(s, 1^x) \mid 0 \leqslant x \leqslant 2(|u_s| + |v_s|), x \neq |u_s| + |v_s|\} \cup M \end{array}$$
.

En effet, on a $R_1 = R_2$ ssi pour tout s, $(s, 1^{|u_s|+|v_s|}) \in M$, ssi pour tout s, $s \notin SOL$ d'après l'observation 2.

Pour conclure la preuve, il nous reste à montrer que R_1 et R_2 sont définissables par des transducteurs. Le transducteur définissant la transduction R_1 est donné sur la Figure 1.4. Il y a un traitement spécial du dernier indice car les mots u_s et v_s se terminent par #.

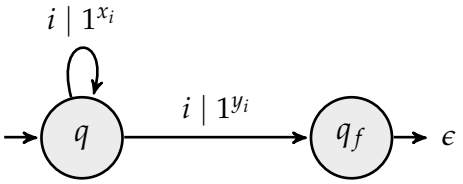


FIGURE 1.4 – *Transducteur T_1 réalisant R_1, pour tout $i \in \{1, \ldots, n\}$, pour tout $x_i \in \{0, \ldots, 2(|u_i| + |v_i|)\}$ et pour tout $y_i \in \{0, \ldots, 2(|u_i| + |v_i| + 1)\}$.*

Classe de transducteurs	domaine partiel	domaine total
SFT	NL-complet	NL
fNFT	PSpace-complet	NL
2SFT	PSpace-complet	PSpace-complet

TABLE 1.1 – *Problème d'équivalence pour différentes classes de transducteurs fonctionnels (NL désigne la classe NLogSpace).*

Nous expliquons maintenant comment construire un transducteur pour R_2. Comme R_2 est une union de deux relations, on peut simplement définir un transducteur pour chaque membre de l'union et prendre l'union disjointe de chaque transducteur. Pour la relation $\{(s, 1^x) \mid 0 \leqslant x \leqslant 2(|u_s| + |v_s|), x \neq |u_s| + |v_s|\}$, il faut faire une disjonction de cas : soit on est strictement plus petit que $|u_s| + |v_s|$, soit on est strictement plus grand. Dans le premier cas, à chaque symbole i lu en entrée (excepté le dernier), il faut sortir un mot 1^x avec $x \leqslant |u_i| + |v_i|$. Le dernier indice nécessite un traitement spécial car u_s et v_s se terminent par #. On utilise ce cas pour obtenir l'inégalité stricte ($x < |u_s| + |v_s|$) escomptée. L'autre cas ($x > |u_s| + |v_s|$) se traite de manière similaire. Pour la relation M, le transducteur lit en entrée une séquence d'indices $s = i_1 \ldots i_k$. Il va deviner deviner une position i et une position j dans les mots u_s et v_s (en utilisant le non-déterminisme) et vérifier que la $i^{\text{ème}}$ lettre de u_s est bien différente de la $j^{\text{ème}}$ lettre de v_s (pour cela, il va en fait deviner deux indices α et β dans $\{1, \ldots, k\}$ correspondant à ces indices i et j). On peut ensuite vérifier qu'on peut produire la sortie de façon correcte puisque l'on sait si l'on se trouve avant ou après i et j. ■

Transducteurs fonctionnels

Lorsque l'on se restreint à des transducteurs fonctionnels, alors le problème de l'équivalence est décidable pour toutes les classes que nous avons présentées précédemment. Ceci explique en partie l'intérêt porté aux transducteurs fonctionnels. Ces résultats sont synthétisés dans la Table 1.1.

Nous détaillons à présent les résultats présentés dans cette table. Pour décider si deux transducteurs *fonctionnels* sont équivalents, on peut procéder ainsi : on teste d'une part s'ils ont le même domaine, et d'autre part si le transducteur défini par l'union disjointe des deux transducteurs est encore fonctionnel. Le premier problème est bien connu pour les différentes classes de transducteurs vues précédemment, puisqu'il s'agit du problème d'équivalence d'automates finis, unidirectionnels ou bidirectionnels. On sait par exemple que tester l'équivalence de deux automates finis déterministes est NL-complet et que ce problème devient PSpace-complet si les automates sont non déterministes ou s'ils sont déterministes et bidirectionnels.

La fonctionnalité des transducteurs unidirectionnels a largement été étudiée [51, 44, 8] et il a été démontré récemment dans [40] que ce problème est dans la classe NL. La preuve utilise des machines à compteurs pour déterminer, de façon non déterministe, un mot d'entrée u, deux exécutions ρ_1 et ρ_2 sur ce mot u, et une position de sortie i tels que $\text{out}^u(\rho_1)[i] \neq \text{out}^u(\rho_2)[i]$. On obtient ainsi les résultats des deux premières lignes de la table.

Concernant les transducteurs bidirectionnels, Gurari a montré que le problème de l'équivalence est PSpace-complet [43]. Là aussi, sa preuve utilise des machines à compteurs pour deviner un témoin de non équivalence. Pour montrer que le problème est encore PSpace-difficile dans le cas où les transducteurs sont déterministes, on peut procéder comme suit. On réduit le problème du vide de l'intersection de n automates finis déterministes $A_1, \ldots, A_n$ au problème d'équivalence de deux transducteurs déterministes bidirectionnels. Le premier transducteur T_1 réalise la fonction f_1 suivante : pour tout mot u, $f_1(u) = \#$ si u appartient à l'intersection des langages des A_i, et $f_1(u) = \$$ sinon. Il est facile de construire T_1 de taille linéaire dans la taille des A_i : celui-ci va simplement lire n fois le mot en simulant successivement chacun des A_i. Si un des A_i n'accepte pas le mot u, T_1 retourne le symbole \$ et sinon, il retourne le symbole #. Le second transducteur T_2 envoie tous les mots d'entrée u sur le symbole \$. Par conséquent, on a immédiatement que T_1 et T_2 sont équivalents ssi l'intersection des langages des A_i est vide.

Transducteurs finiment valués

La frontière entre décidabilité et indécidabilité pour le problème d'équivalence se situe en réalité bien au-delà des transducteurs fonctionnels. On peut en effet considérer la classe des transducteurs finiment valués, définie comme les transducteurs pour lesquels il existe un entier k tel que pour tout mot d'entrée, le nombre de mots de sortie associés à ce mot est au plus k. Il

a été démontré dans [21, 55, 27] que le problème de l'équivalence de transducteurs unidirectionnels finiment valués est décidable. Il est également affirmé dans [21] que le problème d'équivalence est même décidable pour les transducteurs bidirectionnels finiment valués.

1.3 Définissabilité des transductions en logique

Un formalisme, basé sur la logique monadique du second-ordre (MSO), a été introduit par Courcelle dans [18] pour définir des transformations de graphes. Dans cette section, nous allons restreindre ce formalisme à des structures de mots finis, afin de définir des transductions de mots. Le lecteur intéressé pourra se référer au livre [19] pour plus de détails et de résultats sur ce formalisme dans le contexte plus général des transformations de graphes.

1.3.1 Mots, structures et MSO

Un mot w sur un alphabet Σ peut être vu comme une structure logique de domaine $\{1, \ldots, |w|\}$ (les positions du mots), sur le vocabulaire contenant les prédicats unaires $\sigma(x)$ pour chaque lettre $\sigma \in \Sigma$, chacun interprété par l'ensemble des positions de w étiquetées par σ, et le prédicat binaire $x \preceq y$ interprété par l'ordre naturel entre les positions du mots. Nous rappelons que MSO sur les mots, est l'extension de la logique du premier ordre (FO) avec la quantification sur des ensembles de positions. Dans le cas des structures de mots, la syntaxe est donnée par la grammaire suivante :

$$\varphi \ ::= \ \exists x\ \varphi \mid \exists X\ \varphi \mid \sigma(x) \mid x \preceq y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \qquad\qquad \sigma \in \Sigma$$

où x appartient à un ensemble de variables dites du premier ordre (interprétées par des positions du mots) et X appartient à un ensemble de variables dites du second ordre (interprétées par des ensembles de positions). Le lecteur est renvoyé à la littérature, par exemple [52], pour plus de détails sur la sémantique de MSO. On notera $w \models \varphi$ le fait qu'une formule φ (sans variables libres, dite close) est satisfaite par un mot w (vu comme une structure).

Une formule close φ définit un langage $L_\varphi = \{w \in \Sigma^* \mid w \models \varphi\}$. On dira qu'un langage L est *MSO-définissable* s'il existe une formule close de MSO φ telle que $L = L_\varphi$. Le théorème suivant, souvent appelé théorème de Büchi, établit un lien entre les automates finis et la logique MSO :

Théorème 1.3.1 (Büchi, Elgot, Trakhtenbrot [30, 53, 13])**.** *Soit $L \subseteq \Sigma^*$ un langage. L est régulier si et seulement si L est MSO-définissable.*

Ce théorème, dont la preuve est effective (on peut à partir d'un automate calculer une formule équivalente et réciproquement), a eu de nombreuse implications, notamment dans le domaine de la vérification [54].

Lorsque φ a des variables libres du premier ordre $x_1, \ldots, x_k$ et du second ordre $X_1, \ldots, X_m$, on étend la satisfaction aux mots w étendus d'une interprétation ν de ces variables (ν est une fonction associant à chaque variable du premier ordre une position de w et à chaque variable du second ordre un ensemble de positions de w). On notera $w, \nu \models \varphi$ lorsque le mot w satisfait la formule φ en utilisant l'interprétation ν des variables libres.

Exemple 1.3.2. *La formule $min(x) =_{def} \forall y\ x \preceq y$ est satisfaite par tout mot w et la première position x de w. De manière similaire, on peut définir $max(x)$ satisfaite par la dernière position.*

La formule $S(x,y) =_{def} x \preceq y \wedge \neg\exists z\ x \prec z \prec y$ définit la relation successeur entre les positions.

Considérons maintenant la formule suivante :

$$\psi(X) =_{def} \forall x \forall y\ S(x,y) \rightarrow [(x \in X) \leftrightarrow (y \notin X)]\ .$$

Elle est satisfaite, sur tout mot w, par exactement deux ensembles de positions X : les positions paires et les positions impaires.

La formule

$$\varphi =_{def} \exists X\ \psi(X) \wedge \forall x\ min(x) \rightarrow x \in X \wedge max(x) \rightarrow x \notin X$$

est satisfaite par tout mot de longueur paire. Cela montre que le langage régulier des mots de longueur paire est MSO-définissable.

Ajoutons quelques contraintes à cette formule :

$$\begin{aligned} \varphi_{aa} \quad =_{def} \quad & \exists X\ \psi(X) \wedge \forall x\ min(x) \rightarrow x \in X \wedge max(x) \rightarrow x \notin X \\ & \wedge \forall x\ (x \in X \rightarrow a(x)) \wedge (x \notin X \rightarrow a(x))\ . \end{aligned}$$

Elle définit le langage des mots de longueur paire ne contenant que des a, ce qui est dénoté par l'expression rationnelle $(aa)^*$.

Modifions légèrement cette formule :

$$\begin{aligned} \varphi_{ab} \quad =_{def} \quad & \exists X\ \psi(X) \wedge \forall x\ min(x) \rightarrow x \in X \wedge max(x) \rightarrow x \notin X \\ & \wedge \forall x\ (x \in X \rightarrow a(x)) \wedge (x \notin X \rightarrow b(x))\ . \end{aligned}$$

Elle définit maintenant le langage $(ab)^*$. A la différence du langage $(aa)^*$, le langage $(ab)^*$ peut être simplement défini par la formule du premier ordre suivante :

$$\begin{aligned} \varphi^{FO}_{ab} \quad =_{def} \quad & \forall x(min(x) \rightarrow a(x)) \wedge (max(x) \rightarrow b(x)) \\ & \wedge \forall y\ S(x,y) \rightarrow [(a(x) \rightarrow b(y)) \wedge (b(x) \rightarrow a(y))]\ . \end{aligned}$$

Plus généralement, on peut se demander quand une formule MSO est équivalente (au sens sémantique) à une formule de FO. Ce problème, décidable, est connu comme le problème de la FO-définissabilité.

1.3.2 Transducteurs MSO

Dans cette section, nous définissons les transducteurs MSO. Leur sémantique, qui est assez technique, est donnée de manière intuitive. Le lecteur est renvoyé vers [18, 31, 34] pour une définition plus détaillée.

Dans un transducteur MSO, la structure logique correspondant au mot de sortie est définie par une interprétation MSO d'un nombre fixé k de copies de la structure logique correspondant au mot d'entrée. Dans la suite, afin d'alléger le texte, nous parlerons de mot plutôt que de structure, tout en gardant en tête le fait qu'ils sont vus comme des structures. Ce mécanisme de copie est formalisé par le fait que les positions du mot de sortie sont des copies (de 1 à k) des positions du mot d'entrée. La position correspondant à la copie c d'une position d'entrée x sera notée x^c. Un MSO transducteur ne garde pas toutes les copies pour définir le mot de sortie, mais un sous-ensemble pour chaque c, défini par une formule MSO notée $\phi^c_{pos}(x)$, à une variable libre x, interprétée sur le mot d'entrée. Autrement dit, l'élément x^c sera dans le mot de sortie si et seulement si le mot d'entrée et la position x satisfont la formule $\phi^c_{pos}(x)$. Par exemple, supposons que nous prenions deux copies du mot d'entrée et que dans la première copie, nous gardions toutes les positions étiquetées par a, et dans la seconde copie, toutes les positions étiquetées par b. Ceci sera spécifié par les deux formules $\phi^1_{pos}(x) = a(x)$ et $\phi^2_{pos}(x) = b(x)$.

Les lettres du mot de sortie ainsi que l'ordre entre ses différentes positions sont définis par des formules MSO avec une et deux variables libres (du premier ordre) respectivement, interprétées, comme pour les formules $\phi^c_{pos}(x)$, sur le mot d'entrée. Par exemple, sur l'alphabet $\Sigma = \{a, b\}$, pour exprimer le fait que toutes les positions du mot de sortie sont étiquetées par a, on définit les formules $\phi^c_a(x) = \top$ et $\phi^c_b(x) = \bot$ pour toute copie c. L'ordre relatif entre les positions du mot de sortie met en relation des copies, possiblement d'index différents, de positions du mot d'entrée. Par conséquent, l'ordre est défini par des formules MSO de la forme $\phi^{c,d}_{\preceq}(x, y)$ pour toutes copies $1 \leqslant c, d \leqslant k$.

Enfin, une formule MSO close ϕ_{dom} définit le domaine de la fonction comme tous les mots qui satisfont ϕ_{dom}. Finalement, un MSO transducteur sur un alphabet Σ est un 5-uplet

$$T = (k, \phi_{dom}, (\phi^c_{pos}(x))_{1\leqslant c\leqslant k}, (\phi^c_a(x))_{1\leqslant c\leqslant k, a\in\Sigma}, (\phi^{c,d}_{\preceq}(x, y))_{1\leqslant c,d\leqslant k}) \,.$$

La structure de sortie n'est pas forcément un mot de manière générale (par exemple si les formules $\phi^{c,d}_{\preceq}(x,y)$ définissent un cycle), mais pour être un transducteur MSO, on demande qu'en plus les structures de sortie correspondent bien à des mots, ce qui est une propriété décidable (non expliquée dans ce chapitre).

Avant de donner quelques exemples de transducteurs MSO, remarquons que tout mot d'entrée de longueur M sera transformé, par un transducteur MSO, en un mot de sortie de longueur au plus kM, où k est une constante qui dépend uniquement du transducteur (c'est le nombre de copies pour être précis).

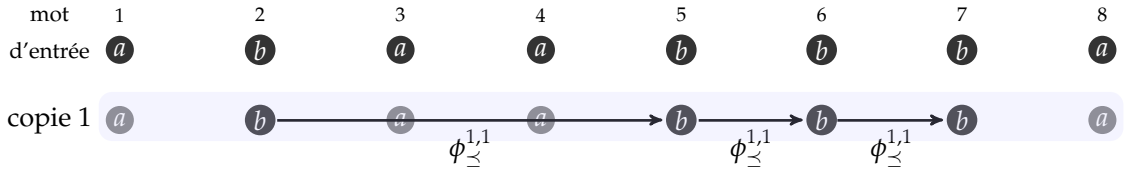


FIGURE 1.5 – *Transduction f_{eff} définie par T_{eff}.*

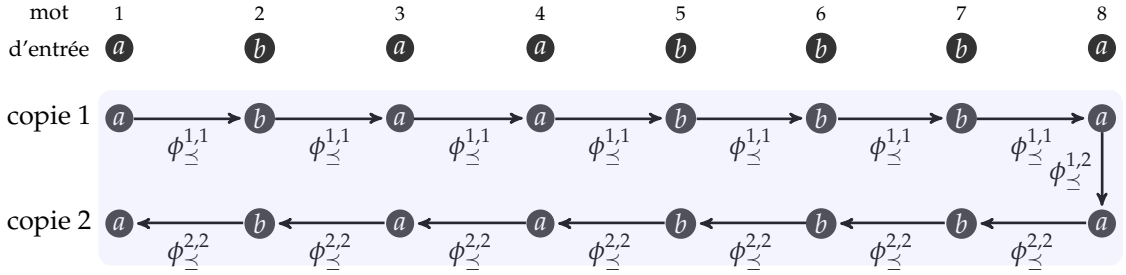


FIGURE 1.6 – *Transduction f_{mir} définie par T_{mir}.*

1.3.3 Exemples de transducteurs MSO

Notre premier exemple est un transducteur MSO T_{eff} qui réalise la fonction f_{eff} illustré dans la Figure 1.5 (seulement la relation successeur entre les positions est représentée). Les copies des positions du mot d'entrée qui ne sont pas gardées dans la sortie, *i.e.* qui sont filtrées par les formules $\phi^{c}_{pos}(x)$, sont représentées par des nœuds floutés. Le transducteur T_{eff} est défini par le tuple

$$T_{\text{eff}} = (1, \phi_{dom} \equiv \top, \phi^{1}_{pos}(x) \equiv \neg a(x), (\phi^{1}_{\sigma}(x) \equiv \sigma(x))_{\sigma\in\Sigma}, \phi^{1,1}_{\preceq}(x,y) \equiv x \preceq y)\ .$$

Comme deuxième exemple, considérons la fonction f_{mir}. Pour la réaliser avec un transducteur MSO (Figure 1.6), nous avons besoin de deux copies

du mot d'entrée. Les symboles sont gardés à l'identique, en revanche, l'ordre entre les positions est inversé dans la seconde copie. Nous devons aussi exprimer le fait que toutes les positions de la première copie sont avant les positions de la deuxième copie. Formellement, nous avons le transducteur T_{mir}, pour $k = 2$, défini par les formules suivantes :

$$\begin{array}{rclrcl} \phi_{dom} & \equiv & \top & \phi^c_{pos}(x) & \equiv & \top \\ \phi^c_a(x) & = & a(x) & \phi^c_b(x) & = & b(x) \\ \phi^{1,1}_{\preceq}(x,y) & \equiv & x \preceq y & \phi^{1,2}_{\preceq}(x,y) & \equiv & \top \\ \phi^{2,1}_{\preceq}(x,y) & \equiv & \bot & \phi^{2,2}_{\preceq}(x,y) & \equiv & y \preceq x \end{array}$$

pour tout copie $c \in \{1, 2\}$. Nous remarquons qu'étant donné un mot d'entrée u, l'ordre entre les positions du mot de sortie est égal à la relation binaire $O = \{(i^c, j^d) \mid 1 \leqslant c, d \leqslant 2, i, j \in dom(u), u \models \phi^{c,d}_{\preceq}(i,j)\}$. Uniquement la relation successeur induite par O est représentée sur la figure.

1.3.4 Théorèmes de Büchi pour les fonctions de mots

Fonctions régulières. La première correspondance entre automates et logique pour les transductions de mots a été prouvée par Engelfriet et Hoogeboom pour la classe des fonctions régulières :

Théorème 1.3.3 ([31]). *Une fonction f est régulière (définissable par un transducteur bidirectionnel) ssi elle est définissable par un transducteur MSO.*

La preuve de ce résultat est techniquement complexe, nous en donnons les principaux arguments. Pour prouver que toute fonction régulière est MSO-définissable, l'astuce est de partir d'un transducteur bidirectionnel déterministe et de construire un transducteur MSO qui, étant donné un mot d'entrée u, va produire un mot de sortie qui correspond à la suite des transitions sur ce mot (l'exécution du transducteur sur ce mot). Le nombre de copies correspond au nombre maximal de fois qu'une position d'entrée est visitée par le transducteur bidirectionnel. Ce MSO transducteur est ensuite composé avec un autre MSO transducteur qui va uniquement garder les mots de sortie des transitions. Les MSO transducteurs étant clos par composition [18], cela nous donne le résultat. Notons que pour définir le premier MSO transducteur (celui qui produit les exécutions du transducteur bidirectionnel), il est nécessaire de pouvoir exprimer, en MSO, étant données une position d'entrée x et une transition t, que la transition t va être tirée à la position x par l'exécution du transducteur bidirectionnel sur le mot. Ceci est une propriété régulière (on peut facilement faire un automate bidirectionnel tournant sur des mots avec une position marquée

x et simulant le transducteur bidirectionnel pour décider cette propriété), et d'après le théorème de Büchi, elle est MSO définissable. Nous faisons donc ici appel au théorème de Büchi sur les langages.

La réciproque, c'est-à-dire aller des transducteurs MSO vers les transducteurs bidirectionnels, est plus complexe, et passe par un modèle intermédiaire de transducteurs bidirectionnels réalisant des "sauts MSO" $\varphi(x, y)$, où $\varphi(x, y)$ est une formule MSO définissant une fonction des positions x vers des positions y. Intuitivement, une telle machine, dont la tête de lecture est à la position x, peut se déplacer directement (sauter) à une position y dès lors que $\varphi(x, y)$ est satisfaite sur le mot d'entrée. Passer d'un transducteur MSO à un tel modèle est assez facile : les formules de sauts sont obtenues en exprimant la relation successeur de l'ordre défini par les formules $\phi^{c,d}_{\preceq}(x, y)$, ordre restreint aux positions qui sont conservées par les formules $\phi^{c}_{pos}(x)$. Ensuite, l'idée est de passer des "sauts" à des "petits pas" (de longueur unitaire). Pour cela, il faut enrichir les états du transducteur bidirectionnel avec suffisamment d'information (calculée à partir du mot d'entrée) pour qu'à tout moment, le transducteur puisse décider quelles sont les formules de la forme $\exists y\ \varphi(x, y)$ qui sont satisfaites ou non, où x est la position courante. Lorsqu'une telle formule est satisfaite, le transducteur doit alors se déplacer à (l'unique) y qui satisfait la formule de saut. Cette partie est non triviale, et basée de nouveau sur le théorème de Büchi.

Fonctions rationnelles. En utilisant une restriction adéquate des transducteurs MSO, il est aussi possible de démontrer un théorème de Büchi pour les fonctions rationnelles. Intuitivement, cette restriction, appelée "préservant l'ordre", demande que, dans la représentation graphique qui représente le mot de sortie (voir par exemple les Figures 1.5 et 1.6), il n'y ait pas d'arêtes de la droite vers la gauche. Autrement dit, l'ordre de l'entrée doit être préservé en sortie. Par exemple, le transducteur MSO de la Figure 1.5 préserve l'ordre alors que celui utilisé pour la Figure 1.6 non. Cette restriction a été utilisée dans [11, 34] pour démontrer une correspondance entre fonctions rationnelles (définies par des transducteurs unidirectionnels) et transducteurs MSO préservant l'ordre.

Dans ce chapitre, nous allons utiliser un autre formalisme plus simple, mais équivalent aux transducteurs MSO préservant l'ordre [46]. Nous les appellerons des transducteurs MSO gauche-droite. Un *transducteur MSO gauche-droite* est un tuple $T = (K, \phi_{dom}, (\phi_v(x))_{v \in K})$ tel que $K \subseteq \Sigma^*$ est un ensemble **fini** de mots, ϕ_{dom} est une formule MSO close (sur Σ), et $\phi_v(x)$ est une formule MSO à une variable libre. La relation définie par T est l'ensemble des paires (u, w) tel que $u \models \phi_{dom}$ et u se décompose en n

symboles $u = \sigma_1 \dots \sigma_n$ et w se décompose en n mots de K, $w = v_1 \dots v_n$, $v_i \in K$, et pour tout $i \in \{1, \dots, n\}$, $u \models \phi_{v_i}(i)$.

Par exemple, la fonction f_{eff} est définie par $K = \{\epsilon, b\}$, $\phi_{dom} = \top$, $\phi_\epsilon(x) = a(x)$ et $\phi_b(x) = b(x)$. Nous pouvons maintenant énoncer le théorème de Büchi pour les fonctions rationnelles :

Théorème 1.3.4 ([11, 34])**.** *Une fonction f est rationnelle si et seulement si elle est définissable par un transducteur MSO gauche-droite.*

Voici les arguments principaux de la preuve. Étant donné un transducteur MSO gauche-droite, on construit un transducteur fini dont les états calculent assez d'information pour être capable de décider quelles sont les formules $\phi_v(x)$ satisfaites par la position courante x. Si dans un état q en lisant une lettre σ, l'information q et la lettre σ permettent de déduire que $\phi_v(x)$ est satisfaite par la position courante x, le transducteur produit la sortie v. La difficulté est de calculer cette information, qui doit être finie : il suffit de prendre ce qui s'appelle les MSO-types jusqu'à un niveau de quantification borné, notion bien connue en théorie des modèles finis et qui sort du cadre de ce chapitre.

La réciproque est plus aisée : étant donné un transducteur fini T, on prend K l'ensemble des mots de sortie pouvant apparaître sur des transitions de T, et toute formule $\phi_v(x)$ exprime le fait qu'il existe une exécution acceptante qui tire une transition t sur la position x, dont la sortie est v. Ceci est une propriété régulière et qui est donc MSO-définissable par le théorème de Büchi. Attention, la construction précédente est correcte si le transducteur T n'est pas ambigu, *i.e.* qu'il existe au plus une exécution acceptante sur tout mot d'entrée. Ceci peut être supposé sans perte de généralité : il est connu que toute fonction rationnelle est définissable par un transducteur non-ambigu [10, 49].

Nous illustrons la puissance des connections entre transducteurs à états et logique par l'application suivante. Le problème consiste, étant donné un transducteur MSO, à décider si ce transducteur MSO est *équivalent* à un transducteur MSO gauche-droite. Autrement dit, si jamais le transducteur MSO de départ faisait des retours en arrière, est-ce que ces retours sont réellement nécessaires ? Par exemple, la fonction g qui à a^n associe a^{2n} peut être réalisée par le transducteur MSO T_{mir} de la Figure 1.6, restreint à l'alphabet $\{a\}$. La passe retour effectuée par ce transducteur n'est pas nécessaire : en lisant un a d'entrée on peut produire deux a en sortie et réaliser cette fonction par une passe gauche-droite. Autrement dit, la fonction g est réalisable par le transducteur MSO gauche-droite T_g avec $K = \{aa\}$, $\phi_{dom} = \forall x.a(x)$ et $\phi_{aa}(x) = \top$.

Décider ce problème de définissabilité au niveau syntaxique de la logique ne semble pas chose aisée. Grâce aux correspondances avec les transducteurs à états, la question revient alors à se demander, étant donné un transducteur bidirectionnel réalisant une fonction, si cette fonction est également réalisable par un transducteur unidirectionnel. Les transducteurs à états étant plus adaptés à des raisonnements algorithmiques, cette question a pu être résolue positivement dans [37], ce qui donne le théorème suivant :

Corollaire 1.3.5. *Le problème suivant est décidable : soit f une fonction donnée par un transducteur MSO, est-ce que f est définissable par un transducteur MSO gauche-droite ?*

1.4 Définissabilité en logique du premier ordre

Dans cette section, nous allons étudier des problèmes de définissabilité de transduction dans des fragments logiques, notamment ici la logique du premier ordre. Nous commençons par rappeler comment ces problèmes sont résolus dans le cas des langages, et passons ensuite aux transductions séquentielles. Enfin, nous ferons un rapide état de l'art pour le cas des fonctions rationnelles et régulières.

1.4.1 Le cas des langages de mots finis

Le problème de définissabilité prend ses origines en logique et a été largement étudié dans le cadre des langages réguliers de mots finis. Il consiste à décider, étant donnés une formule MSO close φ et un fragment $\mathcal{F}$ de MSO (par exemple, FO), à décider si φ est équivalente à une formule de $\mathcal{F}$ (voir par exemple [52, 28] pour des résultats sur le sujet). Une des techniques puissantes pour décider des problèmes de définissabilité logique consiste à se ramener à un problème de définissabilité entre classes d'automates, et se base trois ingrédients correspondants à avoir les propriétés suivantes :

1. l'existence d'une correspondance (effective) entre le fragment logique $\mathcal{F}$ considéré et une sous-classe $\mathcal{C}$ des automates finis,
2. la préservation de l'appartenance à $\mathcal{C}$ par minimisation. Autrement dit, il faut que partant d'un automate $A \in \mathcal{C}$, l'automate minimal M pour le langage $L(A)$ satisfasse $M \in \mathcal{C}$,
3. la décidabilité du problème d'appartenance d'un automate à la classe $\mathcal{C}$.

En effet, lorsqu'on a ces trois propriétés, pour décider si une formule MSO close φ est équivalente à une formule de $\mathcal{F}$, il suffit de (i) convertir φ en automate fini A (par le théorème de Büchi), (ii) minimiser A en un automate M et (iii) décider si $M \in \mathcal{C}$ est vrai (ce qui est possible par la propriété 3). Si de plus on veut effectivement calculer une formule de $\mathcal{F}$, il suffit de convertir M en une formule de $\mathcal{F}$, ce qui est possible par la propriété 1.

Pour la correction de cet algorithme, supposons que φ soit équivalente à une formule de $\mathcal{F}$. Alors d'après la propriété 1, il existe un automate $A \in \mathcal{C}$ dont le langage L est celui défini par φ. D'après la propriété 2, l'automate (déterministe) minimal pour L est dans $\mathcal{C}$, et de nouveau d'après 1, on peut le convertir en une formule de $\mathcal{F}$ définissant le même langage. L'algorithme va donc retourner 'oui'. Réciproquement, supposons que l'algorithme retourne 'oui'. Alors, comme toutes les transformations faites dans l'algorithme préservent le langage de départ (celui défini par la formule MSO), on obtient bien un automate de $\mathcal{C}$ qui le reconnaît, et d'après la propriété 1, cela implique que ce langage est définissable par une formule de $\mathcal{F}$.

FO-définissabilité. Dans le cas du fragment premier-ordre de MSO, nous exposons une sous-classe d'automates $\mathcal{C}$ telle que les propriétés $1-3$ sont satisfaites, ce qui implique la décidabilité du problème de définissabilité de FO à partir de MSO.

Propriété 1**.** Elle est due à Schützenberger, McNaughton et Papert [50, 47], qui ont établi une correspondance entre FO et les automates *apériodiques* (qui peuvent être supposés déterministes et complets). Un automate est apériodique s'il existe un entier n_0 tel que pour $n \geqslant n_0$, toute paire d'états (p, q) et tout mot $u \in \Sigma^*$, on a $p \xrightarrow{u^n} q$ si et seulement si $p \xrightarrow{u^{n+1}} q$ (l'existence d'une exécution de p à q sur u^n est équivalente à l'existence d'une exécution de p à q sur u^{n+1}). De manière équivalente, nous pouvons définir une congruence $\sim_A$ sur Σ^*, pour tout automate déterministe A complet, comme $u \sim_A v$ si et seulement si pour tout état q, on a $q.u = q.v$, où $q.u$ (resp. $q.v$) est l'état atteint par A en lisant u (resp. v) à partir de q. Il est facile de vérifier que $\sim_A$ est une congruence (exercice), appelée la *congruence de transition* de A. On peut alors reformuler la définition de l'apériodicité (pour les automates déterministes complets) comme ceci :

Définition 1.4.1. *Un automate (déterministe et complet) A est* apériodique *si il existe un entier naturel n_0 tel que pour tout $n \geqslant n_0$ et $u \in \Sigma^*$, on a $u^n \sim_A u^{n+1}$.*

On dira aussi que la congruence de transition de A est apériodique.

Propriété 2. Elle énonce que l'apériodicité des congruences de transition doit être préservée par minimisation, ce qui est bien le cas. Pour le voir, rappelons que l'automate déterministe complet minimal pour tout langage régulier L est unique et isomorphe à l'automate dont les états sont des classes pour la congruence de Myhill-Nerode $\equiv_L$ définie par

$$u \equiv_L v \text{ si et seulement si } u^{-1}L = v^{-1}L$$

où $u^{-1}L$ est le *résiduel de L par u*, défini par le langage $u^{-1}L = \{w \mid uw \in L\}$ (et de manière similaire pour $v^{-1}L$). Si on note $[u]$ la classe d'un mot u, et pour $U \subseteq \Sigma^*$, $[U] = \{[u] \mid u \in U\}$, l'automate minimal est alors donné par l'ensemble d'états $\Sigma/_{\equiv_L}$ (les classes d'équivalence, en nombre fini d'après le théorème de Myhill-Nerode, pour tout langage régulier L), l'état initial $[\epsilon]$, les états finaux $[L]$, et la relation de transition $[u] \xrightarrow{\sigma} [u\sigma]$ pour tout $u \in \Sigma^*$ et $\sigma \in \Sigma$.

L'automate minimal M pour un langage régulier L n'a pas simplement la propriété d'être unique et minimal en nombre d'états, mais a aussi la propriété d'avoir la congruence de transition la plus grossière. Autrement dit, on a le lemme suivant :

Lemme 1.4.2. *Pour tout automate A reconnaissant L et pour tout $u, v \in \Sigma^*$, si $u \sim_A v$, alors $u \sim_M v$.*

Démonstration. Supposons en effet que $u \sim_A v$ et démontrons que $u \sim_M v$. Soit $\alpha \in \Sigma^*$. On sait que dans M, $[\alpha] \xrightarrow{u} [\alpha u]$. On veut démontrer que $[\alpha] \xrightarrow{v} [\alpha u]$, autrement dit, que $\alpha u \equiv_L \alpha v$. En revenant à la définition de $\equiv_L$, il faut donc démontrer que $(\alpha u)^{-1}L = (\alpha v)^{-1}L$, autrement dit, que pour tout w, $\alpha uw \in L$ ssi $\alpha vw \in L$. Puisque $u \sim_A v$ et que $\sim_A$ est une congruence, on sait que $\alpha u \sim_A \alpha v$. En particulier si q_0 est l'état initial de A, alors $q_0.(\alpha u) = q_0.(\alpha v)$. Pour conclure, puisque l'automate A est déterministe, on obtient que $\alpha uw \in L$ ssi $q_0.(\alpha uw) \in F$ (F étant les états finaux de A) ssi $(q_0.\alpha u).w \in F$ ssi $(q_0.\alpha v).w \in F$ ssi $\alpha vw \in L$. ■

En particulier, d'après ce lemme, si $u^n \sim_A u^{n+1}$ pour un certain n, alors $u^n \sim_M u^{n+1}$, ce qui implique que dès lors que A est apériodique, M l'est aussi, assurant la propriété 2.

Propriété 3. Étant donné un automate, il peut être décidé en PSpace s'il est apériodique [29]. Cette complexité est d'ailleurs optimale, puisque le problème est également PSpace-difficile, même si l'automate est déterministe.

1.4.2 Les cas des fonctions séquentielles

Dans cette section, nous allons étendre les résultats vus précédemment des langages aux fonctions séquentielles. Nous dirons qu'une fonction f est FO-définissable si elle est réalisée par un MSO-transducteur gauche-droite dont toutes les formules sont du premier ordre. Nous l'appellerons alors FO-transducteur gauche-droite. Nous allons détailler les étapes permettant de démontrer le théorème suivant :

Théorème 1.4.3. *Le problème suivant est décidable : soit f une fonction séquentielle donnée par un MSO-transducteur gauche-droite, est-ce que f est définissable par un FO-transducteur gauche-droite ?*

Il faut d'abord remarquer que les trois propriétés données dans le cas des langages, pourvu qu'ils existent dans le cas des transducteurs à états finis, nous donnent un algorithme pour le problème de ce théorème.

Propriété 1 : correspondance logique-transducteurs

Comme on l'a vu, la première propriété est l'existence d'une correspondance entre la logique (ici les FO-transducteurs) et les transducteurs à états. Comme pour les langages, nous définissons les *transducteurs apériodiques* comme la sous-classe des transducteurs dont l'automate sous-jacent (l'automate obtenu en oubliant les mots de sortie) est apériodique. Il existe alors une correspondance connue [11] entre FO-transducteurs et transducteurs (unidirectionnels) apériodiques, qui a été renforcée au cas des fonctions séquentielles dans [46] :

Théorème 1.4.4 ([11, 46])**.** *Soit f une fonction rationnelle (resp. séquentielle). Alors f est FO-définissable ssi elle est reconnaissable par un transducteur apériodique (resp. transducteur séquentiel apériodique).*

Propriété 2 : minimisation

Pour la propriété 2 (la minimisation), là encore nous pouvons faire appel à des travaux existants [16, 17] qui ont montré l'existence d'une procédure de minimisation des transducteurs séquentiels, aboutissant à un transducteur séquentiel unique et minimal. Comme pour les automates, l'idée est de fusionner des états "équivalents", *i.e.* des états à partir desquels on accepte le même ensemble de mots. Une différence par rapport aux automates est qu'avant de pouvoir fusionner des états, on doit potentiellement déplacer certains (préfixes de) mots d'une transition à l'autre dans le transducteur.

Exemple. Afin de mieux comprendre le phénomène précédent, prenons un exemple simple, donné en Figure 1.7. Le transducteur séquentiel représenté en Figure 1.7(a) définit la fonction séquentielle f qui à a^{2n} ($n > 0$) associe a^{n+1} et à ba^{2n} ($n \geqslant 0$) associe ba^n. Les états 1 et 3 ne sont clairement pas équivalents, à cause du domaine (l'un est accepteur, l'autre non). Observons maintenant les états 1 et 4, on ne peut pas dire qu'ils soient équivalents, si on considère la fonction définie à partir d'eux. Pour formaliser cette notion, notons, pour tout transducteur T reconnaissant une fonction f et un état q de ce transducteur, f_q la fonction "droite" de l'état q, qui à u associe v dès lors qu'il existe une exécution de T sur u à partir de l'état q, jusqu'à un état final, et produisant v. Dans l'exemple, nous avons $f_1 : a^{2n+1} \mapsto a^{n+1}$ pour tout $n \geqslant 0$ et $f_4 : a^{2n+1} \mapsto a^n$ pour tout $n \geqslant 0$. Même si f_1 et f_4 ont le même domaine de définition, on a $f_1 \neq f_4$. On peut aussi se rendre compte que $f_3 \neq f_2$.

Par contre, nous pouvons observer qu'à partir de l'état 1, nous allons toujours commencer par produire *au moins* un a. On peut alors décaler la production de ce a aux arêtes entrantes de l'état 1. Cela donne un transducteur séquentiel équivalent, qui définit donc toujours la même fonction f. Il est donné à la Figure 1.7(b). Dans ce transducteur, il est facile de voir que $f_1 = f_4$ et $f_2 = f_3$. Par conséquent, nous pouvons fusionner les états 1 et 4, ainsi que les états 2 et 3, ce qui donne le transducteur séquentiel de la Figure 1.7(c), qui est par ailleurs minimal en nombre d'états, et qui produit "le plus tôt possible" ce qu'il peut produire.

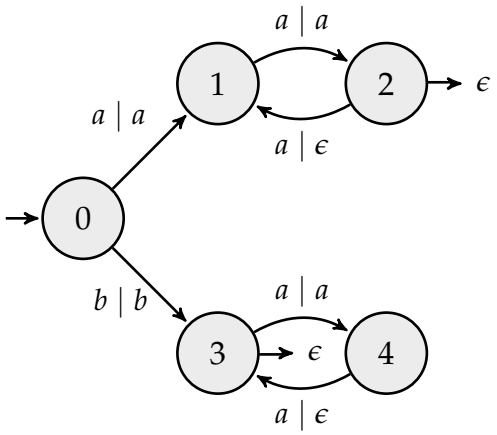
Congruence syntaxique. Comme pour le théorème de Myhill-Nerode et les langages, la minimisation dans le cas des fonctions repose sur une congruence sur Σ^* définie pour toute fonction f. Premièrement, pour formaliser la notion de "plus tôt possible" décrite dans l'exemple, on définit la fonction

$$\widehat{f}(u) = \bigwedge \{f(uw) \mid w \in u^{-1}\mathrm{dom}(f)\} \quad \text{avec } \textstyle\bigwedge \varnothing \text{ égal à } \epsilon.$$

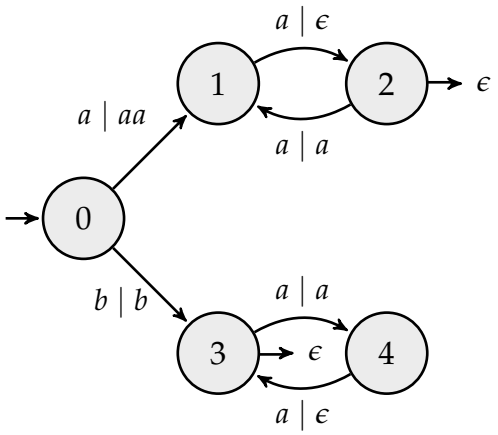
qui retourne le plus long préfixe commun des sorties définies sur toutes les continuations possibles de u dans le domaine de f. Étant donné un mot d'entrée u, on peut alors définir *la fonction résiduelle de f par u* (qui est partielle en général), notée $u^{-1}f$, comme

$$u^{-1}f : w \in u^{-1}\mathrm{dom}(f) \mapsto \widehat{f}(u)^{-1}f(uw)$$

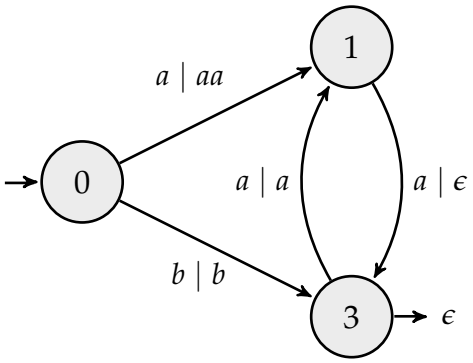
où $\widehat{f}(u)^{-1}f(uw)$ est le mot $f(uw)$ auquel on a retiré $\widehat{f}(u)^{-1}$ (qui en est un préfixe). Partant de là, pour toute fonction f, sa *congruence syntaxique* $\equiv_f$



(a) Un transducteur séquentiel avec $f_1 \neq f_4$ et $f_3 \neq f_2$.



(b) Transducteur séquentiel équivalent à celui de la Figure 1.7(a) avec $f_1 = f_4$ et $f_2 = f_3$.



(c) Transducteur minimal équivalent à celui de la Figure 1.7(a).

FIGURE 1.7 – *Illustration de la minimisation de transducteurs séquentiels.*

est définie par

$$u \equiv_f v \text{ si et seulement si } u^{-1}f = v^{-1}f \ .$$

Nous laissons en exercice le fait de vérifier que $\equiv_f$ est bien une congruence (droite). On a alors l'équivalent du théorème de Myhill-Nerode pour les fonctions séquentielles :

Théorème 1.4.5 ([16, 17])**.** *Une fonction f est séquentielle si et seulement si $\equiv_f$ est d'index fini.*

Minimisation. Étant donnée une fonction séquentielle f, on définit le transducteur T_f sur l'ensemble d'états $\Sigma^*/_{\equiv_f}$ (qui est fini d'après le théorème précédent), d'état initial $[\epsilon]$, d'états finaux $[\mathrm{dom}(f)]$, avec la fonction de transition

$$[u] \xrightarrow{\sigma|\widehat{f}(u)^{-1}\widehat{f}(u\sigma)} [u\sigma]$$

et la fonction terminale $[u] \mapsto \widehat{f}(u)^{-1}f(u)$. Ce transducteur est bien défini car $\equiv_f$ est une congruence droite. Ce transducteur séquentiel a la propriété d'être minimal en nombre d'états :

Théorème 1.4.6 ([16, 17])**.** *Pour toute fonction séquentielle f, le transducteur T_f est minimal et son automate sous-jacent est unique.*

L'unicité n'est obtenue que pour l'automate sous-jacent : il est parfois possible de déplacer des mots de sortie apparaissant sur certaines transitions vers d'autres transitions tout en conservant la structure de l'automate sous-jacent et la fonction définie (voir [17] pour plus de détails).

Nous ne donnons pas dans ce cours un algorithme effectif de minimisation et référons le lecteur à [46] pour des détails. Nous mentionnons que la minimisation se fait en temps polynomial, comme dans le cas des automates déterministes, via raffinements successifs de relations de congruences. Ce qui nous intéresse particulièrement, c'est le fait que le transducteur séquentiel minimal est apériodique si et seulement si il existe un transducteur séquentiel apériodique définissant la fonction (propriété 2). Pour prouver ce résultat, notons $\sim_T$ la congruence de transition de l'automate sous-jacent à T, pour tout transducteur séquentiel T. Nous avons alors le lemme suivant :

Lemme 1.4.7. *Soit f une fonction séquentielle et T un transducteur séquentiel définissant f. Si $u \sim_T v$, alors $u \sim_{T_f} v$.*

Démonstration. Supposons que $u \sim_T v$ et montrons que $u \sim_{T_f} v$. Par définition de $\sim_{T_f}$, nous devons démontrer que pour tout mot α, $[\alpha] \xrightarrow{u}_{T_f} [\alpha u]$ si et

seulement si $[\alpha] \xrightarrow{v}_{T_f} [\alpha u]$. Autrement dit, il faut démontrer que $\alpha u \equiv_f \alpha v$, *i.e.* $(\alpha u)^{-1} f = (\alpha v)^{-1} f$.

Soit $w \in \Sigma^*$. Premièrement, démontrons que les domaines de ces deux fonctions sont égaux. On a en effet que $w \in \mathrm{dom}((\alpha u)^{-1} f)$ ssi $w \in (\alpha u)^{-1}\mathrm{dom}(f)$ (par définition de $(\alpha u)^{-1} f$), ssi $q_0.(\alpha u w) \in F$ (où q_0 est l'état initial de T et F son ensemble d'états finaux), ssi $(q_0.\alpha u).w \in F$ (puisque l'automate sous-jacent de T est déterministe), ssi $(q_0.\alpha v).w \in F$ (car $u \sim_T v$ implique en particulier $q_0.\alpha u = q_0.\alpha v$), ssi $q_0.(\alpha v w) \in F$, ssi $w \in \mathrm{dom}((\alpha v)^{-1}\mathrm{dom}(f))$.

Deuxièmement, supposons que $w \in \mathrm{dom}((\alpha u)^{-1} f) = \mathrm{dom}((\alpha v)^{-1} f)$. Notons $s = ((\alpha u)^{-1} f)(w)$ et $s' = ((\alpha v)^{-1} f)(w)$, et démontrons que $s = s'$. Regardons ce qui se passe dans le transducteur T lorsqu'on lit $\alpha u w$ et $\alpha v w$. Puisque $u \sim_T v$, on a en particulier que $q_0.\alpha u = q_0.\alpha v =_{def} q$. Il existe donc w_1, w_2 tels que $q_0 \xrightarrow{\alpha u|w_1}_T q$ et $q_0 \xrightarrow{\alpha v|w_2}_T q$. Notons également

$$\gamma = \bigwedge \{\rho t(q_f) \in \Sigma^* \mid q_f \in F \wedge \exists \rho' \in \Sigma^*\ q \xrightarrow{\rho'|\rho}_T q_f\}$$

où t est la fonction terminale de T. Le mot γ est le plus long préfixe commun de toutes les sorties pouvant être produites à partir de l'état q. Enfin, notons $f_q(w)$ la sortie produite par T à partir de l'état q en lisant w. On a alors :

$$\begin{array}{lcllcl} \widehat{f}(\alpha u) & = & w_1\gamma & \widehat{f}(\alpha v) & = & w_2\gamma \\ f(\alpha u w) & = & w_1 f_q(w) & f(\alpha v w) & = & w_2 f_q(w)\,. \end{array}$$

De ces équations on peut déduire

$$\begin{array}{rcl} s & = & \widehat{f}(\alpha u)^{-1} f(\alpha u w) \\ & = & (w_1\gamma)^{-1} w_1 f_q(w) \\ & = & \gamma^{-1}(w_1^{-1} w_1) f_q(w) \\ & = & \gamma^{-1}(w_2^{-1} w_2) f_q(w) \\ & = & (w_2\gamma)^{-1} w_2 f_q(w) \\ & = & \widehat{f}(\alpha v)^{-1} f(\alpha v w) \\ & = & s'\,. \end{array}$$

Ceci conclut la preuve. ■

Nous pouvons alors directement en déduire que si T est apériodique, *i.e.* si $\sim_T$ est apériodique, alors T_f l'est également.

Propriété 3 : test d'appartenance à la classe des apériodiques

Il est possible de tester, étant donné un transducteur séquentiel T, s'il est apériodique. En effet, par définition de l'apériodicité d'un transducteur, il est apériodique ssi son automate sous-jacent est apériodique, ce qui est testable en espace polynomial comme on l'a mentionné pour les langages.

La recette : décider la FO-définissabilité d'une fonction séquentielle

Nous pouvons maintenant mettre toutes les propriétés ensemble pour obtenir un algorithme de décision pour le problème du Théorème 1.4.3 :

Entrée Un MSO-transducteur gauche-droite φ définissant une fonction séquentielle f

1. convertir φ en transducteur séquentiel T
2. minimiser T en T_f
3. tester l'apériodicité de T_f

Prouvons la correction de cet algorithme. Si φ est équivalente à un FO-transducteur, alors d'après le Théorème 1.4.4, f est définissable par un transducteur apériodique et séquentiel T. D'après le lemme 1.4.7, T_f est apériodique, et l'algorithme retourne oui. Réciproquement, si l'algorithme retourne 'oui', alors T_f est apériodique, donc f est FO-définissable d'après le Théorème 1.4.4, puisque les différentes transformations opérées par l'algorithme préservent f.

Exemple. La fonction f définie par le transducteur de la Figure 1.7(a) n'est pas FO-définissable car le transducteur minimal de la Figure 1.7(c) n'est pas apériodique. En effet, sur la famille de mots $(a^n)_{n\geqslant 0}$, on a l'existence d'une exécution de l'état 1 à l'état 1 uniquement pour les mots de longueur paire.

1.4.3 Fonctions rationnelles et régulières

Le cas des fonctions rationnelles est plus ardu, car pour définir une fonction rationnelle, le non-déterminisme de l'automate sous-jacent est nécessaire en général, comme on l'a vu dans la Section 1.2. Or, l'unicité de l'automate minimal est perdue pour les automates non-déterministes. Néanmoins, il a été démontré qu'il existait un transducteur pouvant être obtenu de manière canonique à partir de n'importe quel transducteur fonctionnel dans [48]. Sans donner plus de détails, l'idée principale est d'ajouter aux transducteurs séquentiels la capacité de calculer de l'information (finie)

sur le suffixe à la position courante lorsqu'un mot est lu de gauche à droite. Nous appellerons cette information *information en avant*. Il a été démontré par [48] qu'une information unique et minimale suffisait à réaliser toute fonction rationnelle par un transducteur séquentiel étendu avec cette information en avant. Autrement dit, pour toute fonction rationnelle f, en enrichissant le mot d'entrée avec une information canonique et minimale, il est toujours possible de calculer f de gauche à droite de manière séquentielle. Combiné aux techniques de minimisation des fonctions séquentielles, ce résultat a permis d'obtenir un modèle canonique et minimal pour les fonctions rationnelles, néanmoins sans avoir l'unicité. Il a finalement été démontré dans [36, 35, 46] que cet objet canonique préservait certaines propriétés structurelles du transducteur de départ, notamment le fait d'être apériodique, permettant ainsi de décider pour les fonctions rationnelles le problème de FO-définissabilité (entre autres).

Pour les fonctions régulières, le problème de FO-définissabilité est ouvert. Des résultats partiels ont été obtenus pour une sémantique plus forte des transducteurs, la sémantique *origine*. L'idée est d'ajouter en plus à la sémantique une fonction o de toute position du mot de sortie vers une position du mot d'entrée (celle qui l'a générée). Deux transducteurs équivalents ne le sont pas nécessairement avec la sémantique origine car les fonctions d'origine sont en général différentes, pour une même paire de mot entrée/sortie. Il a été démontré dans [11] qu'étant donné un MSO-transducteur définissant une fonction, il est décidable de savoir s'il existe un FO-transducteur définissant la même fonction et les mêmes fonctions d'origine pour chaque paire entrée/sortie.

1.5 Perspectives

Nous avons présenté un ensemble de résultats importants, reliant des présentations de transductions fondées sur les automates, la logique et l'algèbre. Il existe cependant de nombreux autres résultats liés à ces sujets, ainsi qu'un certain nombre de perspectives de recherches, que nous décrivons ci-dessous.

Extensions. Tout d'abord, les résultats énoncés dans ce chapitre portent sur les transductions des mots finis vers les mots finis. Un certain nombre de travaux a consisté à chercher à les généraliser à des structures plus complexes comme les mots infinis, les arbres et les mots bien parenthésés. Le cas des mots infinis a été étudié dans [2]. Par ailleurs, une série de travaux a considéré des transformations des mots parenthésés (qui permettent de re-

présenter des documents XML) vers les mots, en utilisant les transducteurs à pile visible [41, 23]. On peut également citer les transductions d'arbres, pour lesquelles plusieurs modèles de transducteurs existent [32, 33].

Sémantiques alternatives. La difficulté du problème d'équivalence de transducteurs provient du fait que lorsque l'on compare deux transducteurs sur un même mot d'entrée, les mots de sortie peuvent être produits de manière asynchrone. Une notion d'origine a été introduite dans [11] pour pallier cette difficulté. On définit l'origine d'une position p_o du mot de sortie comme la position p_i du mot d'entrée qui est à l'origine de p_o, c'est-à-dire la position d'entrée telle que lorsque le symbole de cette position a été lu, le symbole de p_o a été produit par le transducteur. La sémantique avec origine consiste à ajouter cette information aux paires (u, w) de la transduction. Si deux transducteurs sont équivalents pour la sémantique avec origine, alors ils le sont pour la sémantique classique, mais le contraire n'est pas vrai. Cette nouvelle sémantique permet de simplifier l'analyse de nombreux problèmes liés aux transducteurs. Dans [11], il est montré que le problème de FO-définissabilité est décidable pour les fonctions régulières, pour la sémantique origine. Différents problèmes de transducteurs d'arbres ont également été étudiés avec la sémantique origine dans [39]. Plus récemment, il a été montré que l'équivalence du point de vue de la sémantique avec origine de transducteurs bidirectionnels *non-déterministes* est PSpace-complet [12]. Pour finir, mentionnons qu'une notion de resynchronisation, qui est une généralisation de la sémantique avec origine, a été introduite dans [38].

Transducteurs à registres. Un autre modèle a été introduit par Alur et Černý pour représenter des transducteurs : il s'agit d'automates déterministes unidirectionnels équipés d'un nombre fini de variables appelées registres, que l'on peut utiliser pour enregistrer des parties du mot produit en sortie (ces variables ne peuvent pas être testées). Les différentes classes de transductions présentées dans ce chapitre trouvent une présentation dans ce modèle, en restreignant le modèle à des mises à jour sans copie des registres. Si l'on autorise les copies de registres, on obtient une classe de fonctions strictement plus expressive que les fonctions régulières, pour laquelle le problème d'équivalence est encore décidable, et étroitement lié au problème d'équivalence de langages HDT0L [42, 9]. Un problème ouvert important consiste à identifier la complexité de ce problème.

Une série de travaux s'est intéressée à la minimisation du nombre de registres dans ces transducteurs à registres. Étant donné un transducteur à

registres et un entier k, il s'agit de déterminer s'il existe un transducteur à registres équivalent ayant au plus k registres. Ce problème a été résolu pour différentes sous-classes [26, 25], et pour une sous-classe plus large mais avec des modèles non-déterministes [5]. Le cas général constitue un problème ouvert important.

Langages de spécification. Les expressions régulières constituent une présentation alternative des langages réguliers qui est très utile en pratique. Pour la classe des fonctions régulières, une notion d'expressions régulières de transformations a été proposée dans [3], la preuve d'équivalence reposant sur le modèle des transducteurs à registres. Un langage déclaratif de spécification de transformations fondé sur ces expressions a ensuite été introduit dans [1]. Récemment, des transformations directes des transducteurs bidirectionnels vers les expressions régulières de transformations ont été proposées dans [7], ainsi que dans [24], qui donne aussi une généralisation aux mots infinis. Enfin, une logique spécialement conçue pour définir des transductions avec origine a été définie dans [22]. Il y est démontré que toutes les fonctions régulières sont exprimables dans cette logique, et que toute transduction non fonctionnelle exprimable dans cette logique peut être uniformisée en une fonction régulière.

Représentations algébriques. Comme on l'a vu, la présentation algébrique des transductions est un outil important pour caractériser par exemple des sous-classes logiques comme FO. Cependant, les outils actuels, fondés sur un transducteur réalisant la transduction, se limitent au cadre des fonctions rationnelles, et il semble très difficile de les étendre aux fonctions régulières. Une piste de recherche alternative consiste à utiliser des notions ne dépendant que de la transduction, et non de la machine la réalisant. Un premier travail dans cette direction a été publié récemment [14].

Remerciements. Nous remercions chaleureusement les relecteurs de ce chapitre, pour leurs corrections et nombreuses suggestions, qui ont permis d'en améliorer la lecture.

Bibliographie

[1] R. ALUR, L. D'ANTONI et M. RAGHOTHAMAN : Drex : A declarative language for efficiently evaluating regular string transformations. *In POPL 2015*, p. 125–137. ACM, 2015.

[2] R. ALUR, E. FILIOT et A. TRIVEDI : Regular transformations of infinite strings. *In LICS 2012*, p. 65–74. IEEE Computer Society, 2012.

[3] R. ALUR, A. FREILICH et M. RAGHOTHAMAN : Regular combinators for string transformations. *In CSL-LICS 2014*, p. 9 :1–9 :10. ACM, 2014.

[4] F. BASCHENIS, O. GAUWIN, A. MUSCHOLL et G. PUPPIS : One-way definability of sweeping transducers. *In FSTTCS 2015*, vol. 45 de *LIPIcs*, p. 178–191. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.

[5] F. BASCHENIS, O. GAUWIN, A. MUSCHOLL et G. PUPPIS. : Minimizing resources of sweeping and streaming string transducers. *In ICALP 2016*, vol. 55 de *LIPIcs*, p. 114 :1–114 :14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[6] F. BASCHENIS, O. GAUWIN, A. MUSCHOLL et G. PUPPIS : Untwisting two-way transducers in elementary time. *In LICS 2017*, p. 1–12. IEEE Computer Society, 2017.

[7] N. BAUDRU et P.-A. REYNIER : From two-way transducers to regular function expressions. *In DLT 2018*, vol. 11088 de *Lecture Notes in Computer Science*, p. 96–108. Springer, 2018.

[8] M.-P. BÉAL, O. CARTON, C. PRIEUR et J. SAKAROVITCH : Squaring transducers : an efficient procedure for deciding functionality and sequentiality. *Theoret. Comput. Sci*, 292(1):45–63, 2003.

[9] M. BENEDIKT, T. DUFF, A. SHARAD et J. WORRELL : Polynomial automata : Zeroness and applications. *In LICS 2017*, p. 1–12. IEEE Computer Society, 2017.

[10] J. BERSTEL : *Transductions and context-free languages*, vol. 38 de *Teubner Studienbücher : Informatik*. Teubner, 1979.

[11] M. BOJANCZYK : Transducers with origin information. *In ICALP 2014 (2)*, vol. 8573 de *Lecture Notes in Computer Science*, p. 26–37. Springer, 2014.

[12] S. BOSE, A. MUSCHOLL, V. PENELLE et G. PUPPIS : Origin-equivalence of two-way word transducers is in PSPACE. *In FSTTCS 2018*, vol. 122 de *LIPIcs*, p. 22 :1–22 :18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.

[13] J. R. BÜCHI : Weak second-order arithmetic and finite automata. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 6(1–6):66–92, 1960.

[14] M. CADILHAC, O. CARTON et C. PAPERMAN : Continuity and rational functions. *In ICALP 2017*, vol. 80 de *LIPIcs*, p. 115 :1–115 :14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2017.

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

[16] C. CHOFFRUT : A generalization of Ginsburg and Rose's characterization of G-S-M mappings. *In ICALP 1979*, vol. 71 de *Lecture Notes in Computer Science*, p. 88–103. Springer, 1979.

[17] C. CHOFFRUT : Minimizing subsequential transducers : a survey. *Theoret. Comput. Sci.*, 292(1):131–143, 2003.

[18] B. COURCELLE : Monadic second-order definable graph transductions : A survey. *Theoret. Comput. Sci.*, 126:53–75, 1994.

[19] B. COURCELLE et J. ENGELFRIET : *Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach*, vol. 138 de *Encyclopedia of mathematics and its applications*. Cambridge University Press, 2012.

[20] K. CULIK et J. KARHUMAKI : The equivalence problem for single-valued two-way transducers (on NPDT0L languages) is decidable. *SIAM J. Comput.*, 16(2):221–230, 1987.

[21] K. CULIK II et J. KARHUMÄKI : The equivalence of finite valued transducers (on HDT0L languages) is decidable. *Theoret. Comput. Sci.*, 47(3):71–84, 1986.

[22] L. DARTOIS, E. FILIOT et N. LHOTE : Logics for word transductions with synthesis. *In LICS 2018*, p. 295–304. ACM, 2018.

[23] L. DARTOIS, E. FILIOT, P.-A. REYNIER et J.-M. TALBOT : Two-way visibly pushdown automata and transducers. *In LICS 2016*, p. 217–226. ACM, 2016.

[24] V. DAVE, P. GASTIN et S. N. KRISHNA : Regular transducer expressions for regular transformations. *In LICS 2018*, p. 315–324. ACM, 2018.

[25] L. DAVIAUD, I. JECKER, P.-A. REYNIER et D. VILLEVALOIS : Degree of sequentiality of weighted automata. *In FoSSaCS 2017*, vol. 10203 de *Lecture Notes in Computer Science*, p. 215–230. Springer, 2017.

[26] L. DAVIAUD, P.-A. REYNIER et J.-M. TALBOT : A generalized twinning property for minimisation of cost register automata. *In LICS 2016)*, p. 857–866. ACM, 2016.

[27] R. de SOUZA : On the decidability of the equivalence for k-valued transducers. *In DLT 2008*, vol. 5257 de *Lecture Notes in Computer Science*, p. 252–263. Springer, 2008.

[28] V. DIEKERT et P. GASTIN : First-order definable languages. *In* J. FLUM, E. GRÄDEL et T. WILKE, éds : *Logic and Automata : History and Perspectives*, vol. 2 de *Texts in Logic and Games*, p. 261–306. Amsterdam University Press, 2008.

[29] V. DIEKERT, P. GASTIN et M. KUFLEITNER : A survey on small fragments of first-order logic over finite words. *Internat. J. Found. Comput. Sci.*, 19(3):513–548, 2008.

[30] C. C. ELGOT : Decision problems of finite automata design and related arithmetics. *Trans. Amer. Math. Soc.*, 98(1):21–51, 1961.

[31] J. ENGELFRIET et H. J. HOOGEBOOM : MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.

[32] J. ENGELFRIET et S. MANETH : Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inform. and Comput.*, 154(1):34–91, 1999.

[33] J. ENGELFRIET et S. MANETH : Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003.

[34] E. FILIOT : Logic-automata connections for transformations. *In ICLA 2015*, vol. 8923 de *Lecture Notes in Computer Science*, p. 30–57. Springer, 2015.

[35] E. FILIOT, O. GAUWIN et N. LHOTE : Aperiodicity of rational functions is PSPACE-complete. *In FSTTCS 2016*, vol. 65 de *LIPIcs*, p. 13 :1–13 :15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[36] E. FILIOT, O. GAUWIN et N. LHOTE : First-order definability of rational transductions : an algebraic approach. *In LICS 2016*, p. 387–396. ACM, 2016.

[37] E. FILIOT, O. GAUWIN, P.-A. REYNIER et F. SERVAIS : From two-way to one-way finite state transducers. *In LICS 2013*, p. 468–477. IEEE Computer Society, 2013.

[38] E. FILIOT, I. JECKER, C. LÖDING et S. WINTER : On equivalence and uniformisation problems for finite transducers. *In ICALP 2016*, vol. 55 de *LIPIcs*, p. 125 :1–125 :14, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[39] E. FILIOT, S. MANETH, P.-A. REYNIER et J.-M. TALBOT : Decision problems of tree transducers with origin. *Inform. and Comput.*, 261:311–335, 2018.

[40] E. FILIOT, N. MAZZOCCHI et J.-F. RASKIN : A pattern logic for automata with outputs. *In DLT 2018*, vol. 11088 de *Lecture Notes in Computer Science*, p. 304–317. Springer, 2018.

[41] E. FILIOT, J.-F. RASKIN, P.-A. REYNIER, F. SERVAIS et J.-M. TALBOT : Visibly pushdown transducers. *J. Comput. System Sci.*, 2018.

[42] E. FILIOT et P.-A. REYNIER : Copyful streaming string transducers. *In RP 2017*, vol. 10506 de *Lecture Notes in Computer Science*, p. 75–86. Springer, 2017.

[43] E. M. GURARI : The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM J. Comput.*, 11(3):448–452, 1982.

[44] E. M. GURARI et O. H. IBARRA : A note on finite-valued and finitely ambiguous transducers. *Math. Syst. Theory*, 16(1):61–66, 1983.

[45] 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.

[46] N. LHOTE : *Definability and Synthesis of Transductions*. Thèse de doctorat, Université de Bordeaux, Université Libre de Bruxelles, 2018.

[47] R. MCNAUGHTON et S. PAPERT : *Counter-free automata*. M.I.T. Press, 1971.

[48] C. REUTENAUER et M.-P. SCHÜTZENBERGER : Minimization of rational word functions. *SIAM J. Comput.*, 20(4):669–685, 1991.

[49] J. SAKAROVITCH : *Elements of Automata Theory*. Cambridge University Press, 2009.

[50] M.-P. SCHÜTZENBERGER : On finite monoids having only trivial subgroups. *Inform. and Control*, 8(2):190–194, 1965.

[51] M. P. SCHÜTZENBERGER : Sur les relations rationnelles. *In 2nd GI Conference on Automata Theory and Formal Languages*, vol. 33 de *Lecture Notes in Computer Science*, p. 209–213. Springer, 1975.

[52] H. STRAUBING : *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel and Berlin, 1994.

[53] B. A. TRAKHTENBROT : Finite automata and logic of monadic predicates (in Russian). *Dokl. Akad. Nauk*, 140:326–329, 1961.

[54] M. Y. VARDI et P. WOLPER : An automata-theoretic approach to automatic program verification. *In LICS 1986*, p. 332–344. IEEE Computer Society, 1986.

[55] A. WEBER : Decomposing finite-valued transducers and deciding their equivalence. *SIAM J. Comput.*, 22(1):175–202, 1993.

[56] A. WEBER et R. KLEMM : Economy of description for single-valued transducers. *Inform. and Comput.*, 118(2):327–340, 1995.