Devoir maison n°1bis - À rendre le 02 Novembre 2025 (pour les 5/2 surtout)

Exercice 1 : Arbres de construction de mots

Dans tout cet exercice on considère des expressions régulières et des mots sur un alphabet Σ fixé, qui sera $\{\alpha, \beta\}$ dans les exemples. En guise d'exemple on utilisera les expressions régulières suivantes.

$$f \overset{\text{déf}}{=} (\alpha\cdot\varepsilon)^\star|((\beta\cdot\alpha)|\emptyset) \text{ et } e_{\text{ex}} \overset{\text{déf}}{=} f^\star$$

Le mot $w = \alpha\alpha\beta\alpha$ est dans le langage de e_{ex} en tant que concaténation de trois mots dans le langage de f : $\alpha\alpha$, $\beta\alpha$ et ε. On remarque que w est aussi dans le langage de e_{ex} en tant que concaténation de trois autres mots dans le langage de f : α, α et $\beta\alpha$.

Ainsi, pour une expression régulière donnée, un même mot admet potentiellement plusieurs "constructions" qui justifient qu'il est dans le langage de cette expression régulière. Dans cet exercice on introduit la notion d'arbres de construction permettant de représenter la manière dont un mot est construit selon une expression régulière (comprendre : quels choix ont été faits lors de la lecture de l'expression régulière pour obtenir ce mot). On étudie ensuite des propriétés de ces arbres pour retrouver le lemme de l'étoile sans passer par les automates.

On définit par induction structurelle les **arbres de construction**, notés $\mathcal{A}$, de la manière suivante :

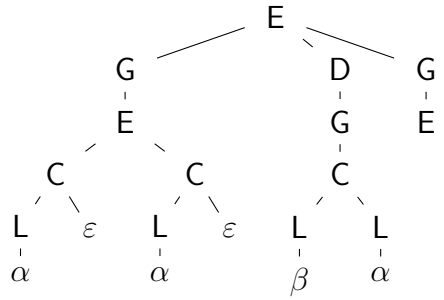
- $\varepsilon \in \mathcal{A}$;
- $l \in \mathcal{A}$, pour $l \in \Sigma$;
- Si $a \in \mathcal{A}$ alors $\mathsf{G}(a) \in \mathcal{A}$ et $\mathsf{D}(a) \in \mathcal{A}$;
- Si $a \in \mathcal{A}$ et $b \in \mathcal{A}$ alors $\mathsf{C}(a, b) \in \mathcal{A}$;
- Si $n \in \mathbb{N}$ et $(a_1, a_2, \ldots, a_n) \in \mathcal{A}^n$ alors $\mathsf{E}(a_1, a_2, \ldots, a_n) \in \mathcal{A}$.

G et D ont été choisis pour désigner Gauche et Droite ; E pour désigner Étoile, C pour désigner Concaténation.

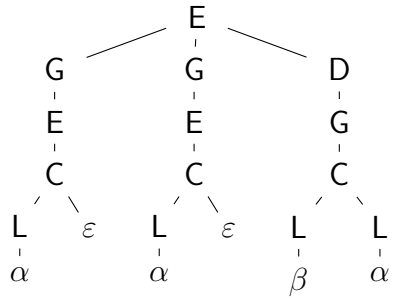
Définitions OCaml. Un fichier `compagnon_arbres_construction.ml` est fourni avec cet exercice. On y trouvera les définitions des types mentionnés dans ce sujet ainsi que les exemples de l'énoncé. Un arbre de construction est représenté en OCaml au moyen du type ci-dessous.

```
type a_cons =
  | Eps                    (* ε *)
  | L of char              (* Une lettre de Σ *)
  | G of a_cons            (* G(a) *)
  | D of a_cons            (* D(a) *)
  | C of a_cons * a_cons   (* C(a, b) *)
  | E of a_cons list       (* E(a_1, a_2, ..., a_n) *)
```

Les figures 1a et 1b représentent deux arbres de construction sur l'alphabet $\Sigma = \{\alpha, \beta\}$. Ces deux arbres de construction correspondent aux deux constructions données en début d'exercice pour justifier que $\alpha\alpha\beta\alpha$ est dans le langage de e_{ex}.



(a) arbre de construction a^1_{ex}



(b) arbre de construction a^2_{ex}

FIGURE 1 – Exemples d'arbres de construction

On définit, le **mot associé** à un arbre de construction a, noté[♣] $p(a)$, par induction sur les arbres de construction :
- $p(\varepsilon) \overset{\text{déf}}{=} \varepsilon$;
- $p(l) \overset{\text{déf}}{=} l$ pour $l \in \Sigma$;
- $p(\mathsf{G}(a)) \overset{\text{déf}}{=} p(\mathsf{D}(a)) \overset{\text{déf}}{=} p(a)$ pour $a \in \mathcal{A}$;
- $p(\mathsf{C}(a,b)) \overset{\text{déf}}{=} p(a)\cdot p(b)$ pour $a \in \mathcal{A}$ et $b \in \mathcal{A}$;
- $p(\mathsf{E}(a_1, a_2, \ldots, a_n)) \overset{\text{déf}}{=} p(a_1)\cdot p(a_2)\cdot \ldots p(a_n)$ pour $n \in \mathbb{N}$ et $(a_1, a_2, \ldots, a_n) \in \mathcal{A}^n$[♡].

Par exemple, le mot associé à l'arbre de construction a^1_{ex} de la figure 1a est $p(a^1_{\text{ex}}) = \alpha\alpha\beta\alpha$, et c'est aussi celui associé à a^2_{ex}.

Définitions OCaml. On représente les mots au moyen du type `type mot = char list`.

Q. 1 Définir une fonction `mot : a_cons -> mot` prenant en paramètre un arbre de construction et retournant le mot qui lui est associé. On assurera une complexité algorithmique linéaire en la taille de l'arbre de construction passé en paramètre.

On définit le fait qu'un arbre de construction $a \in \mathcal{A}$ soit un **modèle** d'une expression régulière $e \in \mathscr{E}_{reg}$, ce que l'on note $a \models e$, par induction sur les expressions régulières :
- $\varepsilon \models e$ si et seulement $e = \varepsilon$;
- $l \models e$ si et seulement $e = l$;
- $\mathsf{G}(b) \models e$ si et seulement si e est de la forme $f|g$ et $b \models f$;
- $\mathsf{D}(b) \models e$ si et seulement si e est de la forme $f|g$ et $b \models g$;
- $\mathsf{C}(b,c) \models e$ si et seulement si e est de la forme $f\cdot g$ et $b \models f$ et $c \models g$;
- $\mathsf{E}(a_1, a_2, \ldots, a_n) \models e$ si et seulement si e est de la forme $f^\star$ et $\forall i \in [\![1, n]\!], a_i \models f$.

Par exemple, l'arbre de construction a^1_{ex} de la figure 1a est modèle de l'expression régulière e_{ex}, mais ce n'est pas un modèle de l'expression régulière $((\alpha\cdot\varepsilon)^\star|(\beta\cdot\alpha))^\star$

Q. 2 Définir une fonction `est_modele : a_cons -> regexp -> bool` prenant en paramètres un arbre de construction a et une expression régulière e et testant si $a \models e$.

♣. p comme production
♡. Avec le cas particulier, pour $n = 0$, que $p(a_1)\cdot p(a_2)\cdot \ldots p(a_n) = \varepsilon$

Solution

```
let rec est_modele (a: a_cons) (e: regexp): bool =
  match a, e with
  | Eps, Epsilon          -> true
  | L(l), Lettre(c)       -> l = c
  | G(b), Union(g, h)     -> est_modele b g
  | D(b), Union(g, h)     -> est_modele b h
  | C(b, c), Concat(g, h) -> (est_modele b g) && (est_modele c h)
  | E(l), Etoile(f)       -> List.for_all (fun b -> est_modele b f) l
  | _                     -> false
```

Dans la suite, lorsque e est une expression régulière, on note $\mathcal{M}(e)$ **l'ensemble de ses modèles**, à savoir :

$$\mathcal{M}(e) \overset{\text{déf}}{=} \{a \in \mathcal{A} \mid a \models e\}.$$

Q. 3 Pour f et g deux expressions régulières, exprimer $\mathcal{M}(f \cdot g)$, $\mathcal{M}(f|g)$ et $\mathcal{M}(f^\star)$ en fonction de $\mathcal{M}(f)$ et $\mathcal{M}(g)$.

Solution

$$\begin{aligned}
\mathcal{M}(f \cdot g) &= \{\mathsf{C}(a,b) \mid a \in \mathcal{M}(f), b \in \mathcal{M}(g)\} \\
\mathcal{M}(f|g) &= \{\mathsf{G}(a) \mid a \in \mathcal{M}(f)\} \cup \{\mathsf{D}(b) \mid b \in \mathcal{M}(g)\} \\
\mathcal{M}(f^\star) &= \{\mathsf{E}(a_1, a_2, \ldots, a_n) \mid n \in \mathbb{N}, (a_1, a_2, \ldots, a_n) \in \mathcal{M}(f)^n\}
\end{aligned}$$

Q. 4 Démontrer que pour toute expression régulière $e \in \mathscr{E}_{reg}$, $\mathcal{L}(e) = \{p(a) \mid a \in \mathcal{M}(e)\}$.

Solution

On montre ce résultat par induction structurelle. Soit $P(e) : \mathcal{L}(e) = \{p(a) \mid a \in \mathcal{A}, a \models e\}$.

- Montrons $P(\varepsilon)$. D'une part : $\mathcal{L}(\varepsilon) = \{\varepsilon\}$. D'autre part : $\mathcal{M}(\varepsilon) = \{a \in \mathcal{A} \mid a \models \varepsilon\} = \{\varepsilon\}$, or $p(\varepsilon) = \varepsilon$. Finalement $\{p(a) \mid a \in \mathcal{A}, a \models \varepsilon\} = \{\varepsilon\}$ d'où le résultat.
- Montrons $P(\emptyset)$. D'une part : $\mathcal{L}(\emptyset) = \emptyset$. D'autre part : $\mathcal{M}(\emptyset) = \{a \in \mathcal{A} \mid a \models \emptyset\} = \emptyset$. Finalement $\{p(a) \mid a \in \mathcal{A}, a \models \emptyset\} = \emptyset$ d'où le résultat.
- Soit $l \in \Sigma$, montrons $P(l)$. D'une part : $\mathcal{L}(l) = \{l\}$. D'autre part : $\mathcal{M}(l) = \{a \in \mathcal{A} \mid a \models l\} = \{l\}$, or $p(l) = l$. Finalement $\{p(a) \mid a \in \mathcal{A}, a \models l\} = \{l\}$ d'où le résultat.
- Soit $(e, f) \in \mathscr{E}_{reg}^2$, supposons $P(e)$ et $P(f)$, montrons que $P(e|f)$. D'une part : $\mathcal{L}(e|f) = \mathcal{L}(e) \cup \mathcal{L}(f)$. D'autre part : $\mathcal{M}(e|f) = \{a \in \mathcal{A} \mid a \models e|f\} = \{\mathsf{G}(b) \mid b \models e\} \cup \{\mathsf{D}(b) \mid b \models f\}$. Finalement :

  $$\begin{aligned}
  \{p(a) \mid a \in \mathcal{A}, a \models e|f\} &= \{p(\mathsf{G}(b)) \mid b \models e\} \cup \{p(\mathsf{D}(b)) \mid b \models f\} \\
  &= \{p(b) \mid b \models e\} \cup \{p(b) \mid b \models f\} \\
  &= \mathcal{L}(e) \cup \mathcal{L}(f) \qquad \text{par hypothèses } P(e) \text{ et } P(f)
  \end{aligned}$$

  D'où le résultat.
- Soit $(e, f) \in \mathscr{E}_{reg}^2$, supposons $P(e)$ et $P(f)$, montrons que $P(e \cdot f)$. D'une part : $\mathcal{L}(e \cdot f) =$

$\mathcal{L}(e)\cdot\mathcal{L}(f)$. D'autre part : $\mathcal{M}(e\cdot f) = \{\mathsf{C}(b,c) \mid b \models e, c \models f\}$. Finalement :

$$\begin{aligned}\{p(a) \mid a, a \models e\cdot f\} &= \{p(\mathsf{C}(b,c)) \mid b \models e, c \models f\}\\ &= \{p(b)\cdot p(c) \mid b \models e, c \models f\}\\ &= \{p(b) \mid b \models e\}\cdot\{p(c) \mid c \models f\}\\ &= \mathcal{L}(e)\cdot\mathcal{L}(f) \qquad \text{par hypothèses } P(e) \text{ et } P(f)\end{aligned}$$

D'où le résultat.

- Soit $e \in \mathscr{E}_{reg}$, supposons $P(e)$, montrons que $P(e^\star)$. D'une part : $\mathcal{L}(e^\star) = \mathcal{L}(e)^\star$. D'autre part : $\mathcal{M}(e^\star) = \{E(a_1, a_2, \ldots, a_n) \mid n \in \mathbb{N}, (a_i)_{i\in[\![1,n]\!]} \in \mathcal{M}(e)^n\}$. Finalement :

$$\begin{aligned}\{p(a) \mid a, a \models e^\star\} &= \{p(E(a_1, a_2, \ldots, a_n)) \mid n \in \mathbb{N}, (a_i)_{i\in[\![1,n]\!]} \in \mathcal{M}(e)^n\}\\ &= \{p(a_1)\cdot p(a_2)\cdot\ldots\cdot p(a_n) \mid n \in \mathbb{N}, (a_i)_{i\in[\![1,n]\!]} \in \mathcal{M}(e)^n\}\\ &= \bigcup_{n\in\mathbb{N}} \{p(a_1)\cdot p(a_2)\cdot\ldots\cdot p(a_n) \mid (a_i)_{i\in[\![1,n]\!]} \in \mathcal{M}(e)^n\}\\ &= \bigcup_{n\in\mathbb{N}} \{p(a_1) \mid a_1 \in \mathcal{M}(e)\}\cdot\{p(a_2) \mid a_2 \in \mathcal{M}(e)\}\cdot\ldots\{p(a_n) \mid a_n \in \mathcal{M}(e)\}\\ &= \bigcup_{n\in\mathbb{N}} \underbrace{\mathcal{L}(e)\cdot\mathcal{L}(e)\cdot\ldots\cdot\mathcal{L}(e)}_{n \text{ fois}}\\ &= \mathcal{L}(e)^\star\end{aligned}$$

D'où le résultat.

On dit d'un arbre de construction $a \in \mathcal{A}$ qu'il est **sans étoile productive**, dès lors que pour tout sous-arbre b de l'arbre a, s'il existe $n \in \mathbb{N}$, tel que $b = \mathsf{E}(a_1, a_2, \ldots, a_n)$ alors $\forall i \in [\![1, n]\!], p(a_i) = \varepsilon$.
On rappelle que la **taille** d'une expression régulière $e \in \mathscr{E}_{reg}$, notée $\|e\|$ dans cet exercice, est le nombre de ses connecteurs, constantes et lettres comptées avec multiplicité.

Q. 5 Montrer que pour toute expression régulière $e \in \mathscr{E}_{reg}$, pour tout arbre de construction $a \in \mathcal{A}$ qui est un modèle de e, si a est sans étoile productive, alors $|p(a)| \leqslant \|e\|$.

Solution

Pour $e \in \mathscr{E}_{reg}$, on définit la propriété $P(e)$ par :

$$\forall a \in \mathcal{M}(e), a \text{ est sans étoile productive} \Rightarrow |p(a)| \leqslant \|e\|.$$

Montrons, par induction structurelle sur l'espace $\mathscr{E}_{reg}$, que $\forall e \in \mathscr{E}_{reg}, P(e)$, ce qui est exactement ce que l'on souhaite démontrer.

- Montrons $P(\varepsilon)$. Soit $a \in \mathcal{M}(\varepsilon)$, alors $a = \varepsilon$ et $p(a) = \varepsilon$, ainsi $|p(a)| = 0 \leqslant \|\varepsilon\| = 1$.
- Montrons $P(\emptyset)$. $\mathcal{M}(e) = \emptyset$ et donc $P(\emptyset)$ est trivialement vraie.
- Soit $l \in \Sigma$, montrons $P(l)$. Soit $a \in \mathcal{M}(l)$, alors $a = l$ et $p(a) = l$, ainsi $|p(a)| = 1 \leqslant \|l\| = 1$.
- Soit $(e, f) \in \mathscr{E}_{reg}^2$, supposons $P(e)$ et $P(f)$, montrons que $P(e|f)$. Soit $a \in \mathcal{M}(e|f)$, sans étoile productive, alors $a = \mathsf{G}(b)$ avec $b \models e$ ou $a = \mathsf{D}(b)$ avec $b \models f$. Sans perdre en généralité, supposons $a = \mathsf{G}(b)$ avec $b \models e$. Puisque a est sans étoile productive, b est sans étoile productive en tant que sous-arbre de a, ainsi par hypothèse inductive $P(e)$ appliquée au modèle b : $|p(b)| \leqslant \|e\|$. Or $|p(a)| = \|p(\mathsf{G}(b))\| = |p(b)| \leqslant \|e\| \leqslant \|e\| + \|f\| + 1 = \|e|f\|$, d'où le résultat.
- Soit $(e, f) \in \mathscr{E}_{reg}^2$, supposons $P(e)$ et $P(f)$, montrons que $P(e\cdot f)$. Soit $a \in \mathcal{M}(e\cdot f)$, sans étoile productive, alors $a = \mathsf{C}(b, c)$ avec $b \models e$ et $c \models f$. Puisque a est sans étoile productive, b et c le sont aussi en tant que sous-arbres de a, ainsi par hypothèse inductive $P(e)$ appliquée au

modèle b : $|p(b)| \leqslant \|e\|$ et par hypothèse inductive $P(f)$ appliquée au modèle c : $|p(c)| \leqslant \|f\|$. Or $|p(a)| = |p(\mathsf{C}(b,c))| = |p(b)\cdot p(c)| = |p(b)| + |p(c)| \leqslant \|e\| + \|f\| \leqslant \|e\| + \|f\| + 1 = \|e\cdot f\|$, d'où le résultat.

- Soit $e \in \mathscr{E}_{reg}$, supposons $P(e)$, montrons que $P(e^\star)$. Soit $a \in \mathcal{M}(e^\star)$, sans étoile productive, soient donc $n \in \mathbb{N}$ et $(a_1, a_2, \ldots, a_n) \in \mathcal{M}(e)^n$, $a = \mathsf{E}(a_1, a_2, \ldots, a_n)$. Puisque a est sans étoile productive, $\forall i \in [\![1, n]\!], p(a_i) = \varepsilon$. Or $|p(a)| = |p(\mathsf{E}(a_1, a_2, \ldots, a_n))| = |p(a_1)\cdot p(a_2)\cdot \ldots \cdot p(a_n)| = |\varepsilon| = 0 \leqslant \|e^\star\|$, d'où le résultat.

Q. 6 Montrer que pour tout arbre de construction $a \in \mathcal{A}$ et tout sous-arbre b de a, il existe deux mots x et y tels que :

(i) $p(a) = x\cdot p(b)\cdot y$;

(ii) pour tout arbre de construction b', l'arbre a' obtenu en remplaçant, dans a, le sous-arbre b par b' vérifie $p(a') = x\cdot p(b')\cdot y$.

Solution

Soit $b \in \mathcal{A}$ un arbre de construction. Pour a un arbre de construction, on définit la propriété $P(a)$ par :

$$b \text{ est un sous-arbre de } a \Rightarrow \text{ il existe deux mots } x \text{ et } y \text{ tels que } (i) \text{ et } (ii).$$

Dans toute cette preuve si a est un arbre de construction, b est un sous-arbre de a et b' un arbre de construction, alors a' dénote l'arbre obtenu en remplaçant le sous-arbre b par b' dans l'arbre a.

On démontre alors $\forall a \in \mathcal{A}, P(a)$ par induction structurelle.

Dans tous les cas de la preuve par induction on doit traiter le sous-cas où $b = a$, traitons donc une fois pour toute ce cas ici et dans la suite on supposera que b est un sous-arbre strict des arbres considérés. En pareil cas ($b = a$), en considérant $x \overset{\text{déf}}{=} y \overset{\text{déf}}{=} \varepsilon$, il est clair que $p(a) = x\cdot p(b)\cdot y$ et que pour tout arbre b', $p(a') = p(b') = x\cdot p(b')\cdot y$.

- Montrons $P(\varepsilon)$. ε n'a pas de sous-arbre strict ainsi ce cas est traité par le cas $b = a$.
- Soit $l \in \Sigma$, montrons $P(l)$. l n'a pas de sous-arbre strict ainsi ce cas est traité par le cas $b = a$.
- Soit $c \in \mathcal{A}$ tel que $P(c)$, notons $a = \mathsf{G}(c)$ et montrons $P(a)$. Si b est un sous-arbre strict de a, alors b est un sous-arbre de c. Par hypothèse d'induction $P(c)$, soient deux mots x et y tels que (i) et (ii) pour c. Montrons alors (i) et (ii) pour a.
  (i) $p(\mathsf{G}(c)) = p(c) = x\cdot p(b)\cdot y$;
  (ii) Soit alors b' un arbre de construction, puisque b est un sous-arbre strict de a, $a' = \mathsf{G}(c')$ où c' est l'arbre obtenu en remplaçant le sous-arbre b de c par b'. Ainsi $p(a') = p(\mathsf{G}(c')) = p(c') = x\cdot p(b')\cdot y$ par (ii) pour c.
- De même pour $P(\mathsf{D}(c))$.
- Soient $c \in \mathcal{A}$ et $d \in \mathcal{A}$ tels que $P(c)$ et $P(d)$, notons $a = \mathsf{C}(c,d)$, montrons $P(a)$. Si b est un sous-arbre strict de a, b est un sous-arbre de c ou d sans perdre en généralité plaçons nous dans le premier cas. Par hypothèse d'induction $P(c)$, soient deux mots x et y tels que (i) et (ii) pour c. Notons alors $x' \overset{\text{déf}}{=} x$ et $y' \overset{\text{déf}}{=} y\cdot p(d)$ et montrons (i) et (ii) pour a avec de tels x' et y'.
  (i) $p(\mathsf{C}(c,d)) = p(c)\cdot p(d) = x\cdot p(b)\cdot y\cdot p(d) = x'\cdot p(b)\cdot y'$;
  (ii) Soit alors b' un arbre de construction, $a' = \mathsf{C}(c', d)$ où c' est l'arbre obtenu en remplaçant le sous-arbre b de c par b'. Ainsi $p(a') = p(\mathsf{C}(c', d)) = p(c')\cdot p(d) = x\cdot p(b')\cdot y\cdot p(d) = x'\cdot p(b')\cdot y'$ par (ii) pour c.
- Soient $n \in \mathbb{N}$, $(a_1, a_2, \ldots, a_n) \in \mathcal{A}^n$ tels que $\forall i \in [\![1, n]\!], P(a_i)$, notons $a = \mathsf{E}(a_1, a_2, \ldots, a_n)$ montrons $P(a)$. Si b est un sous-arbre strict de $\mathsf{E}(a_1, a_2, \ldots, a_n)$, b est un sous-arbre de a_i pour un certain $i \in [\![1, n]\!]$. Par hypothèse d'induction $P(a_i)$, soient deux mots x et y tels que

(i) et (ii) pour a_i. Notons alors $x' \overset{\text{déf}}{=} p(a_1)\cdot\ldots\cdot p(a_{i-1})\cdot x$ et $y' \overset{\text{déf}}{=} y\cdot p(a_{i+1})\cdot\ldots\cdot p(a_n)$ et montrons (i) et (ii) pour a avec de tels x' et y'.

- (i) $p(\mathsf{E}(a_1,a_2,\ldots,a_n)) = p(a_1)\cdot\ldots\cdot p(a_{i-1})\cdot p(a_i)\cdot p(a_{i+1})\cdot\ldots\cdot p(a_n) = p(a_1)\cdot\ldots\cdot p(a_{i-1})\cdot x\cdot p(b)\cdot y\cdot p(a_{i+1})\cdot\ldots\cdot p(a_n) = x'\cdot p(b)\cdot y'$;
- (ii) Soit alors b' un arbre de construction, $a' = \mathsf{E}(a_1,\ldots,a_{i-1},a'_i,a_{i+1},\ldots,a_n)$ où a'_i est l'arbre obtenu en remplaçant le sous-arbre b de a_i par b'. Ainsi $p(a') = p(a_1)\cdot\ldots\cdot p(a_{i-1})\cdot p(a'_i)\cdot p(a_{i+1})\cdot\ldots\cdot p(a_n) = p(a_1)\cdot\ldots\cdot p(a_{i-1})\cdot x\cdot p(b')\cdot y\cdot p(a_{i+1})\cdot\ldots\cdot p(a_n) = x'\cdot p(b')\cdot y'$ par (ii) pour c.

Ainsi $\forall a\in\mathcal{A}, P(a)$ d'où le résultat.

De la même manière on peut montrer le résultat suivant, que l'on admettra dans la suite.

Proposition 0.1

Si a est modèle d'une expression régulière e et si $b = E(a_1,a_2,\ldots,a_n)$ est un sous-arbre de a, alors en notant $b' = E(a_1,a_2,\ldots,a_{i-1},\underbrace{a_i,a_i,\ldots,a_i}_{k\text{ fois}},\ldots,a_n)$, et a' l'arbre obtenu en remplaçant, dans a, le sous-arbre b par b', a' est modèle de e. Précisons que ceci est vrai pour tout $i\in[\![1,n]\!]$ et pour tout $k\in\mathbb{N}$, y compris 0.

Q. 7 Définir une fonction `iter_etoile : a_cons -> int -> int -> a_cons option` prenant en paramètres un arbre a, un entier i et un entier k et retournant l'arbre a' obtenu en remplaçant le premier (dans l'ordre préfixe) sous-arbre de a de la forme $\mathsf{E}(a_1,a_2,\ldots,a_n)$ par l'arbre $\mathsf{E}(a_1,a_2,\ldots,a_{i-1},\underbrace{a_i,a_i,\ldots,a_i}_{k\text{ fois}},\ldots,a_n)$ (on peut supposer que $i\leqslant n$). Si aucun sous-arbre de l'arbre a n'est de la forme $\mathsf{E}(\ldots)$, la fonction devra retourner `None`.

Solution

Première version, proche du programme.

```
(** Calcule la liste l' obtenue en recopiant k fois le i-ème élément de la
    liste l. *)
let rec iter_i_k_fois (l: 'a list) (i: int) (k: int): 'a list =
  match l, i with
  | []   , _ -> failwith "i est trop grand pour la première étoile trouvée"
  | x::l', 0 -> (List.init k (fun _ -> x)) @ l'
  | x::l', _ -> x :: (iter_i_k_fois l' (i-1) k)

let rec iter_etoile (a: a_cons) (i: int) (k: int): a_cons option =
  match a with
  | Eps | L _ -> None
  | C(g, d) ->
    begin
      match iter_etoile g i k with
      | None ->
        begin match iter_etoile d i k with
          | None      -> None
          | Some(d') -> Some(C(g, d'))
        end
      | Some(g') -> Some(C(g', d))
    end
```

```
  | G(b) ->
    begin
      match iter_etoile b i k with
      | None -> None
      | Some(b') -> Some(G(b'))
    end
  | D(b) ->
    begin
      match iter_etoile b i k with
      | None -> None
      | Some(b') -> Some(D(b'))
    end
  | E(l) ->
    Some(E(iter_i_k_fois l i k))
```

Une seconde version, plus légère, au moyen de la fonction `option_cases`.

```
let option_cases (n: 'b) (f: 'a -> 'b) (o: 'a option): 'b =
  match o with
  | None -> n
  | Some(a) -> f a

let rec iter_etoile (a: a_cons) (i: int) (k: int): a_cons option =
  match a with
  | Eps | L _ -> None
  | C(g, d) ->
    option_cases
      (option_cases None (fun d' -> Some(C(g, d'))) (iter_etoile d i k))
      (fun g' -> Some(C(g', d)))
      (iter_etoile g i k)
  | G(b) -> option_cases None (fun b' -> Some(G(b'))) (iter_etoile b i k)
  | D(b) -> option_cases None (fun b' -> Some(D(b'))) (iter_etoile b i k)
  | E(l) -> Some(E(iter_i_k_fois l i k))

```

Q. 8 En utilisant la fonction `iter_etoile` de la **Q. 7**, proposer un jeu de tests permettant de se persuader de la véracité de la propriété 0.1.

Q. 9 Déduire de la propriété 0.1 que si L est un langage régulier, alors il existe $N \in \mathbb{N}$, tel que pour tout mot $w \in L$ de longueur strictement supérieure N, il existe trois mots $(x, y, z) \in (\Sigma^\star)^3$ tels que :

a) $x \cdot y \cdot z = w$;
b) $y \neq \varepsilon$;
c) $\forall k \in \mathbb{N}, x \cdot y^k \cdot z \in L$.

Solution

Soit L un langage régulier, soit donc $e \in \mathscr{E}_{reg}$ tel que $\mathscr{L}(e) = L$. Soit $N = \|e\|$. Par disjonction de cas :

- Si L ne contient aucun mot de longueur strictement supérieure à N alors le résultat de l'énoncé est trivialement vrai.
- Sinon, soit m un mot de longueur strictement supérieure à N. De **Q. 4**, soit a un arbre de

construction modèle de e tel que $p(a) = m$. Par contraposée de **Q. 5**, a n'est pas sans étoile productive, en effet $|p(a)| = |m| > N = \|e\|$. Par définition de "être sans étoile productive", soient $n \in \mathbb{N}$, $i \in [\![1, n]\!]$ et b un sous-arbre de a de la forme $\mathsf{E}(a_1, a_2, \ldots, a_n)$ avec $p(a_i) \neq \varepsilon$. Nommons $y = p(a_i)$. b étant un sous-arbre de a, de **Q. 6**, $p(b)$ est un facteur de $p(a)$, soient donc x_0 et z_0 tels que (i) et (ii) pour a.

$$\begin{aligned} m = p(a) &= x_0 \cdot p(b) \cdot z_0 \\ &= \underbrace{x_0 \cdot p(a_1) \cdot \ldots \cdot p(a_{i-1})}_{\overset{\text{déf}}{=} x} \cdot \underbrace{p(a_i)}_{=y} \cdot \underbrace{p(a_{i+1}) \cdot \ldots p(a_n) \cdot z_0}_{\overset{\text{déf}}{=} z}. \end{aligned}$$

De tels x, y et z vérifient bien les propriétés $a)$ et $b)$. Par ailleurs, soit $k \in \mathbb{N}$, et a' l'arbre obtenu en remplaçant le sous-arbre b de a par le sous-arbre $b' \overset{\text{déf}}{=} \mathsf{E}(a_1, a_2, \ldots, a_{i-1}, \underbrace{a_i, a_i \ldots, a_i}_{k \text{ fois}}, a_{i+1} \ldots, a_n)$. De (ii) pour a : $p(a') = x_0 \cdot p(b') \cdot z_0 = x_0 \cdot p(a_1) \cdot \ldots \cdot p(a_{i-1}) \cdot \underbrace{p(a_i) \cdot p(a_i) \cdot \ldots \cdot p(a_i)}_{k \text{ fois}} p(a_{i+1}) \cdot \ldots \cdot p(a_n) z_0 = x y^k z$. Du résultat admis dans l'énoncé, a' est modèle de e et donc $p(a') \in L$, ce qui conclut la preuve.

Q. 10 Définir une fonction `trouve_xyz : a_cons -> (mot * mot * mot) option` prenant en paramètre un arbre de construction a et retournant un triplet de mots (x, y, z) (avec $y \neq \varepsilon$) tel que pour tout $k \in \mathbb{N}$, xy^kz est un mot du langage de tout expression régulière dont a est un modèle. Si aucun tel triplet n'existe, la fonction devra renvoyer `None`.

Solution

```
type reponse =
  | Mot of mot
  | XYZ of mot * mot * mot

let concat_reponses (r1: reponse) (r2: reponse): reponse =
  match r1, r2 with
  | Mot(u), Mot(v) -> Mot(u @ v)
  | XYZ(x, y, z), Mot(v) -> XYZ(x, y, z @ v)
  | Mot(u), XYZ(x, y, z) -> XYZ(u @ x, y, z)
  | XYZ(x, y, z), XYZ(x', y', z') -> XYZ(x, y, z @ x' @ z')

exception Trouve_NV of mot      (* Un mot non vide a été trouvé *)

(** (trouve_mot_ou_xyz a) calcule un triplet de mots (x, y, z) avec y non
    vide tel que pour tout k (x y^k z) est un mot du langage de tout
    expression régulière dont a est un modèle. Si aucun tel triplet ne peut
    être trouvé, retourne la production de l'arbre a. *)
let trouve_xyz (a: a_cons): (mot * mot * mot) option =
  let rec trouve_mot_ou_xyz (a: a_cons) : reponse =
  match a with
  | Eps -> Mot([])
  | L(l) -> Mot([l])
  | G(b) | D(b) -> trouve_mot_ou_xyz b
  | C(b, c) -> concat_reponses (trouve_mot_ou_xyz b) (trouve_mot_ou_xyz c)
  | E(l) ->
```

```
    try
      List.iter (fun b ->
          match trouve_mot_ou_xyz b with
          | Mot([])      -> ()
          | Mot(w)       -> raise (Trouve_NV w)
          | XYZ(x, y, z) -> raise (Trouve_NV(x @ y @ z))
        ) l; Mot([])
    with
    | Trouve_NV(m) -> XYZ([], m, [])
  in match trouve_mot_ou_xyz a with
  | Mot(_) -> None
  | XYZ(x, y, z) -> Some(x, y, z)

```

Q. 11 Déduire de la question **Q. 9** que le langage $L \overset{\text{déf}}{=} \{\alpha^{n^2} \mid n \in \mathbb{N}\}$ n'est pas un langage régulier.

Solution

Supposons par l'absurde que le langage L soit régulier. Soit donc un entier N tel que fourni par la **Q. 9**. Considérons alors le mot $w = \alpha^{(N+1)^2} \in L$. Puisqu'il est de longueur strictement supérieure à N, il existe trois mots x, y, z tels que

a) $x \cdot y \cdot z = w$;
b) $y \neq \varepsilon$;
c) $\forall k \in \mathbb{N}, x \cdot y^k \cdot z \in L$.

Notons alors $l = |y|$. D'après *b)*, $l \neq 0$. D'après *c)*, $\forall k \in \mathbb{N}, a^{(N+1)^2+(k-1)l} \in L$, à savoir : $\forall k \in \mathbb{N}, \exists p_k \in \mathbb{N}, (N+1)^2 + (k-1)l = p_k^2$. De $l > 0$, la fonction $k \mapsto p_k$ est strictement croissante, à valeurs dans $\mathbb{N}$, il existe donc k_0 tel que $p_{k_0} > \frac{(l-1)}{2}$. On remarque alors que

$$\begin{aligned} p_{k_0+1}^2 &\geqslant (p_{k_0} + 1)^2 \\ &\geqslant p_{k_0}^2 + 2p_{k_0} + 1 \\ &> p_{k_0}^2 + l \\ &= (N+1)^2 + ((k_0+1) - 1)l \end{aligned}$$

On en déduit que $p_{k_0}^2 < (N+1)^2 + ((k_0+1)-1)l < p_{k_0+1}^2$, ce qui est absurde.