M1 MPRI : Automates et Applications

Cours 2
Langages réguliers d'arbres

kn@lri.fr

31 mars 2021

Plan

1 Introduction

2 Arbres n-aires

3 Automates d'arbres

4 Automates descendants

Dans l'épisode précédant

université PARIS-SACLAY

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, …

Dans l'épisode précédant

université PARIS-SACLAY

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, …

Qu'est-ce qu'un mot sur un alphabet Σ ?

Dans l'épisode précédant

université PARIS-SACLAY

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, ...

Qu'est-ce qu'un mot sur un alphabet Σ ? Une suite $w = abc \ldots$

Dans l'épisode précédant

université PARIS-SACLAY

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, ...

Qu'est-ce qu'un mot sur un alphabet Σ ? Une suite $w = abc\ldots$

Peut-on généraliser ?

Dans l'épisode précédant

université PARIS-SACLAY

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, ...

Qu'est-ce qu'un mot sur un alphabet Σ ? Une suite $w = abc\ldots$

Peut-on généraliser ?

Dans l'épisode précédant

Nous avons vu les automates de mots finis ainsi que leur utilité. Ils offrent un modèle d'exécution et plus généralement un moyen de calcul effectif pour des opérations d'un « programme » de haut niveau : les expressions régulières.

Ces dernières ont une utilité pratique : vérifier des chaînes de caractères, rechercher dans du texte, ...

Qu'est-ce qu'un mot sur un alphabet Σ ? Une suite $w = abc\ldots$

Peut-on généraliser ?

Si on autorise chaque symbole à avoir plusieurs successeurs, on obtient un arbre.

Avec des arbres ?

université PARIS-SACLAY

- Formellement, qu'est-ce qu'un arbre ?
- Quel « langage de haut niveau » peut ont avoir sur les arbres ? et quel problème « de la vraie vie^{TM} » ils permettent de résoudre ?
- Quel modèle de calcul pour ce langage de haut niveau ?

Plan

université PARIS-SACLAY

1 Introduction

2 Arbres n-aires

3 Automates d'arbres

4 Automates descendants

Alphabet gradué

université PARIS-SACLAY

Définition (Alphabet gradué)

Un alphabet gradué est un couple $(\Sigma, |\cdot|)$ *où* Σ *est un ensemble de symboles et* $|\cdot| : \Sigma \to \mathbb{N}$ *est une fonction appelée arité.*

La notation $|\cdot|$ signifie que l'on appelle la fonction en mettant son argument à la place du point. Par exemple $|a| = 0$, $|f| = 4$, ...

On note $\Sigma_k = \{a \in \Sigma \mid |a| = k\}$.

Dans la suite, on parlera d'un alphabet gradué Σ en laissant la fonction d'arité implicite, pour ne pas alourdir les énoncés. On notera aussi $\Sigma = \{f^k, g^n, a\}$ pour indiquer que $|f| = k$, $|g| = n$ et $a = 0$.

Arbre n-aire

université PARIS-SACLAY

Définition

Un arbre t est une fonction $t : S \to \Sigma$, où $S \subseteq \mathbb{N}_1^*$ est un ensemble de suites d'entiers appelées positions (ou chemins) et vérifiant les propriété suivantes :

Chemin vide : $\epsilon \in S$

Clôture par préfixe : $i_, \cdots i_{n-1} i_n \in S, \Rightarrow i_0 \cdots i_{n-1} \in S$

Bonne formation : $\forall p \in S, i \in \mathbb{N}, pi \in S \Rightarrow i \leq |t(p)|$

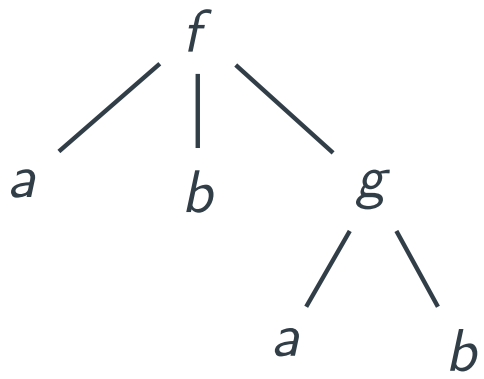
Le domaine de t (l'ensemble S) est noté $Dom(t)$.

Attention : $\mathbb{N}_1$ dénote l'ensemble des entiers privés de 0, et $\mathbb{N}_1^*$ représente l'ensemble des suites de tels entiers (ici * est l'étoile de Kleene).

Exemple

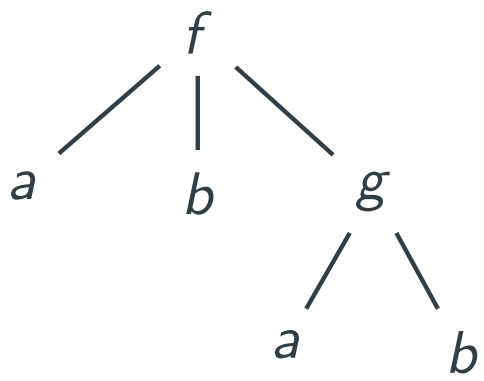
On considère « l'arbre » représenté graphiquement ci-dessous.

Comment le représenter avec la définition formelle ?



Exemple

On considère « l'arbre » représenté graphiquement ci-dessous.
Comment le représenter avec la définition formelle ?

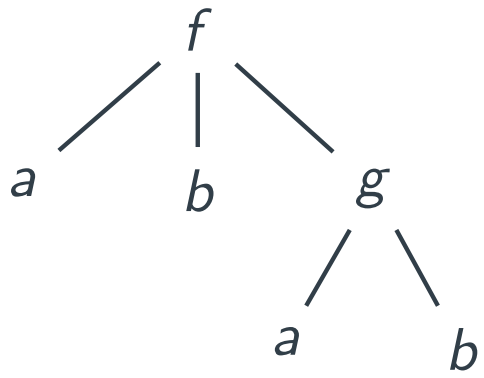


$$t : \begin{array}{lcl} \epsilon & \mapsto & f \\ 1 & \mapsto & a \\ 2 & \mapsto & b \\ 3 & \mapsto & g \\ 31 & \mapsto & a \\ 32 & \mapsto & b \end{array}$$

Les fils sont *ordonnés*. Le chemin ϵ s'appelle la racine. Tout chemin p tel que $|t(p)| = 0$ s'appelle une *feuille*.

Exemple

On considère « l'arbre » représenté graphiquement ci-dessous.
Comment le représenter avec la définition formelle ?
Comment le représenter de façon compacte ?

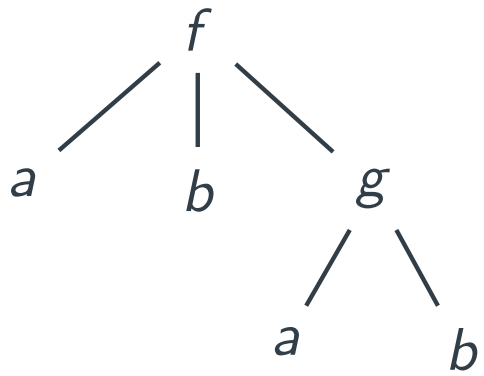


$$t : \begin{array}{lcl} \epsilon & \mapsto & f \\ 1 & \mapsto & a \\ 2 & \mapsto & b \\ 3 & \mapsto & g \\ 31 & \mapsto & a \\ 32 & \mapsto & b \end{array}$$

Les fils sont *ordonnés*. Le chemin ϵ s'appelle la racine. Tout chemin p tel que $|t(p)| = 0$ s'appelle une *feuille*.

Exemple

On considère « l'arbre » représenté graphiquement ci-dessous.
Comment le représenter avec la définition formelle ?
Comment le représenter de façon compacte ?



$$t : \begin{array}{lcl} \epsilon & \mapsto & f \\ 1 & \mapsto & a \\ 2 & \mapsto & b \\ 3 & \mapsto & g \\ 31 & \mapsto & a \\ 32 & \mapsto & b \end{array} \qquad f(a, b, g(a, b))$$

Les fils sont *ordonnés*. Le chemin ϵ s'appelle la racine. Tout chemin p tel que $|t(p)| = 0$ s'appelle une *feuille*.

Utilité de ce formalisme ?

université PARIS-SACLAY

Définissons les concepts suivants :

- Les fils d'un chemin p

Utilité de ce formalisme ?

université PARIS-SACLAY

Définissons les concepts suivants :

- Les fils d'un chemin p : $\mathrm{ch}(p) = \{pi \mid 1 \leq i \leq |t(p)|\}$
  Cet ensemble est vide si p est une feuille.
- Le(s) parent(s) d'un chemin p

Utilité de ce formalisme ?

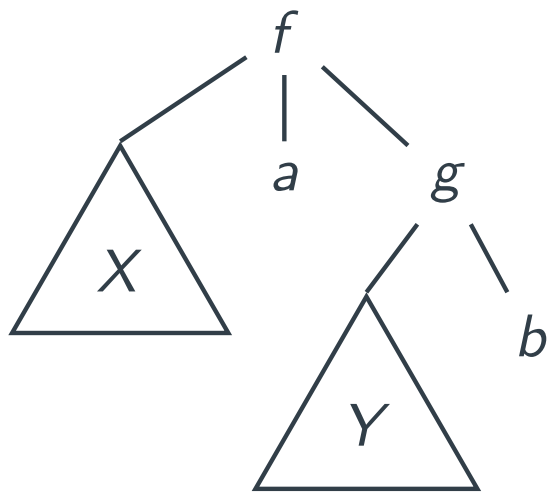
université PARIS-SACLAY

Définissons les concepts suivants :

- Les fils d'un chemin p : $\mathrm{ch}(p) = \{pi \mid 1 \leq i \leq |t(p)|\}$
  Cet ensemble est vide si p est une feuille.
- Le(s) parent(s) d'un chemin p : $\mathrm{par}(p)\{q \mid \exists i \in \mathbb{N}_1, qi = p\}$
  Cet ensemble est
  - vide si $p = \epsilon$
  - un singleton sinon
- Tous les chemins ayant le symbole a : $\mathrm{lab}_a(t) = \{p \mid t(p) = a\}$

Arbres avec des « trous » ?

Dans le contexte des mots (par exemple dans le lemme de la pompe) On peut écrire de façon commode « $w = xyz$ pour dire que w est composé d'un préfixe x, d'un facteur y et d'un suffixe z.
On a souvent besoin de faire de même pour les arbres, mais c'est plus compliqué :



Ensemble de tous les arbres

université PARIS-SACLAY

Définition (Ensemble des arbres)

Soit Σ un alphabet gradué et $\mathcal{X}$ un ensemble de symboles appelés variables tel que $\Sigma \cap \mathcal{X} = \varnothing$. On appelle $\mathcal{T}(\Sigma, \mathcal{X})$ l'ensemble défini inductivement par :

- $\forall a \in \Sigma_0, a \in \mathcal{T}(\Sigma, \mathcal{X})$
- $\forall x \in \mathcal{X}, x \in \mathcal{T}(\Sigma, \mathcal{X})$
- $\forall f \in \Sigma, \forall t_1, \ldots, t_{|f|} \in \mathcal{T}(\Sigma, \mathcal{X}), f(t_1, \ldots, t_{|f|}) \in \mathcal{T}(\Sigma, \mathcal{X})$

On notera simplement $\mathcal{T}(\Sigma)$ l'ensemble $\mathcal{T}(\Sigma, \varnothing)$

Vocabulaire

université PARIS-SACLAY

- un arbre avec des variables est souvent appelé un terme
- un arbre avec au plus une seule occurrence de chaque variable est dit *linéaire*
- un arbre linéaire de $\mathcal{T}(\Sigma, \{\Box\})$ est appelé un *contexte* (c'est un arbre avec un unique « trou » dénoté par $\Box$)

Sous-arbre

Une notion importante est la notion de sous-arbre :

Définition (Sous-arbre)

Soit $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Pour tout $p \in Dom(t)$, on définit le sous-arbre de t enraciné en p et on note $t|_p$ l'arbre défini par :

- $Dom(t|_p) = \{u \mid pu \in Dom(t)\}$
- $\forall u \in Dom(t|_p), t|_p(u) = t(pu)$

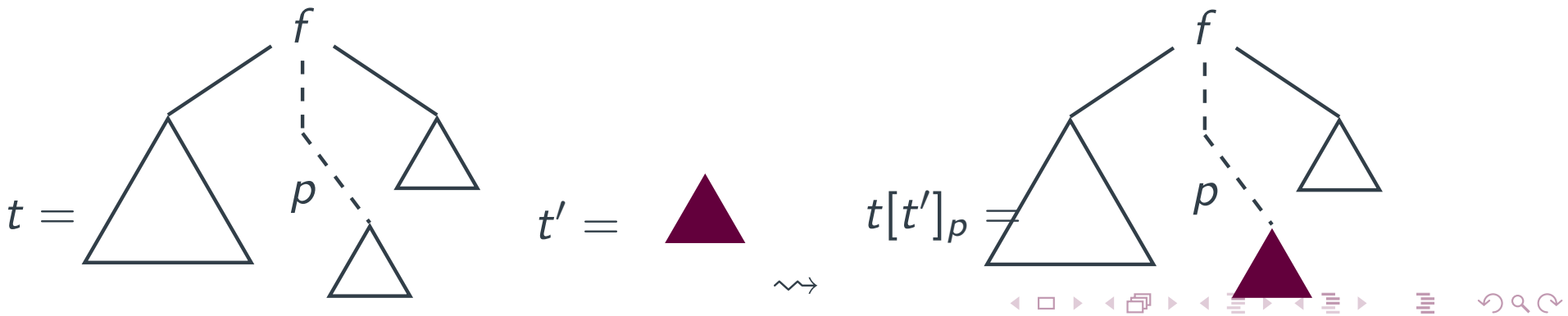
Remplacement de sous-arbre

Il est courant, étant donné un chemin de vouloir remplacer un sous-arbre par un autre.

Définition (Substitution de sous-arbre)

Soit $t, t' \in \mathcal{T}(\Sigma, \mathcal{X})$. L'arbre t dans lequel le sous-arbre en p a été remplacé par t', noté $t[t']_p$ est défini par :

- $Dom(t[t']_p) = (Dom(t) \setminus \{pu \mid pu \in Dom(t)\}) \cup \{pu \mid u \in Dom(t)'\}$
- $\forall q \in Dom(t[t']_p), t[t']_p(q) = \begin{cases} t'(u) & \forall u \text{ tel que } q = pu \\ t(q) & \text{sinon} \end{cases}$



Plan

1 Introduction

2 Arbres n-aires

3 Automates d'arbres

4 Automates descendants

Automate d'arbres non-déterministe

université PARIS-SACLAY

Définition (Automate d'arbre non-déteriniste)

Un automate d'arbre ascendant non-déterministe, NFTA (non-deterministic finite tree automaton) est un $4-$ *uplet* $\mathcal{A} = (Q, \Sigma, \delta, F)$ *défini par*

- *un ensemble d'états* Q
- *un alphabet gradué* Σ
- *un ensemble d'états acceptants* F
- *une fonction de transition* $\delta : \mathcal{T}(\Sigma, Q) \to \mathcal{P}_f(Q)$ *de la forme :*

$$f(q_{i_1}, \ldots, q_{i_k}) \mapsto q$$

avec pour $f \in \Sigma$ *et* $|f| = k$.

Automate d'arbres non-déterministe (2)

université PARIS-SACLAY

La fonction δ prend en argument un symbole de Σ et la liste des états dans lequel doit être chacun de sous-arbre, et renvoie un ensemble d'états pour le nœud courant.
Si δ renvoie toujours un unique état, l'automate est déterministe.
Remarque : dans ce cours, si on ne précise pas explicitement, l'automate d'arbre est considéré comme ascendant (i.e. la définition précédente).

Run d'un automate d'arbre

université PARIS-SACLAY

Définition

Une exécution (ou run) d'un automate d'arbre $\mathcal{A} = (Q, \Sigma, \delta, F)$ *pour un arbre* $t \in \mathcal{T}(\Sigma)$ *est une fonction* $r : dom(t) \to Q$ *telle que*

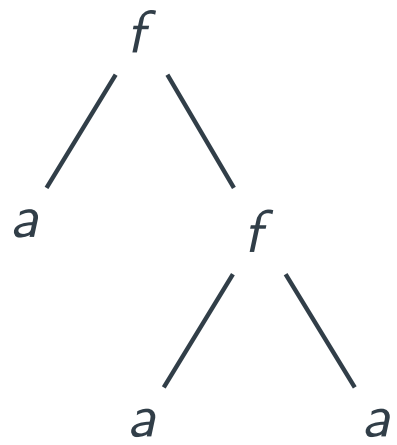
$\forall p \in dom(t),\ r(p) \in \delta(t(p)(r(p1), \ldots, r(pk)))$, *avec* $|t(p)| = k$.

Un run est dit acceptant si et seulement si $r(\epsilon) \in F$.

Exemple

On considère l'automate $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_1\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

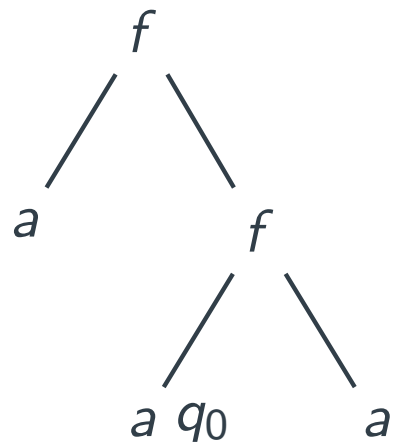
$$\delta : \begin{array}{rcl} a & \mapsto & \{q_0\} \\ f(q_0, q_0) & \mapsto & \{q_1\} \\ f(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$



Exemple

On considère l'automate $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_1\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{rcl} a & \mapsto & \{q_0\} \\ f(q_0, q_0) & \mapsto & \{q_1\} \\ f(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$

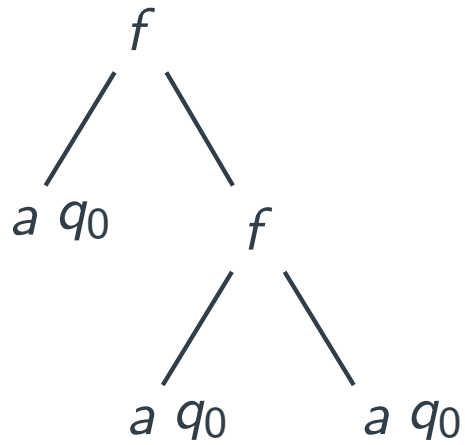


Exemple

université PARIS-SACLAY

On considère l'automate $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_1\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{rcl} a & \mapsto & \{q_0\} \\ f(q_0, q_0) & \mapsto & \{q_1\} \\ f(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$

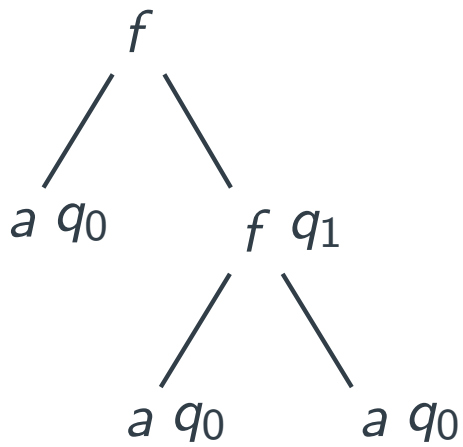


Exemple

université PARIS-SACLAY

On considère l'automate $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_1\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \blacktriangleright \quad \begin{aligned} a &\mapsto \{q_0\} \\ f(q_0, q_0) &\mapsto \{q_1\} \\ f(q_0, q_1) &\mapsto \{q_1\} \end{aligned}$$

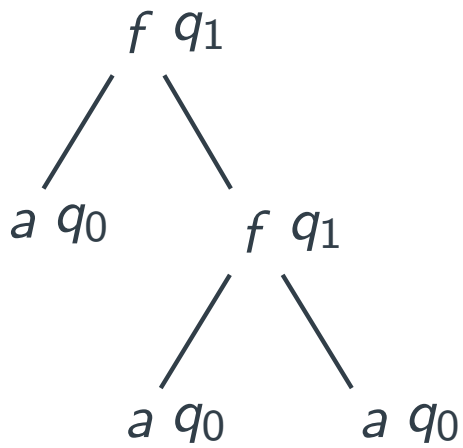


Exemple

université PARIS-SACLAY

On considère l'automate $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_1\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{rcl} a & \mapsto & \{q_0\} \\ f(q_0, q_0) & \mapsto & \{q_1\} \\ \blacktriangleright \quad f(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$



Exemple (2)

université PARIS-SACLAY

Les automates « commencent » leur run aux feuilles, qui servent d'état initial.

Dans l'exemple précédent, l'automate est incomplet. On peut le compléter comme dans le cas des mots :

$$\delta : \begin{array}{rcl} a & \mapsto & \{q_0\} \\ f(q_0, q_0) & \mapsto & \{q_1\} \\ f(q_0, q_1) & \mapsto & \{q_1\} \\ f(q_1, q_0) & \mapsto & \{q_\perp\} \end{array} \quad \begin{array}{rcl} f(q_0, q_\perp) & \mapsto & \{q_\perp\} \\ f(q_\perp, q_0) & \mapsto & \{q_\perp\} \\ f(q_1, q_\perp) & \mapsto & \{q_\perp\} \\ f(q_\perp, q_1) & \mapsto & \{q_\perp\} \\ f(q_\perp, q_\perp) & \mapsto & \{q_\perp\} \end{array}$$

Le nombre d'états rajouter étant polynomial (il faut considérer tous les $f(\ldots, q_\perp, \ldots)$, on donnera souvent des automates incomplets).

Exemple (3)

université PARIS-SACLAY

On se donne l'automate $\mathcal{A}_{\text{prop}} = (\{q_0, q_1\}, \{\vee^2, \wedge^2, \neg^1, \mathsf{F}, \mathsf{T}\}, \delta, \{q_1\})$ reconnaisant les formules propositionnelles vraies :

$$\delta : \begin{array}{rcl} \mathsf{F} & \mapsto & \{q_0\} \\ \mathsf{T} & \mapsto & \{q_1\} \\ \neg(q_0) & \mapsto & \{q_1\} \\ \neg(q_1) & \mapsto & \{q_0\} \\ \wedge(q_0, q_0) & \mapsto & \{q_0\} \\ \wedge(q_0, q_1) & \mapsto & \{q_0\} \end{array} \qquad \begin{array}{rcl} \wedge(q_1, q_0) & \mapsto & \{q_0\} \\ \wedge(q_1, q_1) & \mapsto & \{q_1\} \\ \vee(q_1, q_1) & \mapsto & \{q_1\} \\ \vee(q_0, q_0) & \mapsto & \{q_0\} \\ \vee(q_1, q_0) & \mapsto & \{q_1\} \\ \vee(q_0, q_1) & \mapsto & \{q_1\} \end{array}$$

Exemple d'automate non déterministe

université PARIS-SACLAY

Les exemples précédents sont déterministes. Les exemples pour le non déterminisme des automates de mots se transposent facilement. On considère l'automate reconnaissant les arbres sur $\Sigma = \{f^2, g^2, \#\}$ ayant comme racine le motif $f(f(x,y),z)$ ou $f(z,f(y,z))$, pour x, y, z quelconque dans $\mathcal{T}(\Sigma)$: $(\{q_0, q_1, q_2\}, \Sigma, \delta, \{q_2\})$.

$$\delta : \begin{array}{rcl} \# & \mapsto & \{q_0\} \\ f(q_0,q_0) & \mapsto & \{q_0,q_1\} \\ g(q_0,q_0) & \mapsto & \{q_0\} \end{array} \qquad \begin{array}{rcl} f(q_1,q_0) & \mapsto & \{q_2\} \\ f(q_0,q_1) & \mapsto & \{q_2\} \end{array}$$

Déterminisation

université PARIS-SACLAY

Théorème

Soit $\mathcal{A}$ un automate d'arbre non-déterministe ascendant. Il existe un automate d'arbre déterministe ascendant reconnaissant le même langage.

La preuve et la procédure de déterminisation sont les mêmes que pour le cas des mots.

Déterminisation (2)

Si $\mathcal{A} = (Q, \Sigma, \delta, F)$, on définit $\mathcal{A}_{\text{det}} = (Q_{\text{det}}, \Sigma, \delta_{\text{det}}, F_{\text{det}})$ par :

- $Q_{\text{det}} = \mathcal{P}(Q)$
- $F_{\text{det}} = \{s \in Q_{\text{det}} \mid s \cap F \neq \varnothing\}$
- $\delta_{\text{det}} : f(s_1, \ldots, s_k) \mapsto \{q \in \delta(f(q_1, \ldots, q_k)) \,|q_1 \in s_1, \ldots, q_k \in s_k \in\}$

Attention, cette construction ajoute plein d'états inutiles que l'on peut ensuite supprimer sans changer le langage.

Minimisation

université PARIS-SACLAY

Théorème (Myhill-Nerode)

*Soit $\mathcal{A}$ un automate d'arbre **déterministe** ascendant. Il existe $\mathcal{A}_{min}$ avec $L_{\mathcal{A}_{min}} = L_{\mathcal{A}}$ tel que pour tout $\mathcal{A}'$, $|\mathcal{A}'| < |\mathcal{A}_{min}| \Rightarrow L_{\mathcal{A}'} \neq L_{\mathcal{A}_{min}}$*

Même algorithme de minimisation que dans le cas des mots (exercice).

Propriétés de clôtures

université PARIS-SACLAY

Les langages d'arbres reconnaisables sont clos par union, intersection et complémentaire.

- Pour l'union : on réunit juste les deux automates (union de l'ensemble des états, union de l'ensemble des états finaux et union de δ).
- Pour l'intersection : la construction produit fonctionne comme dans le cas des mots.
- Pour le complémentaire : on peut déterminiser, compléter l'automate (avec un état puits) et inverse état acceptants et non acceptants.

Lemme de la pompe pour les arbres

université PARIS-SACLAY

Lemme (Lemme de la pompe)

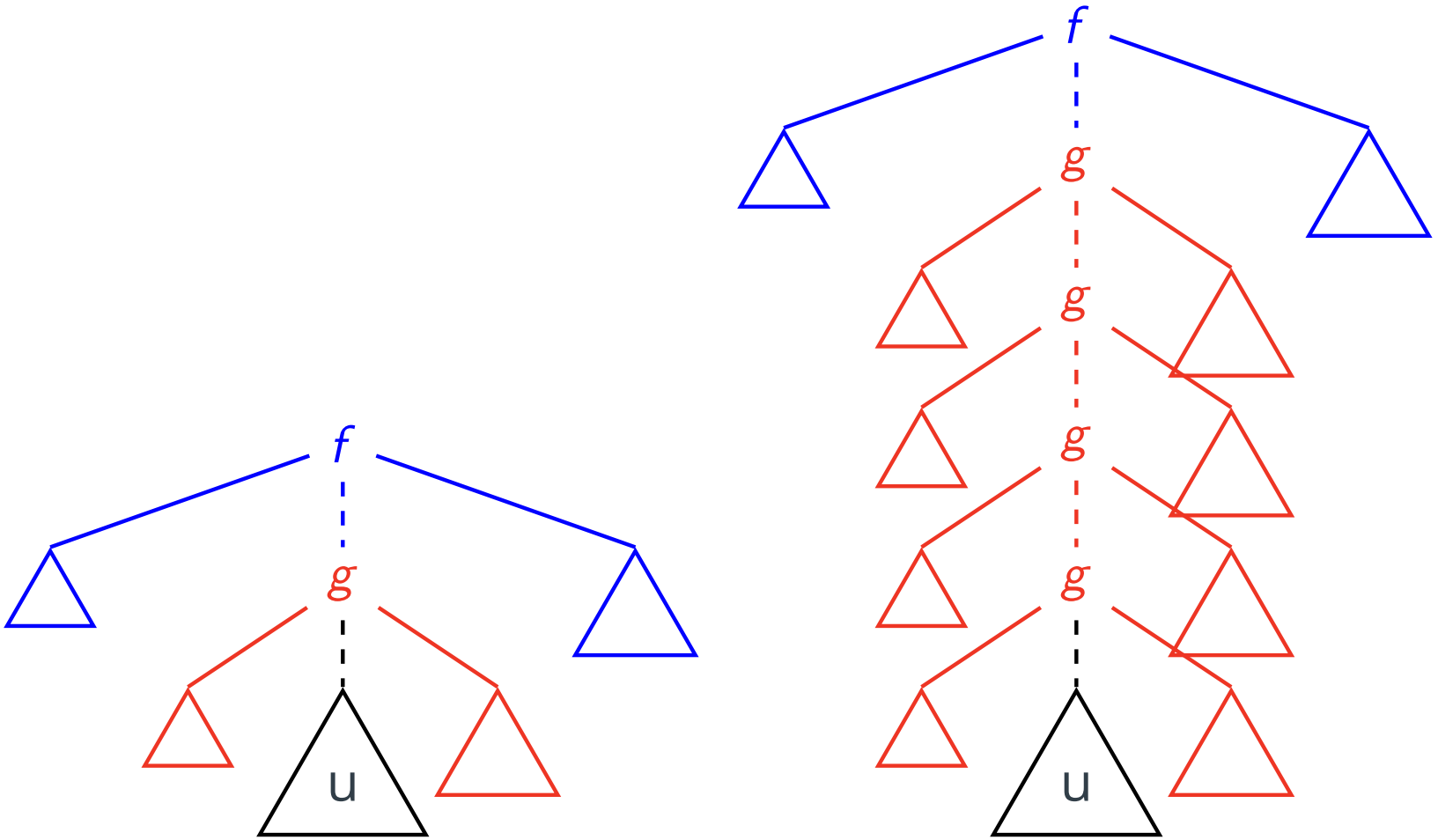
Soit T un langage régulier d'arbre sur Σ. Il existe $p \geq 1$ tel que pour pour tout $t \in T$ tel que $\text{hauteur}(t) \geq p$, il existe un contexte $C \in \mathcal{T}(\Sigma, \{\Box\})$ et un contexte non trivial $C' \in \mathcal{T}(\Sigma, \{\Box\})$ et un arbre $u \in \mathcal{T}(\Sigma)$ tel que $t = C[C'[u]]$ et pour tout $n \geq 0$, $C[C'^n[u]] \in T$.

La fonction $\text{hauteur}(t)$ est définie inductivement sur $t \in \mathcal{T}(\Sigma)$ par :

- $\text{hauteur}(x) = 1, \forall x \in \Sigma_0$ (les feuilles sont de hauteur 1)
- $\text{hauteur}(f(t_1, \ldots, t_k)) = 1 + \max\{\text{hauteur}(t_i) \mid 1 \leq i \leq |f|\}$

Un contexte est un arbres avec un unique $\Box$ à l'intérieur. Le contexte trivial est $\Box$.

Lemme de la pompe illustré



$C[\ ]$: contexte depuis la racine

$C'[\ ]$: contexte intermédiaire $\Rightarrow$ partie que l'on peut pomper en restant dans le langage.

Plan

1 Introduction

2 Arbres n-aires

3 Automates d'arbres

4 Automates descendants

Parcours de l'arbre

université PARIS-SACLAY

Dans le cas des mots, il semble naturel de commencer par le début du mot lors du calcul d'un run. On remarque qu'on pourrait sans difficulté partir de la fin (un état acceptant) puis lire les transitions à l'envers en lisant le mot en sens inverse.
Dans le cas des arbres, la reconnaissance commence aux feuilles, ce qui semble peu naturel. Peut-on « exécuter » l'automate en partant de la racine ?

Automate d'arbres descendant non-déterministe

université PARIS-SACLAY

Définition (Automate d'arbre non-déteriniste)

Un automate d'arbre ascendant non-déterministe, NTDTA (non-deterministic top-down finite tree automaton) est un $4-$ *uplet* $\mathcal{A} = (Q, \Sigma, \delta, I)$ *défini par*

- *un ensemble d'états* Q
- *un alphabet gradué* Σ
- *un ensemble d'états initiaux* I
- *une fonction de transition* $\delta : Q \times \Sigma \to \mathcal{P}_f(Q^*)$ *de la forme :*

$$q, f \mapsto \{(q_{1_1}, \ldots, q_{1_k}), \ldots, (q_{m_1}, \ldots, q_{m_k})\}$$

avec pour $f \in \Sigma$ *et* $|f| = k$.

Automate d'arbres descendant non-déterministe (2)

université PARIS-SACLAY

La fonction δ prend en argument un état et dit pour chaque f de Σ dans quel état poursuivre l'exécution sur les fils de f.

Si δ renvoie toujours un singleton, l'automate est déterministe.

Run d'un automate d'arbre descendant

université PARIS-SACLAY

Définition

Une exécution (ou run) d'un automate d'arbre $\mathcal{A} = (Q, \Sigma, \delta, I)$ *pour un arbre* $t \in \mathcal{T}(\Sigma)$ *est une fonction* $r : dom(t) \to Q$ *telle que*

$\forall p \in dom(t),\ (r(p1), \ldots, r(pk)) \in \delta(r(p), t(p))$*, avec* $|t(p)| = k$*.*

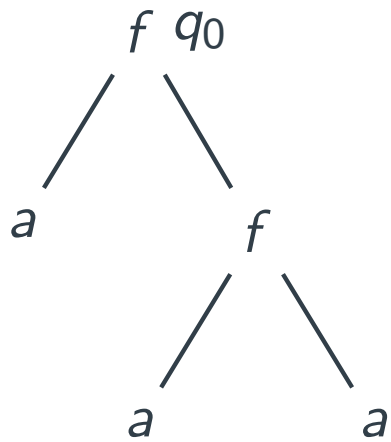
Un run est dit acceptant si et seulement si $r(\epsilon) \in I$*.*

Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{lcl} q_0, f & \mapsto & \{(q_1, q_0)\} \\ q_0, a & \mapsto & \{\} \\ q_1, a & \mapsto & \{\} \end{array}$$

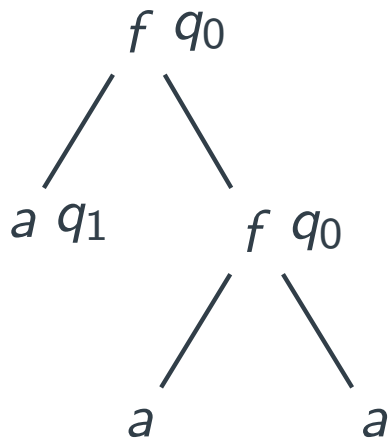


Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \quad \begin{array}{lcl} \blacktriangleright \quad q_0, f & \mapsto & \{(q_1, q_0)\} \\ q_0, a & \mapsto & \{\} \\ q_1, a & \mapsto & \{\} \end{array}$$

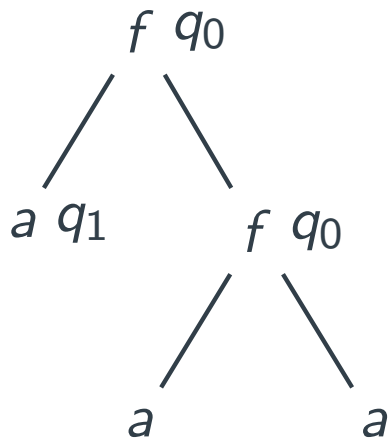


Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{lll} q_0, f & \mapsto & \{(q_1, q_0)\} \\ q_0, a & \mapsto & \{\} \\ \blacktriangleright\ q_1, a & \mapsto & \{\} \end{array}$$

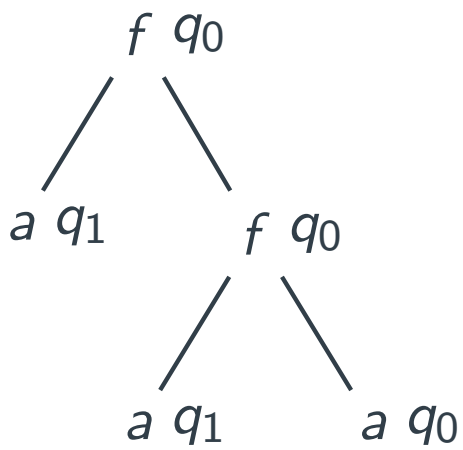


Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{lll} \blacktriangleright & q_0, f & \mapsto & \{(q_1, q_0)\} \\ & q_0, a & \mapsto & \{\} \\ & q_1, a & \mapsto & \{\} \end{array}$$

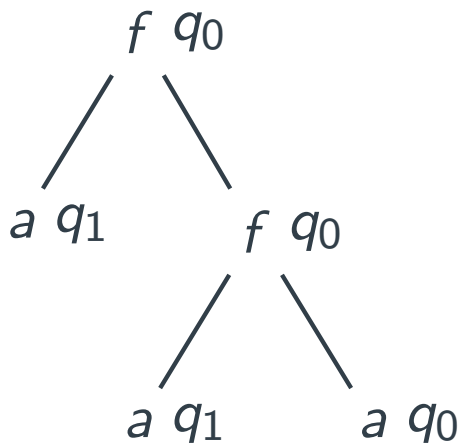


Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \begin{array}{lll} q_0, f & \mapsto & \{(q_1, q_0)\} \\ q_0, a & \mapsto & \{\} \\ \blacktriangleright \quad q_1, a & \mapsto & \{\} \end{array}$$

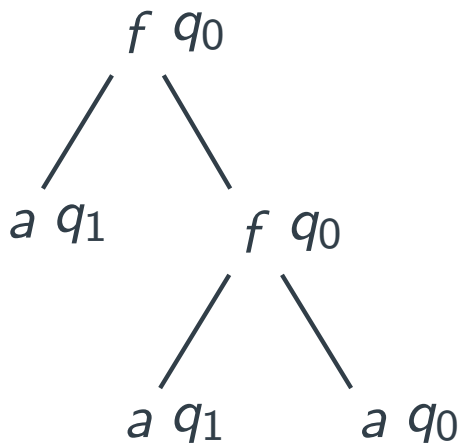


Exemple

université PARIS-SACLAY

On considère l'automate descendant $\mathcal{A} = (\{q_0, q_1\}, \{f^2, a\}, \delta, \{q_0\})$ reconnaissant les « peignes » c'est à dire les arbres binaires linéraires de feuilles a et de nœud interne f. Le run sur l'arbre $t = f(a, f(a, a))$ est donné ci-dessous.

$$\delta : \blacktriangleright \quad \begin{array}{lcl} q_0, f & \mapsto & \{(q_1, q_0)\} \\ q_0, a & \mapsto & \{\} \\ q_1, a & \mapsto & \{\} \end{array}$$



Bottom-up vs Top-down

université PARIS-SACLAY

Théorème (Équivalence NTDTA-NTA)

L'ensemble des langages reconnnus par un automate ***non-déterministe descendant*** *est exactement l'ensemble des langages reconnaissable par un automate ascendant.*

Preuve : on remplace états finaux et initiaux et on lit les transitions à l'envers.

Top-down déterminstes

université PARIS-SACLAY

Théorème (Non-équivalence DTDTA-NTA)

L'ensemble des langages reconnnus par un automate ***déterministe descendant*** *est strictement inclus dans l'ensemble des langages reconnaissable par un automate ascendant.*

Donc : $TD_{\text{det}} \subsetneq TD_{\text{nondet}} = BU_{\text{nondet}} = BU_{\text{det}}$

Contre-exemple

université PARIS-SACLAY

Sur $\Sigma = \{f^2, a, b\}$ le langage $\{f(a,b), f(b,a)\}$ n'est pas reconnaissable par un automate déterministe descendant.
En effet, si l'automate commence en q_0 sur f, il doit à aller à gauche dans un état q_1 et à droite dans un état q_2. Hors, q_1 doit reconnaitre a ou b et de même pour q_2. Mais dans ce cas l'automate reconnaît aussi $f(a,a)$ et $f(b,b)$ qui ne sont pas dans le langage.

Intuitivement, un automate TD déterministe doit décider en fonction uniquement du chemin parcourus par la racine dans quel état aller pour chacun de ses fils.