

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

Definitions

► $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$

Definitions

► $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$

► $\rightarrow := \bigcup_{(a,b) \in D} \xrightarrow{(a,b)}$

Definitions

- ▶ $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$
- ▶ $\rightarrow := \bigcup_{(a,b) \in D} \xrightarrow{(a,b)}$
- ▶ $\xrightarrow{*}$ la cloture transitive et réflexive de \rightarrow
($x \xrightarrow{*} m \iff x \rightarrow m_1 \rightarrow \dots \rightarrow m_{n-1} \rightarrow m$)

Definitions

- ▶ $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$
- ▶ $\rightarrow := \bigcup_{(a,b) \in D} \xrightarrow{(a,b)}$
- ▶ $\xrightarrow{*}$ la cloture transitive et réflexive de \rightarrow
($x \xrightarrow{*} m \iff x \rightarrow m_1 \rightarrow \dots \rightarrow m_{n-1} \rightarrow m$)
- ▶ $\delta(x)$ successeurs de x par \rightarrow (ou successeurs directs de x par $\xrightarrow{*}$)

Definitions

- ▶ $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$
- ▶ $\rightarrow := \bigcup_{(a,b) \in D} \xrightarrow{(a,b)}$
- ▶ $\xrightarrow{*}$ la cloture transitive et réflexive de \rightarrow
($x \xrightarrow{*} m \iff x \rightarrow m_1 \rightarrow \dots \rightarrow m_{n-1} \rightarrow m$)
- ▶ $\delta(x)$ successeurs de x par \rightarrow (ou successeurs directs de x par $\xrightarrow{*}$)
- ▶ $|x|_l$ nombre d'occurences de la lettre l dans x .

Definitions

- ▶ $x \xrightarrow{(a,b)} x' \iff x = uav \text{ et } x' = ubv$
- ▶ $\rightarrow := \bigcup_{(a,b) \in D} \xrightarrow{(a,b)}$
- ▶ $\xrightarrow{*}$ la cloture transitive et réflexive de \rightarrow
($x \xrightarrow{*} m \iff x \rightarrow m_1 \rightarrow \dots \rightarrow m_{n-1} \rightarrow m$)
- ▶ $\delta(x)$ successeurs de x par \rightarrow (ou successeurs directs de x par $\xrightarrow{*}$)
- ▶ $|x|_l$ nombre d'occurences de la lettre l dans x .
- ▶ Le langage de la grammaire formelle G est :

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

Vérification de preuve

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

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

$$m_k = u_{k+1} a_{i_{k+1}} v_{k+1} \xrightarrow{(a_{i_{k+1}}, b_{i_{k+1}})} m_{k+1} = u_{k+1} b_{i_{k+1}} v_{k+1}$$
$$j_k := |u_k|$$

```
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 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

Chemin de S à m dans (Σ^*, \rightarrow) .

Preuve automatique d'appartenance d'un mot

Chemin de S à m dans (Σ^*, \rightarrow) .

Parcours en largeur ! Successeurs directs $\mathcal{S}(x)$ d'un mot x par \rightarrow ?

Preuve automatique d'appartenance d'un mot

Chemin de S à m dans (Σ^*, \rightarrow) .

Parcours en largeur ! Successeurs directs $\mathcal{S}(x)$ d'un mot x par \rightarrow ?

Algorithme de Knuth-Morris-Pratt pour le calcul de $\mathcal{S}_{(a,b)}(x)$
successeurs directs de x par $\xrightarrow{(a,b)}$.

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

Réduire la complexité d'un tel parcours ?

Eviter les chemins passant par certains mots.

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

Retourner le chemin pour la preuve d'appartenance du mot.

Amélioration pour les grammaires croissantes

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

$$x \xrightarrow{*} x' \implies |x| \leq |x'|$$

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

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

interdit plus sophistiquée dans le cas général.

$l \in \Sigma$, $s_x(l) \supset |\delta(x)|_l$ nombres d'occurrences de l dans les successeurs de x .

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

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

interdit plus sophistiquée dans le cas général.

$l \in \Sigma$, $s_x(l) \supset |\delta(x)|_l$ nombres d'occurrences de l dans les successeurs de x .

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

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

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

interdit plus sophistiquée dans le cas général.

$l \in \Sigma$, $s_x(l) \supset |\delta(x)|_l$ nombres d'occurrences de l dans les successeurs de x .

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

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

Pour les grammaires croissantes, $s_x(l) \supset [|x|, \infty[$

Calcul de s_x - Description des mots

Description d'un mot x (par ses nombres d'occurrences des lettres).

Ensemble de ces descriptions $Q \text{ fini} \subset (\mathcal{P}(\mathbb{N}))^\Sigma$.

Soit $cat(x) \in Q$ la description de x ,

$\forall l \in \Sigma, |x|_l \in cat(x)(l)$.

Calcul de s_x - Description des mots

Description d'un mot x (par ses nombres d'occurrences des lettres).

Ensemble de ces descriptions $Q \text{ fini} \subset (\mathcal{P}(\mathbb{N}))^\Sigma$.

Soit $cat(x) \in Q$ la description de x ,

$\forall l \in \Sigma, |x|_l \in cat(x)(l)$.

Exemple : pour $Q = \{\{0\}, \mathbb{N}^*\}^\Sigma$,

$$\begin{aligned} cat(\underline{abcba}) : a &\mapsto \mathbb{N}^*, b \mapsto \mathbb{N}^*, \\ c &\mapsto \mathbb{N}^*, d \mapsto \{0\}, \\ S &\mapsto \{0\} \end{aligned}$$

Calcul de s_x - Description des mots

Description d'un mot x (par ses nombres d'occurrences des lettres).

Ensemble de ces descriptions $Q \text{ fini} \subset (\mathcal{P}(\mathbb{N}))^\Sigma$.

Soit $cat(x) \in Q$ la description de x ,

$\forall l \in \Sigma, |x|_l \in cat(x)(l)$.

Exemple : pour $Q = \{\{0\}, \mathbb{N}^*\}^\Sigma$,

$$\begin{aligned} cat(\underline{abcba}) : a &\mapsto \mathbb{N}^*, b \mapsto \mathbb{N}^*, \\ c &\mapsto \mathbb{N}^*, d \mapsto \{0\}, \\ S &\mapsto \{0\} \end{aligned}$$

Partition de Σ^* par $x \sim y \iff cat(x) = cat(y)$.

Calcul de s_x - Le graphe (Q, A_0)

Graphe (Q, A_0) où $A_0 = \text{cat}(\rightarrow)$, cat homomorphisme de graphes.

$$x \rightarrow x' \implies (\text{cat}(x), \text{cat}(x')) \in A_0$$

Calcul de s_x - Le graphe (Q, A_0)

Graphe (Q, A_0) où $A_0 = \text{cat}(\rightarrow)$, cat homomorphisme de graphes.

$$x \rightarrow x' \implies (\text{cat}(x), \text{cat}(x')) \in A_0$$

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

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

A_0 dérivations possibles d'une description $q \in Q$ à une autre $q' \in Q$.

Calcul de s_x

$x \xrightarrow{*} m \implies \text{cat}(m)$ accessible depuis $\text{cat}(x)$ dans tout (Q, A) où $A \supset A_0$.

Calcul de s_x

$x \xrightarrow{*} m \implies \text{cat}(m)$ accessible depuis $\text{cat}(x)$ dans tout (Q, A) où $A \supset A_0$.

La contraposée :

Si $\text{cat}(m)$ n'est pas accessible depuis $\text{cat}(x)$ dans un graphe (Q, A) majorant (Q, A_0) , alors $\neg(x \xrightarrow{*} m)$.

Calcul de s_x

$x \xrightarrow{*} m \implies \text{cat}(m)$ accessible depuis $\text{cat}(x)$ dans tout (Q, A) où $A \supset A_0$.

La contraposée :

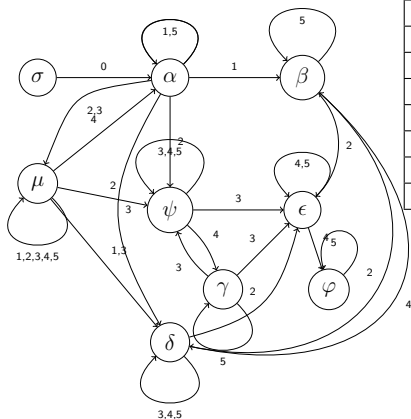
Si $\text{cat}(m)$ n'est pas accessible depuis $\text{cat}(x)$ dans un graphe (Q, A) majorant (Q, A_0) , alors $\neg(x \xrightarrow{*} m)$.

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

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

$D =$

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

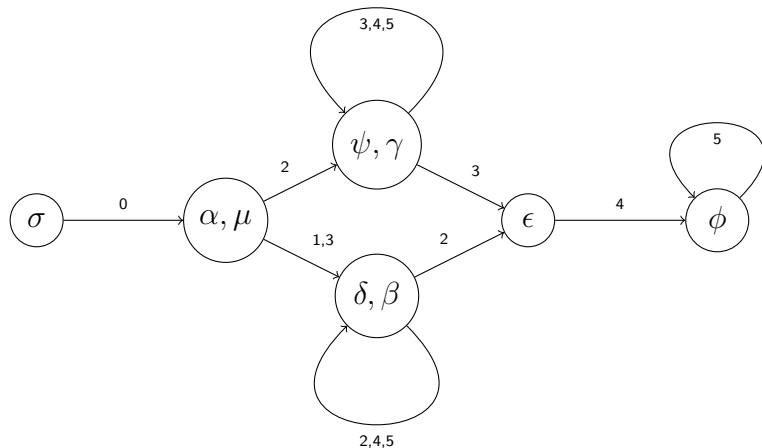


(a) Le graphe A

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



$\neg (\underline{aaabakab} \xrightarrow{*} \underline{akkcckaaakck})$ (ψ pas accessible depuis δ)

Calcul des arêtes A du graphe, conditions nécessaires

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

- $cat(a) \preceq cat(x)$ où
 $\forall \alpha, \beta \in Q, (\alpha \preceq \beta) \iff \forall l \in \Sigma, \max \alpha(l) \leq \min \beta(l).$

Calcul des arêtes A du graphe, conditions nécessaires

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

- ▶ $\text{cat}(a) \preceq \text{cat}(x)$ où
 $\forall \alpha, \beta \in Q, (\alpha \preceq \beta) \iff \forall l \in \Sigma, \max \alpha(l) \leq \min \beta(l).$
- ▶ $\text{cat}(x) - \text{cat}(a) \preceq \text{cat}(y)$ où
 $\forall \alpha, \beta \in Q, \alpha - \beta : l \mapsto$
$$\begin{cases} \{0\} & \text{si } q'(l) = \mathbb{N} \text{ ou } q(l) = \{0\} \\ \mathbb{N} & \text{sinon} \end{cases}$$

Calcul des arêtes A du graphe, conditions nécessaires

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

- ▶ $\text{cat}(a) \preceq \text{cat}(x)$ où
 $\forall \alpha, \beta \in Q, (\alpha \preceq \beta) \iff \forall l \in \Sigma, \max \alpha(l) \leq \min \beta(l).$
- ▶ $\text{cat}(x) - \text{cat}(a) \preceq \text{cat}(y)$ où
 $\forall \alpha, \beta \in Q, \alpha - \beta : l \mapsto$
$$\begin{cases} \{0\} & \text{si } q'(l) = \mathbb{N} \text{ ou } q(l) = \{0\} \\ \mathbb{N} & \text{sinon} \end{cases}$$
- ▶ $\forall l \in \Sigma, \text{cat}(b)(l) = \mathbb{N}^* \implies \text{cat}(y)(l) = \mathbb{N}^*.$

Calcul des arêtes A du graphe, conditions nécessaires

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

- ▶ $\text{cat}(a) \preceq \text{cat}(x)$ où
 $\forall \alpha, \beta \in Q, (\alpha \preceq \beta) \iff \forall l \in \Sigma, \max \alpha(l) \leq \min \beta(l).$
- ▶ $\text{cat}(x) - \text{cat}(a) \preceq \text{cat}(y)$ où
 $\forall \alpha, \beta \in Q, \alpha - \beta : l \mapsto$
$$\begin{cases} \{0\} & \text{si } q'(l) = \mathbb{N} \text{ ou } q(l) = \{0\} \\ \mathbb{N} & \text{sinon} \end{cases}$$
- ▶ $\forall l \in \Sigma, \text{cat}(b)(l) = \mathbb{N}^* \implies \text{cat}(y)(l) = \mathbb{N}^*.$
- ▶ $\forall l \in \Sigma, \text{cat}(x)(l) = \{0\}$ et
 $\text{cat}(b)(l) = \{0\} \implies \text{cat}(y)(l) = \{0\}.$

Calcul des arêtes A du graphe (3)

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

$$A = \bigcup_{(a,b) \in D} A_{(a,b)}.$$

Conclusion

En précalculant la matrice d'accessibilité de A (avec Floyd-Warshall), élimine x en $\mathcal{O}(|x|)$.

Conclusion

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

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

Conclusion

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

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

Prétraitement couteux

Conclusion

En précalculant la matrice d'accessibilité de A (avec Floyd-Warshall), élimine x en $\mathcal{O}(|x|)$.

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

Prétraitement coûteux

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