Vérification et preuve automatique d'appartenance d'un mot à une grammaire formelle.

Ulysse Durand

Les grammaires formelles

$G = (T, N_t, S, D)$ où :

- T est l'alphabet des terminaux
- N_t est l'alphabet des non terminaux
- $S \in N_t$ est l'axiome
  Notons $\Sigma := N_t \cup T$
- $D \subset (\Sigma^\star)^2$ est l'ensemble des règles de dérivation.

Definitions

- $\forall x, x' \in \Sigma^\star, x \overset{(a,b)}{\rightarrow} x' \iff \exists u, v \in \Sigma^\star / x = uav$ et $x' = ubv$
- $\rightarrow := \bigcup_{d \in D} \overset{d}{\rightarrow}$ et on note $\overset{*}{\rightarrow}$ la cloture transitive et réflexive de $\rightarrow$
- $\delta(x) := \{y \in \Sigma^\star / x \overset{*}{\rightarrow} y\}$
- $|x|_l := |\{i \in \mathbb{N} \mid x_i = l\}|$ est le nombre d'occurences de la lettre l dans x.
- Alors le langage de la grammaire formelle G est le suivant :

$$\mathcal{L}(G) := \delta(S) \cap T^\star$$

Vérification de preuve

$$D = \{d_1, \ldots, d_n\} = \{(a_1, b_1), \ldots, (a_n, b_n)\}$$

$$S \xrightarrow{d_{i_1}} m_1 \xrightarrow{d_{i_2}} \ldots \xrightarrow{d_{i_p}} m_p$$

$$m_{k|[1,j_k-1]} a_{i_k} m_{k|[|a_{i_k}|+j_k,|m_k|]} \xrightarrow{(a_{i_k},b_{i_k})} m_{k|[1,j_k-1]} b_{i_k} m_{k|[|a_{i_k}|+j_k,|m_k|]}$$

```
type 'e preuveformelle =
(('e caractere list)*int*int) list
```

`[...;(` m_k `,` i_k `,` j_k `);...]`

Vérification de preuve - Exemple

$G = (T, N_t, S, D)$ où :

- $T = \{\underline{a}, \underline{b}, \underline{c}\}$
- $N_t = \{\underline{S}, \underline{B}\}$
- $D = \{(\underline{S}, \underline{aBSc})_1, (\underline{S}, \underline{abc})_2, (\underline{Ba}, \underline{aB})_3, (\underline{Bb}, \underline{bb})_4\}$

alors $\underline{\text{ aabbcc }}$ est dans $\mathcal{L}(G)$ car
$\underline{S} \rightarrow_1 \underline{aBSc} \rightarrow_2 \underline{aBabcc} \rightarrow_3 \underline{aaBbcc} \rightarrow_4 \underline{aabbcc}$.

En Ocaml :

```
let unepreuve = [(mot "aBSc",0,0);(mot "aBabcc",1,2);
(mot "aaBbcc",2,1);(mot "aabbcc",3,2)]
```

Preuve automatique d'appartenance d'un mot

Cherchons comment dériver $\underline{\ S\ }$ en un mot m donné.

$$G_n := \{x \in \Sigma^\star \mid S(\bigcup_{0 \leq k \leq n} \rightarrow^k)x\}$$

$$G_{n+1} = \bigcup_{x \in G_n} \mathcal{S}(x) \text{ où}$$

$$\mathcal{S}(x) := \{y \in \Sigma^\star \mid x \rightarrow y\} = \bigcup_{d \in D} \{y \in \Sigma^\star \mid x \xrightarrow{d} y\}$$

Algorithme de Knuth-Morris-Pratt pour le calcul de $\mathcal{S}(x)$

On fait en réalité un parcours en largeur du graphe $(\Sigma^\star, \rightarrow)$ depuis $\underline{\ S\ }$ pour trouver un chemin vers m.

Preuve automatique d'appartenance d'un mot - Le parcours en largeur

On va éviter certains mots dans le parcours (ceux n'ayant aucune chance de dériver en m), et l'arrêter en arrivant sur *m*.

```
valide : ('e caractere list) -> bool
interdit : ('e caractere list) -> bool
```

On garde en mémoire le chemin parcouru.

Amélioration pour les grammaires croissantes

$$\forall (a, b) \in D, |a| \leq |b|$$

$$\forall x, x' \in \delta(S), x \xrightarrow{*} x' \implies |x| \leq |x'|$$

```
let interditcroiss x =
Array.length x > Array.length m
```

Amélioration dans le cas général : déduction sur le nombre d'occurence de chaque lettre

Cherchons une fonction `interdit` plus sophistiquée.

Pour tout $l \in \Sigma, x \in \Sigma^\star$, cherchons un ensemble $s_x(l)$ qui majore $|\delta(x)|_l$ ($= \{|y|_l \mid y \in \delta(x)\}$).

Ainsi, $x \xrightarrow{*} m \implies \forall l \in \Sigma, |m|_l \in s_x(l)$

$$\exists l \in \Sigma / |m|_l \notin s_x(l) \implies \neg(x \xrightarrow{*} m)$$

$$\boxed{\forall x \in \delta(S), \exists l \in \Sigma, |m|_l \notin s_x(l) \implies m \notin \delta(x)}.$$

Alors `interdit x` devra renvoyer vrai.

Calcul de s_x - Description des mots

On cherche à associer une description unique à un mot x (par ses nombres d'occurences des lettres).

L'ensemble de ces descriptions sera $Q \subset (\mathcal{P}(\mathbb{N}))^\Sigma$ tel que :

$$\forall q, q' \in Q, \exists l \in \Sigma / q(l) \cap q'(l) = \emptyset$$
$$\text{et } \forall m \in \delta(S), \exists q \in Q / \forall l \in \Sigma, |m|_l \in q(l)$$

Ainsi, grâce à ces deux axiomes, on a existence et unicité d'une telle description pour un mot x donné.

$\forall x \in \delta(S), \exists! q \in Q / \forall l \in \Sigma, |x|_l \in q(l)$, notons cette description $cat(x)$.

On a ainsi partitionné $\Sigma^\star$ par $x \sim y \iff cat(x) = cat(y)$.

Calcul de s_x - Le graphe (Q, A_0)

Constuisons le graphe dont l'ensemble des sommets est Q et les arêtes sont les dérivations possibles d'une description $q \in Q$ à une autre $q' \in Q$.

$$\begin{aligned}&(q, q') \in A_0 \\ &\iff \exists x, x' \in \Sigma^\star / x \to x' \text{ et } \forall l \in \Sigma, |x|_l \in q(l) \text{ et } |x'|_l \in q(l) \\ &\iff \exists x, x' \in \Sigma^\star / x \to x' \text{ et } cat(x) = q \text{ et } cat(x') = q'\end{aligned}$$

On a en fait $A_0 = cat(\to)$.

Pour tout (Q, A) majorant (Q, A_0), cat est un homomorphisme du graphe $(\Sigma^\star, \to)$ vers (Q, A).
$(x \to y \implies (cat(x), cat(y)) \in A)$.

Calcul de s_x

Ainsi, si $cat(x) = q$, et $cat(x') = q'$, alors $x \xrightarrow{*} x' \implies q'$ est accessible depuis q dans (Q, A).

Voilà une condition nécessaire pour que $x \xrightarrow{*} m$.

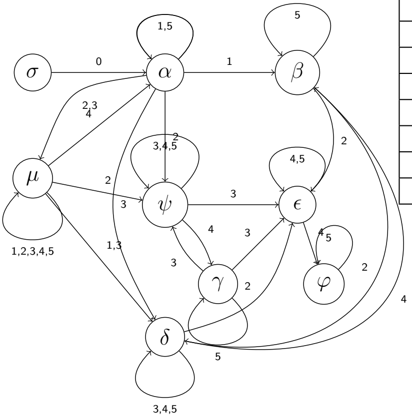
On prend $s_x / \forall l \in \Sigma$,

$$s_x(l) = \bigcup_{y / x \xrightarrow{*} y} cat(y)(l) = \bigcup_{q \text{ accessible depuis } cat(x) \text{ dans A}} q(l).$$

Si $cat(m)$ n'est pas accessible depuis $cat(x)$ dans un graphe (Q, A) majorant (Q, A_0), alors on peut interdire le parcours passant par x, `interdit x` renverra vrai.

Les états q sous la forme $q \in Q = \{\{0\}, \mathbb{N}^*\}^{\Sigma}$

$D =$
$\{(S, \underline{abc})_0, (\underline{abc}, \underline{ab})_1, (\underline{b}, \underline{k})_2, (\underline{c}, \underline{ak})_3, (\underline{kak}, \underline{aa})_4, (\underline{a}, \underline{aaa})_5\}$

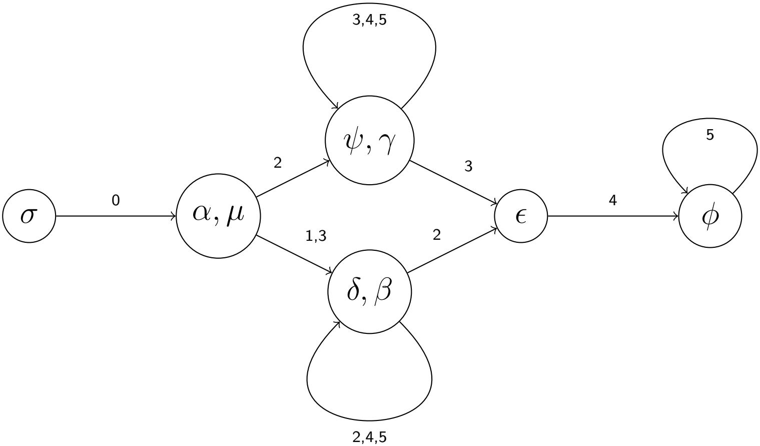


(a) Le graphe A

q	q(a)	q(b)	q(c)	q(k)	q(S)
α	$\mathbb{N}^*$	$\mathbb{N}^*$	$\mathbb{N}^*$	$\{0\}$	$\{0\}$
β	$\mathbb{N}^*$	$\mathbb{N}^*$	$\{0\}$	$\{0\}$	$\{0\}$
γ	$\mathbb{N}^*$	$\{0\}$	$\mathbb{N}^*$	$\{0\}$	$\{0\}$
δ	$\mathbb{N}^*$	$\mathbb{N}^*$	$\{0\}$	$\mathbb{N}^*$	$\{0\}$
ϵ	$\mathbb{N}^*$	$\{0\}$	$\{0\}$	$\mathbb{N}^*$	$\{0\}$
φ	$\mathbb{N}^*$	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$
ψ	$\mathbb{N}^*$	$\{0\}$	$\mathbb{N}^*$	$\mathbb{N}^*$	$\{0\}$
μ	$\mathbb{N}^*$	$\mathbb{N}^*$	$\mathbb{N}^*$	$\mathbb{N}^*$	$\{0\}$
σ	$\{0\}$	$\{0\}$	$\{0\}$	$\{0\}$	$\mathbb{N}^*$

(b) Ses sommets

Les états q sous la forme $q \in Q = \{\{0\}, \mathbb{N}^*\}^\Sigma$



Ainsi, nous pouvons tout de suite affirmer qu'il est impossible de dériver <u>aaabakab</u> en <u>akkcckaaakck</u> (en effet, ψ n'est pas accessible depuis δ)

Calcul des arêtes A du graphe (1)

Pour une dérivation (a, b) fixée, notons

$A_{(a,b)} := \{(q, q') \in A \mid \exists x, x' \in \Sigma^\star / x \overset{(a,b)}{\rightarrow} x'$ et $cat(x) =$
q et $cat(x') = q'\}$

Pour calculer les $(q, q') \in A_{(a,b)}$(majorant $A_{0(a,b)}$), trouvons des conditions nécessaires telles que

$\exists x, y / cat(x) = q$ et $cat(y) = q$ et $x \overset{(a,b)}{\rightarrow} y$.

Calcul des arêtes A du graphe (2)

$$(\exists x, y / cat(x) = q \text{ et } cat(y) = q \text{ et } x \overset{(a,b)}{\rightarrow} y) \implies$$

- x doit contenir les lettres de a, donc $cat(a) \preceq cat(x)$ où $\forall \alpha, \beta \in Q, (\alpha \preceq \beta) \iff \forall l \in \Sigma, \max \alpha(l) \leq \min \beta(l)$.
- y doit contenir au moins les lettres de x moins celles de a, donc $cat(x) - cat(a) \preceq cat(y)$ où $\forall \alpha, \beta \in Q, \alpha - \beta : l \mapsto$ $\begin{cases} \{0\} & \text{si } q'(l) = \mathbb{N} \text{ ou } q(l) = \{0\} \\ \mathbb{N} & \text{sinon} \end{cases}$
- Pour $l \in \Sigma$, si l est dans b, l sera dans y, donc $cat(b)(l) = \mathbb{N}^* \implies cat(y)(l)\mathbb{N}^*$.
- Pour $l \in \Sigma$, si x ne contient pas l et b non plus, y ne contiendra pas l, donc $cat(x)(l) = \{0\}$ et $cat(b)(l) = \{0\} \implies cat(y)(l) = \{0\}$.

Calcul des arêtes A du graphe (3)

On en déduit

$$
\begin{array}{l}
A_{(a,b)} = \\
\{(q, q') \in Q^2 \mid \\
\left\{ \begin{array}{l}
cat(a) \preceq q \text{ et} \\
q - cat(a) \preceq q' \text{ et} \\
\forall l \in \Sigma, \\
\qquad (cat(b)(l) = \mathbb{N}^* \implies q'(l) = \mathbb{N}^*) \text{ et} \\
\qquad (q(l) = \{0\} \text{ et } cat(b)(l) = \{0\} \implies q'(l) = \{0\})
\end{array} \right. \\
\}
\end{array}
$$

et $A = \bigcup_{q \in Q} \bigcup_{d \in D} A_{q,d}$.

Conclusion

En précalculant la matrice d'accessibilité de A (avec Floyd-Warshall), on a une fonction `elimine x` s'executant en $\mathcal{O}(|x|)$.

On arrive alors dans certains cas à réduire le temps de recherche d'une preuve d'appartenance du mot m au langage de notre grammaire.

Mais, seulement 3 appels à `inderdit` réussis pour dériver S en aaakaaak de profondeur 5...

Prétraitement couteux

Extension probablement possible (mais aussi peu utile ?) pour $Q \subset \{2\mathbb{N}, 2\mathbb{N}+1\}^{\Sigma}$