Cours 3/3/21

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

$O(|V| + |E|)$

Algorithme linéaire pour calculer

$Attr_0(F)$ $\quad F \subseteq V$

- File Q de sommets
- Tableaux R, donc de sommets
- Tableau C (counter) :

$0 \leq C(v) \leq |Post(v)|$

On suppose que Pre, Post sont disponibles (sous forme de tableau de listes).

R : à la fin c'est $Attr_0(F)$

donc : sommets déjà traités

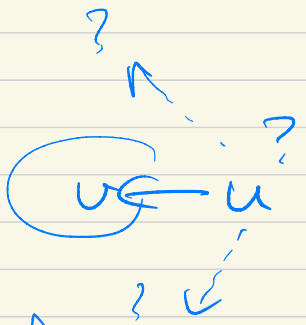
Q : : sommets en cours de traitement

C : utilisé juste pour V_1

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Initialisation :

Pour tout $v \in V$ faire
    $c(v) := |Post(v)|$;
    done(v) := false
$Q \leftarrow \phi$; $R \leftarrow \phi$;
Pour tout $v \in F$ faire
    $Q \leftarrow$ enfiler(Q, v)
    done(v) := true ;
Tant que $(Q = \phi)$ faire
    $v \leftarrow$ defiler(Q)
    $R := R \cup \{v\}$
    Pour tout $(u \in Pre(v))$ faire
        si $(u \in V_0$ & done(u) = false$)$
        alors $Q \leftarrow$ enfiler(Q, u)
            done(u) := true
        si $(u \in V_1$ & done(u) = false$)$
        alors $c(u) := c(u) - 1$;
            si $c(u) = 0$ alors
            $Q \leftarrow$ enfiler(Q, u)
            done(u) := true
Retourner(R)



Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Propriétés

① À cause de "donc", chaque sommet peut rentrer au plus 1 fois dans Q.

↝ Enfiler(Q, u) est exécuté au plus 1 fois.

⇒ chaque arête ($u \in Pre(v)$) sera considérée au plus 1 fois

Tout autre opération : $O(1)$

⇒ complexité $O(|V| + |E|)$

② Pour tout sommet $u \in V$:

- si $u \in V_0$ et u est rajouté à R, alors $Post(u) \cap R = \emptyset$, ou $u \in F$.

↑ int.

- si $u \in V_1$ et u est rajouté à R, alors :

$Post(u) \subseteq R$, ou $u \in F$

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

(2) $\Rightarrow$ $R \subseteq Attr_0(F)$

Stratégie d'attracteur pour P_0 :

$\sigma_0(u) = v$ si u a été rajouté à Q lors du traitement de v

(3) Pour tout sommet $u \in V$ on a :

- si $u \in V_0$ et $Post(u) \cap R \neq \emptyset$, alors l'algo rajoute u à R
- si $u \in V_1$ et $Post(u) \subseteq R$, alors l'algo rajoute u à R.

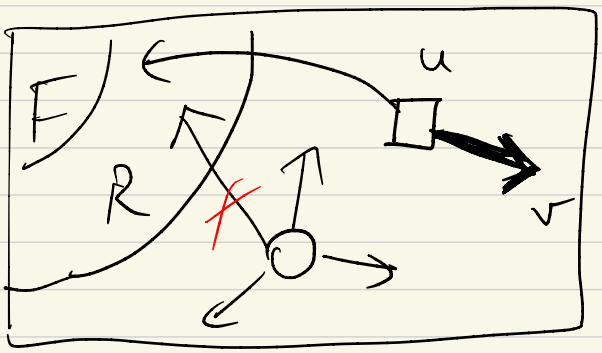
$c(u) = 0$

$\rightsquigarrow$ u sera rajouté à Q, donc à R (plus tard)

Les $u \in V \setminus R$ ont la propriété :

- $u \in V_0$: $Post(u) \subseteq V \setminus R$
- $u \in V_1$: $Post(u) \cap (V \setminus R) \neq \emptyset$

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0



$\sigma_1(u) = v$

$R = W_0$ région gagnante de P_0 (atteindre F)

$V \setminus R = W_1$ — " — de P_1 (éviter F)

Application ① jeux pour la bisimulation

2 systèmes de transitions finis

$A_i = (Q_i, \Sigma, q_{0,i}, \delta_i)$

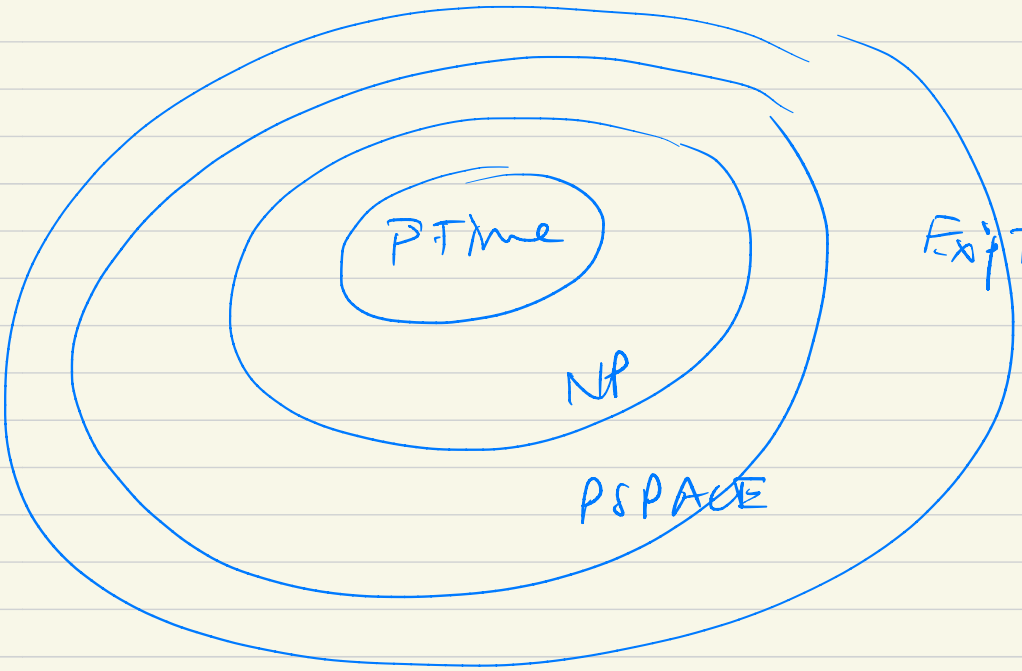
$\delta_i \subseteq Q_i \times \Sigma \times Q_i$

[Avec états finaux on peut demander ↵ $L(A_1) = L(A_2)$]

DFA (dét.) → minimisation PTime

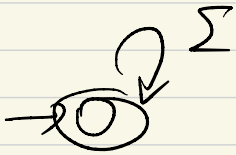
NFA (non-dét.) → PSPACE-c.

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0



PSPACE : espace poly

Pb. PSPACE-c :



Input NFA A

Q : $L(A) = \Sigma^*$?

| universalité

Input $A_1, \dots, A_n$ (N/D)FA

Q : $L(A_1) \cap \dots \cap L(A_n) \neq \emptyset$?

cas spécial de $L(A_1) \stackrel{?}{=} L(A_2)$

A_1, A_2 NFA

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

2 STT $A_i = (Q_i, \Sigma, q_{0,i}, \delta_i)$

$\delta_i \subseteq Q_i \times \Sigma \times Q_i$

Une rel de bisimulation R entre A_1, A_2 est une rel. binaire $R \subseteq Q_1 \times Q_2$

tq :

(1) $(q_{0,1}, q_{0,2}) \in R$

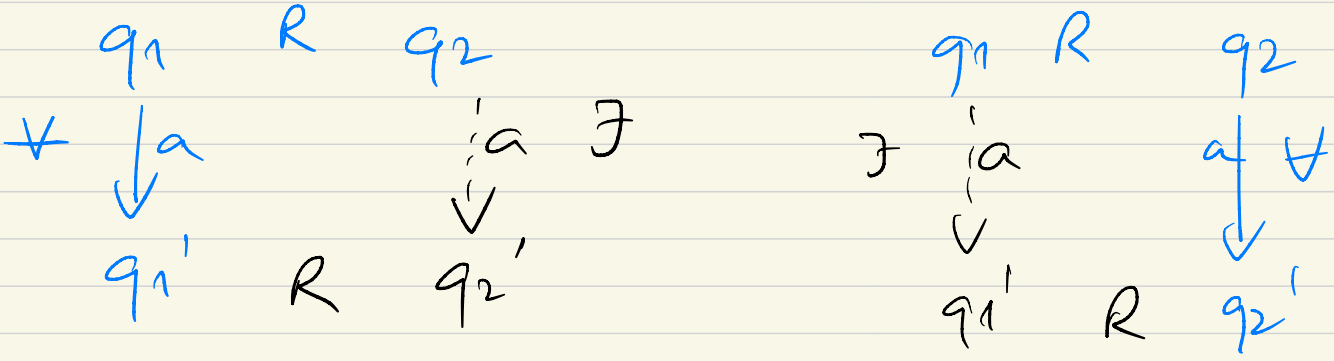
(2) $\forall (q_1, q_2) \in R \quad \forall a \in \Sigma$:

(2.1) $\forall\ q_1 \xrightarrow{a} q_1' \quad \exists\ q_2 \xrightarrow{a} q_2'$:

$(q_1', q_2') \in R$

(2.2) $\forall\ q_2 \xrightarrow{a} q_2' \quad \exists\ q_1 \xrightarrow{a} q_1'$:

$(q_1', q_2') \in R$



(3) Avec F_1, F_2 étab finaux :

$R \subseteq (F_1 \times F_2) \cup ((Q_1 \setminus F_1) \times (Q_2 \setminus F_2))$

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Jeu de bisimulation associé à A_1, A_2 :

- arène : $\underbrace{(Q_1 \times Q_2)}_{= V_0} \cup \underbrace{(Q_1 \times Q_2 \times \Sigma \times \{1,2\})}_{= V_1}$

coup :

$(q_1, q_2) \longrightarrow (q_1', q_2', a, i)$ si
$\in V_0$

ou $\begin{cases} i=1, & q_1 \xrightarrow[1]{a} q_1', \quad q_2' = q_2 \\ i=2, & q_2 \xrightarrow[2]{a} q_2', \quad q_1' = q_1 \end{cases}$

$(q_1, q_2, a, i) \to (q_1', q_2')$ si

ou $\begin{cases} i=1, & q_2 \xrightarrow[2]{a} q_2', \quad q_1' = q_1 \\ i=2, & q_1 \xrightarrow[1]{a} q_1', \quad q_2' = q_2 \end{cases}$

P_0 : Spoiler (veut montrer qu'il n'y a pas de bisim.)

P_1 : Duplicator (veut montrer qu'il y a une bisim.)

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Po gagne si on arrive dans

F ↙ | (q_1, q_2, a, i) tq.
| $i=1$ et $q_2 \not\xrightarrow{a}_2$, ou
| $i=2$ et $q_1 \not\xrightarrow{a}_1$

P1 gagne toute partie infinie, ainsi que les parties qui finissent dans états puits (q_1, q_2) :

$\forall a \in \Sigma \quad q_1 \not\xrightarrow{a}_1, \; q_2 \not\xrightarrow{a}_2$

Propriété :

$(q_{0,1}, q_{0,2})$ gagnant pour Po $\Leftrightarrow$ il n'y a pas de relation de bisimulation entre A_1, A_2

$R = \{(q_1, q_2) : (q_1, q_2)$ gagnant pour P1 dans le jeu de bisimulation$\}$ est bisimulation

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Application ②

Automates finis $\begin{cases} \text{dét.} \to \text{unique calcul} \\ \text{non-dét.} \to \exists \\ \text{alternants} \end{cases}$

A NFA, $w \in \Sigma^*$

$w \in L(A)$ s'il ex. calcul acc. de A sur w

Dual : $w \in L(A)$ si tous les calculs de A sur w sont acceptants

Un automate alternant

$A = (Q, \Sigma, q_0, \delta, F)$

Q états, $F \subseteq Q$, $q_0 \in Q$

$\delta : Q \times \Sigma \to Bool^+(Q)$

↓

formules bool. sans nég. sur Q

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Ex

$\varphi = (q_1 \wedge q_2) \vee q_3 \quad \in Bool_+(Q)$

$p \xrightarrow{a} (q_1 \wedge q_2) \vee q_3$

choix universel — choix non-dét.

$R \subseteq Q$, $\varphi \in Bool_+(Q)$

On écrit $\boxed{R \models \varphi}$ si R satisfait φ, c-ad si φ est vraie pour la valuation

$$val(q) = \begin{cases} \text{true} & \text{si } q \in R \\ \text{false} & \text{sinon} \end{cases}$$

$\{q_1, q_2\} \models \varphi$, $\{q_3\} \models \varphi$

$\{q_1, q_2, q_3\} \models \varphi$

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

$A = (Q, \Sigma, \delta, q_0, F)$

$w = a_1 \dots a_n \in \Sigma^*$

Jeu $G(A, w)$:

- arène $(Q \times \{1, \dots, n\}) \cup (2^Q \times \{1, \dots, n\})$
$=: V_0 \quad =: V_1$

- cmpl :

$(q, i) \longrightarrow (R, i)$ si $R \subseteq Q$

est tq. $R \vdash \delta(q, a_i)$

$(R, i) \longrightarrow (q', i+1)$ tq.

$q' \in R$

On suppose que $\forall q \in Q, \forall a \in \Sigma$:

$\delta(q, a) \neq$ false, true

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

Condition de victoire pour P_0 dans $G(A,w)$

$\rightarrow$ atteindre $F \times \{n\}$

$w \in L(A) \overset{\text{def}}{\iff}$ P_0 gagne ce jeu d'accessibilité à partir de $(q_0, 0)$

Ex. $L = \{ ww : w \in \{a,b\}^n \}$

Tout NFA A tq. $L(A) = L$ est de taille exp. (A doit mémoriser la première moitié).

Mais il ex. un aut. alt. B de taille poly tq. $L(B) = L$

B : pour chaque $1 \le i \le n$:

la $i$ème lettre = la $(n+i)$ème lettre

$\rightsquigarrow$ $Q = \{1, \dots, n\}^2 \times \Sigma \cup \{q_0\}$

$\cup \{1, \dots, n\}$

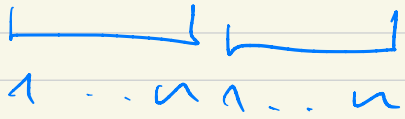
Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

$$\delta(q_0, \underset{b}{a}) = 1 \wedge (1, 0, \underset{b}{a})$$

$$\delta(1, \underset{b}{a}) = 2 \wedge (2, 0, \underset{b}{a})$$

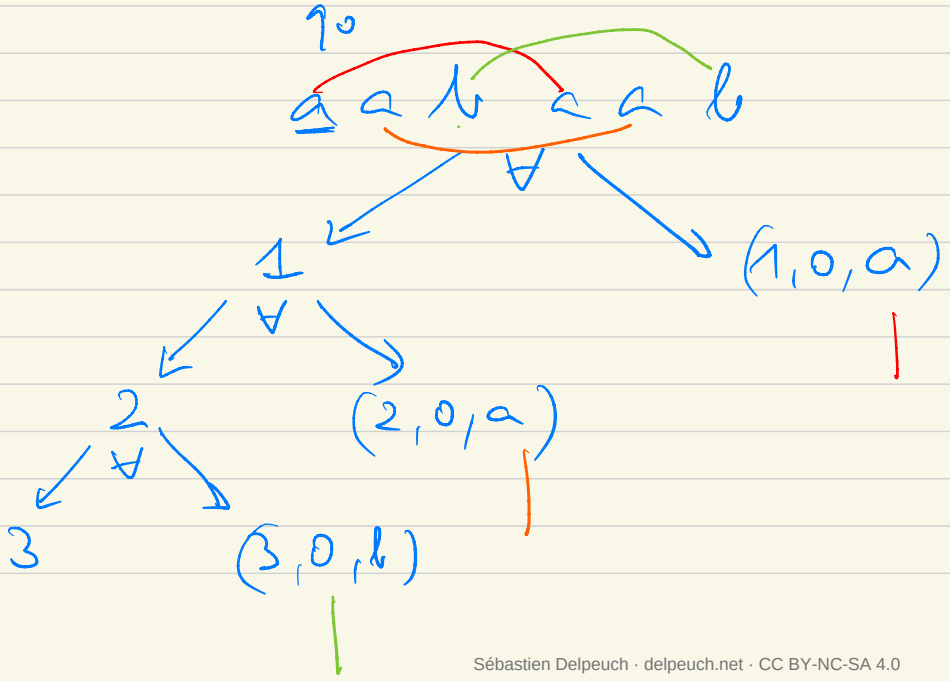
$$\vdots$$

$$\delta(n, \underset{b}{a}) = n \wedge (n, 0, \underset{b}{a})$$



$$\delta((i, 0, a), c) = (i, 1, a) \quad , c \in (a, b)$$

$$\delta((i, j, a), c) = \begin{cases} (i, j+1, a) & \text{si } j \neq n \\ \text{true} & \text{si } j \geq n, c = a \end{cases}$$



Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0

exp. exp.

DFA = NFA = AFA aut. forms

det. non-det. alternant

$A_1, A_2, \dots$ An NFA de taille $\leq m$

$A_i = (Q_i, \Sigma, q_{0,i}, \delta_i, F)$

$\Downarrow$

AFA A tq.

$L(A) = \bigcap_{i=1}^{n} L(A_i)$ de taille $\mathcal{O}(n \cdot m)$

$Q = \bigcup_{i=1}^{n} Q_i \times \{1, \dots, n\} \cup \{q_0\}$

$\delta(q_0, a) = \bigwedge_{i=1}^{n} \bigvee_{q_{0,i} \xrightarrow{a}_i q_i} (q_i, i)$

intersection des A_i

$\delta((q,i), a) = \bigvee_{q \xrightarrow{a}_i p} (p, i)$

$F = \bigcup F_i \times \{i\}$

Sébastien Delpeuch · delpeuch.net · CC BY-NC-SA 4.0