Propriétés quantitatives des mots et des arbres. Applications aux langages XML.

Benjamin Monmege
sous la direction de Benedikt Bollig, Paul Gastin et Marc Zeitoun

Laboratoire Spécification et Vérification - ENS de Cachan
61, avenue du Président Wilson 94235 CACHAN Cedex, France

Stage de M2 de février à juillet 2010

Table des matières

Introduction

Le contexte général. La vérification automatique est aujourd'hui un domaine de recherche central en informatique, particulièrement dans le cadre booléen. Les besoins actuels évoluent vers une analyse plus fine, *quantitative*.

L'extension de la vérification à des domaines quantitatifs a commencé depuis une quinzaine d'années dans un cadre spécifique, celui des systèmes probabilistes. Il existe cependant beaucoup d'autres propriétés quantitatives intéressantes hors du champ des systèmes probabilistes. Concrètement, on peut vouloir par exemple évaluer le coût nécessaire à l'exécution d'une tâche, la durée de vie d'un appareil, les besoins énergétiques d'une application, la fiabilité d'un programme, ou le nombre de réponses sélectionnées par une requête dans une base de données.

Pour modéliser des systèmes prenant en compte l'aspect quantitatif de façon générique, il est naturel d'utiliser une notion d'automates à poids développée depuis les années 60 [Sch61, Sak03, KVD09], version quantitative des classiques automates finis. Le résultat du calcul d'un automate à poids sur un mot n'est plus une valeur booléenne distinguant les comportements acceptés de ceux rejetés : à la place, un calcul sur un mot produit un poids, c'est-à-dire une valeur dans une structure algébrique adaptée, appelée semi-anneau.

Le problème étudié. Bien que l'étude théorique des automates à poids soit avancée, les applications à la spécification et la vérification débutent à peine. L'axe central est ainsi de développer des connexions entre langages de spécification et modèles de systèmes dans le cadre quantitatif, pour aboutir à des algorithmes de vérification. La lignée des résultats classiques, dans le cadre booléen, a commencé avec l'équivalence entre logique monadique du second ordre (MSO) et automates finis, et a naturellement donné naissance à des généralisations sur les arbres.

De façon surprenante, une caractérisation logique des automates à poids n'a été établie que récemment [DG07], en termes d'une logique MSO avec poids restreinte, qui capture exactement les séries formelles reconnaissables (c'est-à-dire celles qui représentent le comportement des automates à poids).

La contribution proposée. Au lieu de restreindre la logique, on se propose de définir des modèles d'automates étendus qui la reflètent naturellement. Plus précisément, on capture la quantification universelle du premier ordre par un mécanisme de *jetons* dans un automate à double sens. En s'inspirant de la théorie des automates boustrophédons et avec jetons sur les mots et les arbres [EH99, BSSS06, Boj08], on a donc défini des généralisations avec poids qui préservent leurs connexions naturelles avec la logique.

Par suite, on a introduit et étudié d'autres formalismes logiques afin de capturer la nouvelle classe d'automates ainsi définie : l'une d'elle est donnée par la logique du *premier ordre* augmentée de la *clôture transitive bornée*. En raison du lien de cette logique avec les automates à jetons dans le cadre des langages d'arbres [EH07], il est particulièrement naturel et intéressant d'établir une telle connexion dans le cadre quantitatif.

Au cours de la preuve d'un résultat d'expressivité reliant des automates à jetons et la logique du premier ordre munie d'une clôture transitive bornée – à la fois sur les mots et sur les arbres – on a été amené à introduire une notion d'automates *imbriqués* à poids, comme cela a été fait dans le cadre booléen pour les arbres [tCS10]. Une motivation supplémentaire est le lien fort qui relie cette classe d'automates avec le langage de requêtes émanant du standard XPath. On a finalement proposé une extension quantitative du langage XPath, qu'on a reliée à la logique du premier ordre.

Les arguments en faveur de sa validité. La piste d'extension de la logique possède un double avantage : d'une part, elle ne restreint pas la logique à une classe qui, dans le contexte quantitatif, peut s'avérer limitée pour les applications ; d'autre part, elle s'appuie sur des outils

développés dans le cadre des langages de requêtes dans des documents XML, qui constitue l'une des principales applications visées. Ainsi, étant donnée un format de requêtes, on peut répondre plus facilement à un certain nombre de problèmes de décision, tels que la satisfiabilité d'une requête, le *model checking* ou l'inclusion d'une requête dans une autre, grâce au modèle d'automates proposés (actuellement avec des hypothèses fortes sur le semi-anneau sous-jacent).

Les perspectives. De nombreuses questions restent ouvertes après ce stage. L'une d'elles concerne le pan des problèmes de décision : les différents liens entre les langages de requêtes (XPath ou MSO) et les modèles d'automates apportent de nouvelles perspectives dans la résolution des problèmes de décisions, qu'il faut désormais considérer.

Par ailleurs, et d'un point de vue plus théorique, notre travail a mis au jour de nombreuses classes d'automates et de logiques, que nous n'avons pas encore pu relier, faute de robustesse ou d'expressivité des unes et des autres. L'une d'elles est de considérer le modèle d'automates à jetons forts, qu'il faudrait relier à une clôture transitive non bornée.

Ce rapport se découpe en quatre parties. La première dresse le paysage quantitatif dans lequel notre étude voyage : on rappelle ainsi brièvement la définition des notions de semi-anneau, de polynômes et de séries formelles. La deuxième partie motive notre travail en explicitant une extension quantitative du langage de requêtes XPath que nous avons développée : on étudie de nombreux exemples originaux permettant de se familiariser avec le pouvoir d'expression d'un tel langage. Les deux parties suivantes présentent les nouveaux modèles d'automates pondérés, ainsi que les résultats d'expressivité que nous avons démontrés, reliant ces machines avec des fragments de logique. La troisième partie se focalise ainsi sur les mots, tandis que la quatrième aborde les arbres, et mentionnera les difficultés rencontrées, inhérentes aux structures branchantes. Faute de place, les preuves détaillées ont été placées en annexe de ce rapport.

Remerciements et déroulement du stage

Je tiens à remercier très chaleureusement Benedikt Bollig, Paul Gastin et Marc Zeitoun pour leur soutien tout au long de ce stage de recherche, leur patience et leur rigueur. Leur exigence scientifique et le temps qu'ils m'ont accordé m'ont permis de m'épanouir durant ce stage. Merci également à tous les membres du LSV pour leur accueil durant ces cinq mois.

Même si mon stage a officiellement débuté mi-février 2010, nous avons commencé à travailler ensemble dès le mois de novembre 2009, à raison d'une demi-journée par semaine. Cette organisation nous a donné l'opportunité dès le mois de février de soumettre une première contribution pour la conférence ICALP'10, soumission qui a été acceptée par le comité de sélection. Ce rapport est donc composé non seulement de résultats présentés et publiés à ICALP, mais aussi de résultats nouveaux développés et prouvés durant le reste du stage. Plus précisément, les résultats de la section 3 sur les mots apparaissent dans le papier [BGMZ10], alors que les généralisations quantitatives de XPath, présentée ici en section 2, ainsi que les résultats sur les arbres (section 4), sont des résultats postérieurs.

Ce stage m'a par ailleurs permis de découvrir plus en détail le quotidien du métier de chercheur, à travers de nombreux déplacements que j'ai pu faire. J'ai ainsi pu présenter nos résultats lors de l'atelier WATA à Leipzig et de la conférence internationale ICALP à Bordeaux. J'ai également pu assister à l'école *Quantitative Model Checking* à Copenhague et être présent lors de réunions de projets (Quasimodo, DOTS) et de conférences internationales généralistes (FLoC à Edimbourg).

1 Paysage quantitatif

Afin de comprendre quelle structure algébrique est nécessaire sur l'ensemble des poids, décrivons pour commencer quelques exemples d'utilisation de *systèmes pondérés*. On considère ici des automates finis classiques auquel on ajoute sur chacune des transitions un poids provenant d'un ensemble S. Imaginons dans un premier temps qu'on veuille modéliser la quantité de ressources nécessaire pour leur exécution : on interprète alors le poids sur les transitions comme la ressource nécessaire pour traverser cette transition. On aura dans ce cas la ressource globale nécessaire pour traverser un chemin de cet automate en sommant les poids des différentes transitions empruntées. On peut alors être intéressé par minimiser les ressources nécessaires pour lire un mot donné, c'est-à-dire minimiser les poids des chemins étiquetés par ce mot. Dans un second temps, on peut interpréter des poids réels comme des probabilités d'emprunter une transition, auquel cas on multipliera les différents poids le long d'un chemin pour obtenir la probabilité de suivre ce chemin. Ainsi, la probabilité totale de lire un mot donné et d'atteindre un ensemble d'états distingués sera obtenue en additionnant les différents poids des chemins étiquetés par ce mot et terminant dans cet ensemble d'états.

Ces deux exemples montrent qu'on a naturellement besoin de munir l'ensemble des poids considéré de deux opérations internes, l'une permettant de calculer le poids d'un chemin, et l'autre permettant de rassembler les poids de plusieurs chemins étiquetés par le même mot. C'est ce que permet la structure de semi-anneau qu'on définit ci-dessous.

Commençons par rappeler qu'un monoïde est une structure $\mathbb{S} = (S, \otimes, e)$ avec S un ensemble, $\otimes : S \times S \to S$ une loi de composition interne associative et $e \in S$ un élément neutre pour $\otimes$. Un monoïde est dit commutatif si pour tous $a, b \in S$, $a \otimes b = b \otimes a$.

Définition 1.1.

Un semi-anneau est une structure $\mathbb{S} = (S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ telle que
- $(S, \oplus, \mathbf{0})$ est un monoïde commutatif,
- $(S, \odot, \mathbf{1})$ est un monoïde,
- $\odot$ est distributive sur $\oplus$

  $\forall a, b, c \in S \quad a \odot (b \oplus c) = (a \odot b) \oplus (a \odot c) \quad$ et $\quad (a \oplus b) \odot c = (a \odot c) \oplus (b \odot c)$
- $\mathbf{0}$ est un élément absorbant pour $\odot$: $\forall a \in S \quad a \odot \mathbf{0} = \mathbf{0} \odot a = \mathbf{0}$

Un semi-anneau est dit commutatif si le monoïde $(S, \odot, \mathbf{1})$ est commutatif.

Pour faire le lien avec l'introduction de cette section, on utilisera l'opération $\odot$ du semi-anneau pour calculer le poids d'un chemin en fonction des différentes transitions empruntées, alors qu'on utilisera $\oplus$ pour calculer le poids global de l'ensemble des exécutions relatives à un mot donné. **Dans toute la suite de ce rapport, on supposera (sauf mention contraire) que les semi-anneaux considérés sont commutatifs.**

Exemple 1.1. Il existe de nombreuses structures algébriques qui se révèlent être des semi-anneaux :
- **Nat** $= (\mathbb{N}, +, \cdot, 0, 1)$ l'ensemble des entiers naturels
- **Int** $= (\mathbb{Z}, +, \cdot, 0, 1)$ l'ensemble des entiers relatifs
- **Prob** $= ([0, +\infty[, +, \cdot, 0, 1)$ le semi-anneau des probabilités (qu'on ne limite pas à l'intervalle $[0, 1]$ pour des raisons de bonne définition de l'opération $+$),
- **Bool** $= (\mathbb{B}, \vee, \wedge, \bot, \top)$ l'algèbre des booléens,
- **Trop** $= (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ le semi-anneau tropical,
- **Lang** $= (2^{\Sigma^*}, \cup, \cdot, \varnothing, \{\varepsilon\})$ le semi-anneau des langages sur un alphabet Σ : il n'est pas commutatif,

– **Mat** $= (S^{P\times P}, \oplus, \odot, \Omega, I)$ le semi-anneau des matrices[1] carrées indexées par l'ensemble fini P, à coefficients dans un semi-anneau $\mathbb{S} = (S, \oplus, \odot, \mathbf{0}, \mathbf{1})$ dans lequel on étend les opérations d'addition et de multiplication aux matrices de manière classique, Ω étant la matrice nulle et I la matrice identité. Là-encore, ce semi-anneau n'est pas commutatif.

Décrivons plus en détail deux semi-anneaux particuliers que nous allons utiliser dans les sections suivantes. Pour cela, fixons un semi-anneau $\mathbb{S}$ ainsi qu'un alphabet fini Σ. Considérons l'ensemble des fonctions totales du monoïde Σ^* (muni de la concaténation) dans le semi-anneau $\mathbb{S}$, communément appelées séries formelles sur $\mathbb{S}$. On peut naturellement additionner deux telles fonctions f et g, en additionnant deux par deux chacune des images de chaque mot w : $(f \oplus g)(w) = f(w) \oplus g(w)$. De plus, on peut également définir une multiplication $f \odot g$ (communément appelée produit de Hadamard) : $(f \odot g)(w) = f(w) \odot g(w)$.

Avec les notations précédentes, on note $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ l'ensemble des fonctions de Σ^* dans $\mathbb{S}$: muni de l'addition et de la multiplication décrites ci-dessus, de la série identiquement nulle et de la série identiquement égale à $\mathbf{1}$ (qui est donc la série caractéristique du langage Σ^*), il s'agit d'un semi-anneau. Traditionnellement, on l'appelle semi-anneau des séries formelles à coefficients dans le semi-anneau $\mathbb{S}$.

Si on se restreint à l'ensemble des fonctions dont le support (l'ensemble des mots qui n'ont pas une image nulle) est fini, on définit le semi-anneau des polynômes à coefficients dans $\mathbb{S}$, qu'on note $\mathbb{S}\langle\Sigma^*\rangle$.

Exemple 1.2. La série définie par $f(w) = |w|$ si $|w| \leq 4$, et $f(w) = 0$ sinon, est un polynôme de $\mathbf{Nat}\langle\Sigma^*\rangle$. La série définie par $f(w) = |w|_a - |w|_b$ n'est pas un polynôme sur le semi-anneau **Int** (où $|w|_a$ est le nombre d'occurrences de la lettre a dans le mot w, pour toute lettre a).

2 À la recherche d'un langage de requêtes quantitatives

On s'intéresse à définir un langage de requêtes quantitatives sur des documents XML. On commence donc par définir la classe d'arbres sur laquelle on travaille, puis on étend les langages booléens, en leur donnant une sémantique quantitative. Dans un premier temps, on utilise comme structure de données des arbres finis planaires d'arité non bornée. On spécialisera plus tard certaines extensions aux cas des arbres binaires.

Définition 2.1.

> Soit Σ un alphabet fini. Un Σ-arbre (fini planaire d'arité non bornée) est une fonction partielle $t : \mathbb{N}^* \longrightarrow \Sigma$ dont le domaine $\mathrm{dom}(t)$ est non vide, fini, clos par préfixe et tel que si $n(i+1) \in \mathrm{dom}(t)$ alors $ni \in \mathrm{dom}(t)$. Les éléments du domaine sont appelés des *nœuds* et l'*étiquette* du nœud u est l'élément $t(u)$. La suite vide ε est appelée *racine* de t, et toutes les suites maximales sont appelées des *feuilles*. Pour $u, v \in \mathrm{dom}(t)$, on écrit $u < v$ si $v = uw$ pour une suite non vide $w \in \mathbb{N}^*$ (i.e. si u est un *ancêtre* de v dans l'arbre), et on note $u \prec v$ si $u = wi$ et $v = wj$ pour une suite $w \in \mathbb{N}^*$ et deux entiers naturels $i < j$ (i.e. si v est un *frère droit* de u). On définit les ordres larges $\leq$ et $\preccurlyeq$ de manière naturelle. Étant donné un nœud u de l'arbre t, on note $\mathrm{subtree}(t, u)$ le sous-arbre de t, dont u est la racine. On note $\mathcal{T}_\Sigma$ l'ensemble des Σ-arbres.

Dans ce cadre, un *arbre binaire* est un Σ-arbre dont chaque nœud interne u (i.e. qui n'est pas une feuille) possède exactement deux successeurs, $u0$ et $u1$.

1. Si A et B sont des ensembles tels que B est fini, on note A^B l'ensemble des vecteurs de taille $|B|$ à valeurs dans A. Si $v \in A^B$, pour tout $b \in B$, on note v_b la valeur associée dans A.

On peut aisément étendre la notion de série formelle au cas des arbres. On note ainsi $\mathbb{S}\langle\langle\mathcal{T}_\Sigma\rangle\rangle$ le *semi-anneau des séries formelles d'arbres* sur $\mathbb{S}$, ainsi que $\mathbb{S}\langle\mathcal{T}_\Sigma\rangle$ le semi-anneau des polynômes d'arbres sur $\mathbb{S}$.

Dans l'optique d'une navigation dans un arbre, on peut distinguer quatre directions privilégiées $\leftarrow, \rightarrow, \uparrow$ et $\downarrow$. Pour un arbre t fixé, on définit ainsi quatre relations binaires $\mathcal{R}^t_{\text{step}}$ pour $\text{step} \in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$:

$$\mathcal{R}^t_{\leftarrow} = \{(u,v) \in \text{dom}(t)^2 \mid \exists w \in \text{dom}(t)\ \exists i \in \mathbb{N} \quad u = w(i+1) \wedge v = wi\} \qquad \mathcal{R}^t_{\rightarrow} = (\mathcal{R}^t_{\leftarrow})^{-1}$$
$$\mathcal{R}^t_{\uparrow} = \{(u,v) \in \text{dom}(t)^2 \mid \exists i \in \mathbb{N} \quad u = vi\} \qquad \mathcal{R}^t_{\downarrow} = (\mathcal{R}^t_{\uparrow})^{-1}$$

Dans toutes les définitions suivantes, on fixe un alphabet Σ fini, et un semi-anneau commutatif $\mathbb{S}$.

2.1 Logique du premier ordre

On fixe un ensemble infini $\text{Var} = \{x, y, x_1, x_2, \ldots\}$ de variables du premier ordre.

Définition 2.2.

L'ensemble des formules du premier ordre sur le semi-anneau $\mathbb{S}$ et l'alphabet Σ, noté $\text{wFO}(\mathbb{S}, \Sigma)$ (ou plus simplement wFO), est donné par la grammaire suivante :

$$\varphi ::= s \mid P_a(x) \mid x \leq y \mid x \preccurlyeq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi$$

où $s \in \mathbb{S}$, $a \in \Sigma$ et $x, y \in \text{Var}$.

Pour $\varphi \in \text{wFO}(\mathbb{S}, \Sigma)$, notons $\text{Free}(\varphi)$ l'ensemble des variables libres apparaissant dans φ. Si $\text{Free}(\varphi) = \varnothing$, alors on dit que φ est une phrase. Pour un ensemble fini $\mathcal{V} \subseteq \text{Var}$ et un arbre t, une $(\mathcal{V}, t)$-valuation est une fonction σ qui assigne à chaque variable du premier ordre dans $\mathcal{V}$ un élément de $\text{dom}(t)$. Pour $x \in \text{Var}$ et $u \in \text{dom}(t)$, on note $\sigma[x \mapsto u]$ la $(\mathcal{V} \cup \{x\}, t)$-valuation qui envoie x sur u et qui coïncide avec σ sur les autres variables.

Dans la suite, on encode une paire (t, σ), avec σ une $(\mathcal{V}, t)$-valuation, par un $\Sigma_\mathcal{V}$-arbre, avec $\Sigma_\mathcal{V} := \Sigma \times \{0,1\}^\mathcal{V}$. On manipule un $\Sigma_\mathcal{V}$-arbre ρ comme un couple (t, σ) où $t(u) = a \in \Sigma$ et $\sigma(u) = v \in \{0,1\}^\mathcal{V}$ pour toute position $u \in \text{dom}(\rho)$, si $\rho(u) = (a, v)$. On dit que (t, σ) est un couple *valide* si, pour chaque variable du premier ordre $x \in \mathcal{V}$, la composante correspondant à x dans σ contient exactement un 1. Si (t, σ) est valide, alors on peut aisément interpréter σ comme une $(\mathcal{V}, t)$-valuation qui envoie chaque variable du premier ordre $x \in \mathcal{V}$ vers l'unique position contenant un 1 dans la composante de σ correspondant à x.

Pour définir la sémantique d'une formule φ de $\text{wFO}(\mathbb{S}, \Sigma)$, on fixe $\mathcal{V}$ un ensemble fini de variables tel que $\text{Free}(\varphi) \subseteq \mathcal{V}$. La sémantique de φ sur $\mathcal{V}$ est une série formelle $[\![\varphi]\!]_\mathcal{V} \in \mathbb{S}\langle\langle\mathcal{T}_{\Sigma_\mathcal{V}}\rangle\rangle$, donnée par la règle suivante : si (t, σ) n'est pas valide, on pose $[\![\varphi]\!]_\mathcal{V}(t, \sigma) = \mathbf{0}$, et sinon $[\![\varphi]\!]_\mathcal{V}$ est définie inductivement [2] dans la Table 1.

On note simplement $[\![\varphi]\!]$ la série formelle $[\![\varphi]\!]_{\text{Free}(\varphi)}$. Dans la suite de ce rapport, et par souci de clarté, on utilisera sans les mentionner certains raccourcis dans l'écriture des formules tels que l'atome $x = y$ en lieu et place de $x \leq y \wedge y \leq x$, ou l'atome $x \prec y$ en lieu et place de $x \preccurlyeq y \wedge \neg(y \preccurlyeq x)$.

On note $\mathbb{S}^{\text{wFO}}\langle\langle\mathcal{T}_\Sigma\rangle\rangle$ la collection de toutes les séries formelles définissables par une phrase de $\text{wFO}(\mathbb{S}, \Sigma)$.

Exemple 2.1. On se place dans l'alphabet $\Sigma = \{a, b\}$.

2. Si le semi-anneau n'est pas commutatif, l'ordre dans lequel est calculé le produit relatif à la quantification universelle importe : on fixe alors l'ordre linéaire préfixe du parcours en profondeur d'un arbre.

$$
\begin{aligned}
[\![s]\!]_\mathcal{V}(t,\sigma) &= s & [\![\neg\varphi]\!]_\mathcal{V}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{si } [\![\varphi]\!]_\mathcal{V}(t,\sigma) = \mathbf{0} \\ \mathbf{0} & \text{sinon} \end{cases} \\
[\![P_a(x)]\!]_\mathcal{V}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{si } t(\sigma(x)) = a \\ \mathbf{0} & \text{sinon} \end{cases} & [\![\varphi_1 \vee \varphi_2]\!]_\mathcal{V}(t,\sigma) &= [\![\varphi_1]\!]_\mathcal{V}(t,\sigma) \oplus [\![\varphi_2]\!]_\mathcal{V}(t,\sigma) \\
[\![x \leq y]\!]_\mathcal{V}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{si } \sigma(x) \leq \sigma(y) \\ \mathbf{0} & \text{sinon} \end{cases} & [\![\varphi_1 \wedge \varphi_2]\!]_\mathcal{V}(t,\sigma) &= [\![\varphi_1]\!]_\mathcal{V}(t,\sigma) \odot [\![\varphi_2]\!]_\mathcal{V}(t,\sigma) \\
[\![x \preccurlyeq y]\!]_\mathcal{V}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{si } \sigma(x) \preccurlyeq \sigma(y) \\ \mathbf{0} & \text{sinon} \end{cases} & [\![\exists x\ \varphi]\!]_\mathcal{V}(t,\sigma) &= \bigoplus_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t, \sigma[x \mapsto u]) \\
& & [\![\forall x\ \varphi]\!]_\mathcal{V}(t,\sigma) &= \bigodot_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t, \sigma[x \mapsto u])
\end{aligned}
$$

TABLE 1 – Sémantique de wFO($\mathbb{S}, \Sigma$)

1. Tout polynôme de $\mathbf{Nat}\langle\mathcal{T}_\Sigma\rangle$ est définissable par une phrase de wFO($\mathbb{S}, \Sigma$), à l'aide d'une disjonction finie de phrases $\varphi_{t,s}$ associant à un arbre t le poids s, dans laquelle la quantification existentielle initiale comporte autant de variables qu'il y a de nœuds dans t :
$$
\varphi_{t,s} = \exists \overline{x}\ \Big(\bigwedge_{u,ui \in \mathrm{dom}(t)} x_u < x_{ui}\ \wedge \bigwedge_{ui,u(i+1) \in \mathrm{dom}(t)} x_{ui} \prec x_{u(i+1)} \\
\wedge \forall y \bigvee_{u \in \mathrm{dom}(t)} y = x_u\ \wedge \bigwedge_{u \in \mathrm{dom}(t)} P_{t(u)}(x_u)\ \wedge\ s \Big)
$$
2. La phrase $\exists x\ \big(P_a(x) \vee (P_b(x) \wedge -1)\big)$ reconnaît la série formelle $|t|_a - |t|_b$ sur le semi-anneau **Int** (où $|t|_a$ est le nombre de nœuds de t étiquetés par a) : en effet, la formule ouverte $P_a(x) \vee (P_b(x) \wedge -1)$ associe à l'arbre (t, σ) la valeur 1 si $t(\sigma(x)) = a$, et -1 si $t(\sigma(x)) = b$, puis la quantification existentielle somme ces valeurs pour tous les nœuds de l'arbre.
3. Sur le semi-anneau **Nat**, la phrase $\forall x\ 2$ calcule $2^{|t|}$.

Dans tout semi-anneau, pour step $\in \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$, on peut facilement définir une formule ρ_{step} (resp. une formule ρ_{step^*}) à deux variables libres x et y telle que $[\![\rho_{\text{step}}]\!](t, [x \mapsto u, y \mapsto v])$ (resp. $[\![\rho_{\text{step}^*}]\!](t, [x \mapsto u, y \mapsto v])$) vaut $\mathbf{1}$ si $(u, v) \in \mathcal{R}^t_{\text{step}}$ (resp. $(u, v) \in (\mathcal{R}^t_{\text{step}})^*$) et $\mathbf{0}$ sinon.

Dans la suite, on utilisera la notation $\varphi_1 \overset{+}{\Longrightarrow} \varphi_2$ comme une macro pour $\neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2)$ (dont la sémantique coïncide avec une implication booléenne dans le cas du semi-anneau **Bool**).

2.2 Extension et sémantique quantitative des langages de type XPath

2.2.1 wXPath

Dans le cadre des Σ-arbres, on donne ici une syntaxe *sympathique* permettant d'exprimer exactement les propriétés booléennes de XPath, étendue en rendant possible l'usage de constantes du semi-anneau à l'intérieur des formules d'état.

Définition 2.3.

wXPath($\mathbb{S}, \Sigma$) (ou plus simplement wXPath) est un langage de requêtes, possédant deux sortes : les requêtes d'état ($\psi, \psi_1, \ldots$) et les requêtes de chemin ($P, Q, \ldots$). Il est défini par la grammaire suivante :

$$
\begin{aligned}
\psi &::= a \mid s \mid \psi \vee \psi \mid \psi \wedge \psi \mid \neg\psi \mid \langle P\rangle\psi \mid [P]\psi \\
P &::= \bullet \mid \text{step} \mid \text{step}^* \mid P/P \mid P \cup P \mid \psi?
\end{aligned}
$$

dans laquelle $a \in \Sigma$, $s \in \mathbb{S}$ et step est un élément quelconque de $D = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$.

Étant donné un Σ-arbre t, chaque requête d'état ψ associe à chaque nœud u de t un poids $[\![\psi]\!]^t(u)$ et chaque requête de chemin P associe à chaque paire (u, v) de nœuds de t un poids $[\![P]\!]^t(u, v)$. Afin de les définir, on commence par traduire les requêtes de wXPath dans wFO$(\mathbb{S}, \Sigma)$ grâce à la Table 2. Par suite, on définit la sémantique d'une requête d'état ψ par $[\![\psi]\!]^t(u) = [\![\mathrm{Tr}_x(\psi)]\!](t, [x \mapsto u])$ et d'une requête de chemin P par $[\![P]\!]^t(u, v) = [\![\mathrm{Tr}_{x,y}(P)]\!](t, [x \mapsto u, y \mapsto v])$.

$$
\begin{array}{rcl@{\qquad}rcl}
\mathrm{Tr}_x(a) = & P_a(x) & & \mathrm{Tr}_{x,y}(\bullet) = & x = y \\
\mathrm{Tr}_x(s) = & s & & \mathrm{Tr}_{x,y}(\mathrm{step}) = & \rho_{\mathrm{step}}(x, y) \\
\mathrm{Tr}_x(\psi_1 \vee \psi_2) = & \mathrm{Tr}_x(\psi_1) \vee \mathrm{Tr}_x(\psi_2) & & \mathrm{Tr}_{x,y}(\mathrm{step}^*) = & \rho_{\mathrm{step}^*}(x, y) \\
\mathrm{Tr}_x(\psi_1 \wedge \psi_2) = & \mathrm{Tr}_x(\psi_1) \wedge \mathrm{Tr}_x(\psi_2) & & \mathrm{Tr}_{x,y}(P_1/P_2) = & \exists z\ (\mathrm{Tr}_{x,z}(P_1) \wedge \mathrm{Tr}_{z,y}(P_2)) \\
\mathrm{Tr}_x(\neg\psi) = & \neg\mathrm{Tr}_x(\psi) & & \mathrm{Tr}_{x,y}(P_1 \cup P_2) = & \mathrm{Tr}_{x,y}(P_1) \vee \mathrm{Tr}_{x,y}(P_2) \\
\mathrm{Tr}_x(\langle P\rangle\psi) = & \exists y\ (\mathrm{Tr}_{x,y}(P) \wedge \mathrm{Tr}_y(\psi)) & & \mathrm{Tr}_{x,y}(\psi?) = & x = y \wedge \mathrm{Tr}_x(\psi) \\
\mathrm{Tr}_x([P]\psi) = & \forall y\ (\mathrm{Tr}_{x,y}(P) \overset{+}{\Longrightarrow} \mathrm{Tr}_y(\psi)) & & &
\end{array}
$$

TABLE 2 – Traduction de wXPath$(\mathbb{S}, \Sigma)$ vers wFO$(\mathbb{S}, \Sigma)$

Remarque 1. La définition de la sémantique de wXPath montre ainsi trivialement que wXPath est un fragment de wFO.

On note step$^+$ la requête de chemin step/step*. On abrège les requêtes $\langle P\rangle\mathbf{1}$ en $\langle P\rangle$. Dans la suite, afin de rester au plus près des habitudes du langage XPath, on dira qu'une requête *sélectionne* des nœuds et leur associe en plus une valeur non nulle du semi-anneau.

Exemple 2.2.

- $\mathbb{S} = \mathbf{Bool}$: on obtient exactement la sémantique traditionnelle de XPath. En particulier, on peut définir les macros suivantes, qu'on utilisera par la suite dans tout semi-anneau :
  - leaf $\equiv \neg\langle\downarrow\rangle$ qui sélectionne les feuilles
  - root $\equiv \neg\langle\uparrow\rangle$ qui sélectionne la racine
  - first $\equiv \neg\langle\leftarrow\rangle$ qui sélectionne le fils gauche de chaque nœud
  - last $\equiv\ \neg\langle\rightarrow\rangle$ qui sélectionne le fils droit de chaque nœud
- $\mathbb{S} = \mathbf{Nat}$:
  - $\langle\downarrow^*\rangle$ calcule la taille du sous-arbre enraciné en le nœud sélectionné
  - $\langle(\rightarrow^+ \cup \leftarrow^+)/\psi?\rangle$ calcule le nombre de frères du nœud sélectionné qui vérifient la propriété ψ, en comptant de plus, le poids de la vérification de ψ
  - $\langle\downarrow^*/\mathrm{leaf}?\rangle$ (ou $\langle\downarrow^*\rangle\mathrm{leaf}$) calcule le nombre de feuilles du sous-arbre enraciné en le nœud sélectionné
  - $[\langle\downarrow^*\rangle\mathrm{leaf}]\big((a \wedge 2) \vee (b \wedge 3)\big)$ calcule $2^{f_a(t)}3^{f_b(t)}$, où $f_a(t)$ (resp. $f_b(t)$) représente le nombre de feuilles de t étiquetées par la lettre a (resp. b)
  - $\downarrow^*/a?/\downarrow^*$ calcule le nombre de lettres a entre deux nœuds sélectionnés par la requête de chemin : en effet, cette formule sélectionne des paires de nœuds ancêtres l'un de l'autre, et compte le poids 1 pour chaque a sur l'unique chemin qui relie les deux nœuds (utilise le non déterminisme pour compter).
- $\mathbb{S} = \mathbf{Trop}$: à cause de l'idempotence de l'opérateur min, on ne peut plus compter par non déterminisme comme précédemment. En particulier, il est impossible de compter des distances *arbitrairement grandes* dans le semi-anneau.
  - $\langle\uparrow^*/\psi?\rangle$ sélectionne les nœuds qui possèdent des ancêtres vérifiant une requête d'état ψ en calculant de plus le coût minimal nécessaire à la vérification de cette propriété
  - $\langle\leftarrow\rangle \overset{+}{\Longrightarrow} 1 \wedge \langle\rightarrow\rangle \overset{+}{\Longrightarrow} 1 \wedge \langle\uparrow\rangle \overset{+}{\Longrightarrow} 1 \wedge \langle\downarrow\rangle \overset{+}{\Longrightarrow} 1$ calcule le nombre de *types de voisin* (fils, frère gauche, frère droit, parent) du nœud sélectionné.

2.2.2 wCondXPath

On a vu, en particulier dans le semi-anneau tropical, qu'il est difficile de compter le long d'un chemin sélectionné par une requête de chemin. Plus généralement, on ne peut pas sélectionner un chemin, en conditionnant sur chaque étape de ce chemin. C'est ce que permet de faire *Conditional XPath*, qu'on étend de manière similaire en une version quantitative dans la définition suivante.

Définition 2.4.

wCondXPath($\mathbb{S}, \Sigma$) (ou plus simplement wCondXPath) est le langage de requêtes défini par la grammaire suivante :

$$\psi ::= a \mid s \mid \psi \vee \psi \mid \psi \wedge \psi \mid \neg\psi \mid \langle P \rangle \psi \mid [P]\psi$$
$$P ::= \bullet \mid \text{step} \mid P/P \mid P \cup P \mid \psi? \mid (\text{step}/\psi?)^*$$

dans laquelle $a \in \Sigma$, $s \in \mathbb{S}$ et step est un élément quelconque de $D = \{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. On a donc ajouté à wXPath toutes les clôtures transitives de la forme $(\text{step}/\psi?)^*$.

La traduction de la requête $(\text{step}/\psi?)^*$ est définie par

$$\text{Tr}_{x,y}((\text{step}/\psi?)^*) = \rho_{\text{step}^*}(x,y) \wedge \forall z \left(\left(\rho_{\text{step}^*}(x,z) \wedge \rho_{\text{step}^*}(z,y) \wedge x \neq z \right) \xRightarrow{+} \text{Tr}_z(\psi) \right)$$

Dans la suite, on utilise la notation $(\text{step}/\psi?)^+$ pour $\text{step}/\psi?/(\text{step}/\psi?)^*$. De plus, on définit le symétrique de l'opération de clôture transitive conditionnelle, grâce à $(\psi?/\text{step})^*$ qui est un raccourci pour $\bullet \cup \psi?/(\text{step}/\psi?)^*/\text{step}$.

Exemple 2.3.

- $\mathbb{S} = \mathbf{Bool}$: on retrouve à nouveau la sémantique traditionelle de CondXPath. En particulier, la clôture transitive conditionnelle permet l'écriture d'opérations *Tant Que*, impossible en XPath.
- $\mathbb{S} = \mathbf{Nat}$:
  - $(\downarrow /(\text{first} \wedge 2)?) \cup (\text{leaf}?/((\text{last} \wedge 2)?/ \uparrow)^*/ \rightarrow /4?)$ sélectionne les paires de nœuds (u, v) telles que v est le successeur immédiat de u dans l'ordre linéaire préfixe du parcours en profondeur de l'arbre, et calcule 2^d où d est la distance entre u et v dans l'arbre (on compte par déterminisme en ajoutant des poids dans la requête)
  - $(\downarrow /\text{first}?) \cup (\text{leaf}?/(\text{last}?/ \uparrow)^*/(\text{last}?/ \uparrow)^*/ \rightarrow)$ sélectionne les paires de nœuds (u, v) telles que v est le successeur immédiat de u dans l'ordre linéaire préfixe du parcours en profondeur de l'arbre, et calcule la distance entre u et v dans l'arbre (on compte grâce au non déterminisme quant au choix du lieu de coupure entre la concaténation des deux clôtures transitives $(\text{last}?/ \uparrow)^*$)
- $\mathbb{S} = \mathbf{Trop}$: on peut désormais calculer des distances, et les minimiser en utilisant des clôtures transitives conditionnelles : $(\uparrow /1?)^*/(\downarrow /1?)^*$ calcule la longueur du plus petit chemin séparant les deux nœuds sélectionnés (sans utiliser les liens de fraternité)
- $\mathbb{S} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$: $\langle(\downarrow /1?)^*/\text{leaf}?\rangle$ calcule la hauteur du sous-arbre enraciné en le nœud sélectionné

Dans le cas booléen, [Mar04] établit que CondXPath est un langage de requête FO-complet, à savoir que non seulement, comme on l'a vu précédemment, on peut traduire chaque requête CondXPath vers une formule de FO, mais aussi que toute formule du premier ordre à une variable libre est équivalente à une requête d'état de CondXPath. Dans le cas quantitatif, nous posons donc comme conjecture qu'il en est de même, sans pour autant l'avoir prouvé jusqu'à maintenant.

2.3 Problèmes décisionnels et calculatoires liés

Remarquons pour commencer que les deux formalismes précédents (wFO et wCondXPath) ont des fonctionnements différents : là où une formule du premier ordre réalise une passe sur un arbre, éventuellement décoré des valuations des variables libres, et y associe un poids, une requête wCondXPath annote chaque nœud (ou chaque paire de nœuds) d'un arbre donné par un poids. Cependant, il est possible, comme on l'a déjà mentionné, de transformer un format dans l'autre, et vice versa. On peut considérer les formules du premier ordre avec une ou deux variables libres, et les considérer comme des requêtes qui associent à chaque nœud (ou couple de nœuds) un poids.

Ainsi, si on s'intéresse au problème de la satisfaisabilité d'une requête, on peut l'exprimer de deux manières équivalentes :

1. Étant donnée une formule φ de wFO sur $\Sigma_{\mathcal{V}}$, existe-t-il un $\Sigma_{\mathcal{V}}$-arbre (t, σ) tel qu'on ait $[\![\varphi]\!](t, \sigma) \neq \mathbf{0}$?
2. Étant donnée une requête d'état ψ de wCondXPath sur $\mathcal{T}_\Sigma$, existe-t-il un Σ-arbre t et un nœud $u \in \mathrm{dom}(t)$ tel que $[\![\psi]\!]^t(u) \neq \mathbf{0}$?

Pour simplifier, on exprime ci-dessous d'autres problèmes, uniquement dans l'un ou l'autre des deux formalismes. On suppose toujours que le semi-anneau est fixé dans ces différents problèmes de décision et de calcul (ce n'est pas une instance du problème).

Appartenance. Étant donnés une formule φ et un arbre t, a-t-on $[\![\varphi]\!](t) \neq 0$ (ce qu'on pourrait écrire $t \vDash \varphi$) ?

Model checking. Étant donnés une formule φ et un langage régulier d'arbres L, existe-t-il $t \in L$ tel que $t \vDash \varphi$?

Calcul. Étant donnés une formule φ et un arbre t, calculer $[\![\varphi]\!](t)$. En particulier, on s'intéresse à des résultats de complexité d'un tel calcul.

Inclusion. Considérons un semi-anneau muni d'une relation d'ordre $\sqsubseteq$. Étant données deux formules φ_1 et φ_2, est-il vrai que pour tout arbre t, $[\![\varphi_1]\!](t) \sqsubseteq [\![\varphi_2]\!](t)$?

Chevauchement. Étant données deux formules φ_1 et φ_2, décider si elles se chevauchent, i.e. s'il existe un arbre t tel que $[\![\varphi_1]\!](t)$ et $[\![\varphi_2]\!](t)$ sont tous les deux non nuls.

3 Le cas des mots : automates et spécification

La section précédente a mis en place des langages de requêtes quantitatives sur les arbres. Afin de pouvoir les étudier (par exemple tester la satisfiabilité), la démarche classique consiste à exprimer ces requêtes dans un modèle de machines, sur lesquelles il est souvent plus simple de raisonner. On se restreint dans cette section au cas des mots. On étudiera le cas des arbres dans la section suivante. On fixe un semi-anneau $\mathbb{S}$ et un alphabet fini Σ.

3.1 Automates pondérés et logique quantitative sur les mots

Définition 3.1.

Un automate pondéré sur $\mathbb{S}$ et Σ est un quadruplet $\mathcal{A} = (Q, \lambda, \mu, \nu)$ où
- Q est un ensemble fini d'états,
- $\mu : \Sigma \to S^{Q \times Q}$ est la fonction de transitions pondérées,
- $\lambda \in S^{1 \times Q}$ est le vecteur ligne de poids initial,
- $\nu \in S^{Q \times 1}$ est le vecteur colonne de poids final.

Afin de définir la sémantique de tels automates, remarquons que l'ensemble des matrices $S^{Q\times Q}$ peut être muni de la structure de semi-anneau comme précisé dans l'exemple 1.1. Ainsi, on peut étendre la fonction μ en un morphisme du semi-anneau $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ dans $S^{Q\times Q}$: l'image d'un mot $w = a_1 \cdots a_n$ est alors donnée par la multiplication matricielle $\mu(w) = \mu(a_1) \odot \cdots \odot \mu(a_n)$.

Définition 3.2.

Soit $\mathcal{A} = (Q, \lambda, \mu, \nu)$ un automate pondéré sur $\mathbb{S}$ et Σ. La sémantique de l'automate $\mathcal{A}$, notée $[\![\mathcal{A}]\!]$, est une série formelle de $\mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ telle que pour tout mot $w \in \Sigma^*$, $[\![\mathcal{A}]\!](w) = \lambda \odot \mu(w) \odot \nu$.

Remarque 2. Remarquons qu'on a défini la sémantique des automates pondérés en termes algébriques, par multiplication de matrices. On aurait aussi pu donner une définition équivalente en terme de calcul progressant de gauche à droite dans le mot donné en entrée.

Exemple 3.1. L'automate de la Figure 1 reconnaît la série formelle $|w|_a - |w|_b$ de l'exemple 1.2 : c'est la représentation graphique (dans laquelle on a ôté les transitions de poids 0) de l'automate dont les états sont $Q = \{1, 2\}$, la fonction de transitions est $\mu(a) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ et $\mu(b) = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}$, le vecteur initial est $\lambda = (1\ 0)$ et le vecteur final est $\nu = (0\ 1)$. Il est non déterministe au sens où plusieurs calculs peuvent accepter chaque mot. Notons que la négation de cette série formelle qui associe la valeur 1 aux mots qui contiennent le même nombre de a que de b, et 0 aux autres arbres, n'est pas reconnaissable par un automate pondéré.

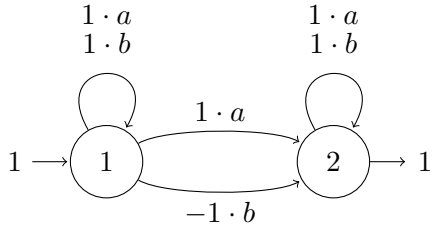


FIGURE 1 – Automate reconnaissant la série $|w|_a - |w|_b$

Spécialisons ensuite la définition 2.2 de la logique du premier ordre au cas des mots.

Définition 3.3.

L'ensemble des formules du premier ordre sur le semi-anneau $\mathbb{S}$ et l'alphabet Σ, noté $\mathrm{wFO}(\mathbb{S}, \Sigma)$ (ou plus simplement wFO), est donné par la grammaire suivante :

$$\varphi ::= s \mid P_a(x) \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi$$

où $s \in \mathbb{S}$, $a \in \Sigma$ et $x, y \in \mathrm{Var}$.

La sémantique de l'ordre $\leq$ est l'ordre linéaire donné par les positions du mot. Le codage précédent d'une $(\mathcal{V}, t)$-valuation σ dans un arbre (t, σ) sera ici simplifié, en disant qu'une $(\mathcal{V}, w)$-valuation σ est encodée comme le mot $\sigma = \sigma_1 \cdots \sigma_n \in (\{0, 1\}^{\mathcal{V}})^*$ avec n la longueur du mot w et σ_i le vecteur dont la composante relative à la variable x contient un 1 si $\sigma(x) = i$, et un 0 sinon. On définit de manière similaire au cas des arbres la notion de couple (w, σ) valide (et dans ce cas, on verra indifféremment σ comme un mot sur $\{0, 1\}^{\mathcal{V}}$ ou comme une $(\mathcal{V}, w)$-valuation).

Exemple 3.2. Sur le semi-anneau **Nat**, la série formelle $[\![\forall x\ 2]\!](w) = 2^{|w|}$ est reconnaissable par l'automate $\mathcal{A} = (\{q\}, (1), \mu, (1))$ pour lequel pour toute lettre $a \in \Sigma$, $\mu(a)_{q,q} = 2$.

Exemple 3.3. Sur le semi-anneau **Nat**, la série formelle $[\![\forall x \forall y\ 2]\!](w) = 2^{|w|^2}$ n'est pas reconnaissable par un automate pondéré. En effet, remarquons que si $\mathcal{A} = (Q, \lambda, \mu, \nu)$ est un automate à poids, alors $[\![\mathcal{A}]\!](u) = O((M|Q|)^{|u|+2})$ pour $M = \max\{\mu(a)_{p,q}, \lambda_p, \nu_q \mid a \in \Sigma, p, q \in Q\}$, puisqu'il y a $O(|Q|^{|u|+1})$ calculs sur u, chacun de poids majoré par $M^{|u|+2}$.

3.2 Clôture transitive et formules en forme normale

On a vu dans l'exemple 3.3 que, pour des raisons combinatoires dans le semi-anneau **Nat**, les automates à poids ne peuvent pas être clos par quantification universelle du premier ordre. Si on veut obtenir un résultat d'expressivité avec une logique, il nous faut donc restreindre, d'une manière ou d'une autre, cette quantification. La première façon de faire, qui donne le résultat fondateur dont nous avons brièvement parlé en introduction, est donné dans [DG07] : l'idée est d'utiliser un fragment de la logique monadique du second ordre, sans quantification universelle du second ordre, et de limiter la quantification universelle du premier ordre à des *formules reconnaissables étagées*, c'est-à-dire des formules dont l'image est finie et dont chaque préimage est un langage rationnel. Nous allons ici réutiliser cette classe de formules étagées mais en introduisant non pas des quantifications du second ordre, mais un opérateur unique de clôture transitive. Ceci reprend l'idée développée pour les arbres dans [NS03].

Étendons dans un premier temps la notion de clôture transitive dans le cadre quantitatif : on autorise donc des formules de la forme $\mathrm{TC}^{<}_{xy}\varphi$. On commence par donner une sémantique à ce nouvel opérateur. La définition suivante est à comparer à celle de la requête de wCondXPath $(\mathrm{step}/\varphi?)^*$ donnée dans la définition 2.4 : la différence majeure est qu'ici on s'autorise les clôtures transitives de formules qui peuvent boucler, ce qui complique la définition. En particulier, le poids donné au couple (i, j) de positions d'un mot dans la formule $\mathrm{TC}^{<}_{xy}\varphi$ ne consistera pas en la somme du poids de l'ensemble des *chemins* de i à j par des sauts $\varphi(x, y)$: on se limite aux chemins simples, c'est-à-dire ceux sans boucle.

Définition 3.4.

Soit $\varphi(x, y)$ une formule contenant au moins deux variables libres x et y. On définit les puissances de φ par $\varphi^1(x, y) = x \leq y \wedge \varphi(x, y)$ et pour $n \geq 2$

$$\varphi^n(x, y) = \exists z_0 \cdots \exists z_n \Big[x = z_0 \wedge y = z_n \wedge \bigwedge_{1 \leq i \leq n} \big(z_{i-1} < z_i \wedge \varphi(z_{i-1}, z_i)\big) \Big] \qquad (1)$$

On introduit alors l'opérateur de clôture transitive $\mathrm{TC}^{<}_{xy}\varphi$ dont la sémantique est donnée par

$$[\![[\mathrm{TC}^{<}_{xy}\varphi](z, t)]\!] = \bigoplus_{n \geq 1} [\![\varphi^n(z, t)]\!]$$

Remarquons que cette somme infinie est bien définie puisque $[\![\varphi^n(x, y)]\!](u, \sigma) = \mathbf{0}$ si $n \geq \max(2, |u|)$. Dans la suite, on abrège $[\![[\mathrm{TC}^{<}_{xy}\varphi](z, t)]\!](u, [z \mapsto i, t \mapsto j])$ en $[\![\mathrm{TC}^{<}_{xy}\varphi]\!](u, i, j)$. Si φ possède d'autres variables libres que x et y, on les appelle *paramètres* de la formule.

Intuitivement, l'opérateur $\mathrm{TC}^{<}_{xy}$ généralise la clôture transitive booléenne d'une relation définie par une formule avec deux variables libres.

Exemple 3.4. Soit $\varphi(x, y) = y \leq x + 1 \wedge \forall z\ 2$ sur le semi-anneau **Nat**. Pour un mot u, on a $[\![\mathrm{TC}^{<}_{xy}\varphi]\!](u, 1, |u|) = \prod_{i=1}^{|u|-1} [\![\varphi]\!](u, i, i+1) = 2^{|u|(|u|-1)}$. Ceci montre que la classe des séries reconnaissables n'est pas fermée par clôture transitive (on peut reformuler ce constat en disant que la clôture transitive permet de simuler une quantification universelle du premier ordre).

On cherche dans la suite une caractérisation des séries reconnaissables comme la classe des séries définie par une formule en forme normale $\mathrm{TC}^{<}_{xy}\varphi$. Au vu du précédent exemple, il faut restreindre la forme de la formule φ pour rester dans le champ des séries reconnaissables. Pour cela, on définit un fragment syntaxique de $\mathrm{wFO}(\mathbb{S}, \Sigma)$, qui sera interprété comme des formules booléennes.

Définition 3.5.

On note $\mathrm{bFO}(\mathbb{S}, \Sigma)$ (ou plus simplement bFO) le fragment syntaxique booléen de $\mathrm{wFO}(\mathbb{S}, \Sigma)$ donné par

$$\varphi ::= \mathbf{0} \mid \mathbf{1} \mid P_a(x) \mid x \leq y \mid \neg\varphi \mid \varphi \wedge \varphi \mid \forall x\, \varphi$$

où $a \in \Sigma$ et $x, y \in \mathrm{Var}$.

On peut vérifier par induction que la sémantique d'une formule bFO prend toutes ses valeurs dans $\{\mathbf{0}, \mathbf{1}\}$: c'est la raison pour laquelle on a interdit l'usage des disjonctions et des quantifications existentielles du premier ordre. On peut introduire des macros définissant la disjonction et la quantification existentielle booléenne : $\varphi \veebar \psi \stackrel{\text{déf}}{=} \neg(\neg\varphi \wedge \neg\psi)$ et $\exists x\varphi \stackrel{\text{déf}}{=} \neg\forall x\neg\varphi$. La sémantique de $\veebar$ and $\exists$ coïncide avec la sémantique classique de la disjonction et de la quantification existentielle dans le semi-anneau des booléens. On est alors prêt pour définir le fragment utile.

Définition 3.6.

Une formule bFO-étagée est une formule obtenue à partir de la grammaire

$$\varphi ::= s \mid \alpha \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi$$

avec $s \in \mathbb{S}$ et $\alpha \in \mathrm{bFO}$. En particulier, les quantifications ne sont autorisées que dans la partie booléenne de la formule.

Par souci de simplicité, on utilise dans la suite une forme particulière pour les formules bFO-étagées, donnée par le lemme suivant.

Lemme 3.1. *Pour tout formule φ bFO-étagée, on peut construire une formule équivalente $\psi = \bigvee_i(\varphi_i \wedge s_i)$ avec $\varphi_i \in \mathrm{bFO}$ et $s_i \in \mathbb{S}$. Plus précisément,* $\mathrm{Free}(\varphi) = \mathrm{Free}(\psi)$ *et* $[\![\varphi]\!]_{\mathcal{V}}(w, \sigma) = [\![\psi]\!]_{\mathcal{V}}(w, \sigma)$ *pour tout $\mathcal{V}$ contenant* $\mathrm{Free}(\varphi)$ *et pour tout couple valide (w, σ).*

Dans la suite, on note $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod})$ l'ensemble des formules de la forme $\mathrm{TC}^{<}_{xy}\varphi$ avec φ une formule bFO-étagée pouvant utiliser de plus les prédicats unaires $x \equiv_\ell m$ pour tout $\ell \in \mathbb{N}$ et $1 \leq m \leq \ell$ (sa sémantique est donnée par $[\![x \equiv_\ell m]\!](w, \sigma) = \mathbf{1}$ si $\sigma(x) \equiv m \mod \ell$ et $\mathbf{0}$ sinon). On dit qu'une série $f \in \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ est $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod})$-définissable, s'il existe une formule $\varphi(x, y)$ sans paramètre telle que pour tout mot u, $f(u) = [\![\mathrm{TC}^{<}_{xy}\varphi]\!](u, 1, |u|)$.

Proposition 3.1. *Soit $f \in \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$. Si f est une série reconnaissable,* i.e., *$f \in \mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$, alors f est $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod})$-définissable.*

Démonstration. En annexe, page 22. □

L'inclusion précédente est en fait une égalité. Pour prouver l'autre égalité (que nous n'utiliserons pas dans la suite de ce rapport), on utilise le fragment défini dans [DG07], comme un fragment intermédiaire. Finalement, on obtient le résultat d'expressivité suivant :

Théorème 3.1. $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod}) = \mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$.

Démonstration. En annexe, page 24. □

La section 3 du papier [BGMZ10] apporte d'autres formalismes logiques équivalents des séries reconnaissables. Par exemple, citons juste que le théorème 3.1 est encore vrai dans le cas d'un semi-anneau non commutatif, quitte à utiliser une classe un peu plus restrictive de formules MSO restreintes, définies dans [DG09].

3.3 Automates imbriqués : extension du pouvoir d'expression

On souhaite définir dans cette partie une classe robuste d'automates, permettant d'exprimer (au moins) toute requête de la logique du premier ordre sur les mots. On donne certaines intuitions intéressantes dans un premier temps, à travers la notion d'automate boustrophédon à jetons. Dans un second temps, on considérera une sous-classe de ces automates, les automates imbriqués, qui possèdent un formalisme plus simple pour réaliser les preuves et qui suffisent pour engendrer toutes les requêtes de wFO. On prouve un résultat d'expressivité qui relie cette classe d'automates à une classe de logique contenant wFO et un opérateur de clôture transitive bornée.

3.3.1 Nécessité de calculer des valeurs plus grandes : automates à jetons

On a déjà vu à plusieurs reprises que les automates pondérés ne sont pas clos par quantification universelle, ou par clôture transitive (exemples 3.3 et 3.4). Cela vient du fait que les automates pondérés ne peuvent pas calculer des valeurs trop grandes. La raison essentielle est qu'ils sont limités à une passe de gauche à droite du mot donné en entrée : même en utilisant leur non déterminisme intrinsèque (qui se traduit par une somme sur les différents calculs de l'automate), dans le semi-anneau **Nat**, ils ne peuvent engendrer des poids supérieurs à une exponentielle en la longueur du mot (exemple 3.3).

Afin de remédier à ce problème, considérons donc dans un premier temps des automates *boustrophédons* pondérés. Ces automates ont donc la possibilité de lire le mot en entrée, et de se déplacer vers la droite ou vers la gauche à chaque instant – en compilant un poids – cette décision dépendant de l'état courant de l'automate et de la lettre à la position courante. Contrairement à la sémantique des automates pondérés, donnée dans la définition 3.2, qu'on obtient de manière algébrique par multiplication matricielle, on est forcé maintenant de définir la sémantique d'un automate boustrophédon pondéré en terme de calculs acceptants. À un automate $\mathcal{A}$ et un mot w, on associe donc un poids calculé par la somme sur l'ensemble des calculs acceptants de $\mathcal{A}$ sur w des poids de ces calculs, eux-mêmes calculés comme le produit des poids des transitions qu'ils contiennent : pour fixer les idées, considérons qu'un calcul acceptant est un calcul qui commence à gauche du mot, termine à droite du mot, et que le poids de ce calcul est calculé en prenant en compte le poids des transitions et le poids de l'état initial et de l'état final du calcul.

Cependant, on voit aisément qu'un mot peut posséder un nombre infini de calculs acceptants (dès que l'automate peut boucler), auquel cas la somme définissant la sémantique de l'automate sur ce mot n'est pas nécessairement définie. Comme dans la définition 3.4 de la clôture transitive, on peut donc résoudre ce problème en considérant dans la sémantique uniquement les calculs simples de l'automate, à savoir ceux qui ne contiennent pas deux configurations identiques (même état rencontré deux fois sur la même position du mot). Mais alors, il n'est pas très difficile de voir que toute série formelle reconnaissable par un automate boustrophédon pondéré peut aussi être reconnue par un automate pondéré, si le semi-anneau est commutatif (généralisation de la traduction à base de *crossing sequences* développée dans le théorème 2.5 et l'exercice 2.18 de [HU79]). En particulier, les automates boustrophédons pondérés ne permettent pas de calculer des poids plus importants que leurs cousins à simple sens.

Cependant, cette classe d'automates a l'avantage de pouvoir aisément être étendue en utilisant un nombre fini de jetons. Un automate à jetons sur les mots (voir [GH96] pour la définition dans le cadre booléen) est ainsi un automate à double sens qui, outre sa capacité à se déplacer

sur le mot d'entrée, peut poser des jetons sur le mot, puis revenir les enlever plus tard. Plusieurs limitations sont apportées afin de ne pas donner trop de pouvoir à ces automates. La première est que les jetons sont numérotés (de 1 à r) et qu'à tout moment de l'exécution, l'ensemble des jetons disposés sur le mot est un sous-ensemble $\{p, p+1, \dots, r-1, r\}$ de $\{1, \dots, r\}$: cela peut s'exprimer en disant qu'à tout instant, on peut uniquement déposer le plus grand jeton non posé, et relever le plus petit jeton posé. Une deuxième restriction concerne la capacité donnée à l'automate pour relever un jeton : les automates à jetons faibles doivent être sur la position du jeton pour le relever, tandis que les automates à jetons forts n'ont pas cette restriction. Dans le cadre booléen, il a été montré qu'aucune de ces variantes ne permet d'engendrer plus de langages que les langages réguliers.

Cependant, il n'est pas difficile de se convaincre qu'il n'en est rien dans le cadre quantitatif. En particulier, si $\mathcal{A}$ est un automate à jetons sur l'alphabet $\Sigma_{\{x\}}$, on peut construire un automate à jetons $\mathcal{B}$ (avec un jeton supplémentaire) qui calcule la série formelle obtenue par quantification universelle de $[\![\mathcal{A}]\!]$: on utilise le jeton supplémentaire pour marquer successivement chacune des positions du mot, puis à chaque position, on simule l'exécution de $\mathcal{A}$ sur l'alphabet Σ en repérant la position de la variable x comme la position du jeton. Par une construction similaire, on voit aisément qu'en fait les automates à jetons peuvent exprimer toutes les formules du premier ordre.

On va voir dans la suite qu'ils peuvent aussi permettre d'exprimer des clôtures transitives. Cependant, le formalisme double sens est peu adapté lorsqu'il s'agit de preuves, en particulier à cause de la restriction de calculs simples dans la définition de la sémantique. On va donc se ramener dans la section suivante au cas des automates à simple sens, en s'autorisant des *appels imbriqués*.

3.3.2 Clôture transitive bornée et automates imbriqués

On considère donc dans cette section une sous-classe restreinte des automates à jetons qu'on a introduit précédemment de manière informelle. On force dans un premier temps les jetons à être *progressants*, au sens où à chaque fois qu'un jeton p est levé d'une position i, l'automate peut soit lever le jeton $p+1$ (s'il existe), soit reposer le jeton p à **droite** de la position i. Dans un second temps, on force une exécution *simple sens imbriquée* des automates : à chaque fois qu'un jeton p est posé, l'automate revient au début du mot, puis reprend une exécution simple sens dans laquelle il ne peut pas toucher au jeton p, à la fin de laquelle il revient à la position du jeton p. On formalise finalement cette définition comme suit :

Définition 3.7.

Pour $r \geq 0$, on définit une classe r-nwA(Σ) (pour *nested weighted automata*) d'automates imbriqués de rang r sur Σ (et $\mathbb{S}$). Un automate imbriqué de rang 0 sur Σ est un automate pondéré sur $\mathbb{S}$ et Σ. Pour $r > 0$, un automate imbriqué de rang r sur Σ est un quadruplet $\mathcal{A} = (Q, \lambda, \mu, \nu)$ où

- Q est un ensemble fini d'états,
- $\mu : Q \times \Sigma \times Q \to (r-1)$-nwA$(\Sigma \times \{0,1\})$ est la fonction de transitions,
- $\lambda \in S^{1 \times Q}$ est le vecteur ligne de poids initial,
- $\nu \in S^{Q \times 1}$ est le vecteur colonne de poids final.

Définissons la sémantique $[\![\mathcal{A}]\!] \in \mathbb{S}\langle\langle \Sigma^* \rangle\rangle$ de $\mathcal{A} = (Q, \lambda, \mu, \nu) \in r$-nwA$(\Sigma)$. Si $r = 0$, alors la série formelle $[\![\mathcal{A}]\!]$ est définie comme dans le cas des automates pondérés (définition 3.2). Pour $r > 0$, un calcul de $\mathcal{A}$ sur $w = w_1 w_2 \cdots w_n$ est une suite d'états $\rho = q_0 \xrightarrow{w_1} q_1 \xrightarrow{w_2} \cdots \xrightarrow{w_n} q_n$. Le poids de ce calcul est défini par

$$\pi(\rho) \overset{\text{déf}}{=} \lambda_{q_0} \odot \Big[\bigodot_{i=1}^{n} [\![\mu(q_{i-1}, w_i, q_i)]\!](w, i) \Big] \odot \nu_{q_n}$$

où $(w,i) \in (\Sigma \times \{0,1\})^*$ est le mot $v = v_1 \cdots v_n$ avec $v_i = (w_i, 1)$ et $v_j = (w_j, 0)$ si $j \neq i$. On définit alors $[\![\mathcal{A}]\!](w)$ comme la somme (finie) des poids des calculs de $\mathcal{A}$ sur w.

On définit nwA comme $\bigcup_{r\geq 0} r$-nwA. On dit qu'une série $f \in \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ est r-nwA-reconnaissable si $f = [\![\mathcal{A}]\!]$ pour un r-nwA $\mathcal{A}$, et qu'elle est nwA-reconnaissable si elle est r-nwA-reconnaissable pour un certain r. On pose enfin $\mathbb{S}^{r\text{-nwA}}\langle\langle\Sigma^*\rangle\rangle$ (resp. $\mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle$) la classe des séries r-nwA-reconnaissables (resp. nwA-reconnaissables) sur Σ et $\mathbb{S}$.

Exemple 3.5. Un 1-nwA reconnaissant la série $w \mapsto 2^{|u|^2}$ sur **Nat** est $\mathcal{A} = (\{p\}, (1), \mu, (1))$ où pour toute lettre $a \in \Sigma$, $\mu(p, a, p)$ est l'automate pondéré défini dans l'exemple 3.2.

On a introduit les automates imbriqués de rang r comme une sous-classe des automates à r jetons. En fait, quitte à se restreindre aux automates à jetons faibles et à considérer des semi-anneaux commutatifs, on peut montrer (cf. [BGMZ10]) que tout automate à r jetons faibles est équivalent à un automate imbriqué de rang r (et la traduction est effective) : il s'agit d'un raffinement de la preuve que les automates boustrophédons pondérés sont équivalents aux automates pondérés simple sens.

Cependant, on ne sait pas, à l'heure actuelle si le modèle d'automates à jetons faibles est plus restrictif que celui à jetons forts. En particulier, dans notre objectif de caractériser logiquement la classe $\mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle$, cela nous force à restreindre l'opérateur de clôture transitive.

Définition 3.8.

Étant donné un entier $N \geq 1$, on introduit l'opérateur de clôture transitive bornée $N\text{-TC}^{<}_{xy}\varphi$ dont la sémantique est déduite de l'égalité

$$N\text{-TC}^{<}_{xy}\varphi = \text{TC}^{<}_{xy}\big((y \leq x + N) \wedge \varphi\big)$$

On restreint ainsi les sauts entre étapes de clôture transitive à être bornés par une constante. On définit alors la logique $\text{wFO+BTC}^{<}(\mathbb{S}, \Sigma)$ grâce à la grammaire

$$\varphi ::= s \mid P_a(x) \mid x \leq y \mid \neg\alpha \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \varphi \mid N\text{-TC}^{<}_{xy}\varphi$$

avec $s \in \mathbb{S}$, $a \in \Sigma$, $x, y \in \text{Var}$, $\alpha \in \text{bFO}$ et N décrivant tous les entiers naturels non nuls. Une série formelle $f \in \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$ est dite $\text{wFO+BTC}^{<}$-définissable s'il existe une formule φ de $\text{wFO+BTC}^{<}(\mathbb{S}, \Sigma)$ telle que pour tout mot $u \in \Sigma^*$, $f(u) = [\![\varphi]\!](u)$. Finalement, on note $\mathbb{S}^{\text{wFO+BTC}^{<}}\langle\langle\Sigma^*\rangle\rangle$ l'ensemble des séries $\text{wFO+BTC}^{<}$-définissables.

On peut montrer que les automates à jetons faibles sont clos par clôture transitive bornée. Dans le modèle d'automates imbriqués, ce résultat devient

Proposition 3.2. $\mathbb{S}^{\text{wFO+BTC}^{<}}\langle\langle\Sigma^*\rangle\rangle \subseteq \mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle$

Démonstration. En annexe, page 24 □

L'autre inclusion est également vraie : on la prouve en étendant par récurrence le résultat de la proposition 3.1, en remarquant qu'on peut reproduire la même preuve en bornant les clôtures transitives utilisées par l'entier $2n$.

Proposition 3.3. $\mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle \subseteq \mathbb{S}^{\text{wFO+BTC}^{<}}\langle\langle\Sigma^*\rangle\rangle$

Démonstration. En annexe, page 25. □

Finalement, grâce aux deux propositions précédentes, on a prouvé

Théorème 3.2. $\mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle = \mathbb{S}^{\text{wFO+BTC}^{<}}\langle\langle\Sigma^*\rangle\rangle$

3.3.3 Conséquences pour la décidabilité

À ce point des résultats, on a donc la capacité de considérer une requête du premier ordre, avec éventuellement l'usage de la clôture transitive bornée, (ou d'une requête wCondXPath sur les mots) et de la traduire dans le modèle des automates imbriqués. Cela permet de résoudre certains problèmes de décision sur les langages de requête par des méthodes de graphes.

Par exemple, considérons une formule φ, dont on veut savoir si elle est satisfaisable (i.e. si le support, qui est l'ensemble des mots de poids non nul, est non vide). Supposons de plus que le semi-anneau est positif, au sens où pour tout $s, s' \in \mathbb{S}$, si $s \oplus s' = \mathbf{0}$ alors $s = s' = \mathbf{0}$ et si $s \odot s' = \mathbf{0}$ alors $s = \mathbf{0}$ ou $s' = \mathbf{0}$. Par exemple, les semi-anneaux **Nat**, **Prob**, **Bool** et **Trop** sont positifs alors que **Int** ne l'est pas. On est alors assuré qu'un mot w possède un poids non nul dans un automate imbriqué donné, si et seulement s'il existe un calcul de poids non nul étiqueté par ce mot, i.e. si et seulement s'il existe un calcul étiqueté par ce mot n'empruntant que des transitions de poids non nul. Dans ce cas, on peut se ramener facilement au cas d'un automate imbriqué sur le semi-anneau booléen en remplaçant chaque poids non nul par $\top$ et chaque poids nul par $\bot$: or les automates imbriqués sur le semi-anneau booléen reconnaissent les langages réguliers (et la traduction est effective), et le problème du vide d'un langage régulier est décidable.

L'extension des questions de décidabilité à des semi-anneaux non positifs est une perspective future de recherche.

4 Extension des résultats aux arbres

Afin d'étendre les résultats précédents aux arbres, rappelons brièvement les qualités utilisées dans les preuves pour permettre une traduction de la logique vers les automates.

Toutes les preuves sont dans un premier temps basées sur l'aspect *navigationnel* des automates considérés : sur les arbres, cela se traduit par le fait qu'il nous faut préférer des modèles d'automates cheminant dans les arbres (*tree-walking automata* et ses extensions) aux classiques automates d'arbres (*top-down automata*) qui sont intrinsèquement des modèles parallèles.

Rappelons dans un second temps que nous avons introduit des modèles restreints d'automates à jetons qui conservent une vision naturelle simple sens : même si les automates imbriqués visitent plusieurs fois chaque position du mot (et même un nombre non borné de fois), ils ne réalisent que des parcours de gauche à droite du mot. Cela nous permet d'éviter le problème de la définition d'une sémantique des automates boustrophédons qui nécessite la notion épineuse de calcul simple.

Finalement, ce qui permet d'obtenir une équivalence entre une logique contenant wFO et un opérateur de clôture transitive, est son lien fort avec la notion d'emboîtement développée dans les automates imbriquées.

On va donc réutiliser dans le cas des arbres ces trois ingrédients principaux afin d'étendre le plus naturellement possible les résultats présentés dans le cas des mots.

4.1 Automates d'arbres pondérés navigants et restriction au parcours en profondeur

La vision navigationnelle des automates boustrophédons dans le cas des arbres a été introduite par Aho et Ullman avec les automates d'arbres cheminants ou *tree-walking automata* (on pourra se référer à l'étude de [Boj08] pour un résumé des différents résultats). Une extension quantitative assez naturelle a récemment été introduite dans [FM09]. Ainsi, en suivant l'intuition des automates à jetons dans le cas des mots, on peut étendre au cas quantitatif les automates cheminant à jeton ou *tree-walking pebble automata* présentés dans [EH99] dans le cas booléen. Il

n'est pas difficile de se persuader alors qu'on peut exprimer toute formule du premier ordre, ainsi qu'un opérateur de clôture transitive, à l'aide de ce formalisme. Cependant, jusqu'à maintenant, nous n'avons pu exhiber aucune réciproque : aucune logique n'exprime exactement le pouvoir de tels automates. Le problème principal se trouve, encore une fois, dans cette notion de calcul simple. Afin d'éviter ce problème, nous allons donc fortement restreindre la classe des automates navigants sur les arbres, en les forçant à parcourir l'arbre à l'aide d'un parcours en profondeur (parcours que savent faire les classiques automates navigants, cf. exemple 1 de [Boj08]).

Dans toute cette section, on se limite à l'étude d'arbres binaires. En particulier, $\mathcal{T}_\Sigma$ désigne désormais l'ensemble des arbres binaires sur l'alphabet Σ. Les résultats se généralisent facilement au cas des arbres d'arité bornée.

Définition 4.1.

Un wDFS sur $\mathbb{S}$ et Σ est un quadruplet (Q, λ, μ, ν) tel que :
- Q est un ensemble fini d'états,
- $\mu : \Sigma \to S^{Q\times D\times Q}$ est la fonction de transitions pondérées, avec $D = \{\rightarrow, \downarrow, \uparrow\}$,
- $\lambda \in S^{1\times Q}$ est le vecteur ligne de poids initial,
- $\nu \in S^{Q\times 1}$ est le vecteur colonne de poids terminal.

On donne la sémantique des wDFS à l'aide de calculs qui suivent un parcours en profondeur, ce qui contraste avec la définition matricielle donnée pour les automates à poids sur les mots (définition 3.2). On code ce parcours en profondeur en ajoutant à chaque nœud trois balises $\{W, S, E\}$: elles encodent le fait qu'on visite ce nœud pour la première fois pour la balise W (on y entre), pour la deuxième fois pour la balise S (on a déjà visité le fils gauche) ou pour la troisième fois pour la balise E (on a visité les deux fils, et on sort de ce sous-arbre). Cependant, on s'autorise des coupes dans le parcours de l'arbre, comme le formalise la définition suivante.

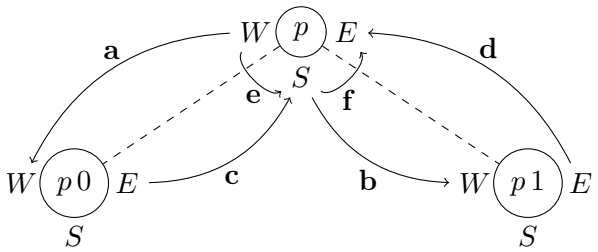


FIGURE 2 – Différentes transitions possibles dans un wDFS entre un nœud p et ses deux enfants

Définition 4.2.

La sémantique d'un wDFS est une série formelle de $\mathbb{S}\langle\langle\mathcal{T}_\Sigma\rangle\rangle$. Soit $\mathcal{A} = (Q, \lambda, \mu, \nu)$ un wDFS sur $\mathbb{S}$ et Σ. Soit $t \in \mathcal{T}_\Sigma$. Un calcul ρ de $\mathcal{A}$ sur t est une suite de configurations $(u_i, q_i, \beta_i)_{0\leq i\leq n}$ avec $u_i \in \mathrm{dom}(t)$, $q_i \in Q$ et $\beta_i \in \{W, S, E\}$ pour tout i vérifiant les propriétés suivantes
- $u_0 = \varepsilon$, $\beta_0 = W$: on débute à la racine,
- $u_n = \varepsilon$, $\beta_n = E$: on termine à la racine,
- pour tout $i \in \{0, \ldots, n-1\}$, l'un des cas suivants est vérifié (cf. la figure 2)
  - **a.** $u_{i+1} = u_i\,0$ et $\beta_i = \beta_{i+1} = W$: on pose $d(u_i, u_{i+1}) = \downarrow$
  - **b.** $u_{i+1} = u_i\,1$, $\beta_i = S$ et $\beta_{i+1} = W$: on pose $d(u_i, u_{i+1}) = \downarrow$
  - **c.** $u_i = u_{i+1}\,0$, $\beta_i = E$ et $\beta_{i+1} = S$: on pose $d(u_i, u_{i+1}) = \uparrow$
  - **d.** $u_i = u_{i+1}\,1$ et $\beta_i = \beta_{i+1} = E$: on pose $d(u_i, u_{i+1}) = \uparrow$
  - **e.** $u_i = u_{i+1}$ et $\beta_{i+1} = S$ et $\beta_i = W$: on pose $d(u_i, u_{i+1}) = \rightarrow$
  - **f.** $u_i = u_{i+1}$ et $\beta_{i+1} = E$ et $\beta_i = S$: on pose $d(u_i, u_{i+1}) = \rightarrow$

On associe à un tel calcul ρ son poids $\pi(\rho)$ étant le produit des poids des transitions empruntées :

$$\pi(\rho) \stackrel{\text{déf}}{=} \lambda_{q_0} \odot \left[\bigodot_{i=1}^{n} \mu(t(u_{i-1}))_{q_{i-1},d(u_{i-1},u_i),q_i}\right] \odot \nu_{q_n}$$

Finalement, à un arbre t, on associe le poids $[\![\mathcal{A}]\!](t)$, étant la somme des poids des calculs de $\mathcal{A}$ sur t.

Exemple 4.1. Soit $\Sigma = \{\wedge, \vee, \top, \bot\}$.

- Dans le semi-anneau **Bool**, on peut construire un wDFS reconnaissant exactement l'ensemble des arbres binaires, représentant une formule close avec les opérateurs $\wedge$ et $\vee$, qui s'évaluent à $\top$ à la racine. Durant le parcours en profondeur, il maintient dans l'état, la valuation courante du sous-arbre relatif à la position courante. Lorsqu'il remonte dans l'arbre grâce à une transition de type **c**, si l'état courant contient $\top$ et que l'étiquette est $\vee$, il n'explore pas le sous-arbre droit, et emprunte une transition de type **f** (même chose si l'état courant contient $\bot$ et que l'étiquette est $\wedge$). Dans les autres cas, au contraire, l'automate *oublie* le résultat de l'exploration du sous-arbre gauche et entame l'exploration du sous-arbre droit (grâce à une transition de type **b**). Remarquons que ce langage ne peut pas être accepté par un wDFS si on n'introduit pas les transitions de type **f**.
- Dans le semi-anneau **Trop**, on peut de plus faire calculer à ce wDFS le coût d'une telle exploration.

Afin de décrire le comportement de cette classe d'automates *simple sens* sur des arbres, il est naturel de généraliser le résultat de la proposition 3.1 en définissant une clôture transitive *simple sens*, qu'on borne d'ores et déjà.

Définition 4.3.

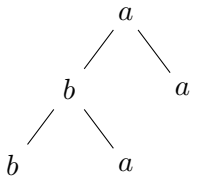
On introduit l'opérateur de clôture transitive bornée $N\text{-TC}^{\lhd}_{xy}\varphi$ pour chaque entier $N \geq 1$. Pour donner sa sémantique, on commence par linéariser les arbres de $\mathcal{T}_\Sigma$ à l'aide d'un ordre $\lhd$ sur leurs nœuds annotés par une balise. Soit $t \in \mathcal{T}_\Sigma$. On définit ainsi l'ordre $\lhd$ sur les éléments de $\mathrm{dom}(t) \times \{W, S, E\}$ comme la fermeture par clôture transitive de

$$(u, W) \lhd (u0, W), \quad (u0, E) \lhd (u, S) \lhd (u1, W), \quad (u1, E) \lhd (u, E)$$
$$\text{et} \quad (u, W) \lhd (u, S) \lhd (u, E)$$

pour tout $u \in \mathrm{dom}(t)$. On note $d_{\mathrm{dfs}}((u, \alpha), (v, \beta))$ la distance minimale dans l'ordre $\lhd$ séparant les nœuds (u, α) et (v, β). Finalement, on définit la sémantique de $N\text{-TC}^{\lhd}_{xy}\varphi$ par

$$[\![[N\text{-TC}^{\lhd}_{xy}\varphi](z_1, z_2)]\!](t, [z_1 \mapsto u, z_2 \mapsto v])$$
$$= \bigoplus_{n \geq 0} \bigoplus_{\substack{(u,W)=(w_0,\alpha_0) \lhd \ldots \lhd (w_n,\alpha_n)=(v,E) \\ \forall i \; d_{\mathrm{dfs}}((w_i,\alpha_i),(w_{i+1},\alpha_{i+1})) \leq N}} \bigodot_{i=1}^{n} [\![\varphi(x, y)]\!](t, [x \mapsto w_{i-1}, y \mapsto w_i])$$

Exemple 4.2. Dans l'arbre de la figure ci-contre, on voit que $d_{\mathrm{dfs}}((\varepsilon, W), (01, S)) = 4$, puisqu'on a le chemin suivant dans l'ordre $\lhd$: $(\varepsilon, W) \lhd (0, W) \lhd (0, S) \lhd (01, W) \lhd (01, S)$.



Comme dans le cas des mots, on note $\mathrm{BTC}^{\lhd}_{\mathrm{step}}(\mathrm{bFO{+}mod})$ l'ensemble des formules de la forme $N\text{-TC}^{<}_{xy}\varphi$ avec $N \geq 1$ et φ une formule bFO-étagée pouvant utiliser de plus les prédicats unaires $x \equiv_\ell m$ pour tout $\ell \in \mathbb{N}$ et $1 \leq m \leq \ell$ (sa sémantique est donnée par $[\![x \equiv_\ell m]\!](t, \sigma) = \mathbf{1}$

si la profondeur du nœud $\sigma(x)$ est congrue à m modulo ℓ et $\mathbf{0}$ sinon). On dit qu'une série $f \in \mathbb{S}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$ est $\mathrm{BTC}^{\triangleleft}_{\mathrm{step}}$(bFO+mod)-définissable, s'il existe $N \geq 1$ et une formule $\varphi(x,y)$ sans paramètre telle que pour tout arbre t, $f(t) = [\![N\text{-}\mathrm{TC}^{<}_{xy}\varphi]\!](t,\varepsilon,\varepsilon)$.

Théorème 4.1. *Soit $\mathbb{S}$ un semi-anneau et soit $f \in \mathbb{S}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$. Si f est une série reconnaissable par un* wDFS*, alors f est* $\mathrm{BTC}^{\triangleleft}_{\mathrm{step}}$(bFO+mod)*-définissable.*

Démonstration. En annexe, page 26. □

4.2 Imbrication et lien avec la logique

Comme dans le cas des mots (définition 3.8), on considère alors la logique $\mathrm{wFO{+}BTC}^{\triangleleft}(\mathbb{S},\Sigma)$ contenant la logique du premier ordre agrémentée de l'ensemble des opérateurs de clôture transitive bornée (pour $N \geq 1$). Une série formelle $f \in \mathbb{S}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$ est dite $\mathrm{wFO{+}BTC}^{\triangleleft}$-définissable s'il existe une formule φ de $\mathrm{wFO{+}BTC}^{\triangleleft}(\mathbb{S},\Sigma)$ telle que pour tout arbre $t \in \mathcal{T}_\Sigma$, $f(t) = [\![\varphi]\!](t)$. Finalement, on note $\mathbb{S}^{\mathrm{wFO{+}BTC}^{\triangleleft}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$ l'ensemble des séries $\mathrm{wFO{+}BTC}^{\triangleleft}$-définissable.

Du fait de la forme simple sens des wDFS, on peut transposer au cas des arbres la définition 3.7 des automates imbriqués sur les mots.

Définition 4.4.

Pour $r \geq 0$, on définit une classe r-nDFS(Σ) d'automates imbriqués de rang r sur Σ (et $\mathbb{S}$). Un automate imbriqué de rang 0 sur Σ est un wDFS sur $\mathbb{S}$ et Σ. Pour $r > 0$, un automate imbriqué de rang r sur Σ est un quadruplet $\mathcal{A} = (Q, \lambda, \mu, \nu)$ où

- Q est un ensemble fini d'états,
- $\mu : Q \times \Sigma \times D \times Q \to (r-1)$-nDFS($\Sigma \times \{0,1\}$) est la fonction de transitions,
- $\lambda \in S^{1\times Q}$ est le vecteur ligne de poids initial,
- $\nu \in S^{Q\times 1}$ est le vecteur colonne de poids final.

Définissons la sémantique $[\![\mathcal{A}]\!] \in \mathbb{S}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$ de $\mathcal{A} = (Q, \lambda, \mu, \nu) \in r$-nDFS($\Sigma$). Si $r = 0$, alors la série formelle $[\![\mathcal{A}]\!]$ est définie comme dans le cas des wDFS. Pour $r > 0$, un calcul de $\mathcal{A}$ sur un arbre t est une suite de configurations $(u_i, q_i, \alpha_i)_{0\leq i\leq n}$ respectant les conditions de la définition 4.2. Le poids de ce calcul est défini par

$$\pi(\rho) \stackrel{\text{déf}}{=} \lambda_{q_0} \odot \left[\bigodot_{i=1}^{n} [\![\mu(q_{i-1}, t(u_{i-1}), d(u_{i-1}, u_i), q_i)]\!](t, u_{i-1})\right] \odot \nu_{q_n}$$

où $(t, u_i) \in \mathcal{T}_{\Sigma\times\{0,1\}}$ est l'arbre τ de domaine $\mathrm{dom}(t)$ tel que pour tout $u \in \mathrm{dom}(t)$, $\tau(u) = (t(u), 0)$ si $u \neq u_i$ et $\tau(u_i) = (t(u_i), 1)$. On définit alors $[\![\mathcal{A}]\!](t)$ comme la somme (finie) des poids des calculs de $\mathcal{A}$ sur t.

Exemple 4.3. Sur le semi-anneau **Nat**, la requête $\psi = \mathrm{root} \wedge [\langle\downarrow^*\rangle \mathrm{leaf}](\langle(\uparrow/2?)^*\rangle \mathrm{root})$ sélectionne la racine d'un arbre t et lui associe le poids 2^d avec d la somme des profondeur des feuilles de t. La série associée (qui associe le poids précédent à un arbre donné) est reconnaissable par le 1-nDFS qui marque successivement chaque feuille de l'arbre, et lance pour chacune d'entre elles un wDFS qui calcule 2^p, avec p la profondeur de la feuille marquée.

Finalement, on note $\mathbb{S}^{\mathrm{nDFS}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$ la classe des séries formelles reconnaissables par un nDFS. On peut alors obtenir le résultat d'expressivité suivant :

Théorème 4.2. $\mathbb{S}^{\mathrm{nDFS}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle = \mathbb{S}^{\mathrm{wFO{+}BTC}^{\triangleleft}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$

Démonstration. En annexe, page 27 □

Tout comme dans le cas des mots, on peut déduire de ce théorème des procédures de décision pour des requêtes de $\mathrm{wFO{+}BTC}^{\triangleleft}$, et même de wCondXPath.

Conclusion et ouverture

Comme on l'a vu dans ce rapport, la définition, l'exploration et la vérification de langages de requêtes quantitatives débutent à peine. Dans une volonté d'étendre des résultats déjà vrais dans le cas booléen, il est intéressant de considérer des machines séquentielles telles que les automates, et de les étendre au domaine quantitatif. Cependant, des difficultés et des limitations propres au cas quantitatif entrent vite en jeu : impossibilité de calculer des grandes valeurs, sémantique délicate à définir dans le cas d'automates à double sens...

On a ainsi réussi à apporter une sémantique satisfaisante à une version quantitative de CondXPath, et à traduire ce langage de requêtes dans les formules du premier ordre, puis dans un modèle d'automates d'arbres cheminants avec jetons, avec une restriction imitant l'atout simple sens du cas des mots. Il reste à explorer les propriétés de tels automates plus en détail, afin de fournir des résultats de décidabilité et de complexité aux problèmes naturels qu'on peut se poser dans wCondXPath.

Comme on l'a brièvement mentionné au cours de ce rapport, une piste naturelle à explorer dans le futur, dans le cas des arbres, serait d'utiliser des extensions quantitatives des automates cheminants à jetons afin d'exprimer facilement les quantifications du premier ordre. Le défaut majeur actuel du modèle tel qu'on l'a défini pendant le stage est qu'on ne parvient pas à trouver une classe de logique qui exprime exactement le pouvoir de ces automates. Par ailleurs, ce modèle permettrait facilement de considérer des arbres d'arité non bornée, ce qui est particulièrement intéressant par rapport aux motivations liées aux documents XML.

On peut remarquer que les modèles d'automates double-sens et leur généralisation avec jetons sont problématiques dans le cadre quantitatif, du fait des boucles de poids non nul qui entraînent potentiellement des poids non définis. Une manière originale d'effacer ce problème peut être de se placer dans un semi-anneau complet (tel que le semi-anneau des probabilités **Prob**) : cela nous amènerait à considérer des méthodes totalement différentes de celles développées dans ce rapport, telles qu'elles ont récemment été appliquées dans [BG09].

Finalement, une piste intéressante, afin d'obtenir un grand nombre de procédures de décision supplémentaires, serait d'*algébriser* les modèles d'automates considérés. En effet, en s'inspirant de la présentation faite dans [BR10], sur des semi-anneaux qu'on peut inclure dans des corps, on peut décider (ou prouver l'indécidabilité) de nombreux problèmes en considérant des propriétés purement algébriques des automates pondérés. Dans le cadre double-sens, de telles propriétés n'ont pas encore été découvertes, à notre connaissance. Cela constituerait une avancée importante, tant pour la définition de la sémantique de ces automates, que pour l'application à la vérification.

Références

[BG09] B. Bollig and P. Gastin. Weighted versus probabilistic logics. In *Proceedings of the 13th International Conference on Developments in Language Theory (DLT'09)*, volume 5583 of *LNCS*, pages 18–38. Springer, 2009.

[BGMZ10] B. Bollig, P. Gastin, B. Monmege, and M. Zeitoun. Pebble weighted automata and transitive closure logics. In *Proceedings of ICALP'10*, pages 587–598. Springer, 2010. Rapport de recherche disponible à http://www.lsv.ens-cachan.fr/Publis/RAPPORTS_LSV/rapports.

[Boj08] M. Bojańczyk. Tree-walking automata. In *LATA'08*, volume 5196 of *LNCS*, pages 1–17. Springer, 2008.

[BR10] Jean Berstel and Christophe Reutenauer. *Noncommutative Rational Series With Applications*. Disponible en ligne sur http://www-igm.univ-mlv.fr/%7Eberstel/LivreSeries/RationalSeries9April2010.pdf, 2010.

[BSSS06] M. Bojańczyk, M. Samuelides, T. Schwentick, and L. Segoufin. Expressive power of pebble automata. In *Automata, Languages and Programming*, volume 4051 of *LNCS*, pages 157–168. Springer, 2006.

[DG07] M. Droste and P. Gastin. Weighted automata and weighted logics. *Theoretical Computer Science*, 380(1-2) :69–86, 2007.

[DG09] M. Droste and P. Gastin. Weighted automata and weighted logics. In W. Kuich, H. Vogler, and M. Droste, editors, *Handbook of Weighted Automata*, EATCS Monographs in Theoret. Comput. Sci. Springer, 2009. To appear.

[EH99] J. Engelfriet and H. J. Hoogeboom. Tree-walking pebble automata. In *Jewels Are Forever, Contributions to Theoretical Computer Science in Honor of Arto Salomaa*, pages 72–83. Springer, 1999.

[EH07] J. Engelfriet and H. J. Hoogeboom. Automata with nested pebbles capture first-order logic with transitive closure. *Log. Meth. in Comput. Sci.*, 3, 2007.

[FM09] Z. Fülöp and L. Muzamel. Weighted Tree-Walking Automata. *Acta Cybernetica*, 19(2) :275 – 293, 2009.

[GH96] N. Globerman and D. Harel. Complexity results for two-way and multi-pebble automata and their logics. *Theoretical Computer Science*, 169 :161–184, 1996.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, 1979.

[KVD09] W. Kuich, H. Vogler, and M. Droste, editors. *Handbook of Weighted Automata*. EATCS Monographs in Theoret. Comput. Sci. Springer, 2009.

[Mar04] Maarten Marx. Conditional XPath, the first order complete XPath dialect. In *PODS '04 : Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 13–22. ACM, 2004.

[NS03] F. Neven and T. Schwentick. On the power of tree-walking automata. *Information and Computation*, 183(1) :86–103, 2003.

[Sak03] J. Sakarovitch. *Éléments de théorie des automates*. Vuibert Informatique, 2003.

[Sch61] M. P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4 :245–270, 1961.

[tCS10] B. ten Cate and L. Segoufin. Transitive closure logic, nested tree walking automata, and XPath. *J. ACM*, 2010. To appear. Short version in PODS'08.

[Tho82] W. Thomas. Classifying regular events in symbolic logic. *Journal of Computer and System Sciences*, 25(3) :360–376, 1982.

Annexes

Proposition (3.1). *Soit $\mathbb{S}$ un semi-anneau et soit $f \in \mathbb{S}\langle\langle\Sigma^*\rangle\rangle$. Si f est une série reconnaissable,* i.e.*, $f \in \mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$, alors f est $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod})$-définissable.*

Démonstration. Soit $\mathcal{A} = (Q, \lambda, \mu, \nu)$ un automate à poids reconnaissant f, avec $Q = \{1, \dots, n\}$. Pour $d \geq 1$ et $p, q \in Q$, on définit une formule $\psi^d_{p,q}(x)$ bFO-étagée qui calcule le poids du facteur de longueur d qui débute en position x (si un tel facteur existe), lorsque l'automate $\mathcal{A}$ part dans l'état p et arrive dans l'état q :

$$\psi^d_{p,q}(x) \stackrel{\text{déf}}{=} \bigvee_{v=v_1\cdots v_d} \Big(\mu(v)_{p,q} \wedge \bigwedge_{1\leq i\leq d} P_{v_i}(x+i-1)\Big)$$

Plus formellement, pour tout mot u et pour toutes positions i, j dans le mot, on a

$$[\![\psi^d_{p,q}(x)]\!](u,i) = \begin{cases} \mu(u[i..i+d-1])_{p,q} & \text{si } i+d-1 \leq |u|, \\ \mathbf{0} & \text{sinon.} \end{cases} \tag{2}$$

Pour montrer le théorème, il s'agit de construire une formule $\varphi(x, y)$ sans paramètre de bFO+mod, telle que $[\![\mathcal{A}]\!](u) = [\![\mathrm{TC}^{<}_{xy}\varphi]\!](u, 1, |u|)$. L'idée, inspirée par [Tho82], consiste à forcer la clôture transitive à emprunter les points intermédiaires $z_\ell = \ell n + q_\ell$, avec $1 \leq q_\ell \leq n$, pour les valeurs successives de ℓ, afin de coder l'ensemble des calculs de $\mathcal{A}$ empruntant l'état q_ℓ juste avant de lire la lettre en position $\ell n + 1$. Afin de rendre cela également possible pour $\ell = 0$, on suppose dans la suite, sans perte de généralité, que $\lambda(1) = \mathbf{1}$ et $\lambda(q) = \mathbf{0}$ pour $q \neq 1$, *i.e.*, l'unique état initial ayant un poids non nul est $q_0 = 1$.

On considère donc des tranches $[\ell n + 1, (\ell + 1)n]$ de positions dans le mot sur lequel on souhaite évaluer la formule (la dernière tranche pouvant être incomplète, selon la taille du mot). Chaque position y est située dans exactement une telle tranche. On peut donc écrire $\langle y\rangle = \ell n + 1$ pour la première position de la tranche abritant y, de même que $[y] \stackrel{\text{déf}}{=} y + 1 - \langle y\rangle \in Q$ pour le *décalage* correspondant dans la tranche. Notons que, pour $q \in Q$, $[y] = q$ peut être exprimé dans bFO+mod par la formule $y \equiv_n q$. Ainsi, on utilisera librement $[y]$ et $\langle y\rangle$ (qui vaut $y + 1 - [y]$) comme des macros dans les formules.

La formule $\mathrm{TC}^{<}_{xy}\varphi$ va passer par les positions x et y marquées $\bullet$ dans la Figure 3, et calculer le poids du facteur de longueur n entre les positions $\langle x\rangle$ et $\langle y\rangle - 1$, en empruntant respectivement les états $[x]$ et $[y]$ juste avant ces positions.

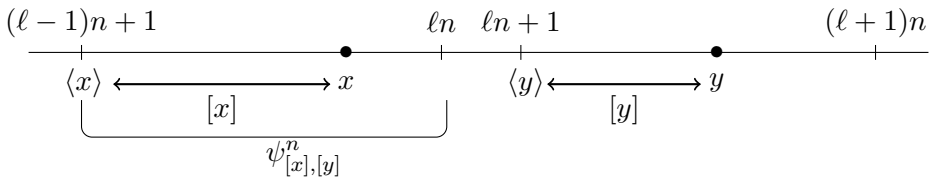


FIGURE 3 – Positions empruntées par la formule $\mathrm{TC}^{<}_{xy}\varphi$

Pour calculer de tels poids, il faut donc utiliser des formules de la forme $\psi^d_{p,q}(\langle x\rangle)$. Cependant, celles-ci ne sont pas formellement des formules bFO-étagées, à cause de la quantification existentielle du premier ordre cachée utilisée dans la définition de $\langle x\rangle$. On remédie à ce problème en construisant des formules bFO-étagées $\theta^d_{p,q}(x)$ telles que $[\![\theta^d_{p,q}(x)]\!] = [\![\psi^d_{p,q}(\langle x\rangle)]\!]$:

$$\theta^d_{p,q}(x) \stackrel{\text{déf}}{=} \bigvee_{v=v_1\cdots v_d} \Big(\mu(v)_{p,q} \wedge \bigwedge_{1\leq i\leq d} P_{v_i}(\langle x\rangle + i - 1)\Big)$$

Cette écriture a pour effet de transférer les quantifications existentielles cachées dans la partie booléenne de la formule, là où elles ne connaissent pas de restriction.

On construit finalement la formule φ en distinguant les cas où x est proche ou loin de la dernière position du mot. À cette fin, on utilise last comme une macro (définissable en bFO) désignant la dernière position du mot.

$$\varphi(x,y) = (\langle x\rangle + 2n \leq \text{last}) \wedge (\langle y\rangle = \langle x\rangle + n) \wedge \Big(\bigvee_{q_1,q_2\in Q} [x] = q_1 \wedge [y] = q_2 \wedge \theta^n_{q_1,q_2}(x)\Big)$$
$$\vee\, (\langle x\rangle + 2n > \text{last}) \wedge (y = \text{last}) \wedge \Big(\bigvee_{\substack{q_1,q_2\in Q\\ 1\leq d\leq 2n}} [x] = q_1 \wedge y - \langle x\rangle + 1 = d \wedge \theta^d_{q_1,q_2}(x) \wedge \nu(q_2)\Big)$$

La définition implique que pour tout mot u et pour toutes positions i, j,

$$[\![\varphi(x,y)]\!](u,i,j) = \begin{cases} \mu(u[\langle i\rangle..\langle j\rangle - 1])_{[i],[j]} & \text{si } \langle j\rangle = \langle i\rangle + n \wedge \langle i\rangle + 2n \leq \text{last},\\ \delta^{[i]} \odot \mu(u[\langle i\rangle..j]) \odot \nu & \text{si } j = \text{last} \wedge \langle i\rangle + 2n > \text{last},\\ \mathbf{0} & \text{sinon,}\end{cases} \tag{3}$$

où δ^ℓ est le vecteur ligne avec un $\mathbf{1}$ en position ℓ et $\mathbf{0}$ partout ailleurs. Ainsi, par définition de l'opérateur $\mathrm{TC}^<_{xy}$,

$$[\![\mathrm{TC}^<_{xy}\varphi(x,y)]\!](u,1,|u|) = \bigoplus_{m\geq 1} [\![\varphi^m(x,y)]\!](u,1,|u|)$$

On va montrer qu'il y a une unique valeur non nulle dans cette somme, qui est $[\![\varphi^m(x,y)]\!](u,1,|u|)$ avec $m = \max(1, \lfloor \frac{|u|-1}{n} \rfloor)$.

Soit $1 = i_0, \ldots, i_m = |u|$ une suite de positions choisies comme valuation des variables $z_0, \ldots, z_m$ dans la formule (1), menant à une valeur non nulle de $[\![\varphi^m(x,y)]\!](u,1,|u|)$. Le premier cas de (3) implique que $\langle i_\ell\rangle = \langle i_{\ell-1}\rangle + n$ pour tout $\ell \in \{1,\ldots,m-1\}$, de telle sorte que $\langle i_\ell\rangle = \ell n + 1$ si $\ell < m$.

- **1er cas** : $|u| \leq 2n$. On conclue par (1) que $[\![\varphi^1(x,y)]\!](u,1,|u|) = [\![\varphi(x,y)]\!](u,1,|u|) = \lambda \odot \mu(u) \odot \nu$ (en appliquant le second cas de (3)). De plus, $[\![\varphi^m(x,y)]\!](u,1,|u|) = \mathbf{0}$ pour $m \geq 2$, puisque si $1 < i_1 < |u|$ alors $[\![\varphi(x,y)]\!](u,i_0,i_1) = \mathbf{0}$ (le premier cas de (3) est exclus puisque $\langle i_0\rangle + 2n > |u|$ et le second cas aussi puisque $i_1 \neq |u|$). Finalement, on trouve

$$[\![\mathrm{TC}^<_{xy}\varphi(x,y)]\!](u,1,|u|) = \lambda \odot \mu(u) \odot \nu = [\![\mathcal{A}]\!](u)$$

- **2ème cas** : $|u| > 2n$. Alors $[\![\varphi(x,y)]\!](u,1,|u|) = \mathbf{0}$. Ainsi, $m \geq 2$ et $\langle i_\ell\rangle = \ell n + 1$ pour $\ell < m$. Par (3), $[\![\varphi^m(x,y)]\!](u,1,|u|) \neq \mathbf{0}$ implique que $\langle i_{m-2}\rangle + 2n \leq |u| < \langle i_{m-1}\rangle + 2n$, ce qu'on peut réécrire $m = \lfloor \frac{|u|-1}{n} \rfloor$. On en déduit alors que $[\![\mathrm{TC}^<_{xy}\varphi(x,y)]\!](u,1,|u|) = [\![\varphi^m(x,y)]\!](u,1,|u|)$ pour $m = \lfloor \frac{|u|-1}{n} \rfloor$. La valuation $1 = i_0 < i_1 < \ldots < i_m = |u|$ est uniquement définie par la suite des décalages $([i_1], \ldots, [i_{m-1}]) \in Q^{m-1}$, puisqu'alors $i_\ell = \langle i_\ell\rangle - 1 + [i_\ell] = \ell n + [i_\ell]$ pour $\ell < m$.
  Les choix possibles pour cette séquence induisent une somme dans l'évaluation suivante de $[\![\varphi^m(x,y)]\!](u,1,|u|)$, dans laquelle on renomme la suite $([i_1], \ldots, [i_{m-1}])$ par le vecteur $\overrightarrow{q} = (q_1, \ldots, q_{m-1})$. On rappelle qu'on a supposé que $q_0 = 1$ est l'unique état initial de

poids non nul.

$$
\begin{aligned}
&[\![\varphi^m(x,y)]\!](u,1,|u|)\\
&\quad= \bigoplus_{\overrightarrow{q}\in Q^{m-1}} \Big[\bigodot_{\ell=1}^{m-1} [\![\varphi]\!]\big(u,(\ell-1)n+q_{\ell-1},\ell n+q_\ell-1\big)\Big] \odot [\![\varphi]\!](u,(m-1)n+q_{m-1},|u|)\\
&\quad= \bigoplus_{\overrightarrow{q}\in Q^{m-1}} \Big[\bigodot_{\ell=1}^{m-1} \mu\big(u[(\ell-1)n+1..\ell n]\big)_{q_{\ell-1},q_\ell}\Big] \odot \big(\delta^{q_{m-1}} \odot \mu(u[(m-1)n+1..|u|]) \odot \nu\big)\\
&\quad= \delta^{q_0} \odot \mu\big(u[1..(m-1)n]\big) \odot \mu(u[(m-1)n+1..|u|]) \odot \nu\\
&\quad= \lambda \ \odot \mu(u) \odot \nu\\
&\quad= [\![\mathcal{A}]\!](u)
\end{aligned}
$$

□

Théorème (3.1). $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod}) = \mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$.

Démonstration. Afin de démontrer l'inclusion $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod}) \subseteq \mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$, on définit un modèle intermédiaire de logique, fragment d'une extension quantitative de la logique monadique du second ordre. En particulier, on s'autorise l'utilisation de variables monadiques du second ordre $\mathrm{VAR} = \{X, Y, \ldots\}$, et les quantifications existentielles sur de telles variables : il est facile de voir que les automates pondérés ne sont pas clos par quantifications universelles du second ordre, si on définit leur sémantique comme un produit sur tous les sous-ensembles de positions dans le mot. Cependant, comme on l'a vu dans la section 3.1, il est également nécessaire de restreindre l'usage des quantifications universelles du premier ordre.

On définit l'ensemble $\mathrm{RMSO}(\mathbb{S},\Sigma)$ de formules de la logique monadique du second ordre restreinte par la grammaire suivante :

$$
\begin{aligned}
\varphi ::= s \mid P_a(x) \mid \neg P_a(x) \mid x \le y \mid \neg(x \le y) \mid x \in X \mid \neg(x \in X) \mid\\
\varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x\, \varphi \mid \forall x\, \alpha \mid \exists X\, \varphi
\end{aligned}
$$

où $s \in \mathbb{S}$, $a \in \Sigma$, $x, y \in \mathrm{Var}$, $X \in \mathrm{VAR}$ et α une formule bFO-étagée.

La section 4 de [DG07] apporte alors l'égalité entre les formules $\mathrm{RMSO}(\mathbb{S},\Sigma)$-définissables et $\mathbb{S}^{\mathrm{rec}}\langle\langle\Sigma^*\rangle\rangle$.

Afin de montrer l'inclusion voulue, il suffit donc de justifier que $\mathrm{TC}^{<}_{\mathrm{step}}(\mathrm{bFO{+}mod}) \subseteq \mathrm{RMSO}(\mathbb{S},\Sigma)$, ce qui se fait aisément en exprimant la clôture transitive et le modulo par une quantification existentielle du second ordre. □

Proposition (3.2). $\mathbb{S}^{\mathrm{wFO{+}BTC}^{<}}\langle\langle\Sigma^*\rangle\rangle \subseteq \mathbb{S}^{\mathrm{nwA}}\langle\langle\Sigma^*\rangle\rangle$

Démonstration. On montre le résultat par induction sur la formule $\varphi \in \mathrm{wFO{+}BTC}^{<}(\mathbb{S},\Sigma)$. Les formules de bFO sont reconnaissables par des automates pondérés, donc par des automates imbriqués de rang 0.

Remarquons qu'une série r-nwA-reconnaissable est également $(r+1)$-nwA-reconnaissable, ce qui permet, étant donnés deux automates imbriqués, de les supposer tous les deux du même rang. La clôture par disjonction de deux automates imbriqués de rang r se réalise alors par union disjointe des deux automates, tandis que la clôture par conjonction se montre par produit synchrone (on suppose ici que le semi-anneau est commutatif : voir [BGMZ10] pour une preuve dans le cas où le semi-anneau est non commutatif).

Les clôtures par quantification se font en ajoutant un niveau d'imbrication. Soit $\mathcal{A}$ un automate imbriqué de rang r qui reconnaît la série $[\![\varphi(x)]\!]$ sur l'alphabet $\Sigma \times \{0,1\}$. On construit

un automate imbriqué $\mathcal{B}_\forall$ (resp. $\mathcal{B}_\exists$) de rang $r+1$ sur l'alphabet Σ qui reconnaît la série $[\![\forall x\ \varphi]\!]$ (resp. $[\![\exists x\ \varphi]\!]$). L'automate $\mathcal{B}_\forall$ est défini par $(\{p\}, (\mathbf{1}), \mu_\forall, (\mathbf{1}))$ avec pour toute lettre $a \in \Sigma$, $\mu_\forall(p, a, p) = \mathcal{A}$: pour chaque position du mot, il lance l'automate $\mathcal{A}$ avec la position courante marquée. L'automate $\mathcal{B}_\exists$, quant à lui, est défini par $(\{p_1, p_2\}, (\mathbf{1}\ \ \mathbf{0}), \mu_\exists, \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix})$ avec pour toute lettre $a \in \Sigma$,

$$\mu_\exists(p_1, a, p_1) = \mathbf{1} \qquad \mu_\exists(p_2, a, p_2) = \mathbf{1}$$
$$\mu_\exists(p_1, a, p_2) = \mathcal{A} \qquad \mu_\exists(p_2, a, p_1) = \mathbf{0}$$

Dans lequel, on note $k \in \mathbb{S}$ un automate pondéré dont la sémantique est la série formelle constante égale à k.

Finalement, montrons la clôture par clôture transitive bornée. Soit $\mathcal{A}$ un automate imbriqué de rang r sur l'alphabet $\Sigma \times \{0,1\}^2$ qui reconnaît la série $[\![\varphi(x,y)]\!]$. On construit un automate $\mathcal{B}$ imbriqué de rang $r+1$ sur l'alphabet $\Sigma \times \{0,1\}^2$ qui reconnaît la série $[\![[N\text{-TC}^<_{xy}\varphi](z,t)]\!]$. Cet automate possède $N+2$ états qu'on note $p_0, \ldots, p_{N-1}, q_0, q_f$. Seul l'état q_0 possède un poids initial non nul, égal à $\mathbf{1}$, et seul l'état q_f possède un poids final non nul, égal à $\mathbf{1}$ également.

- Dans une première phase, l'automate $\mathcal{B}$, tout en restant dans l'état q_0, cherche la position du mot abritant la valuation de la variable z (qui doit nécessairement apparaître avant celle de la variable t sinon la valeur renvoyée est $\mathbf{0}$) : cela est fait grâce à la transition $\mu(q_0, (a,0,0), q_0) = \mathbf{1}$.
- Dans une deuxième phase, on réalise une succession de sauts de longueur bornée par N, et on calcule le poids de ces sauts en utilisant des variantes paramétrées de l'automate $\mathcal{A}$. Remarquons ainsi qu'on peut construire un automate $\widetilde{\mathcal{A}}_k$ (avec $0 \leq k \leq N$) imbriqué de rang r sur l'alphabet $\Sigma \times \{0,1\}$ tel que $[\![\widetilde{\mathcal{A}}_k]\!](w, [x \mapsto i]) = [\![\varphi(x,y)]\!](w, [x \mapsto i, y \mapsto i+k])$. Ainsi, lorsqu'on est dans un état p_0 (ou q_0 sur la position abritant la valuation de la variable z, juste après la phase d'initialisation), on devine de façon non déterministe la distance du prochain saut, et on appelle l'automate $\widetilde{\mathcal{A}}_k$ pour calculer le poids du saut. On le fait grâce aux transitions suivantes :

  $$\mu(p_0, (a,0,0), p_k) = \widetilde{\mathcal{A}}_k \quad \text{pour } 1 \leq k \leq N$$
  $$\mu(q_0, (a,1,0), p_k) = \widetilde{\mathcal{A}}_k \quad \text{pour } 1 \leq k \leq N$$

  Entre les positions-étapes de la clôture transitive, on avance donc dans le mot en décroissant la distance contenue dans l'état :

  $$\mu(p_k, (a,0,0), p_{k-1}) = \mathbf{1} \quad \text{pour } 1 \leq k \leq N$$

- Finalement la troisième phase consiste à s'arrêter dès qu'on trouve la position abritant la valuation de t, de vérifier qu'on est dans l'état p_0 (ou q_0 auquel cas $z = t$ et on calcule le poids $[\![\varphi(z,z)]\!]$) et de parcourir la fin du mot dans l'état q_f :

  $$\mu(p_0, (a,0,1), q_f) = \mathbf{1}$$
  $$\mu(q_0, (a,1,1), q_f) = \widetilde{\mathcal{A}}_0$$
  $$\mu(q_f, (a,0,0), q_f) = \mathbf{1}$$

□

Proposition (3.3). $\mathbb{S}^{\text{nwA}}\langle\langle\Sigma^*\rangle\rangle \subseteq \mathbb{S}^{\text{wFO+BTC}^<}\langle\langle\Sigma^*\rangle\rangle$

Démonstration. On montre la propriété par récurrence sur le rang d'imbrication de l'automate.

Si $\mathcal{A}$ est un automate imbriqué de rang 0, alors la preuve de la proposition 3.1 permet de conclure, puisqu'on peut facilement remplacer l'opération de clôture transitive par $2n$-TC$^<$, étant donnée que la formule 3 interdit tout saut de taille supérieure à $2n$.

Supposons désormais que tout automate imbriqué de rang $r-1$ puisse s'exprimer par une formule de wFO+BTC$^<$(𝕊,). Soit $\mathcal{A}$ un automate imbriqué de rang r, sur l'alphabet Σ. On applique l'hypothèse de récurrence pour chaque automate $\mu(p,a,q)$ dans la définition de l'automate $\mathcal{A}$: on obtient donc un nombre fini de formules $\zeta_{p,a,q}(x)$ avec une variable libre supplémentaire x. Ces formules nous permettent d'écrire un équivalent des formules $\psi^d_{p,q}$ et $\theta^d_{p,q}$ dans lesquelles on remplace le poids $\mu(v)_{p,q}$ par les formules ζ adéquates :

$$\theta^d_{p_0,p_d}(x) \overset{\text{déf}}{=} \bigvee_{p_1,\dots,p_{d-1}\in Q} \bigwedge_{0\le i<d} \bigvee_{a\in\Sigma} \zeta_{p_i,a,p_{i+1}}(\langle x\rangle + i) \wedge P_a(\langle x\rangle + i)$$

La différence majeure se trouve dans le fait qu'on a besoin de l'ensemble des états intermédiaires pour calculer le poids du sous-mot de longueur d, puisqu'il n'y a plus de propriétés matricielles suffisantes, comme dans le cas des automates pondérés. Le reste de la preuve est exactement identique en tous points. □

Théorème (4.1). *Soit $\mathbb{S}$ un semi-anneau et soit $f \in \mathbb{S}\langle\langle\mathcal{T}_\Sigma\rangle\rangle$. Si f est une série reconnaissable par un* wDFS*, alors f est* BTC$^{\lhd}_{\text{step}}$(bFO+mod)*-définissable.*

Démonstration. On reprend les idées de la démonstration de la proposition 3.1. On note $Q = \{0,\dots,n-1\}$ les états du wDFS reconnaissant la série f : dans la suite, on suppose que n est pair. Les positions qui remplacent les positions $\ell n+1$ du mot seront ici les nœuds à profondeur multiple de $n+1$. Comme un calcul de l'automate peut passer au plus trois fois par chaque nœud p, il faut $3n$ positions pour coder les états juste avant de passer par les nœuds (p,W), (p,S) et (p,E). De plus, ces encodages doivent respecter l'ordre $\lhd$. Ainsi, pour la balise W du nœud p, on associe l'état $i \in \{0,\dots,n-1\}$ au nœud $p0^{i+1}$. Pour la balise S, on associe l'état $i \in \{0,\dots,n/2-1\}$ au nœud $p01^{n/2-1-i}$ et l'état $i \in \{n/2,\dots,n-1\}$ au nœud $p10^{i-n/2}$. Enfin, pour la balise E, on associe l'état $i \in \{0,\dots,n-1\}$ au nœud $p1^{n-1-i}$. On a résumé cette définition dans la figure 4.

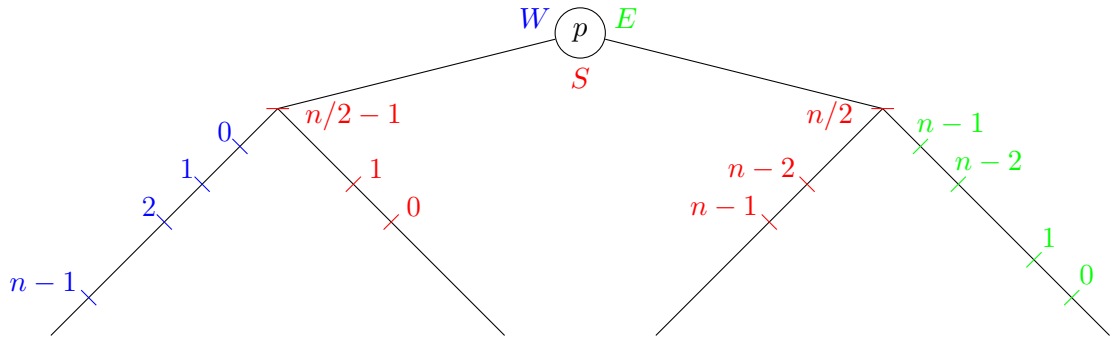


FIGURE 4 – Encodage des états aux différents passages du nœud p

Étant donnée une position x, on peut définir en logique du premier ordre avec l'opérateur modulo une formule $\langle x\rangle$ associant le plus grand nœud p de profondeur nulle modulo $n+1$, ancêtre de n. Disposant de x et $\langle x\rangle$, on peut alors savoir si la position $\langle x\rangle$ code un état ou pas, et si oui, on peut calculer la balise (W, S ou E) correspondante ainsi que l'état par lequel l'automate

passe au niveau de cette balise : on note $B_W(x)$, $B_S(x)$, $B_E(x)$, $\mathcal{E}_i(x)$ (pour $i \in \{0, \dots, n-1\}$) les macros ainsi définies.

Pour un calcul de l'automate donné, soit $(p_i, \beta_i)_{0 \leq i \leq m}$ la suite de nœuds de profondeur nulle modulo $n+1$ traversés par ce calcul : on note q_i l'état correspondant au nœud (p_i, β_i) dans le calcul. La formule qu'on construit est de la forme $2(n+1)\text{-TC}^{\lhd}_{xy}\varphi$, et les étapes intermédiaires empruntées par la clôture transitive sont exactement les nœuds $(u_i)_{0 \leq i \leq m}$ tels que $\langle u_i \rangle = p_i$, $B_{\beta_i}(u_i)$ et $\mathcal{E}_{q_i}(u_i)$.

Étant données deux étapes successives x et y de la clôture transitive, on peut écrire une formule $\psi(x, y)$ qui calcule le poids de l'ensemble des calculs de l'automate du nœud $(\langle x \rangle, \beta_1)$ (si $B_{\beta_1}(x)$) dans un état q_1 tel que $\mathcal{E}_{q_1}(x)$, au nœud $(\langle y \rangle, \beta_2)$ (si $B_{\beta_2}(y)$) dans un état q_2 tel que $\mathcal{E}_{q_2}(y)$. Remarquons que, puisque les calculs de l'automate doivent suivre l'ordre $\lhd$, l'ensemble des calculs précédents ne passe par aucun nœud de profondeur nulle modulo $n+1$, et donc cet ensemble est fini. De plus, le fait que ces calculs restent « locaux » garantit qu'on peut écrire cette formule en logique du premier ordre.

On suppose dans un premier temps que les arbres considérés vérifient tous la propriété que : tout nœud de profondeur nulle modulo $n+1$ possède les $3n$ successeurs décrits dans la figure 4. Alors, on peut décrire facilement la formule φ :

$$\begin{aligned}\varphi(x,y) = \quad & x = \varepsilon \wedge \langle y \rangle = \varepsilon \wedge B_W(y) \wedge \Big(\bigvee_{q \in Q} \mathcal{E}_q(y) \wedge \lambda_q \Big) \\ & \vee \mathrm{Succ}(x,y) \wedge \Big(\bigvee_{q_1, q_2 \in Q} \mathcal{E}_{q_1}(x) \wedge \mathcal{E}_{q_2}(y) \wedge \theta(x,y) \Big) \\ & \vee y = \varepsilon \wedge \langle x \rangle = \varepsilon \wedge B_E(x) \wedge \Big(\bigvee_{q \in Q} \mathcal{E}_q(x) \wedge \nu_q \Big)\end{aligned}$$

avec $\mathrm{Succ}(x, y)$ qui s'assure que les nœuds $(\langle x \rangle, \beta_1)$ et $(\langle y \rangle, \beta_2)$ (avec $B_{\beta_1}(x)$, $B_{\beta_2}(y)$) sont deux successeurs possibles dans l'ordre $\lhd$ (plus précisément qu'il existe de plus un chemin entre ces deux nœuds qui ne rencontrent aucun autre nœud de profondeur nulle modulo $n+1$).

Finalement, on étoffe la formule en considérant le cas général où un nœud p de profondeur nulle modulo $n+1$ peut ne pas posséder tous les successeurs, susceptibles d'encoder les états. Plusieurs cas se présentent au niveau d'un tel nœud. Le premier est celui dans lequel p est la racine d'un sous-arbre de hauteur inférieur à n : ce nœud n'est alors pas considéré comme une étape dans la clôture transitive, et le poids des calculs passant dans son sous-arbre est calculé par les étapes précédentes et suivantes. Le second cas est celui dans lequel p possède des successeurs à profondeur nulle modulo $n+1$ à la fois dans son sous-arbre gauche et dans son sous-arbre droit. Dans ce cas, on encode légèrement différemment les états par lesquels passent un calcul au niveau des balises du nœud p : en fait, on les encode sur les chemins qui mènent de p au successeur le plus à gauche du sous-arbre gauche pour la balise W, au successeur le plus à droite du sous-arbre gauche, et le plus à gauche du sous-arbre droit pour la balise S, et au successeur le plus à droite du sous-arbre droit pour la balise E. □

Théorème (4.2). $\mathbb{S}^{\mathrm{nDFS}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle = \mathbb{S}^{\mathrm{wFO+BTC}^{\lhd}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$

Démonstration. Afin de montrer l'inclusion $\mathbb{S}^{\mathrm{nDFS}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle \supseteq \mathbb{S}^{\mathrm{wFO+BTC}^{\lhd}}\langle\langle \mathcal{T}_\Sigma \rangle\rangle$, il suffit de montrer que la classe des séries reconnaissables par des nDFS contient les formules bFO et est close par opérations booléennes, quantifications et clôture transitive bornée. La preuve imite le cas des mots (proposition 3.2).

L'autre inclusion se prouve de manière identique au cas des mots (proposition 3.3), par récurrence en utilisant le résultat de base du théorème 4.1. □