

Alphabets et Mots

⑤ (Alphabet): ensemble fini & non vide de symboles appelés lettres. @ $X_1 = \{a, b, c\}$, $X_2 = \{0, 1\}$, $X_3 = \{\oplus, \otimes, \odot\}$

⑥ (Mots): Un mot $\alpha\beta$ est une suite finie, éventuellement vide, de lettres. On désigne le vide : ϵ .

@ X_1 : babaca, a, bca, ϵ , d ; X_2 : 001, ϵ , 0, 1, 1001.

Concaténation:

- C'est deux mots u, v : mot constitué de u puis de v .
- opérateur de \sqcup est le point.

$$\begin{aligned} @ u = ab, v = bca \rightarrow u.v = abbca \\ u = \epsilon, v = \epsilon \rightarrow u.v = \epsilon \\ u.\epsilon = \epsilon.u = u \end{aligned}$$

- \sqcup est associative $(u.v).w = u.(v.w) \Rightarrow u.v.w$.

⑦ longueur d'un mot, nbr occurrences

- $|u|$ désigne sa longueur
- $|u|_x$ désigne nbr d'occurrences de x dans u .

$$@ |bababa| = 6, |a| = 1, |\epsilon| = 0.$$

Pptés: • $|u.v| = |u| + |v|$

• $\forall n \in \mathbb{N}, |u.v|_n = |u|_n + |v|_n$

Itérat de la concaténat: la nota u^i , $i \in \mathbb{N}$

$$u^0 = \epsilon$$

$$u^{i+1} = u^i \cdot u = u \cdot u^i \text{ pour } i \geq 0.$$

$$@ u = ab, u^3 = ababab$$

Pptés: $\epsilon^m = \epsilon$ pour $m \geq 0$.

Facteur (déf° formelle):

- un mot u est un Facteur du mot v si \exists des mots p et s tq $v = p u s$. (Rq: p, s peuvent être vides)
- si $\exists s$, $v = u s$ alors u est facteur gauche (préfixe).
- si $\exists p$, $v = p u$ alors u est facteur droit (suffixe).
- si u est facteur de v et $u \neq \epsilon$ et $u \neq v$ alors u est un facteur propre de v .

Facteur (déf° non-formelle):

- u facteur de v : u est "sous-chaine" de v .
- u préfixe de v : v commence par u
- u suffixe se termine

- @ ab est facteur de aaba mais il n'en est ni fg ni fd.
- @ ab est fg (dc f.) de aaba MS pas fd.
- @ E est fg et fd de abbab MS ce n'est pas fg propre.

Langage :

- Un langage sur un alphabet X est un ensemble de mots sur X .
- NB: un langage n'est pas nécessairement fini.

langages sur $\{a, b\}$. $X = \{a, b\}$.

$$\begin{aligned} L_1 &= \{ab, ba\} \quad (2 \text{ mots}) ; \quad L_3 = \{\epsilon\} \quad (1 \text{ mot}) \\ L_5 &= \{(ab)^i \mid i \geq 0\} \quad (\infty \text{ mots}) ; \quad L_4 = \{\emptyset\} \quad (0 \text{ mot}) \end{aligned}$$

Concaténat de langage :

si L & L' lang sur X .

$$L \cdot L' = \{u \cdot v \mid u \in L, v \in L'\}$$

$$\begin{aligned} X &= \{a, b, c\} \\ \{a, ab, ba\} \cdot \{c, ca\} &= \left\{ \begin{array}{l} ac, abc, bac,aca, \\ abca, baca \end{array} \right\} \end{aligned}$$

Catégorielle

$$L_1 \cdot (L_2 \cdot L_3) = (L_1 \cdot L_2) \cdot L_3 = L_1 \cdot L_2 \cdot L_3$$

elt neutre : $\{\epsilon\}$

$$\forall L, \{\epsilon\} \cdot L = L \cdot \{\epsilon\} = L$$

elt absorbant:

\emptyset est elt absorbant de ces langages.
 $\forall L, \emptyset \cdot L = L \cdot \emptyset = \emptyset$.

Itérat de ces de langages :

- $L^0 = \{\epsilon\}$
- $L^{i+1} = L^i \cdot L = L \cdot L^i, i \geq 0$.

Clôture de Kleene :

$\bigcup_{i \geq 0} L^i$: clôture de Kleene.

on note $L^* = \bigcup_{i \geq 0} L^i$ (étoile de Kleene)

$$\text{NB: } \circ \{\epsilon\}^* = \{\epsilon\}$$

$$\circ \emptyset^* = \{\epsilon\}, \text{ par définit.}$$

• si L contient un mot non vide alors L^* est infini

$$\{ba\}^2 = \{babab\}$$

$$\{ba\}^* = \{\epsilon, ba, baba, bababa, \dots\}$$

$$\{ba, c\}^* = \{\epsilon, ba, c, babab, bac, cba, cc, \dots\}$$

Monoïde : si $X: \omega\beta$; $X^* = \{ \text{mots sur l'alphabet } X \}$

- X^* : expression simple \rightarrow ens mots utilisant $\omega\beta \times$
- X^* : monoïde libre engendré par X .

Langages Rationnels

Expression rationnelle :

- symboles atomiques : formée par lettres, $a, b, \epsilon, \emptyset$.
- les 3 opérations: $* \cdot +$
- $()$
- $a+b, (a.b)^*, (a+b).(b+c).a^*$

Langage $\xleftarrow{A \leftrightarrow} ER \uparrow$:

Expr Atomiqs

ER	$L \xleftarrow{A \leftrightarrow}$
x (lettre)	$\{xy\}$
\emptyset	ens vide
ϵ	$\{\epsilon\}$

Définition opérateurs

- * clôture de Kleene
- + union
- \cdot concaténation

a

$a+b$

$a \cdot b$

a^*

$(a \cdot b)^*$

$a+b \cdot c$

$(a+b) \cdot c^*$

$\{a\}$

$\{a\} \cup \{b\} = \{a, b\}$

$\{a\} \cdot \{b\} = \{ab\}$

$\{ \epsilon, a, aa, aaa, \dots \}$

$\{ \epsilon, ab, abab, ababab, \dots \}$

$\{a\} \cup (\{b\} \cdot \{c\}) = \{a, bc\}$

$\{a, b\} \cdot (\{c\}^*)$

$= \{a, b, ac, bc, acc, bcc, \dots \}$

Langages Rationnels

Expression rationnelle :

- symboles atomiques : formée par lettres, $a, b, \varepsilon, \emptyset$.
- les 3 opérateurs: $*$. +
- $(\)$
- $a+b, (a.b)^*, (a+b).(b+c).a^*$

Langage $\xleftarrow{A \in} ER$:

Expr. Atomiques

ER	$L \xleftarrow{A}$
x (lettre)	$\{x\}$
\emptyset	lang. vide
ε	$\{\varepsilon\}$

Définition opérateurs

*	clôture de Kleene
+	union
.	concaténation

ER	$L \xleftarrow{A \in}$
a	$\{a\}$
$a+b$	$\{a\} \cup \{b\} = \{a, b\}$
$a.b$	$\{a\} \cdot \{b\} = \{ab\}$
a^*	$\{\varepsilon, a, aa, aaa, \dots\}$
$(a.b)^*$	$\{\varepsilon, ab, abab, ababab, \dots\}$
$a+b.c$	$\{a\} \cup (\{b\} \cdot \{c\}) = \{a, bc\}$
$(a+b).c^*$	$\{a, b\} \cdot (\{c\}^*)$
	$= \{a, b, ac, bc, acc, bcc, \dots\}$

Langage \xleftarrow{A} ER

Chq expression $e \xleftarrow{A}$ un langage $\mathcal{L}(e)$ défini par :

- 1) $\mathcal{L}(\emptyset) = \emptyset, \mathcal{L}(\varepsilon) = \{\varepsilon\}, \mathcal{L}(x) = \{x\} \quad \forall x \in X$
- 2) $\mathcal{L}(e_1) = \mathcal{L}(e_1)$
- 3) $\mathcal{L}(e_1 + e_2) = \mathcal{L}(e_1) \cup \mathcal{L}(e_2)$
- 4) $\mathcal{L}(e_1 \cdot e_2) = \mathcal{L}(e_1) \cdot \mathcal{L}(e_2)$
si $\nexists e'$ et e'' tq $e_1 = e' + e''$ ou $e_2 = e' + e''$
- 5) $\mathcal{L}(e_1^*) = \mathcal{L}(e_1)^*$ si $\nexists e'$ et e'' tq $e_1 = e' + e''$ ou $e_1 = e' \cdot e''$

$\rightarrow \mathcal{L}(e)$: langage dénoté par e .

Lang. dénotés p ER: ER (ou réguliers) $\xrightarrow{\text{représenter langage}}$ @ $\{a^m b^n\}$
tous trop complexes pour être écrit en ER

③ Lang. rationnel (régulier) est un lang. q pt être dénoté par ER.

RAT: famille langages réguliers

RAT est + ptte famille lang q contient $\emptyset, \{\varepsilon\}, \{a\}$; est close par union, concaténation, et de Kleene

③

Tout lang fini \in RAT

Il existe lang non rationnels

Automates finis Déterministes

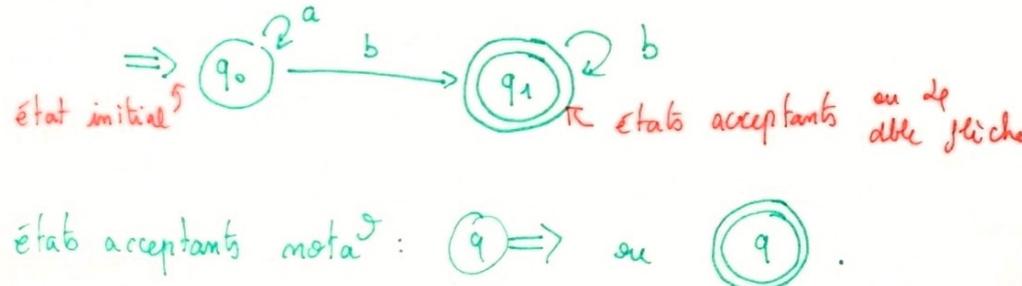
⑤ Un AFD est défini par :

- un alphabet X
- un ensemble non vide Q : ens d'états
- état initial $q_{ini} \in Q$.
- \exists ens $F \subseteq Q$ d'états dits acceptants (ou finals)
- une fonction $\delta: Q \times X \rightarrow Q$: fonction de transition.

Notation: $\delta(q_d, x) = q_a$, moté aussi $q_d \xrightarrow{x} q_a$
 (départ) (arrivée)

@ automate A_1

- $\Sigma = \{a, b\}$
- ens états $Q = \{q_0, q_1\}$
- ens états acceptants $F = \{q_1\}$
- état initial: q_0
- $\delta(q_0, a) = q_0$; $\delta(q_0, b) = q_1$, $\delta(q_1, b) = q_1$,



Automate: "machine à lire des mots"

après lecture complète arrivent au état acceptant.

⑥ @ abb ds automate A_1 :

$$q_0 \xrightarrow{a} \delta(q_0, a) = q_0 \xrightarrow{b} \delta(q_0, b) = q_1 \xrightarrow{b} \delta(q_1, b) = q_1$$

$b a$

$$q_0 \xrightarrow{b} \delta(q_0, b) = q_1 \xrightarrow{a} \delta(q_1, a) \text{ indéfini } \begin{matrix} (\text{échec de}) \\ \text{lecture} \end{matrix}$$

outil: Automatoy.

Extension de la fonction de transition.

f de transition:

$$\hat{\delta}: Q \times X^* \rightarrow Q$$

$$\hat{\delta}(q, \varepsilon) = q$$

$$\forall x \in X, \forall w \in X^*, \hat{\delta}(q, w \cdot x) = \delta(\hat{\delta}(q, w), x)$$

$\rightarrow \hat{\delta}(q, w)$ désigne état atteint en lisant w à partir de q .

$\leadsto qd \ q' = \hat{\delta}(q, w)$, moté $q \xrightarrow{w} q'$

@ A_1 : $\hat{\delta}(q_0, abb) = q_1$, $\hat{\delta}(q_0, ba)$ indéfini

Lang. reconnu P un automate déterministe:

$$A = (X, Q, \delta, q_{ini}, F)$$

$L(A) = \{w \in X^* \mid \hat{\delta}(q_{ini}, w) \in F\}$

Rq: $\varepsilon \in L(A) \Leftrightarrow q_{ini} \in F$

Automates déterministe Complet

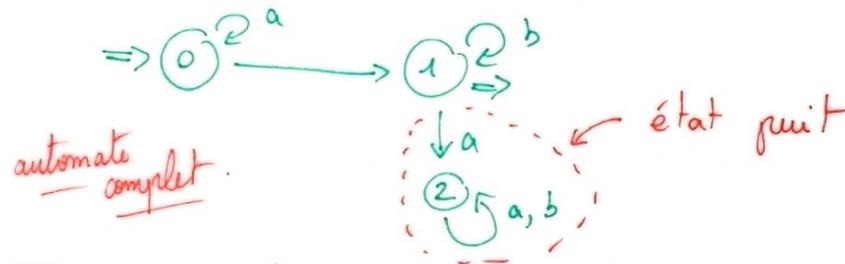
④ Un (ad) A est complet si f est définie
 $\forall (q, x) \in E \quad q \times X$.

@ A₁ n'est pas complet, A₂ est complet.

Δ A₁ n'est pas incorrect.

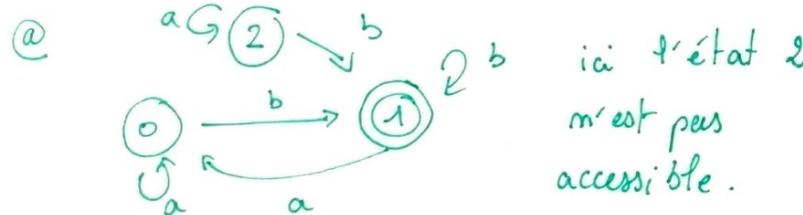
Pté: Tt lang. reconnu p un automate pt ë reconnu p automate complet.

Si un (ad) A n'est pas complet, \exists algo simple \Rightarrow construit complet q reconnaît m langage que A.



Etat accessible :

④ Un état q est accessible si \exists un mot w
 $t_q \hat{f}(q_{ini}, w) = q$



Automate accessible

Un @ est accessible si tous ses états sont accessibles.
 ↳ Tt lang. reconnu p un @ pt ë reconnu p un @ accessible.

Etat co-accessible

Un état q est co-accessible si $\exists \hat{f}(q, w) \in \hat{f}$.
 (ie arrive de l'état acceptant).

Automate co-accessible

Un @ est c-a si tous ses états sont c-a.
 ↳ Tt lang. reconnu p @ pt ë n. p @ c-a.

Automate émondé

Un @ est émondé s'il est ACCESSIBLE ET CO-ACCESSIBLE.
 ↳ Tt lang. reconnu p @ pt ë n. p @ émondé.

Automates finis déterministes

- Un automate fini déterministe (AFD) est un moyen de définir un langage : chq automate déterm. implique un langage reconnu.
- Tl lang. pt-t-il être reconnu p automate ? Non.
- on appelle REC : ens lang. reconnaissables par AFD.
- lien dp lang. réguliers ?
 - lang. rég : gis REC p AFD ?
 - lang. rec p AFD gis rég ?

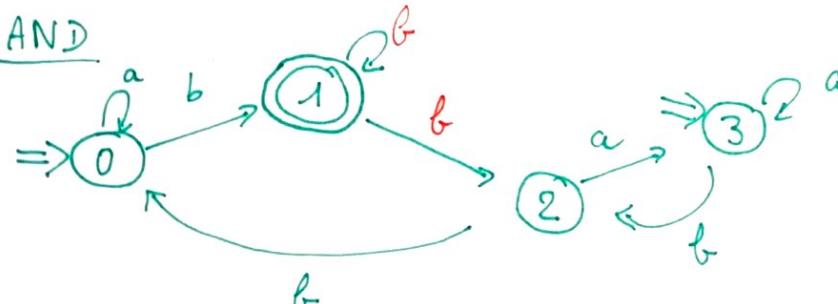
Déterminisme

- Une m^e situat produit gis le m^e effet.
- @ d^e AD : → un seul point de départ ds atm.
- p un état/une lettr : un seul état d'arrivée.

Non-déterminisme

- DS m^e situat, on pt avoir **choix** à faire :
 - il pt y avoir un choix \Leftrightarrow plus points de départ.
 - p état & lettre, il pt y avoir un choix \Leftrightarrow plus états arrivée ($\stackrel{\text{de m}^e}{\text{lettre}}$)

@ AND



④ (AFND)

- un alphabet X
- un ens fini Q : ens états
- ens non vides états initiaux : $I_{ini} \in 2^Q$, $I_{ini} \neq \emptyset$.
- un ss-ens $F \subseteq Q$ d'états dits **acceptants** (ou **finaux** ou **terminaux**)
- fonction f : $Q \times X \rightarrow 2^Q$ appelée f de transis.

$$Q = \{0, 1, 2, 3\}, I_{ini} = \{0, 3\}$$

$$F = \{1\}$$

f	a	b
0	{0}	{1}
1	\emptyset	{1, 2}
2	{3}	{0}
3	{3}	{2}

ens états

Extension de f transit

La f de transis pt être : $\hat{f} : Q \times X^* \rightarrow 2^Q$

$$(q, \varepsilon) \rightarrow \{q\}$$

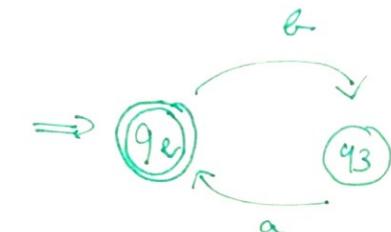
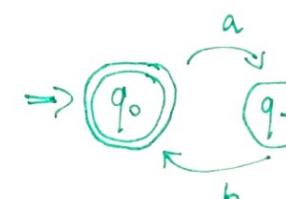
$$(q, w, x) \rightarrow \bigcup_{q' \in \hat{f}(q, w)} \{q' ; x\} \quad (x \in X, w \in X^*)$$

Langage Reconnu

Le lang. reconnu par AND, $A = (X, Q, f, I_{ini}, F)$ est défini par $L(A) = \{w \in X^* \mid \exists q_{ini} \in I_{ini}, \hat{f}(q_{ini}, w) \cap F \neq \emptyset\}$

→ Parfois automates non-déterministes plus faciles.

$$@ (ab)^* + (ba)^*$$



Equivivalence entre AFD & ANFD

Comparaison des outils :

- un and pt il reconnaît lang reconnus p ad ?
- ad ?
- lang nég ? reconn p ad ? par md ?

- REC : ens lang pouvant être reconnus p automates dét.
- REC_{ND} : ens lang atm Non-déf.

$$REC \subseteq REC_{ND}$$

- Pn H atm (AFD) A, \exists un atm (ANFD) And q reconnaît lang.

And défini:

- $Q_{And} = Q_A$ ► $\delta_{And}(q, x) = \{\delta_A(q, x)\}$ si $\delta_A(q, x)$ défini
- $I_{ini\ And} = I_{ini\ A}$ ► $\delta_{And}(q, x) = \emptyset$ si $\delta_A(q, x)$ indéfini
- $F_{And} = F_A$ $\oplus \delta_A(q, x)$.

$$REC_{ND} \subseteq REC$$

- Pn H atm (ANFD) A, \exists un atm (AFD) Ad q reconnaît lang.

Automate déterministe équivalent.

- Pn un automate (AND) A, un atm (AD) équivalent Ad pt à déf:

► $Q_{Ad} = 2^Q_A$ ► $\forall q \in 2^Q_A, \forall x \in X, \delta_{Ad}(q, x) = \bigcup_{e \in q} \delta_A(e, x)$

► $q_{ini} = I_{ini\ A}$

► $F_{Ad} = \{q \in Q_{Ad}, q \cap F_A \neq \emptyset\}$

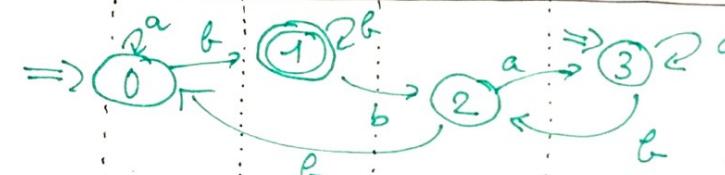
$$REC = REC_{ND}$$

Algo de déterminisation:

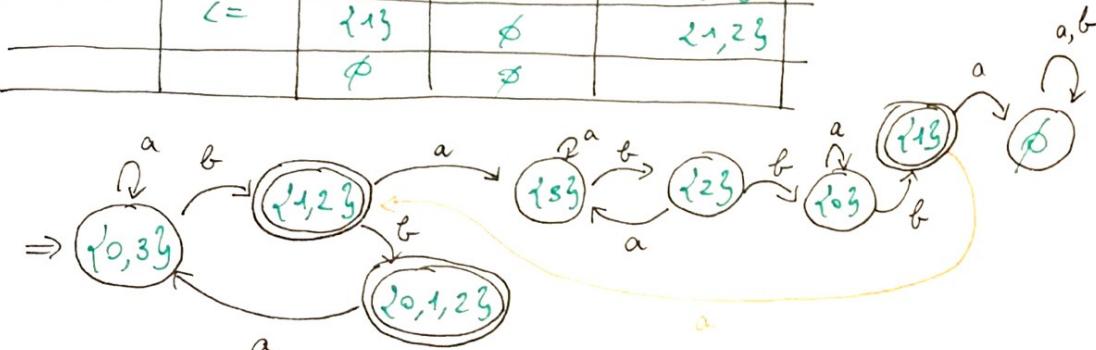
- calculer chq atm (AND) : un (AD) \Leftrightarrow
- algo fournit (AD) accessible & complet.
- A m états \rightarrow au plus posséder 2^m états.

• Algo de déterminisation

init?	accept	state	Transit table
			a b
\Rightarrow	$\{0,3\}$	$\{0,3\}$	$\{1,2,3\}$
	$\{1,2\}$	$\{3\}$	$\{0,1,2,3\}$
	$\{3\}$	$\{3\}$	$\{2\}$
	$\{0,1,2\}$	$\{0,3\}$	$\{0,1,2,3\}$



	$\{2,3\}$	$\{3\}$	$\{0\}$
\Leftarrow	$\{0,3\}$	$\{0\}$	$\{1,2\}$
	\emptyset	\emptyset	$\{1,2,3\}$
	\emptyset	\emptyset	\emptyset



Calcul de déterminisation

Tous les ens q contiennent un état acceptant et des états acceptants.

algo de déterminisation

$$q_{\text{ini}} \leftarrow \text{Init}_A$$

$$Q_{Ad} \leftarrow \{\text{Init}_A\}$$

while $\exists (q, x) \in Q_{Ad} \times X$, $f_{Ad}(q, x)$ is undefined do

$(q, x) \leftarrow$ one of such pair

$$q' \leftarrow \bigcup_{e \in q} f_A(e, x)$$

$$Q_{Ad} \leftarrow Q_{Ad} \cup \{q'\}$$

$$f_{Ad}(q, x) \leftarrow q'.$$

end while

$$\mathcal{F}_{Ad} \leftarrow \{q \in Q_{Ad}, q \cap \mathcal{F}_A \neq \emptyset\}.$$

→ L'atm obtenu est complet & accessible.
(en général, pas minimal)

Stabilité de REC par U, .*

TH (Stabilité ou clôture)

REC est clos par union, concaténation et étoile de Kleene.

② ∀ lang. recon L_1 et L_2 ($\forall L_1, L_2 \in \text{REC}$)

$\Rightarrow L_1 \cup L_2$ est reconnu, $L_1 \cdot L_2$ est reconnu, L_1^* est reconnu.

soit A_1 , un atm L_1 , on suppose posséder un état initial,

$$A_1 = (Q_1, \{\text{Init}_1\}, \mathcal{F}_1, f_1) \text{ et } A_2 = (Q_2, \{\text{Init}_2\}, \mathcal{F}_2, f_2)$$

On va construire un automate non déterministe pr $L_1 \cup L_2, L_1 \cdot L_2, L_1^*$.

$L_1 \cup L_2$ AND

$$\bullet Q_A = Q_1 \cup Q_2 \quad \bullet \text{Init}_A = \{\text{Init}_1, \text{Init}_2\} \quad \bullet \mathcal{F}_A = \mathcal{F}_1 \cup \mathcal{F}_2$$

$$\bullet f_A(q, x) = \begin{cases} f_1(q, x) & \forall q \in Q_1 \\ f_2(q, x) & \forall q \in Q_2 \end{cases} \quad \text{reconnait le lang } L_1 \cup L_2.$$

$L_1 \cdot L_2$ AND

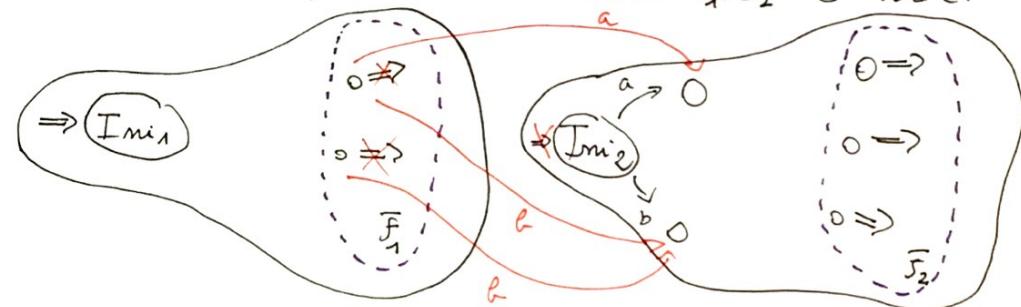
$$\bullet Q_A = Q_1 \cup Q_2.$$

$$\bullet \text{Init}_A = \{\text{Init}_1\} \quad \bullet \mathcal{F}_A = \mathcal{F}_2 \text{ si } \varepsilon \notin L_2$$

$$\text{ou } \mathcal{F}_A = \mathcal{F}_2 \cup \mathcal{F}_1 \text{ si } \varepsilon \in L_2.$$

$$\bullet f_A(q, x) = \begin{cases} f_1(q, x) & \forall q \in Q_1 - \mathcal{F}_1 \\ f_1(q, x) \cup f_2(\text{Init}_2, x) & \forall q \in \mathcal{F}_1 \\ f_2(q, x) & \forall q \in Q_2. \end{cases}$$

reconnait le lang $L_1 \cdot L_2$. Done $L_1 \cdot L_2 \in \text{REC}$.

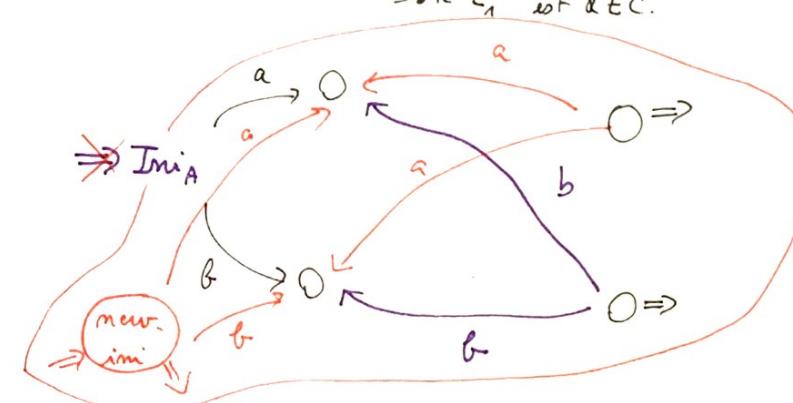


L_1^* AND

$$\bullet Q_A = Q_1 \cup \{\text{new_ini}\}, \quad \text{Init}_A = \{\text{new_ini}\}$$

$$\bullet \mathcal{F}_A = \mathcal{F}_1 \cup \{\text{new_ini}\} \quad \bullet f_A(q, x) = \begin{cases} f_1(q, x) & \forall q \in Q_1 - \mathcal{F}_1 \\ f_1(q, x) \cup f_1(\text{new_ini}, x) & \forall q \in \mathcal{F}_1 \\ f_1(\text{new_ini}, x) & \text{si } q = \text{new_ini} \end{cases}$$

Done L_1^* est à EC.



TH RAT \subseteq REC.

Tout lang rationnel est reconnaissable.

- REC contient \emptyset , $\{\epsilon\}$, $\{x\}$, ($\forall x \in X$)
- REC est close par union, concaténation, étoile de Kleene.

Une équation à une inconnue L.

si A et B sont 2 lang. fixes ; pt-m trouer un lang L vérifiant : $L = AL \cup B$.

Lemme d'Arden

si A et B sont 2 lang. et si $\epsilon \notin A$ alors $L = A^*B$ est le seul lang vérifiant $L = AL \cup B$.

Coro:

si A et B sont 2 lang. régulières et si $\epsilon \notin A$, $A^* \cdot B$, unique solut de $L = AL \cup B$ est aussi un lang. rég.

- Pptis :
- ① si $L = A \cup B$ alors $B \subseteq L$
 - ② si $L = A \cup B$ alors $AL \subseteq L$
 - ③ si $\epsilon \notin A \Rightarrow \epsilon \notin AL$.

On souhaite montrer

- (i) $A^*B \subseteq L$
- (ii) $L \subseteq A^*B$.

(i) P(m): $A^m B \subseteq L$, $m > 0$.

⊗ m=0. $A^0 B = B \subseteq L \Rightarrow P(0)$ vraie

⊗ sup. P(m) vraie pour un certain m.

$$A^{m+1} \cdot B = A \cdot A^m B \text{ ou } A^m B \subseteq L$$

$$A \cdot A^m B \subseteq A \cdot L \subseteq L \Rightarrow A^{m+1} \cdot B \subseteq L$$

$\Rightarrow P(m+1)$ vraie

(P_n) vraie $\forall m > 0$, $A^m B \subseteq L \forall m > 0 \Rightarrow A^*B \subseteq L$.

(ii) P'(m) si w mot, $|w| \leq m$, si $w \in L$ alors $w \in A^*B$, $m > 0$.

⊗ m=0, $|w| \leq 0 \Rightarrow w = \epsilon$

$w \in L$ et $\epsilon \in L$ où $B \subseteq A^* \cdot \emptyset$
 $L = AL \cup B$ $\epsilon \notin A \cdot L$ dc $\epsilon \in B$.

$\Rightarrow w = \epsilon \in A^*B$ dc P'(0) vraie.

⊗ suppos P'(m) vraie pr un certain m,

soit w tq $|w| \leq m+1$, $w \in L = AL \cup B$.

► cas 1, $w \in B$, $w \in A^*B$ ok

► cas 2, $w \notin B$, $w \in AL$,

$\exists u \in A$, $\exists v \in AL$, $w = u \cdot v$, $|u| > 1$

$|v| \leq m$, $v \in AL \Rightarrow v \in L$.

⊗ s'appliq à v, $v \in A^*B$, $w = u \cdot v$, $v \in A^*B$.
 $u, v \in AA^*B \subseteq A^*B$

$w \in A^*B$, $P'(m+1)$ est vraie.

P'(m) vraie $\forall m$,

$\forall w \in L$, $w \in A^*B$.

$\rightarrow A^*B \subseteq L \subseteq A^*B$.

Donc $L = A^*B$.

Lemme d'Arden démontré.

Généralisation Lemme Arden

2 équations & 2 inconnues

si $A_{1,1}, A_{1,2}, A_{2,1}, B_1, B_2$ sont lang rég et si aucun des $A_{i,j}$ ne contient ϵ , alors le système d'équations à 2 inconnues L_1, L_2 :

$$\begin{cases} L_1 = A_{1,1} \cdot L_1 \cup A_{1,2} \cdot L_2 \cup B_1 \\ L_2 = A_{2,1} \cdot L_1 \cup A_{2,2} \cdot L_2 \cup B_2 \end{cases}$$

admet une solut uniq & les lang L_1 et L_2 st réguliers.

Preuve $L_1 = A_{1,1} L_1 \cup A_{1,2} L_2 \cup B_1$.

$$L_1 = A_{1,1}^* (A_{1,2} \cdot L_2 \cup B_1) \quad (\text{comme Arden})$$

$$L_2 = A_{2,1} \cdot A_{1,1}^* (A_{1,2} \cdot L_2 \cup B_1) \cup A_{2,2} L_2 \cup B_2$$

$$L_2 = (A_{2,1} \cdot A_{1,1}^* A_{1,2} \cup A_{2,2}) \cdot L_2 \cup A_{2,1} \cdot A_{1,1}^* B_1 \cup B_2$$

$$L_2 = (A_{2,1} \cdot A_{1,1}^* A_{1,2} \cup A_{2,2})^* (A_{2,1} A_{1,1}^* B_1 \cup B_2) \quad (\text{dm})$$

n équations & n inconnues

Le lemme d'Arden se généralise.

Si A_{ij} et B_i st réguliers et si aucun des A_{ij} ne contient pas ϵ , alors les L_i forment une solut uniq & st rég.

D (Langage L_q):

soit atm déterministe $\mathcal{A} = (X, Q, q_{ini}, F, \delta)$ et $q \in Q$.

$$L_q = \{w \in X^* \mid \hat{\delta}(q, w) \in F\}$$

L_q est l'ens des mots qui lus en partant de q atteignent un état acceptant.

Prop (pour AD): $L(A) = L_{q_{ini}}$.

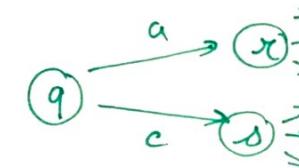
Équation équivalente à un atm, principe

@ $X = \{a, b, c\}$. états q, r, s tq q **ACCEPTANTS**
 $\delta(q, a) = r$, $\delta(q, c) = s$, $\delta(q, b)$ indéfini.

Quels mots atteignent un état acceptant en partant de q ?

- mot vide : ici **non**
- comme b : **non**
- un a suivi mot allant à $\approx \circlearrowright$ **oui**.
- un c $\approx \circlearrowright$ **oui**.

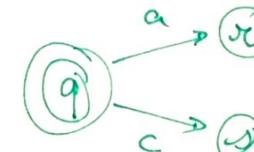
Donc $L_q = \{ab\} \cdot L_r \cup \{cb\} \cdot L_s$,



@ m exemple tq q **ACCEPTANTS**.

- mot vide : **oui**
- cb : **non**
- : **oui**
- : **oui**

$$L_q = \{ab\} \cdot L_r \cup \{cb\} \cdot L_s \cup \{\epsilon\}$$



- On procède de manière pour chaque état automatique.
- une équation pour chaque L_q .
 - parmi les états, on a m équations à ..
 - les facteurs de L_q sont singuliers $\{ \cdot \cdot , n \in X \}$.
(langages)
 - Lemme d'Arden s'applique, il y a sol° unique pour les L_q .
 - les L_q sont réguliers.
 - $L(A) = L_{qini}$ est régulier.

$\text{REC} \subseteq \text{RAT}$:

Pr un automate dit A :

- chaque L_q admet sol° unique qui est lang rationnel.
- $L(A) = L_{qini}$ est un lang régulier.

Donc $\text{REC} \subseteq \text{RAT}$.

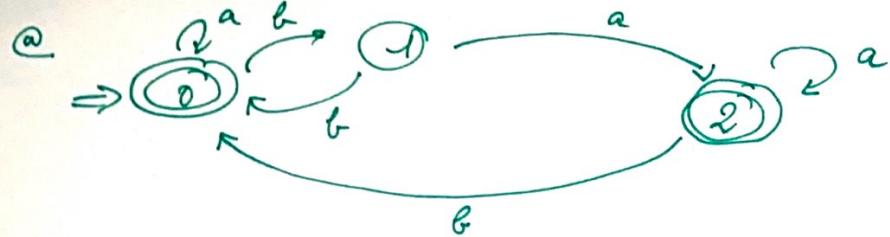
Tu Kleene

RAT = REC

Calcul de l'expression par automate

Equations expressions régulières

- les lang L_q sont réguliers.
- on établit une équation entre expressions régulières.
- ④ $L_q = a \cdot L_1 + b \cdot L_2 + \epsilon$.
- au lieu de $L_q = \{a\}^* \cdot L_1 \cup \{b\}^* \cdot L_2 \cup \{ \epsilon \} \cdot$
- pour calculer expressions régulières, on peut :
- de l'équation $L_1 = \dots$, éliminer L_1 sur la droite de Arden si bonnes
- ...
- on obtient expression pour L_m .
- puis pour chacun des L_i .



$$(0) \quad L_0 = aL_0 + bL_1 + \varepsilon$$

$$(1) \quad L_1 = aL_2 + bL_0$$

$$L_2 = aL_2 + bL_0 + \varepsilon$$

Exemple de Résolution :

$$(0') \quad L_0 = a^*(bL_1 + \varepsilon) \quad \text{Arden}$$

$$L_1 = b.a^*(bL_1 + \varepsilon) + aL_2 \quad (1) \text{ et } (0')$$

$$L_1 = b.a^*b.L_1 + ba^* + aL_2$$

$$(1') \quad L_1 = (ba^*b)^* (ba^* + aL_2) \quad \text{Arden}$$

$$L_2 = L_1 + \varepsilon \quad (1) \text{ et } (2)$$

$$L_2 = (ba^*b)^* (ba^* + aL_2) + \varepsilon$$

$$L_2 = (ba^*b)^* aL_2 + (ba^*b)^* ba^* + \varepsilon.$$

$$(2') \quad L_2 = ((ba^*b)^* a)^* ((ba^*b)^* ba^* + \varepsilon) \quad \text{Arden}$$

$$L_1 = \dots$$

$$L_0 = \dots$$

Autre méthode :

$$L_2 = L_1 + \varepsilon$$

$$L_1 = b.L_0 + a(L_1 + \varepsilon)$$

$$L_1 = aL_1 + bL_0 + a$$

$$(1) \quad L_1 = a^* (bL_0 + a)$$

$$L_0 = aL_0 + ba^*(bL_0 + a) + \varepsilon$$

$$L_0 = (a + ba^*b)L_0 + ba^*a + \varepsilon$$

$$L_0 = (a + ba^*b)^* (ba^*a + \varepsilon).$$

Formes Normales

- **littéral**: tte formule de la forme x ou $\neg x$: variable propositionnelle
- **formule conjonctive** $\varphi_1 \vee \dots \vee \varphi_m$
- **formule disjonctive** $\varphi_1 \wedge \dots \wedge \varphi_m$

- **Forme Normal Disjonctive**:
 ↳ disjunc de conjoncts littéraux $\bigvee_{i=1}^p \bigwedge_{j=1}^{m_i} I_{i,j}$
 $\hookrightarrow (\bigwedge_{i=1}^p \bigwedge_{j=1}^{m_i} I_{i,j}) \vee \dots \vee (\bigwedge_{p=1}^P \bigwedge_{j=1}^{m_p} I_{p,j})$.

(FND) rendre explicite des syntaxes.

$$[v, x] = \bigwedge_{i | v(x_i)=1} x_i \wedge \bigwedge_{i | v(x_i)=0} \neg x_i$$

m est satisfaite que par v . (vaut pr 1 seule valud.)

④ $v = [x_1=0, x_2=1, x_3=0, x_4=1]$

$$\bigwedge_{i | v(x_i)=1} x_i \wedge \bigwedge_{i | v(x_i)=0} \neg x_i$$

$$\{i | v(x_i)=1\} = \{2, 4\}$$

$$\{i | v(x_i)=0\} = \{1, 3\}$$

$$\bigwedge_{i \in \{2, 4\}} x_i \wedge \bigwedge_{i \in \{1, 3\}} \neg x_i$$

$$x_2 \wedge x_4 \wedge \neg x_1 \wedge \neg x_3$$

Si une formule φ sur les variables X est satisfaite exactement par les valeurs v_1, \dots, v_k alors:

$$\varphi \equiv \bigvee_{i=1}^k [v_i, x] = [v_1, x] \vee \dots \vee [v_k, x]$$

⑤ on pose $\varphi = a \wedge (b \vee \neg c)$

a	b	c	$b \vee \neg c$	$a \wedge (b \vee \neg c)$
0	0	0	1	0
0	0	1	0	0
0	1	0	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

$$\begin{aligned} & \rightarrow a \wedge \neg b \wedge \neg c \\ & \rightarrow a \wedge b \wedge \neg c \\ & \rightarrow a \wedge b \wedge c \end{aligned}$$

ainsi $a \wedge (b \vee \neg c) \equiv (a \wedge \neg b \wedge \neg c) \vee (a \wedge b \wedge \neg c) \vee (a \wedge b \wedge c)$.

Avec la sémantique, pour tte formule, on peut construire une autre formule équivalente.

Mise ss (FND) ap ascimatisat en 3 phases :

① Définie \Rightarrow, \Leftarrow :

$$\varphi \Rightarrow \psi \equiv \neg \varphi \vee \psi, \quad \varphi \Leftarrow \psi \equiv (\neg \varphi \vee \psi) \wedge (\varphi \vee \neg \psi)$$

② Pousser la négation au niveau variables:

$$\neg \neg \varphi \equiv \varphi, \quad \neg(\varphi \vee \psi) \equiv \neg \varphi \wedge \neg \psi, \quad \neg(\varphi \wedge \psi) \equiv \neg \varphi \vee \neg \psi$$

③ Distributivité \wedge par rapport \vee :

$$(\varphi \wedge \psi) \wedge \theta \equiv (\varphi \wedge \theta) \vee (\psi \wedge \theta)$$

@ de mise sous FND :

$$\begin{aligned}& \neg(a \Rightarrow b) \vee \neg(c \vee \neg(d \Rightarrow a)) \\& \neg(\neg a \vee b) \vee \neg(c \vee \neg(\neg d \vee a)) \\& (\neg \neg a \wedge \neg b) \vee \neg(c \vee \neg(\neg d \vee a)) \\& (a \wedge \neg b) \vee \neg(c \vee \neg(\neg d \vee a)) \\& (a \wedge \neg b) \vee (\neg c \wedge \neg \neg(\neg d \vee a)) \\& (a \wedge \neg b) \vee (\neg c \wedge (\neg d \vee a)) \\& (a \wedge \neg b) \vee ((\neg c \wedge \neg d) \vee (\neg c \wedge a)) \\& (a \wedge \neg b) \vee (\neg c \wedge \neg d) \vee (\neg c \wedge a)\end{aligned}$$

→ Peut donner une formule exponentielle + grande.

Forme Normale Conjuctive : FNC.

$$\bigwedge_{i=1}^p \bigvee_{j=1}^{m_i} I_{i,j}$$

$$(I_{1,1} \vee \dots \vee I_{1,m_1}) \wedge \dots \wedge (I_{p,1} \vee \dots \vee I_{p,m_p})$$

Ens de contraintes à satisfaire.

C₃: Satisfiabilité

I / Pb de SAT

Une formule φ ds Prop(x), savoir si v q satisfait φ ,

$$\text{i.e. } \llbracket \varphi, v \rrbracket = 1.$$

→ pb diff à résoudre : Théorie de Complexité

→ on ne connaît pas d'algo efficace.

→ Enumérer toutes les valeurs possibles.

NP-complétude de la SAT

Le problème est NP-complet.

• NP : on peut le résoudre en temps polynomial par algo ND.

• complet : Il est NP, peut être codé en temps polynomial déterministe ds ce pb.

Algo ND (non déterministes) :

choisir entre :

code 1

et:

code 2

↳ choisit, exécute le code 1 ou le code 2 sans se tromper pour trouver la solution du pb.

↳ Abstrait.

► Etats limitrophes n'ont pas même couleur :

$$\varphi_3 = \bigwedge_{\{e_1, e_2\} \in L, c \in C} \neg [e_1, c] \vee \neg [e_2, c]. \quad \textcircled{3}$$

• Algo D : 2^m choix possibles.

• Algo ND : m choix possibles.

• Algo NP : on ne sait pas, on admet que l'on ne peut pas faire ça : q^o siècle \$ 1M.

Le pb SAT :

SAT : pb q consiste à résoudre formule ss FNC.

$$(I_{1,1} \vee \dots \vee I_{1,m_1}) \wedge \dots \wedge (I_{p,1} \vee \dots \vee I_{p,m_p})$$

une clause

où I_{ij} : littéraux.

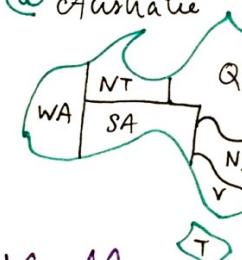
II / Programmer le SAT

→ Il pb a une soluo NP.

→ SAT solving très efficace (certaines conjectures mathématiques)

→ SAT : ens de littéraux, satisfaire ens de clauses
↳ gérer la mémoire.

@ Australie



4 couleurs (toutes cartes dimontrées).

$$C = \{r, v, b\}$$

$$E = \{WA, NT, Q, S, NS, V, T\}$$

L = {{WA, NT}, {WA, SA}, ...} 33 paires états limitrophes.

Variables propositionnelles sont : $[e, c]$

$$[WA, r], [WA, v], [WA, b], \dots$$

Contraintes : ► Chq état a au moins 1 couleur.

$$\varphi_1 = \bigwedge_{e \in E} \bigvee_{c \in C} [e, c] = ([WA, r] \vee [WA, v] \vee [WA, b]) \wedge (\dots)$$

► Aucun état a 2 couleurs (chq état a au plus une couleur)

$$\varphi_2 = \bigwedge_{e \in E, c_1, c_2 \in C | c_1 \neq c_2} \neg [e, c_1] \vee \neg [e, c_2]$$

états = ["WA", "NS", ...] couleurs = ["r", "v", "b"]
 limitrophes = [("WA", "NS"), (" ", " ")].

phi1 = [[encode(e, c) for c in couleurs] for e in états]

phi2 = [[-encode(e, c1), -encode(e, c2)] for e in états]

for c1 in couleurs

for c2 in couleurs

if c1 != c2]

phi3 = [[-encode(e1, c), -encode(e2, c)] for (e1, e2) in voisins for c in couleurs]

minijat2, pip python sat

Contraintes de comptage

Exprimer qu'au moins k var. prop. d'un ass-ens de var

$X = \{x_1, \dots, x_m\}$ st vraies.

- $k=1$, clause $x_1 \vee \dots \vee x_k$
- $k=2$ (\wedge) $x_1 \vee \dots \wedge x_k$

L) X , m élts, si $Y \subseteq X$ alors

- Y contient au moins la élts.
- $\forall Z \subseteq X$ q contient $m-k+1$ élts intersecte Y .

$$\bigwedge_{1 \leq i_1 < \dots < i_{m-k+1} \leq n} \bigvee_{1 \leq j \leq m-k+1} x_{i_j}$$

$n-k$
 parmi élts: au moins 1 de vraie

Comptage au plus: \wedge $\bigvee x_{i_j}$

@ mieux colonie Australie

↳ que chq couleur soit utilisée au moins 2 fois.

o { [e, c] | e $\in E$ }

$\rightarrow 7$ états, $7-2+1=6$.

$$\bigwedge_{E' \subseteq E \text{ et } |E'|=6} \bigvee_{e \in E'} [e, c]$$

phi4 = [[encode(ee, c) for e in états if ee != e
 for c in couleurs for e in états]]

Format DIMACS \rightarrow spécifia pb

- Pour sat-solveur: variables entiers positifs
- le littéral $\neg x_k$: $-k$
- SAT-solveur: $I = [c_1, \dots, c_n]$
 \leftarrow clause

@ [1, -2, 3] représente la clause $x_1 \vee \neg x_2 \vee x_3$

P cnf 5 (nbt
 variables) 2 (nbt clauses)

1	-2	0	\leftarrow clauses
2	5	-1	0