Construction des automates des termes dérivés et termes dérivés cassés

Dans ce chapitre, nous expliciterons une construction pour l'automate des termes dérivés et pour l'automate des termes dérivés cassés. La première construction d'un automate des termes dérivés est donnée dans [5] comme l'application directe de la définition de l'automate. Dans cette méthode, on construit l'ensemble des termes dérivés en dérivant tous les termes dérivés connus par chaque lettre de l'alphabet jusqu'à ne plus en trouver de nouveaux. Les transitions sont construites simultanément à chaque étape par la dérivation par rapport à une lettre. Si l'expression de départ est de taille n alors la méthode d'Antimirov a une complexité de $O(n^5)$. Pour les termes dérivés cassés il est possible d'envisager une méthode similaire, avec une complexité identique.

Il existe également une construction de l'automate des termes dérivés qui se base sur des modifications de l'arbre syntaxique de l'expression. Ces modifications font ressortir une représentation des termes dérivés sous forme de chemins dans un graphe. Cette idée de modifier l'arbre syntaxique d'une expression rationnelle en ajoutant des arcs orientés afin de pouvoir obtenir un automate a d'abord été introduite dans [51] où une telle modification de l'arbre, appelée alors *ZPC-structure* permet de construire l'automate des positions de l'expression.

En se basant sur le résultat que l'automate des termes dérivés est un quotient de l'automate des positions, Champarnaud et Ziadi, dans [18], utilisent également la ZPC-structure pour calculer l'ensemble des termes dérivés –

l'automate Follow (cf. [27]) est également un exemple d'automate qui est un quotient de l'automate des positions qui peut être constuit en utilisant la ZPC-structure – . Les termes dérivés sont alors représentés par des chemins sur la ZPC-structure ([17]) et décrits par une liste de sous-expressions dont la concaténation donne le terme dérivé. Une méthode de Paige et Tarjan développée dans [36] permet d'identifier les listes qui représentent la même expression. Cette méthode permet alors d'identifier les termes dérivés avec une complexité en $O(n^3)$. L'automate des termes dérivés est alors construit en utilisant le calcul, quadratique, des transitions de l'automate des positions de [51]. L'automate des termes dérivés est alors construit en complexité cubique.

L'identification des termes dérivés identiques, qui constitue l'étape la plus coûteuse de [18], peut être améliorée. Ainsi dans [17], le marquage des sous-expressions étoilées identiques de l'expression rationnelle permet de réaliser cette identification en complexité $O(n^2)$. L'identification des sous-expressions est poussée encore plus loin dans [28] où toutes les sous-expressions sont marquées de manière à ce que deux sous-expressions isomorphes partagent la même marque. Cette méthode permet d'améliorer encore la complexité du calcul en $O(ln)$ où l est la longueur littérale de l'expression rationnelle.

Dans [1], Allauzen et Mohri proposent un algorithme pour construire l'automate des termes dérivés d'une expression à partir de l'automate de Thompson (cf. [47]) de cette expression. Dans [1] c'est en fait à la fois l'automate des positions, l'automate des termes dérivés et l'automate Follow qui sont construits à partir de l'automate de Thompson de l'expression en utilisant uniquement des opérations de minimisation et d'élimination de transitions spontanées. Plus précisément, l'automate des termes dérivés est le résultat de l'élimination des transitions spontanées sur la minimisation de l'automate obtenu par élimination des transitions spontanées créées lors d'une concaténation dans la construction de l'automate de Thompson. Cette méthode a la même complexité que l'algorithme de [28] mais permet d'unifier la construction de différents automates à partir d'une expression rationnelle.

Pour le calcul de l'automate des termes dérivés, dans ce chapitre, c'est une méthode très similaire à celle de [28] que nous présentons. Le calcul des termes dérivés est réalisé à partir de chemins dans l'arbre syntaxique de l'expression et nous marquons les états de cet arbre pour identifier les termes dérivés. La méthode que nous allons décrire peut être vue comme une généralisation de l'algorithme de [28] dans le cas où les expressions ne sont pas construites modulo l'associativité du produit. Une autre différence réside dans le fait que, afin de pouvoir adapter notre algorithme au calcul de l'automate des termes dérivés cassés, nous ne séparons pas le calcul des

transitions de celui des états de l'automate, c'est-à-dire que nous n'utilisons pas directement le fait que l'automate des termes dérivés est un quotient de l'automate des positions. Ceci est important car l'automate des termes dérivés cassés n'est pas un quotient de l'automate des positions. Dans le cas de l'automate des termes dérivés, cette différence de conception ne change pas les calculs réalisés car dans les deux cas, c'est l'idée de l'algorithme de [51] – pour les transitions de l'automate des positions – qui est utilisé.

Afin de réaliser l'identification des termes dérivés, et des termes dérivés cassés, nous présentons tout d'abord une méthode de marquage de l'arbre syntaxique d'une expression rationnelle. Ce marquage, avec des réductions que nous décrivons, permet de représenter de manière canonique les concaténations – à droite – de sous-expressions et, ainsi, de les identifier. Cette identification nous permet alors de décrire l'algorithme de construction de l'automate des termes dérivés par l'utilisation de chemins sur l'arbre syntaxique. Pour finir, nous détaillons l'algorithme, adapté du précédent, qui permet de construire l'automate des termes dérivés cassés en temps quadratique.

4.1 Concaténation à droite de sous-expressions

Comme nous l'avons vu dans les chapitres précédents, par les propositions 17 et 26, les termes dérivés et les termes dérivés cassés d'une expression E sont des concaténations à droite de sous-expressions de E. Les sous-expressions de E sont identifiables sur l'arbre T_{E} par des nœuds, représentant un sous-arbre.

Dans cette section, nous présentons une manière de représenter des concaténations à droite de sous-expressions par des mots sur les nœuds de l'arbre syntaxique. Nous montrons ensuite comment l'identification des sous-expressions, par le marquage des sous-expressions égales de l'expression, et l'utilisation d'un système de réécriture simple, permet de donner une représentation canonique des concaténations à droite de sous-expressions.

4.1.1 Sous-expressions sur l'arbre syntaxique

Nous avons déjà défini l'arbre syntaxique T_{E} d'une expression rationnelle E sur A. Comme nous travaillons toujours modulo $\mathbf{T}$, les feuilles de T_{E} sont étiquetées soit par $\mathsf{1}$ soit par une lettre de A. Nous avons noté $\mathrm{fp}(\mathsf{E})$ l'ensemble des *feuilles propres* de T_{E}, c'est-à-dire les feuilles étiquetées par une lettre – ces feuilles propres sont souvent également appelées *positions* de E (cf. [26] par exemple) –.

Afin de décrire l'arbre syntaxique formellement, nous utiliserons des notations empruntées aux expressions rationnelles :
(i) si x est un nœud de T_{E}, étiqueté par $+$ et dont les fils gauche et droit sont respectivement y et z, nous notons $x = y + z$;
(ii) de la même manière, nous notons $x = y \cdot z$ un nœud x, étiqueté par $\cdot$ et dont les fils sont y et z ;
(iii) enfin nous notons $x = y^*$ un nœud étiqueté par $*$ dont le fils est le nœud y.

Ces notations sont raisonnables car les nœuds sont utilisés pour représenter des sous-expressions de E et on a naturellement $[\![y + z]\!] = [\![y]\!] + [\![z]\!]$, $[\![y \cdot z]\!] = [\![y]\!] \cdot [\![z]\!]$ et $[\![y^*]\!] = [\![y]\!]^*$. Nous notons également $\phi(x)$ le père du nœud x.

Par simplification, nous noterons a une feuille x étiquetée par a et $\mathsf{1}$ une feuille étiquetée par $\mathsf{1}$ lorsque seule l'étiquette de ce nœud est importante.

Cependant ces notations ne sont pas de même nature que les précédentes, il s'agit juste de simplifications d'écriture lorsque seule l'étiquette d'un nœud est importante. Ces simplifications ne peuvent pas permettre de décrire l'arbre puisque deux feuilles différentes avec la même étiquette ne sont pas identifiables par cette notation.

Exemple 21. Dans cette section nous posons $A = \{a, b, c\}$ et $\mathsf{E}_1 = ((a \cdot b) \cdot (c^*)) + (b \cdot (c^*))$. La Figure 4.1 montre l'arbre syntaxique de E_1. Les nœuds sont notés $x_1, \ldots, x_{11}$ et les feuilles propres de l'arbre sont : $\mathrm{fp}(\mathsf{E}_1) = \{x_6, x_8, x_9, x_{10}, x_{11}\}$.

On a, par exemple : $[\![x_2]\!] = (a \cdot b) \cdot c^*$ et $[\![x_3]\!] = b \cdot c^*$.

Et, avec nos notations, dans $\mathrm{T}_{\mathsf{E}_1}$, on a $x_2 = x_4 \cdot x_5$ et $x_5 = x_{10}^*$.

Comme nous utilisons les nœuds de l'arbre pour représenter les sous-expressions de E, nous pouvons étendre au domaine des nœuds certaines définitions et notations déjà données sur les expressions rationnelles : le *terme constant* $\mathsf{c}(x)$ d'un nœud x de l'arbre syntaxique T_{E} de l'expression E sur A est le booléen défini par :

$$\mathsf{c}(\mathsf{0}) = 0\,, \quad \mathsf{c}(\mathsf{1}) = 1\,, \quad \forall x \in \mathrm{fp}(\mathsf{E}) \quad \mathsf{c}(x) = 0\ ,$$
$$\mathsf{c}(x + y) = \mathsf{c}(x) \vee \mathsf{c}(y)\,, \quad \mathsf{c}(x \cdot y) = \mathsf{c}(x) \wedge \mathsf{c}(y)\,, \quad \mathsf{c}(x^*) = 1\ .$$

Comme il est évident que $\mathsf{c}(x) = \mathsf{c}([\![x]\!])$, cette définition est pertinente. Cette définition donne directement une méthode de calcul des termes constants de tous les nœuds de l'arbre de bas en haut, c'est en fait la méthode habituelle pour calculer le terme constant d'une expression.

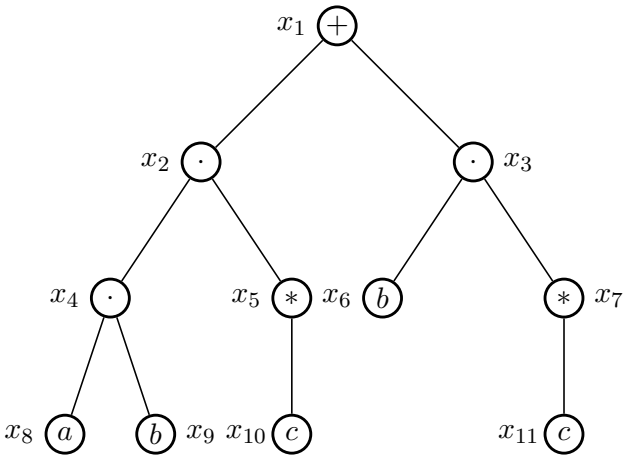

FIGURE 4.1 – L'arbre syntaxique $\mathrm{T}_{\mathsf{E}_1}$

Nous étendons également l'ensemble des *premières positions d'une expression*, défini dans les préliminaires pour la construction de l'automate des positions, à l'ensemble des *premières positions* d'un nœud x :

$$\begin{aligned}
\mathsf{First}(0) = \mathsf{First}(1) &= \emptyset \ , \\
\mathsf{First}(x) &= \{x\}, \quad \forall x \in \mathrm{fp}(\mathsf{E}) \ , \\
\mathsf{First}(y + z) &= \mathsf{First}(y) \cup \mathsf{First}(z) \ , \\
\mathsf{First}(y \cdot z) &= \mathsf{First}(y) \cup \mathsf{c}(y)\mathsf{First}(z) \ , \\
\mathsf{First}(y^*) &= \mathsf{First}(y) \ ,
\end{aligned}$$

De nouveau, nous avons $\mathsf{First}(x) = \mathsf{First}(\llbracket x \rrbracket)$. Nous noterons également $\mathsf{First}(x, a)$ le sous-ensemble de $\mathsf{First}(x)$ des feuilles étiquetées par a.

Exemple 22 (*Ex. 21 cont.*)**.** Dans l'arbre $\mathrm{T}_{\mathsf{E}_1}$, on a : $\mathsf{c}(x_1) = \mathsf{c}(x_2) =$

$\mathsf{c}(x_3) = \mathsf{c}(x_4) = \mathsf{c}(x_6) = \mathsf{c}(x_8) = \mathsf{c}(x_9) = \mathsf{c}(x_{10}) = \mathsf{c}(x_{11}) = 0$ et $\mathsf{c}(x_5) = \mathsf{c}(x_7) = 1$.

Et, par exemple, $\mathsf{First}(x_1) = \{x_8, x_6\}$ et par conséquent : $\mathsf{First}(x_1, a) = \{x_8\}$, $\mathsf{First}(x_1, b) = \{x_6\}$ et $\mathsf{First}(x_1, c) = \emptyset$.

Remarque 6. Pour être précis, le nœud x est lié à l'arbre duquel il est extrait, cependant, en pratique, il pourra être utilisé dans un autre arbre, en particulier l'arbre dont il est la racine, qui est un sous-arbre de l'arbre initial. Comme les fonctions décrites ci-dessus se calculent de bas en haut dans l'arbre, il n'est pas nécessaire de préciser l'arbre dans lequel on effectue le calcul.

Puisque les nœuds de l'arbre syntaxique permettent de représenter les sous-expressions d'une expression E, nous utilisons les mots sur l'alphabet dont les lettres sont les nœuds de T_{E} pour représenter les concaténations à droite de sous-expressions. Les mots sur les nœuds sont définis comme des mots sur l'alphabet dont les lettres sont les nœuds de l'arbre. Afin d'éviter toute confusion, la concaténation sur ces mots est notée :: et le mot $x :: y$ est la concaténation du mot x et du mot y. De même nous choisissons ε comme notation pour le mot vide – pour le distinguer du mot vide 1_{A^*} –.

Les mots sur les nœuds permettent de dénoter une expression, définie inductivement par :

(i) l'expression dénotée par le mot vide est $[\![\varepsilon]\!] = \mathsf{1}$;

(ii) le mot $x_1 :: x_2 :: \cdots :: x_n$ dénote l'expression $[\![x_1 :: x_2 :: \cdots :: x_n]\!] = [\![x_1 :: x_2 :: \cdots :: x_{n-1}]\!] \cdot [\![x_n]\!]$.

De par la définition de l'expression dénotée, les mots sur les nœuds de T_{E} dénotent des concaténations à droite de sous-expressions de E. Cependant cette représentation n'est pas unique et une même concaténation peut être dénotée par plusieurs mots sur les nœuds.

Enfin les notions de terme constant et de premières positions sont étendues aux mots sur les nœuds, de manière à rester cohérent avec les expressions qu'ils dénotent :

(i) $\mathsf{c}(x_1 :: x_2 :: \cdots :: x_n) = \mathsf{c}(x_1) \wedge \mathsf{c}(x_2) \wedge \cdots \wedge \mathsf{c}(x_n)$,

(ii) $\mathsf{First}(\varepsilon) = \emptyset$,

(iii) $\mathsf{First}(x_1 :: x_2 :: \cdots :: x_n) = \mathsf{First}(x_1 :: x_2 :: \cdots :: x_{n-1}) \cup \mathsf{c}(x_1 :: x_2 :: \cdots :: x_{n-1})\mathsf{First}(x_n) = \mathsf{First}(x_1) \cup \mathsf{c}(x_1)\mathsf{First}(x_2 :: \cdots :: x_n)$.

Exemple 23 (*Ex. 21 cont.*)**.**

$$\begin{aligned}
[\![x_8 :: x_9 :: x_5]\!] &= [\![x_8 :: x_9]\!] \cdot [\![x_5]\!] = (a \cdot b) \cdot c^* \ , \\
[\![x_9 :: x_5]\!] &= b \cdot c^* = [\![x_3]\!] \ . \\
\mathsf{c}(x_9 :: x_5) &= \mathsf{c}(x_9) \wedge \mathsf{c}(x_5) = 0 \ , \\
\mathsf{First}(x_9 :: x_5) &= \{x_9\} \ .
\end{aligned}$$

4.1.2 Représentation canonique

En vue de construire l'automate des termes dérivés de E, et comme les termes dérivés – et termes dérivés cassés – sont des concaténations à droite de sous-expressions de E, nous allons les représenter par des mots sur les nœuds de T_{E}. Cependant, pour construire exactement l'ensemble des états de l'automate voulu, il faut être capable d'identifier deux mots sur les nœuds dénotant la même expression. C'est ce que nous allons maintenant faire en proposant une forme canonique pour représenter toute concaténation droite de sous-expressions de E.

Cette idée de représentation canonique des concaténations de sous-expressions rationnelles est sous-jacente dans [28] mais appliquée directement aux mots particuliers qui donnent les termes dérivés. Nous préférons développer une représentation canonique pour toutes les concaténations à droite de sous-expressions afin de pouvoir également appliquer ce résultat pour la construction de l'automate des termes dérivés cassés – qui ne sont pas représentés par les mêmes mots que les termes dérivés –.

Marquer les nœuds

Afin d'avoir une forme canonique pour représenter une concaténation à droite de sous-expressions de E, nous allons 'marquer' les différentes sous-expressions de E. Cette notion de marquage est introduite dans [17] où seules les sous-expressions étoilées sont marquées. Comme dans [28], nous allons repérer tous tous les nœuds représentants des sous-expressions identiques par un même identifiant.

Pour cela nous affectons à chaque nœud x une *marque*, notée $\overline{x}$ telle que, si x et y sont deux nœuds dont les expressions dénotées sont égales – c'est-à-dire que les sous-arbres sont isomorphes –, alors $\overline{x} = \overline{y}$. C'est-à-dire que la marque d'un nœud est directement associée à l'expression rationnelle dénotée par le sous-arbre dont il est la racine. C'est donc une manière de repérer les sous-expressions dans l'arbre syntaxique.

Comme, pour que deux nœuds x et y aient la même marque, il faut, soit que ce soient des feuilles avec la même étiquette, soit qu'ils aient la même étiquette et que leurs fils aient les mêmes marques, il est possible de marquer tout l'arbre en marquant les feuilles en premier puis en remontant dans l'arbre.

Une méthode similaire pour marquer les nœuds de l'arbre est donnée dans [28] par minimisation d'un automate – sans circuits – construit à partir de l'arbre syntaxique : l'arbre syntaxique est transformé en un automate avec un unique état initial dont les transitions sortantes pointent vers les feuilles avec la même étiquette que la feuille en question. Les arêtes de l'arbre sont transformées en transitions ascendantes dont l'étiquette dépend de l'étiquette du père et de la nature du fils (droit ou gauche). L'état final est la racine. En effectuant le quotient de cet automate on récupère les classes d'équivalence des nœuds pour les marques.

Comme les marques représentent les sous-expressions, nous étendons aux marques les définitions proposées précédemment sur les nœuds : $[\![\overline{x}]\!] = [\![x]\!]$ est l'expression rationnelle associée à $\overline{x}$. Il est également naturel d'étendre la définition du terme constant d'un nœud – ou d'une expression – : $\mathsf{c}(\overline{x}) = \mathsf{c}(x)$.

Enfin il est également possible de définir des mots sur les marques de nœuds – les notations sont les mêmes que pour les mots sur les nœuds – et l'expression qu'ils représentent :

$$[\![\overline{x_1} :: x_2 :: \cdots :: \overline{x_n}]\!] = [\![x_1 :: x_2 :: \cdots :: x_n]\!] \ .$$

Il est donc finalement possible de représenter les concaténations droite de sous-expressions par des mots sur les marques.

Exemple 24 (*Ex. 21 cont.*)**.** On pourrait par exemple marquer l'arbre $\mathrm{T}_{\mathsf{E}_1}$ de la manière présentée sur la Figure 4.2.

Exemple 25. La Figure 4.3 montre l'arbre syntaxique de l'expression $((a \cdot b) \cdot c) + (a \cdot (b \cdot c))$. Seules les feuilles x_8 et x_6, x_9 et x_{10} , x_5 et x_{11} partagent la même marque.

Réduction préfixe

Si les marques permettent d'identifier les sous-expressions identiques, elles ne permettent pas directement d'identifier les concaténations de sous-expressions.

On peut noter que si les mots $\overline{x} :: \overline{y}$ et $\overline{z}$ dénotent la même expression, alors il existe sur l'arbre un nœud concaténation marqué $\overline{z}$ dont les fils gauche

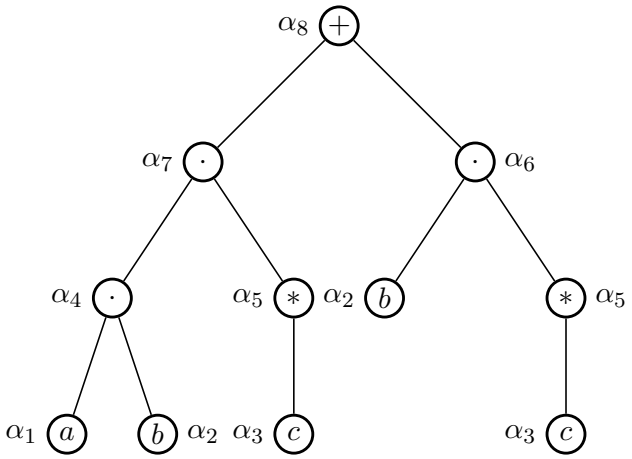

FIGURE 4.2 – Exemple de marquage de l'arbre syntaxique $\mathrm{T}_{\mathsf{E}_1}$

et droit sont marqués respectivement par $\overline{x}$ et $\overline{y}$. Il peut pourtant arriver que $[\![x :: y]\!] = [\![z]\!]$ sans que x et y soient les fils de z ni même qu'ils soient fils d'un même nœud concaténation.

Par exemple, sur la Figure 4.3, $I :: II$ et IV sont des mots de marques qui dénotent la même expression $(a \cdot b)$ et le nœud x_4 est marqué IV et ses fils sont marqués I et II. Par contre, les nœuds x_6 et x_{10} sont aussi marqués respectivement I et II mais ne sont pas les fils d'un même nœud. De même, la même expression $((a \cdot b) \cdot c)$ dénote les mots de marques $I :: II :: III$, $IV :: III$ et VII qui peuvent eux-mêmes être associés à différents mots de nœuds.

C'est pourquoi nous introduisons une réduction sur les mots de marques qui permet d'obtenir une représentation canonique des concaténations à droite de sous-expressions d'une expression rationnelle.

Considérons $\mathcal{R}$ l'ensemble des règles de réécriture sur les mots de marques

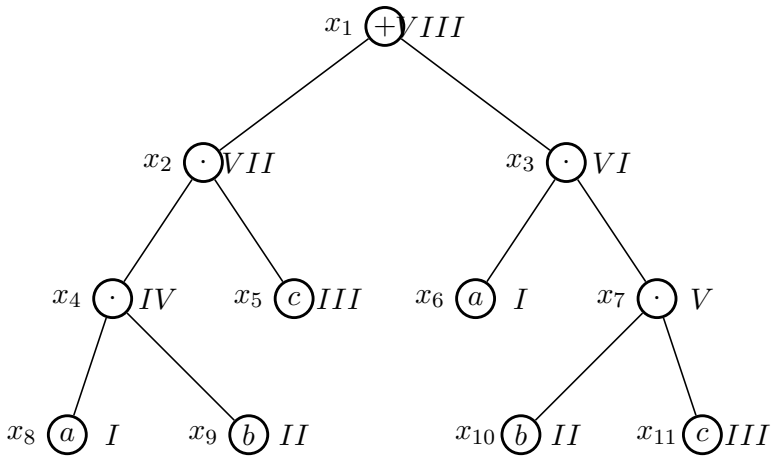

FIGURE 4.3 – Exemple de marquage pour $((a \cdot b) \cdot c) + (a \cdot (b \cdot c))$

$\alpha_1 :: \alpha_2 \to \beta$ avec α_1, α_2 et β telles qu'il existe $y = x_1 \cdot x_2$ avec $\overline{x_1} = \alpha_1$, $\overline{x_2} = \alpha_2$ et $\overline{y} = \beta$ c'est-à-dire des nœuds dont les marques sont respectivement α_1 et α_2 et qui sont fils gauche et droit d'un nœud étiqueté par une concaténation et dont la marque est β.

Notons $\rho\left(\overline{X}\right)$ le résultat de la réécriture préfixe du mot sur les marques $\overline{X}$ par les règles $\mathcal{R}$. Ainsi si les deux premières lettres de $\overline{X}$ correspondent à un nœud concaténation dans l'arbre, on remplace ces deux lettres par la marque du nœud et l'on recommence la réécriture sur le mot ainsi formé. Cette réécriture termine puisqu'elle transforme deux lettres en une seule. De plus on a :

$$\llbracket \rho\left(\overline{X}\right) \rrbracket = \llbracket X \rrbracket \ .$$

Exemple 26 (*Ex. 21 cont.*)**.** Les règles $\mathcal{R}_1$ pour l'arbre $\mathrm{T}_{\mathsf{E}_1}$ sont :

$$\alpha_1 :: \alpha_2 \to \alpha_4 \ , \quad \alpha_2 :: \alpha_5 \to \alpha_6 \ , \quad \alpha_4 :: \alpha_5 \to \alpha_7 \ .$$

Et un exemple de réduction serait :

$$\rho\left(\overline{x_9 :: x_5 :: x_9 :: x_5}\right) = \rho\left(\alpha_2 :: \alpha_5 :: \alpha_2 :: \alpha_5\right) = \alpha_6 :: \alpha_2 :: \alpha_5 \ .$$

Cet exemple montre bien que la réduction est préfixe car $\alpha_2 :: \alpha_5$ n'est réduit qu'en début de mot.

Notons $\overline{1}$ la marque de n'importe quelle feuille étiquetée par 1. Nous appellerons *mot propre sur les marques* un mot dont $\overline{1}$ n'est pas une lettre. Les expressions rationnelles étant, comme toujours, définie modulo les identités triviales **T**, il est possible de transformer un mot non propre sur les marques en un mot propre en éliminant les occurrences de $\overline{1}$ tout en conservant l'expression dénotée.

Pour montrer que $\rho\left(\overline{X}\right)$ est une représentation canonique de $[\![\overline{X}]\!]$ partagée par tous les mots de marques qui dénotent la même expression, nous prouvons le théorème suivant :

Théorème 34. *Soient $\overline{X}$ et $\overline{Y}$ deux mots propres sur les marques de l'arbre syntaxique d'une expression rationnelle.*

$$[\![\overline{X}]\!] = [\![\overline{Y}]\!] \Leftrightarrow \rho\left(\overline{X}\right) = \rho\left(\overline{Y}\right)$$

Démonstration. Prouvons le sens direct par l'absurde : soient $\overline{X} = \overline{x_1} \cdots \overline{x_n}$ et $\overline{Y} = \overline{y_1} \cdots \overline{y_m}$ deux mots propres sur les marques, différents, réduits et tels que $[\![\overline{X}]\!] = [\![\overline{Y}]\!]$.

- Supposons que $n > 1$ et $m > 1$, alors, comme $\overline{X}$ est propre, $[\![\overline{X}]\!] = [\![\overline{x_1} \ldots \overline{x_{n-1}}]\!] \cdot [\![\overline{x_n}]\!]$ donc $[\![\overline{X}]\!]$ est une concaténation dont le terme de droite est $[\![\overline{x_n}]\!]$ et de même $[\![\overline{Y}]\!]$ est une concaténation dont le terme de droite est $[\![\overline{y_m}]\!]$. Il est donc nécessaire que $\overline{x_n} = \overline{y_m}$ et que $\overline{x_1} \cdots \overline{x_{n-1}}$ et $\overline{y_1} \cdots \overline{y_{m-1}}$ soient deux mots, différents, réduits – car un préfixe d'un mot réduit est réduit – et dénotent la même expression. Par hypothèse de récurrence, on peut supposer donc que $n = 1$.
- Si $n = m = 1$ alors $[\![\overline{x_1}]\!] = [\![\overline{y_1}]\!]$ et donc, par définition, $\overline{x_1} = \overline{y_1}$.
- Si $n = 1$ et $m = 2$ alors $[\![\overline{x_1}]\!] = [\![\overline{y_1}]\!] \cdot [\![\overline{y_2}]\!]$. Il existe alors un nœud x_1 tel que $[\![x_1]\!] = [\![\overline{y_1}]\!] \cdot [\![\overline{y_2}]\!]$ et donc x_1 est un nœud concaténation dont le fils gauche représente $[\![\overline{y_1}]\!]$ et le fils droit $[\![\overline{y_2}]\!]$ c'est-à-dire que leurs marques respectives sont $\overline{y_1}$ et $\overline{y_2}$. Contradiction avec l'hypothèse que $\overline{Y}$ est réduit.
- Si $n = 1$ et $m > 2$ alors supposons que $\overline{Y}$ soit un mot de longueur minimale qui dénote la même expression qu'un mot $\overline{X}$ de longueur 1. Dans ce cas $[\![\overline{Y}]\!]$ est une concaténation dont le terme de gauche est $[\![\overline{y_1} \ldots \overline{y_{m-1}}]\!]$, il y a donc un nœud x_1 dans l'arbre dont le fils

gauche dénote cette dernière expression. C'est en contradiction avec l'hypothèse de minimalité.

Comme $\|\rho\left(\overline{X}\right)\| = \|X\|$, la réciproque est directe. □

Nous considérons dans ce chapitre uniquement les mots de nœuds propres car les mots que nous calculons pour représenter les termes dérivés et les termes dérivés cassés sont propres. Cette hypothèse simplifie grandement les réductions pour obtenir un mot de marque canonique pour une expression.

Dans le cas des expressions à multiplicités, les mots qui représentent les termes dérivés et les termes dérivés cassés ne sont pas propres et nous verrons dans le chapitre suivant qu'il faut alors rajouter des règles de transformations, sur les mots de marques, pour obtenir la représentation canonique souhaitée.

4.2 Calcul de $\mathcal{D}(\mathsf{E})$ et $\mathcal{D}_b(\mathsf{E})$

Afin d'établir un algorithme pour calculer l'automate des termes dérivés cassés, nous allons d'abord montrer, dans cette section, comment les mots sur les nœuds et les mots sur les marques permettent de calculer efficacement l'ensemble des termes dérivés et les transitions de l'automate des termes dérivés. Une fois l'algorithme pour les termes dérivés établi, nous détaillerons les modifications qui permettent d'obtenir la construction de l'automate des termes dérivés cassés.

4.2.1 Calculs sur l'arbre syntaxique décoré

Nous présentons ici la méthode de Champarnaud et Ziadi (cf. [17] ou [16] par exemple) qui, inspirée des calculs réalisés sur l'arbre syntaxique pour obtenir l'automate standard (cf. [51]), permet de calculer les termes dérivés d'une expression en complexité quadratique.

Pour ce faire, nous allons définir pour chaque position de l'arbre syntaxique un *chemin* dans celui-ci qui remonte à la racine. Ce chemin permet de collecter quelques nœuds de l'arbre qui définissent alors un mot sur les nœuds qui, nous le verrons par la suite, représente un terme dérivé.

Plus précisément, il est possible de 'décorer' l'arbre syntaxique en ajoutant des arcs orientés, appelés *liens*,

(i) du fils de chaque nœud étiqueté par $*$ à son père ;

(ii) de chaque fils gauche de nœud concaténation au fils droit de ce dernier.

Cette décoration de l'arbre syntaxique est une manière de représenter la *ZPC-structure* décrite dans [51]. La ZPC-structure est constituée de deux

copies orientées de l'arbre syntaxique telles que dans l'une les déplacements sont descendants et dans l'autre ascendants. Il existe des liens entre les deux copies. Dans [51] la ZPC-structure d'une expression rationnelle permet de construire directement l'automate des positions de l'expression.

L'arbre syntaxique décoré que nous avons défini est une manière de voir la ZPC-structure en superposant les deux copies. Par exemple, l'arbre décoré de E_1 est montré sur la Figure 4.4.

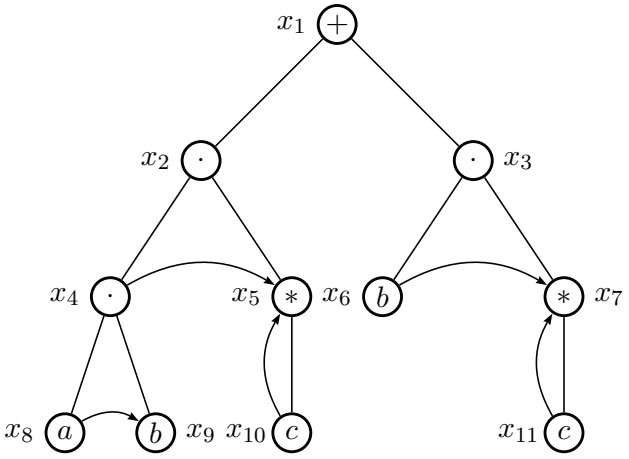

FIGURE 4.4 – Arbre syntaxique décoré de $\mathrm{T}_{\mathsf{E}_1}$

Dans l'arbre décoré, il existe un unique chemin qui part d'un nœud x, qui prend les arcs orientés décrits ci-dessus lorsque c'est possible, et si ce n'est pas possible remonte au père, jusqu'à la racine de l'arbre. Si l'on décrit ce chemin comme un mot dont les lettres sont les nœuds successivement atteints en empruntant un arc orienté (cf. [17]), on obtient le mot $\pi(x)$, que nous définissons formellement ci-dessous.

Soit E une expression rationnelle, T_{E} l'arbre syntaxique qui la représente.

On note r la racine de T_{E}. Le *chemin associé* à un nœud x de T_{E} est le mot sur les nœuds défini par :

$$\pi_r(x) = \begin{cases} \varepsilon & \text{si } x = r, \\ y :: \pi_r(\phi(y)) & \text{si } \phi(x) = x \cdot y, \\ y :: \pi_r(y) & \text{si } y = x^*, \\ \pi_r(\phi(x)) & \text{sinon.} \end{cases}$$

Lorsqu'il n'y a pas de confusion sur l'arbre que l'on considère, on note $\pi(x)$ à la place de $\pi_r(x)$.

Pour tout x, les chemin $\pi(x)$ est donc un mot de nœuds qui sont tous extrémités d'un lien. Comme les expressions – et leurs arbres – sont construites modulo $\mathbf{T}$, le mot $\pi(x)$ est propre pour tout x car les liens pointent soit sur une étoile, soit sur le fils droit d'un nœud concaténation, un tel fils ne peut pas être étiqueté par 1.

Remarque 7. Si la concaténation d'expressions rationnelles est associative, c'est-à-dire que les expressions sont construites modulo $\mathbf{A_p}$, alors l'arbre syntaxique d'une expression est tel qu'aucun nœud concaténation n'a pour fils droit un autre nœud concaténation – les concaténations consécutives sont représentées par un peigne orienté vers la gauche –. En particulier, aucune lettre de $\pi(x)$ ne peut être associée à une concaténation.

C'est pourquoi dans [28] où le produit est associatif, il n'y a pas besoin d'utiliser la réduction préfixe par $\mathcal{R}$ car, dans le calcul des chemins, aucune lettre ne représente une concaténation. En reprenant la Figure 4.3, si le produit est associatif, les deux sous-arbres gauche et droit, de racines respectives x_2 et x_3, représentent la même expression rationnelle et c'est le sous-arbre de racine x_2 qui doit en fait être construit dans les deux cas.

Exemple 27 (*Ex. 21 cont.*)**.** Dans $\mathrm{T}_{\mathsf{E}_1}$ on a les chemins associés suivants :

$$\pi(x_6) = \pi(x_{11}) = x_7, \quad \pi(x_8) = x_9 :: x_5, \quad \pi(x_9) = \pi(x_{10}) = x_5, \quad \pi_{x_4}(x_8) = x_9 \ ,$$

pour lesquels on a, par exemple,

$$[\![\pi(x_8)]\!] = b \cdot c^* \ , \quad [\![\pi(x_6)]\!] = [\![\pi(x_9)]\!] = [\![\pi(x_{10})]\!] = [\![\pi(x_{11})]\!] = c^* \ .$$

Dans l'exemple, x_6 et x_9 ont des chemins différents qui dénotent une même expression rationnelle.

Les chemins associés aux feuilles propres d'un arbre permettent de décrire les termes dérivés de l'expression rationnelle. C'est ce qu'énonce le théorème suivant qui est une réécriture avec nos notations de la Proposition 6 de [17], dont la preuve est également donnée par induction sur la profondeur de l'expression.

Théorème 35 ([17]). *Soit* E *une expression rationnelle dont l'arbre syntaxique* T_{E} *a pour racine* r*. On a :*

$$\frac{\partial}{\partial a}\mathsf{E} = \{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} \ .$$

Démonstration. Prouvons ce théorème par induction sur la profondeur de l'expression E :

- Si $\mathsf{E} = 1$, alors $\mathsf{First}(r,a) = \emptyset$ et $\frac{\partial}{\partial a}\mathsf{E} = \emptyset$.
- Si $\mathsf{E} = b$, avec $b \neq a$, alors $\mathsf{First}(r,a) = \emptyset$ et $\frac{\partial}{\partial a}\mathsf{E} = \emptyset$
- Si $\mathsf{E} = a$, alors $\mathsf{First}(r,a) = \{r\}$ et $\frac{\partial}{\partial a}\mathsf{E} = [\![\pi(r)]\!] = [\![\varepsilon]\!] = 1$.
- Si $\mathsf{E} = \mathsf{F} + \mathsf{G}$, alors $r = x + y$ avec $[\![x]\!] = \mathsf{F}$ et $[\![y]\!] = \mathsf{G}$. On a donc :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \cup \mathsf{First}(y,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l)$ et donc $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!]$. De même, si $l \in \mathsf{First}(y,a)$ alors $[\![\pi_r(l)]\!] = [\![\pi_y(l)]\!]$ et donc par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cup \frac{\partial}{\partial a}\mathsf{G} \ .$$

- Si $\mathsf{E} = \mathsf{F} \cdot \mathsf{G}$, alors $r = x \cdot y$ avec $[\![x]\!] = \mathsf{F}$ et $[\![y]\!] = \mathsf{G}$. On a :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \cup \mathsf{c}(x)\mathsf{First}(y,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l) :: y$ et donc $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!] \cdot \mathsf{G}$. Si $l \in \mathsf{First}(y,a)$ alors $\pi_r(l) = \pi_y(l)$ et donc $[\![\pi_r(l)]\!] = [\![\pi_y(l)]\!]$. Alors, par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cdot \mathsf{G} \cup \mathsf{c}(\mathsf{F})\frac{\partial}{\partial a}\mathsf{G} \ .$$

- Si $\mathsf{E} = \mathsf{F}^*$, alors $r = x^*$ avec $[\![x]\!] = \mathsf{F}$. On a :

  $$\mathsf{First}(r,a) = \mathsf{First}(x,a) \ .$$

  Si $l \in \mathsf{First}(x,a)$, alors $\pi_r(l) = \pi_x(l) :: r$ et alors $[\![\pi_r(l)]\!] = [\![\pi_x(l)]\!] \cdot \mathsf{E}$. Par induction :

  $$\{[\![\pi_r(l)]\!] \mid l \in \mathsf{First}(r,a)\} = \frac{\partial}{\partial a}\mathsf{F} \cdot \mathsf{E} \ .$$

□

Afin de pouvoir dériver par rapport à un mot non vide, et comme nous venons de voir que les dérivations par rapport à une lettre sont des mots de nœuds, nous 'dérivons' les mots de nœuds :

Théorème 36. *Soit* E *une expression rationnelle et* X *un mot sur les nœuds :*

$$\frac{\partial}{\partial a}[\![X]\!] = \{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} \ .$$

Démonstration. Par induction sur la longueur de X. Si $X = x_1$ alors ce théorème est une instance du précédent pour l'expression rationnelle $[\![x_1]\!]$. Pour $X = x_1...x_n$, nous avons par définition :

$$\mathsf{First}(X, a) = \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a) \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\mathsf{First}(x_n, a) \ .$$

L'arbre syntaxique représentant $[\![X]\!]$, dont la racine est notée p, peut être représenté, récursivement, par la concaténation de l'arbre représentant $[\![x_1 :: \cdots :: x_{n-1}]\!]$, dont nous noterons la racine q, et du sous-arbre de T_{E} dont la racine est x_n.

Si $l \in \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a)$, alors $[\![\pi_p(l)]\!] = [\![\pi_q(l)]\!] \cdot [\![x_n]\!]$ et donc

$$\begin{aligned}\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} = {} & \{[\![\pi(l)]\!] \cdot [\![x_n]\!] \mid l \in \mathsf{First}(x_1 :: \cdots :: x_{n-1}, a)\} \\ & \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(x_n, a)\} \ .\end{aligned}$$

C'est à dire :

$$\begin{aligned}\{[\![\pi(l)]\!] \mid l \in \mathsf{First}(X, a)\} &= \left[\frac{\partial}{\partial a}[\![x_1 :: \cdots :: x_{n-1}]\!]\right] \cdot [\![x_n]\!] \cup \mathsf{c}(x_1 :: \cdots :: x_{n-1})\frac{\partial}{\partial a}[\![x_n]\!] \\ &= \frac{\partial}{\partial a}[\![X]\!] \ .\end{aligned}$$

□

En particulier, une instance de ce théorème donne, pour une feuille propre h :

$$\frac{\partial}{\partial a}[\![\pi(h)]\!] = \{[\![\pi(l)]\!] \mid l \in \mathsf{First}(\pi(h), a)\} \ .$$

L'application du Théorème 35 avec cette dernière équation donne l'équivalent du théorème 9 de [18] :

Théorème 37 ([18])**.** *Soit* E *une expression rationnelle, on a :*

$$\mathrm{TD}\,(\mathsf{E}) = \{[\![\pi(l)]\!] \mid l \in \mathrm{fp}(\mathsf{E})\}$$

Exemple 28 (*Ex. 21 cont.*)**.**

$$\begin{aligned}\mathrm{TD}\,(\mathsf{E}_1) &= \{[\![\pi(l)]\!] \mid l \in \mathrm{fp}(\mathsf{E}_1)\} \\ &= \{[\![\pi(x_6)]\!], [\![\pi(x_8)]\!], [\![\pi(x_9)]\!], [\![\pi(x_{10})]\!], [\![\pi(x_{11})]\!]\} \\ &= \{b \cdot c^*, c^*\}\end{aligned}$$

4.2.2 Construction de l'automate des termes dérivés

Nous avons vu dans les sous-sections précédentes toutes les briques qui permettent maintenant de construire l'automate des termes dérivés. Cette construction se divise en deux étapes. La première étape est la construction de l'ensemble des états de l'automate. Cette étape utilise le résultat du Théorème 37 et l'identification canonique des concaténations à droite de sous-expressions décrite dans la section précédente et coïncide avec le calcul des états réalisé par la méthode présentée dans [28]. La deuxième étape est le calcul des transitions de l'automate. Cette étape utilise le Théorème 35 et reprend la méthode de calcul des transitions de l'automate des positions de [51].

Soit E une expression rationnelle dont l'arbre syntaxique T_{E} a pour racine r. Les états de l'automate $\mathcal{D}(\mathsf{E})$ des termes dérivés de E sont l'expression elle-même $\llbracket r \rrbracket$ et les $\llbracket \pi(l) \rrbracket$ pour toute feuille propre l de T_{E}. Il est possible que pour deux feuilles l et l', $\llbracket \pi(l) \rrbracket = \llbracket \pi(l') \rrbracket$. Cependant, il est possible de représenter de manière unique $\llbracket \pi(l) \rrbracket$ en prenant la réduction du mot marqué : $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right)$.

Il est donc possible de considérer que les états de $\mathcal{D}(\mathsf{E})$ ne sont plus les termes dérivés mais les mots réduits sur les marques qui les représentent. L'ensemble des états est construit en calculant les réductions par $\mathcal{R}$ des mots de marques résultant du calcul des chemins pour chaque feuille. L'état initial est $\rho\left(\overline{r}\right)$ et les états finaux sont ceux dont le terme constant est non nul $\mathsf{c}(q) = 1$ – le terme constant a été défini pour les mots de marques –.

Notons $\theta(q)$ l'ensemble des feuilles propres l de T_{E} dont la réduction du mot marqué associé au chemin est q. C'est-à-dire que si l et l' sont dans $\theta(q)$ alors : $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right) = q$.

Nous choisissons de définir $\mathsf{First}(q) = \mathsf{First}(\pi(l))$ où l est un représentant arbitrairement choisi dans $\theta(q)$. Il faut noter que, même si l et l' sont dans $\theta(q)$, les chemins $\pi(l)$ et $\pi(l')$ peuvent être différents et il n'y a alors pas nécessairement égalité de $\mathsf{First}(\pi(l))$ et $\mathsf{First}(\pi(l'))$. Le choix d'un représentant pour définir la fonction First pour toute la classe q est justifiée par la proposition suivante – qui découle directement du fait que si deux feuilles ont des chemins associés à la même expression, alors la dérivation par rapport à une lettre de ces chemins est nécessairement identique – :

Proposition 38. *Soit q un état de l'automate des termes dérivés. Si l et l' sont dans $\theta(q)$ alors :*

$$\{\llbracket \pi(k) \rrbracket \mid k \in \mathsf{First}(\pi(l), a)\} = \{\llbracket \pi(k) \rrbracket \mid k \in \mathsf{First}(\pi(l'), a)\} \quad .$$

Enfin nous pouvons décrire de manière constructive l'ensemble des transitions de l'automate $\mathcal{D}(\mathsf{E})$: soient q_i et q_j deux états, on a :

$$\begin{aligned} \llbracket q_j \rrbracket \in \frac{\partial}{\partial a} \llbracket q_i \rrbracket &\Leftrightarrow \exists l \in \mathsf{First}(\llbracket q_i \rrbracket, a), \quad \llbracket \pi(l) \rrbracket = \llbracket q_j \rrbracket \ , \\ &\Leftrightarrow \exists l \in \mathsf{First}(q_i, a), \quad l \in \theta(q_j) \ . \end{aligned}$$

C'est à dire qu'il y a une transition entre un état q_i et un état q_j étiquetée par a si et seulement si, il existe dans T_{E} une feuille propre l étiquetée par a telle que $l \in \mathsf{First}(q_i)$ et $l \in \theta(q_j)$.

Pour résumer, pour construire l'automate des termes dérivés d'une expression rationnelle E nous suivons les différentes étapes suivantes :

1. calcul pour tout nœud x de l'arbre syntaxique par un parcours de bas en haut :
   - du terme constant $\mathsf{c}(x)$;
   - de l'ensemble des feuilles propres $\mathsf{First}(x)$;
   - de la marque $\overline{x}$;
2. calcul du chemin $\pi(l)$ associé à toute feuille propre l de T_{E} et réduction préfixe de leur version marquée : $\rho\left(\overline{\pi(l)}\right)$;
3. par identification lexicographique, toutes les feuilles l et l' telles que $\rho\left(\overline{\pi(l)}\right) = \rho\left(\overline{\pi(l')}\right)$ sont regroupées dans une même classe. Les classes de cette étape donnent les états de l'automate, il faut garder la composition de chaque classe $\theta(q)$ de feuilles ;
4. pour chaque état q, calcul de $\mathsf{First}(\rho\left(\overline{\pi(l)}\right))$ pour un représentant $l \in \theta(q)$ donné ;
5. chaque état q dont le terme constant $\mathsf{c}(q)$ est non nul est identifié comme un état final ;
6. pour chaque paire d'états (q_i, q_j) calcul des intersections $\mathsf{First}(q_i, a) \cap \theta(q_j)$. Si l'intersection est non vide, alors il y a une transition de q_i à q_j étiquetée a.

Si n est la taille de l'expression rationnelle E et l sa longueur littérale alors, le point 1 est réalisé par un parcours en profondeur sur l'arbre, donc en $O(n)$. Le calcul du chemin pour une feuille donnée est également en $O(n)$ et la réduction préfixe est linéaire. L'étape 2 a donc un complexité de $O(ln)$. L'identification lexicographique des états – étape 3 – est également en $O(ln)$. Finalement les étapes suivantes sont également en $O(ln)$ et l'algorithme obtenu est donc en $O(ln)$, comme l'algorithme de [28].

Exemple 29 (*Ex. 21 cont.*)**.** Nous avons vu que les trois états de $\mathcal{D}(\mathsf{E}_1)$ sont α_8 – le marque de la racine de $\mathrm{T}_{\mathsf{E}_1}$ – qui est initial, $\alpha_2 :: \alpha_5$, qui représente $b \cdot c^*$ et α_5, qui représente c^* et est final.

La Figure 4.5 montre l'automate des termes dérivés de E_1 ainsi construit :

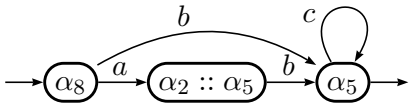

FIGURE 4.5 – L'automate $\mathcal{D}(\mathsf{E}_1)$

Le calcul de l'ensemble des états de l'automate des termes dérivés que nous avons décrit est, à l'associativité du produit près, la méthode exposée dans [28].

En revanche, la méthode de calcul des transitions diffère, conceptuellement, de celle de [28]. Dans [28] les transitions sont calculées presque indépendamment du calcul des états de l'automate. En fait ce sont les transitions de l'automate des positions qui sont calculées et c'est en réalisant le quotient de l'automate des positions, grâce aux classes d'équivalence des états, que sont obtenues les transitions de l'automate des termes dérivés. Dans les faits, la méthode de [28] réalise les mêmes calculs que ceux que dans la méthode que nous présentons car, l'automate des positions est calculé par la méthode exposée dans [51] qui utilise les mêmes chemins sur l'arbre pour construire l'automate des positions que ceux pour calculer les termes dérivés.

Comme l'automate des termes dérivés cassés n'est pas un quotient de l'automate des positions, nous avons choisi de proposer une méthode de calcul pour les termes dérivés qui s'adapte facilement aux autres cas que nous allons considérer – termes dérivés cassés et multiplicités –. C'est pourquoi nous ne séparons pas le calcul des états de celui des transitions.

4.2.3 Termes dérivés cassés

Nous allons maintenant décrire comment modifier la méthode décrite ci-dessus afin de calculer l'automate $\mathcal{D}_b(\mathsf{E})$ des termes dérivés cassés de l'expression rationnelle E.

Cassage d'un mot de nœuds

Comme nous avons représenté les vrais termes dérivés par un mot de nœuds de l'arbre syntaxique – décrit par un chemin sur celui-ci –, et que les vrais termes dérivés cassés sont le cassage des vrais termes dérivés, nous allons nous intéresser à la réalisation d'une sorte de cassage sur les mots de nœuds – qui donne donc, à partir d'un mot de nœuds, un ensemble de mots –. Ce cassage doit représenter le cassage de l'expression dénotée.

Exemple 30. Nous utiliserons maintenant comme exemple courant l'expression $\mathsf{E}_2 = (a^* + a \cdot (b + 1)) \cdot (a + b)$ dont l'arbre syntaxique est présenté sur la Figure 4.6.

Pour tout nœud x de l'arbre T_{E}, définissons l'ensemble des nœuds suivants :

$$\omega(x) = \begin{cases} \{x\} & \text{si } x \text{ est une feuille ou si } x = y^*, \\ \omega(y) & \text{si } x = y \cdot z, \\ \omega(y) \cup \omega(z) & \text{si } x = y + z. \end{cases}$$

C'est-à-dire que $\omega(x)$ est l'ensemble des nœuds obtenus à partir de x dans le sous-arbre dont il est racine en :
(i) continuant dans chaque branche d'un nœud étiqueté $+$;
(ii) continuant dans la branche gauche d'un nœud $\cdot$;
(iii) s'arrêtant sur tout autre nœud – feuille ou nœud $*$ –.

Exemple 31 (*Ex. 30 cont.*)**.**

$$\omega(x_1) = \omega(x_2) = \{x_4, x_9\}\,, \quad \omega(x_3) = \{x_6, x_7\}\,, \quad \omega(x_{10}) = \{x_{11}, x_{12}\}\ .$$

Nous allons maintenant introduire le cassage $\mathrm{B}(X)$ d'un mot X sur les nœuds. Comme un mot sur les nœuds représente une concaténation, l'idée est de 'casser' le premier terme de cette concaténation, c'est-à-dire la première lettre du mot. Si ce cassage donne le mot vide, alors il faut casser également la deuxième lettre et ainsi de suite.

Nous donnons donc une définition inductive :
(i) si $X = \varepsilon$, alors $\mathrm{B}(X) = \{\varepsilon\}$;
(ii) si $X = x_1 :: \cdots :: x_n$ alors :

$$\begin{aligned} \mathrm{B}(X) = &\{y :: \pi_{x_1}(y) :: x_2 ... x_n \mid y \in \omega(x_1), [\![y]\!] \neq 1\} \\ &\cup \{\mathrm{B}(\pi_{x_1}(y) :: x_2 ... x_n) \mid y \in \omega(x_1), [\![y]\!] = 1\} \end{aligned} \tag{4.1}$$

Exemple 32 (*Ex. 30 cont.*)**.**

$$\begin{aligned}\mathrm{B}(x_1) &= \{x_4 :: \pi_{x_1}(x_4), x_9 :: \pi_{x_1}(x_9)\} = \{x_4 :: x_3, x_9 :: x_{10} :: x_3\} \ , \\ \mathrm{B}(\pi(x_9)) &= \mathrm{B}(x_{10} :: x_3) \\ &= \{x_{11} :: \pi_{x_{10}}(x_{11}) :: x_3, \mathrm{B}(\pi_{x_{10}}(x_{12}) :: x_3)\} \\ &= \{x_{11} :: x_3, x_6, x_7\} \ .\end{aligned}$$

Exemple 33 (*Ex. 30 cont.*)**.** Comme pour les termes dérivés, il est possible de voir les opérations qui nous intéressent pour le calcul des termes dérivés cassés de manière graphique sur l'arbre syntaxique.

Ainsi, pour réaliser le cassage des chemins – que nous avons définis à l'aide de liens sur l'arbre dans le cas des termes non cassés –, nous allons introduire la notion de lien secondaires – tous les liens précédemment définis existent sous l'appellation lien primaires –.

Ces liens secondaires partent de chaque nœud duquel part un lien primaire, et, si y est le nœud destination de ce lien primaire, alors les liens secondaires pointent vers les $\omega(y)$. En particulier, les liens secondaires qui partent d'un nœud sous une étoile se confondent avec le lien primaire.

Les chemins cassés sont alors définis comme les chemins qui :

(i) partent du nœud qui nous intéresse ;
(ii) remontent vers la racine ;
(iii) empruntent un lien secondaire dès que possible jusqu'à en avoir emprunté un ne pointant pas vers 1 ; à partir de là :
(iv) empruntent les liens primaires dès que possible.

Comme pour les termes dérivés, chaque chemin est représenté par le mot composé par les extrémités de liens primaires et secondaires. L'ensemble des mots correspondant aux liens secondaires qui partent d'un nœud x forme l'ensemble des chemins cassés de x : $\mathrm{B}(x)$.

En reprenant l'exemple $\mathsf{E}_2 = (a^* + a \cdot (b + 1)) \cdot (a + b)$, la Figure 4.7 montre les liens primaires (en continu) et secondaires (en pointillés).

La définition donnée ci-dessus est justifiée par le théorème suivant :

Théorème 39. *Soit X un mot sur les nœuds de l'arbre syntaxique T_{E} d'une expression rationnelle E. Nous avons :*

$$\mathrm{B}(\llbracket X \rrbracket) = \bigcup_{Y \in \mathrm{B}(X)} \llbracket Y \rrbracket = \llbracket \mathrm{B}(X) \rrbracket$$

Démonstration. Par induction sur la profondeur de $\llbracket X \rrbracket$. Nous supposons que X est propre. Nous avons déjà vu qu'il est facile de transformer un mot en mot propre.

Si $[\![X]\!] = 1$, alors, comme X est propre, $X = \varepsilon$ et $\mathrm{B}(X) = \{\varepsilon\}$ et vérifie bien $\mathrm{B}([\![X]\!]) = \bigcup_{Y \in \mathrm{B}(X)} [\![Y]\!]$.

Si $[\![X]\!] = a$ **ou** $[\![X]\!] = \mathsf{F}^*$ alors $X = x_1$ où x_1 est un nœud étiqueté par a ou par une étoile. On a alors $\omega(x_1) = \{x_1\}$ et donc $\mathrm{B}(X) = \{x_1\}$ ce qui termine ce cas.

Si $[\![X]\!] = \mathsf{F} + \mathsf{G}$: alors $X = x_1$, où $x_1 = y_1 + y_2$, avec y_1 dénotant F et y_2 dénotant G. Dans ce cas, on a $\omega(x_1) = \omega(y_1) \cup \omega(y_2)$. Et comme pour tout $z \in \mathrm{T}_{y_i}$, on a $\pi_{x_1}(z) = \pi_{y_i}(z)$, alors $\mathrm{B}(X) = \mathrm{B}(y_1) \cup \mathrm{B}(y_2)$ et donc $\mathrm{B}([\![X]\!]) = \mathrm{B}([\![y_1]\!]) \cup \mathrm{B}([\![y_2]\!])u$.

Si $[\![X]\!] = \mathsf{F} \cdot \mathsf{G}$: on distingue deux cas suivant la longueur de X :

soit $X = x_1 :: \cdots :: x_n$, avec $n \geqslant 2$. De la définition du cassage, on a :

$$\begin{aligned}\mathrm{B}(x_1 :: x_2 :: ... :: x_n) &= \mathrm{B}(x_1) :: (x_2 :: ... :: x_n) \\ &\quad \cup\ \delta_{\mathrm{B}(x_1)} :: \mathrm{B}(x_2 :: ... :: x_n) \\ &= \mathrm{B}(x_1 :: ... :: x_{n-1}) :: x_n \\ &\quad \cup\ \delta_{\mathrm{B}(x_1 :: ... :: x_{n-1})} \mathrm{B}(x_n)\end{aligned}$$

,

alors, par récurrence, $\mathrm{B}(X) = \mathrm{B}\left([\![x_1 :: \cdots :: x_{n-1}]\!]\right) \cdot [\![x_n]\!] \cup \delta_{\mathrm{B}([\![x_1 :: \cdots :: x_{n-1}]\!])} \mathrm{B}\left([\![x_n]\!]\right)$.

soit $X = x_1$, alors comme $[\![X]\!]$ est une concaténation, le nœud x_1 est une concaténation : $x_1 = y_1 \cdot y_2$ et $\omega(x_1) = \omega(y_1)$. De plus, pour $z \in \mathrm{T}_{y_1}$, $\pi_{x_1}(z) = \pi_{y_1}(z) :: y_2$. On a alors :

$$\begin{aligned}\mathrm{B}(X) = &\{z :: \pi_{y_1}(z) :: y_2 \mid z \in \omega(y_1), [\![z]\!] \neq 1\} \\ &\cup\ \{\mathrm{B}(\pi_{y_1}(z) :: y_2) \mid z \in \omega(y_1), [\![z]\!] = 1\}\end{aligned}$$

ce qui correspond au premier cas avec $X = y_1 :: y_2$.

□

La Propriété 22 permet d'appliquer le théorème ci-dessus aux résultats obtenus pour les termes dérivés et d'obtenir directement :

Théorème 40. *Soit* E *une expression rationnelle dont l'arbre syntaxique* T_{E} *a pour racine* r *et soit* X *un mot sur les nœuds :*

$$\frac{\partial_{\mathsf{b}}}{\partial a}[\![X]\!] = \bigcup_{l \in \mathsf{First}(r,X)} \{[\![X]\!] \mid X \in \mathrm{B}(\pi(l))\} \quad .$$

et :

Théorème 41.

$$\mathrm{TBD}\,(\mathsf{E}) = \bigcup_{l \in \mathrm{fp}(\mathsf{E})} \{[\![X]\!] \mid X \in \mathrm{B}(\pi(l))\} \quad .$$

Construction de l'automate des termes dérivés cassés

Comme dans le cas des termes dérivés, nous venons de montrer que les vrais termes dérivés cassés peuvent être calculés par un mot sur les nœuds de l'arbre. Il est donc possible de représenter de manière unique les états de l'automate des termes dérivés cassés – il faut rajouter le cassage $\mathrm{B}(r)$ de l'expression elle-même, qui sera également l'ensemble des états initiaux – avec des mots réduits sur les marques.

Cependant la méthode diffère un peu de celle des termes dérivés car le cassage du chemin d'une feuille propre l peut donner plusieurs mots de marques réduits différents. Il ne faut donc plus identifier les états de l'automate a une classe de feuilles dont les chemins marqués et réduits sont identique mais directement les identifier aux différents chemins dont la réduction est identique.

C'est-à-dire que pour chaque état q on associe $\theta_b(q)$ l'ensemble des mots X tels qu'il existe une feuille propre l telle que $X \in \mathrm{B}(\pi(l))$ et $\rho\left(\overline{X}\right) = q$. En particulier, dans ce cas, $\mathsf{First}(q)$ est alors défini comme le First d'un représentant, c'est-à-dire qu'on choisit un X arbitrairement dans $\theta_b(q)$ et $\mathsf{First}(q) = \mathsf{First}(X)$.

L'application des théorèmes de la sous-section précédente permet de montrer directement, en appliquant les résultats montrés pour les termes dérivés, qu'il y a une transition dans $\mathcal{D}_b(\mathsf{E})$ de q_i à q_j étiquetée par a si et seulement si il existe une feuille $l \in \mathsf{First}(q_i, a)$ et un mot sur les nœud $X \in \mathrm{B}(\pi(l)) \cap \theta_b(q_j)$.

En résumé pour calculer l'automate des termes dérivés cassés d'une expression rationnelle E nous suivons les différentes étapes suivantes :

1. calcul pour tout nœud x de l'arbre syntaxique – par un parcours de bas en haut – :
   - du terme constant $\mathsf{c}(x)$;
   - de l'ensemble des feuilles propres $\mathsf{First}(x)$;
   - de la marque associée $\overline{x}$;
2. calcul pour tout nœud x de l'arbre – par un parcours de haut en bas – de l'ensemble $\omega(x)$;
3. calcul du chemin $\pi(l)$ associé à toute feuille propre l de T_{E} ; calcul du cassage $\mathrm{B}(\pi(l))$ et réduction préfixe des mots ainsi obtenus, pour

chaque mot X il faut retenir l'ensemble des feuilles l telles que $X \in \mathrm{B}(\pi(l))$;

4. identification des mots X et X' obtenus à l'étape précédente, tels que $\rho\left(\overline{X}\right) = \rho\left(\overline{X'}\right)$. Cette étape donne les états de l'automate, il faut garder la composition de chaque classe $\theta_b(q)$ de mot ;
5. pour chaque état q, calcul de $\mathsf{First}(X)$ pour un représentant $X \in \theta_b(q)$ donné ;
6. chaque état q dont le terme constant $\mathsf{c}(q)$ est non nul est identifié comme un état final ;
7. pour chaque paire d'états (q_i, q_j), il y a une transition étiquetée par a si on trouve une feuille de $\mathsf{First}(q_i, a)$ telle qu'il existe $X \in \mathrm{B}(\pi(l)) \cap \theta_b(q_j)$.

Par rapport à l'algorithme donné pour les termes dérivés, l'algorithme des termes dérivés cassés se distingue par le fait que chaque chemin peut engendrer plusieurs chemins cassés. Cependant ces chemins sont limités par le nombre de liens secondaires, ou plutôt par le nombre de nœuds dans lesquels arrive un lien secondaire, puisqu'un chemin cassé peut être vu comme un chemin 'normal' partant d'un nœud extrémité d'un lien secondaire. Il en résulte que l'algorithme reste quadratique mais en $O(n^2)$.

Exemple 34. Prenons l'exemple très simple de l'expression $\mathsf{E}_3 = (a + b) \cdot (c + d)$ dont l'arbre est présenté sur la Figure 4.8.

Sur cet exemple nous n'avons pas a nous préoccuper de marques particulières puisque tous les sous-arbres sont différents : les noms des nœuds sont les marques.

Il existe un unique lien primaire entre x_2 et x_3. On a $\omega(x_1) = \omega(x_2) = \{x_4, x_5\}$ et $\omega(x_3) = \{x_6, x_7\}$. Il y a donc des liens secondaires entre x_2 et x_6 d'une part et x_7 d'autre part. De plus les mots $x_4 :: x_3$ et $x_5 :: x_3$ sont les états initiaux de l'automate des termes dérivés cassés.

On a également $\pi(x_4) = \pi(x_5) = x_3$ et $\pi(x_6) = \pi(x_7) = \varepsilon$. Le cassage du mot x_3 donne $\mathrm{B}(x_3) = \{x_6, x_7\}$. Les termes dérivés cassés sont donc $\{x_4 :: x_3, x_5 :: x_3, x_6, x_7, \varepsilon\}$.

Comme $\mathsf{First}(x_4 :: x_3) = \mathsf{First}(x_5 :: x_3) = \{x_6, x_7\}$, on peut construire l'automate des termes dérivés cassés de la Figure 4.9.

Remarque 8. Dans la version 1.4 de la plateforme de manipulation d'automates Vaucanson, le calcul de l'automate des termes dérivés et le calcul de l'automate des termes dérivés cassés se font par un algorithme hybride entre l'algorithme de [5] et l'algorithme de [18] : les dérivations par rapport à chaque lettre de l'alphabet sont effectuées pour chaque terme dérivé (cassé)

jusqu'à ce qu'aucun nouveau ne soit créé, comme dans la méthode d'Antimirov. Les termes dérivés (cassés) sont néanmoins représentés par une liste de nœuds de l'arbre comme dans [18]. Le premier élément de la liste est alors décomposé tant qu'il s'agit d'un produit afin de garantir l'unicité de la décomposition – raisonnement similaire à celui que nous adoptons dans la première section de ce chapitre –. L'identification des termes dérivés (cassés) égaux est alors réalisée par une reconnaissance lexicographique de la liste des éléments de la liste un à un.

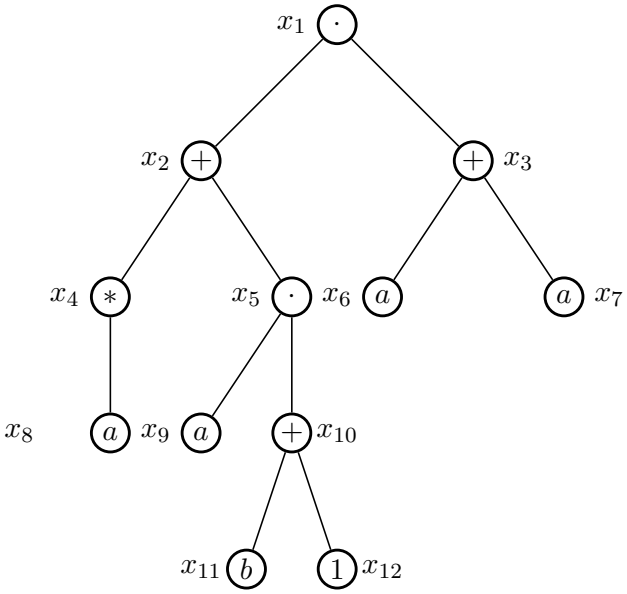

FIGURE 4.6 – l'arbre syntaxique $\mathrm{T}_{\mathsf{E}_2}$

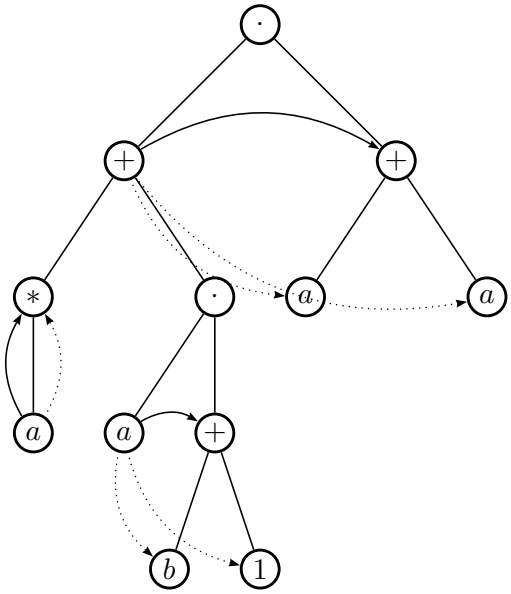

FIGURE 4.7 – l'arbre syntaxique décoré T_{E_2}

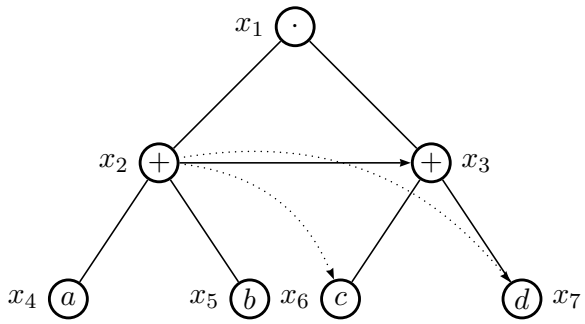

FIGURE 4.8 – L'arbre syntaxique $\mathrm{T}_{\mathsf{E}_3}$

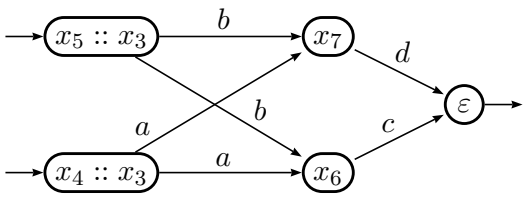

FIGURE 4.9 – L'automate des termes dérivés cassés de E_3