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

- $\forall x, x' \in \Sigma^\star, x \overset{(a,b)}{\to} x' \iff \exists u, v \in \Sigma^\star / x = uav$ et $x' = ubv$
- $\to := \bigcup_{d \in D} \overset{d}{\to}$ et on note $\overset{*}{\to}$ la cloture transitive et réflexive de $\to$
- $\delta(x) := \{y \in \Sigma^\star / x \overset{*}{\to} y\}$
- $|x|_l := |\{i \in \mathbb{N} \mid x_i = l\}|$ 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

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

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

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

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

[..., (i_k , j_k , m_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 = [(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 $\underline{S}$ en un mot m donné.

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

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

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

l'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 - Le graphe A_0 (1)

Considérons le graphe A_0 où les sommets sont dans $Q \subset (\mathcal{P}(\mathbb{N}))^{\Sigma}$ tels 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, $\forall x \in \delta(S), \exists! q \in Q / \forall l \in \Sigma, |x|_l \in q(l)$, notons ce sommet $cat(x)$

Calcul de s_x - Le graphe A_0 (2)

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

$(q, q') \in A_0$

$\iff \exists x, x' \in \Sigma^\star / x \to x'$ et $\forall l \in \Sigma, |x|_l \in q(l)$ et $|x'|_l \in q(l)$

$\iff \exists x, x' \in \Sigma^\star / x \to x'$ et $cat(x) = q$ et $cat(x') = q'$

Pour A un graphe majorant A_0 :

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

Calcul de s_x - Le graphe A_0 (3)

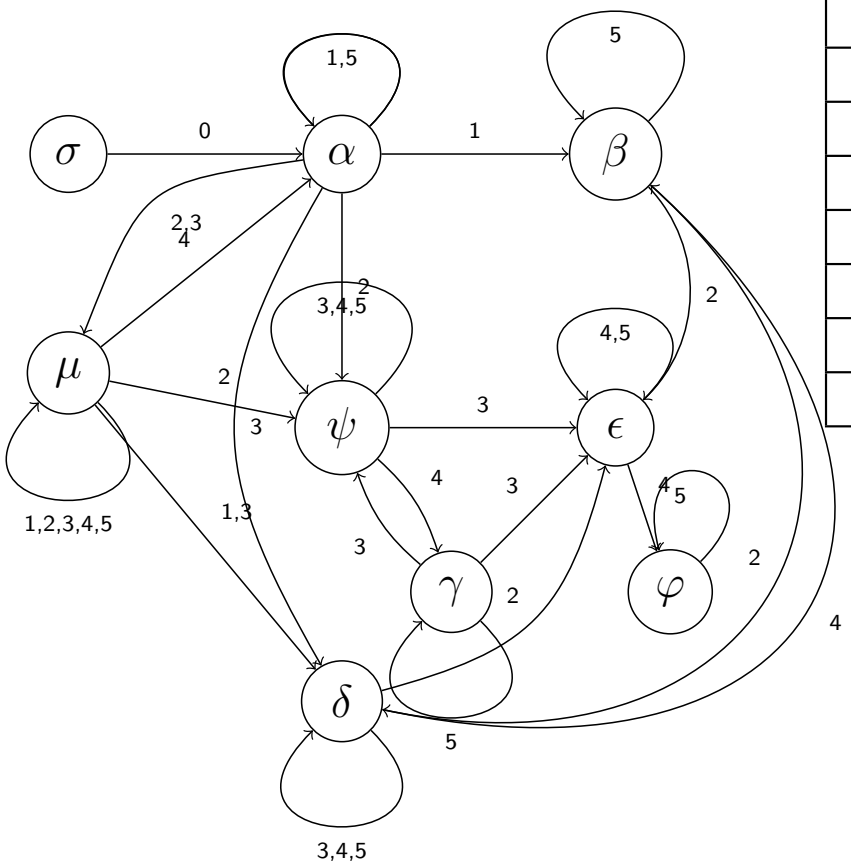
Ainsi, si $cat(x) = q$, et $cat(x') = q'$, alors $x \xrightarrow{*} x' \implies q'$ est accessible depuis q dans A.

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 $q \in Q = \{\{0\}, \mathbb{N}^*\}^{\Sigma}$

$D =$

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

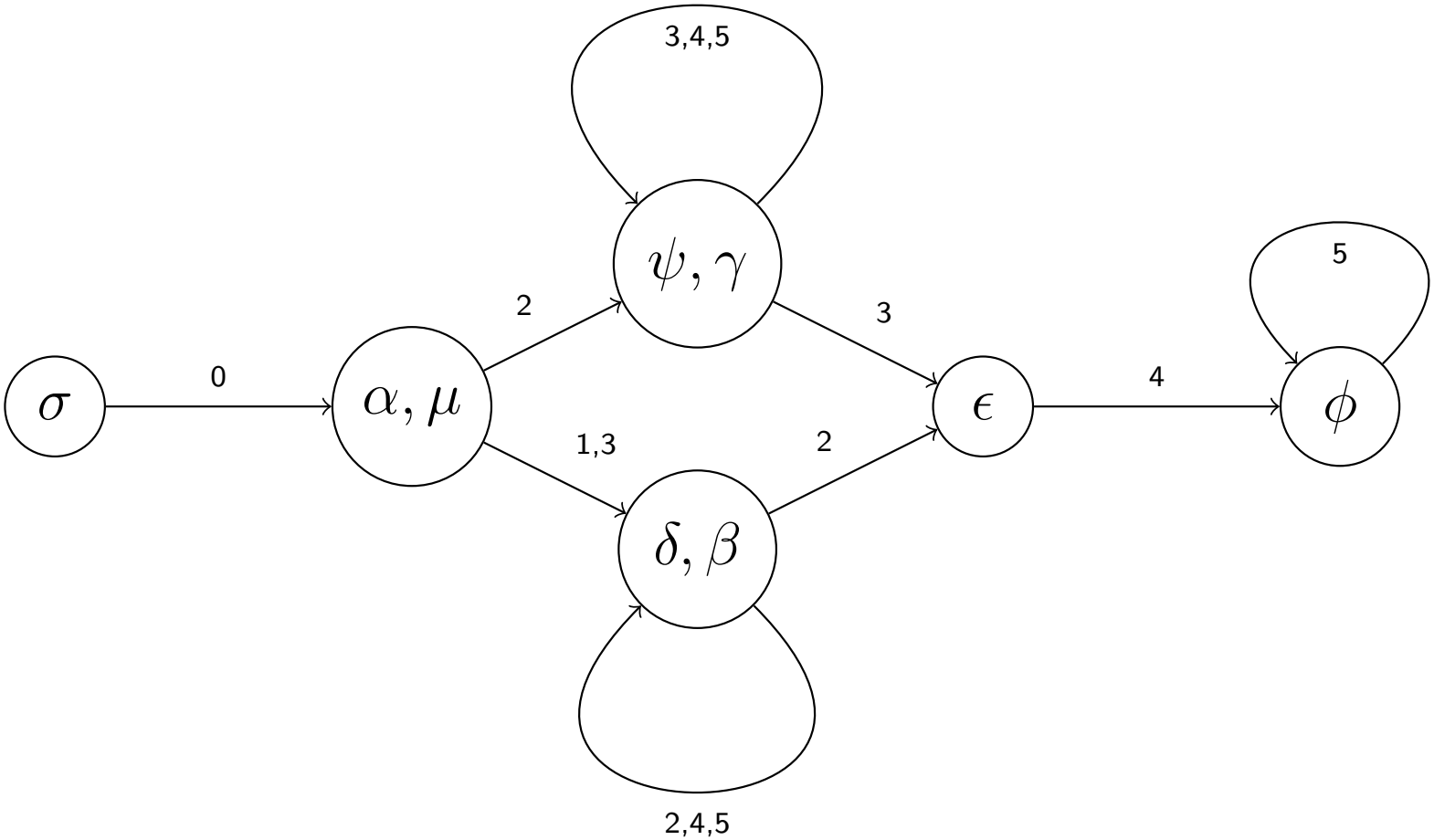


(a) Le graphe A

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 arêtes A du graphe

Pour un sommet de départ fixé et une dérivation fixée.

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

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

$A_{q,(a,b)} \subset$

$$\begin{cases} \emptyset \text{ si } cat(a) \neq q \\ \{(q, q') \in Q^2 \mid \forall l \in \Sigma, (cat(b)(l) = \mathbb{N}^* \implies q'(l) = \mathbb{N}^*) \text{ et} \\ (q(l) = \{0\} \text{ et } cat(b)(l) = \{0\} \implies q(l) = \{0\})\} \text{ sinon} \end{cases}$$

Conclusion

En précalculant la matrice d'accessibilité de A (avec Floyd-Warshall), on a une fonction `elimine x` s'executant 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 $Q \subset \{2\mathbb{N}, 2\mathbb{N}+1\}^{\Sigma}$