

Sur la vérification de preuve et la preuve automatique d'appartenance d'un mot à une grammaire.

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 $\Sigma := N_t \cup T$
- ▶ $D \subset (\Sigma^*)^2$, est l'ensemble des règles de dérivation.

Definitions

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

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

Vérification de preuve

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

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

$$m_k|_{[1, j_k-1]} \textcolor{red}{a} m_k|_{[|a_{i_k}|+j_k, |m_k|]} \xrightarrow{(a_{i_k}, b_{i_k})} m_k|_{[1, j_k-1]} \textcolor{red}{b} m_k|_{[|a_{i_k}|+j_k, |m_k|]}$$

```
type 'e preuveformelle =  
(int*int*('e caractere list)) list  
  
[...,( ik , jk , mk ),...]
```

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 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 = [(0,1,mot "aBSc");(2,3,mot  
"abc");(1,3,mot "aB");(2,4,mot "bb")]
```

Preuve automatique d'appartenance d'un mot

Cherchons comment dériver S en un mot m donné.

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

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

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

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

On fait en réalité un parcours en largeur du graphe (Σ^*, \rightarrow) depuis 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'occurrence de chaque lettre

Cherchons une fonction `interdit` plus sophistiquée.

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

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

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

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

Alors `interdit x` devra renvoyer vrai.

Calcul de s_x - Le graphe A_0 (1)

Considérons le graphe A_0 où les sommets sont dans $Q \subset (\mathcal{P}(\mathbb{N}))^\Sigma$ tels que :

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

Ainsi, $\forall x \in \delta(S), \exists ! q \in Q / \forall l \in \Sigma, |x|_l \in q(l)$, notons ce sommet $cat(x)$

Calcul de s_x - Le graphe A_0 (2)

Les arêtes du graphe sont les dérivation possibles d'une famille majorante $q \in Q$ à une autre $q' \in Q$.

$$(q, q') \in A_0$$

$$\iff \exists x, x' \in \Sigma^*/x \rightarrow x' \text{ et } \forall l \in \Sigma, |x|_l \in q(l) \text{ et } |x'|_l \in q'(l)$$

$$\iff \exists x, x' \in \Sigma^*/x \rightarrow x' \text{ et } \text{cat}(x) = q \text{ et } \text{cat}(x') = q'$$

Pour A un graphe majorant A_0 :

$$\text{On a } \forall l \in \Sigma, s_x(l) = \bigcup_{q \text{ accessible depuis } \text{cat}(x) \text{ dans } A} q(l).$$

Calcul de s_x - Le graphe A_0 (3)

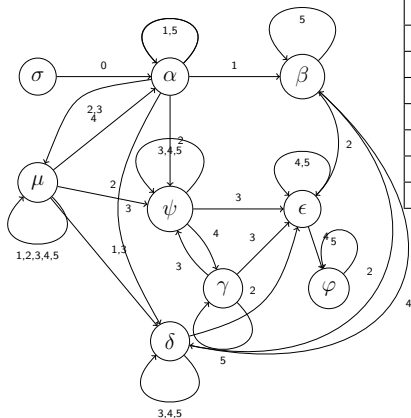
Ainsi, si $cat(x) = q$, et $cat(x') = q'$, alors $x \xrightarrow{*} x' \implies q'$ est accessible depuis q dans A .

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

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

$D =$

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

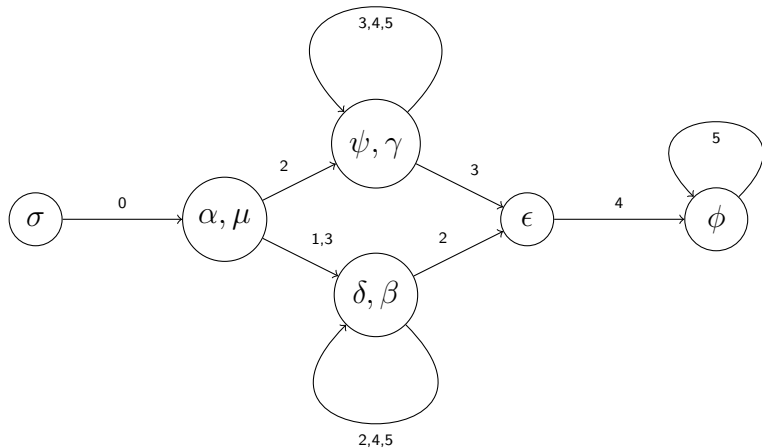


(a) Le graphe A

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 aaabakab en akkcckaaakck (en effet, ψ n'est pas accessible depuis δ)

Calcul des arrêtes A du graphe

Pour un sommet de départ fixé et une dérivation fixée.

$$A_{q,d}(\Sigma) := \{(q, b) \in A \mid \exists x, x' \in \Sigma^* / x \xrightarrow{d} x' \text{ et } \text{cat}(x) = q \text{ et } \text{cat}(x') = q'\}$$

$$\text{Soit } l \in \Sigma, \Sigma' := \Sigma \setminus \{l\}, A = \bigcup_{q \in Q} \bigcup_{d \in D} A_{q,d}(\Sigma).$$

► $\text{cat}(b)(l) = \mathbb{N}^*$

$$A_{q,d}(\Sigma) \subset \{(q, b) \in Q^2 \mid (q_{\Sigma'}, b_{\Sigma'}) \in A_{q,d}(\Sigma') \text{ et } b(l) = \mathbb{N}^*\}$$

► $\text{cat}(b)(l) = \{0\}$ et $q(l) = \{0\}$

$$A_{q,d}(\Sigma) \subset \{(q, b) \in Q^2 \mid (q_{\Sigma'}, b_{\Sigma'}) \in A_{q,d}(\Sigma') \text{ et } b(l) = \{0\}\}$$

► $\text{cat}(b)(l) = \{0\}$ et $q(l) = \Sigma^*$

$$A_{q,d}(\Sigma) \subset \{(q, b) \in Q^2 \mid (q_{\Sigma'}, b_{\Sigma'}) \in A_{q,d}(\Sigma') \text{ et } b(l) = \{0\}\} \\ \cup \{(q, b) \in Q^2 \mid (q_{\Sigma'}, b_{\Sigma'}) \in A_{q,d}(\Sigma') \text{ et } b(l) = \Sigma^*\}$$

Conclusion

En précalculant la matrice d'accessibilité de A (avec Floyd-Warshall), on a une fonction `elimine x` s'exécutant en temps constant.

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

Extension probablement possible pour $Q \subset \{2\mathbb{N}, 2\mathbb{N} + 1\}^\Sigma$