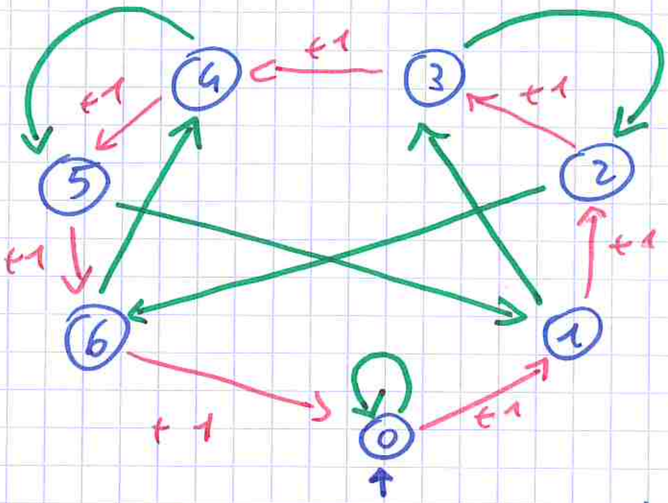
Critère de divisibilité - Automate & Exp. Rat.

ORIGINE

blogdemaths.fr donne le critère humain suivant de % par 7



Très joli car symétrie présente !

$\xrightarrow{+1}$: pour se balader autant que le chiffre en cours (lect. de gauche à droite)

$\rightarrow$: changement de rang (centaine → dizaine → unité par ex.)

On utilise $10 \equiv 3\ [7]$ ou $10 \equiv -4\ [7]$

1438 est-il divisible par 7 ? On va avoir mieux car on aura

$r \in [\![0;6]\!]$ tq $1438 \equiv r\ [7]$.

$$\underbrace{0 \xrightarrow[+1]{} 1 \rightarrow 3}_{1}\ \underbrace{\xrightarrow[+4]{} 0 \rightarrow 0}_{4}\ \underbrace{\xrightarrow[+3]{} 3 \rightarrow 2}_{3}\ \underbrace{\xrightarrow[+8]{} 3}_{8}\ \text{STOP!}$$

Donc $1438 \equiv 3\ [7]$. En fait $1438 = 7 \times 205 + 3$

Pourquoi-comment ça marche ?

$$1438 = \underbrace{(\underbrace{(\underbrace{\overset{x_0}{1} \times 10 + 4}_{x_1}) \times 10 + 3}_{x_2}) \times 10 + 8}_{1438 = x_3}$$

Flèche rouge $\xrightarrow{+1}$ donne $r_0 \in [\![0;6]\!]$ tq $x_0 \equiv r_0\ [7]$

——— verte $\rightarrow$ ——— r_{01} ——— $10x_0 \equiv r_{01}\ [7]$

Flèche rouge $\xrightarrow[+4]{}$ donne $x_1 \in [\![0;6]\!]$ tq $x_1 \equiv x_1$ [7]

En répétant ce m̂ type de raisonnement, on a un procédé général de calcul du reste modulo 7 !

On note au passage que l'on peut faire de m̂ modulo $k \in \mathbb{N} - \{0,1\}$.

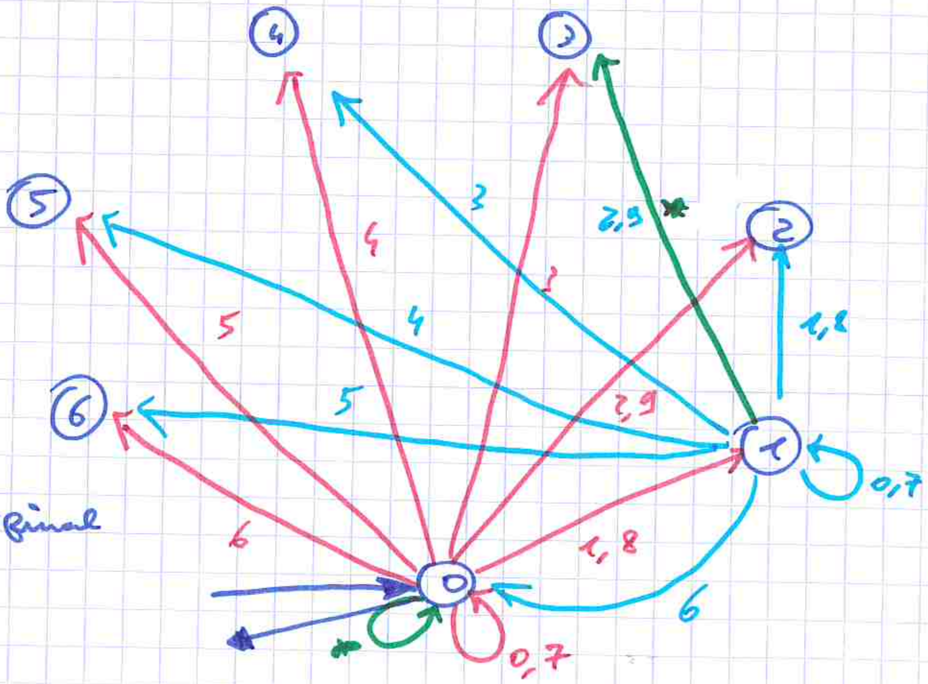
AUTOMATE

On passe à un automate mais si on met juste n en entrée, cela n'en n'est pas un car un coup on choisit une flèche rouge, puis si besoin une flèche verte.

On pourrait utiliser un automate à pile pour stocker un nouveau chiffre analysé et essayer de mimer ce qui a été fait. Voir si c'est ok ! ?!

On va essayer d'avoir $\overbrace{\text{un}}^{\text{tout de m̂}}$ auto. fini, ce qui nous permettra de coder le critère en utilisant une exp. rat.

* Idée 1 (moche) : on change l'entrée 1438 en 1*4*3*8 où chaque * permet de savoir que l'on change de rang. Comme avant, on trouve l'auto. suivant incomplet (seuls les états 0 et 1 ont toutes leurs flèches sortantes !)

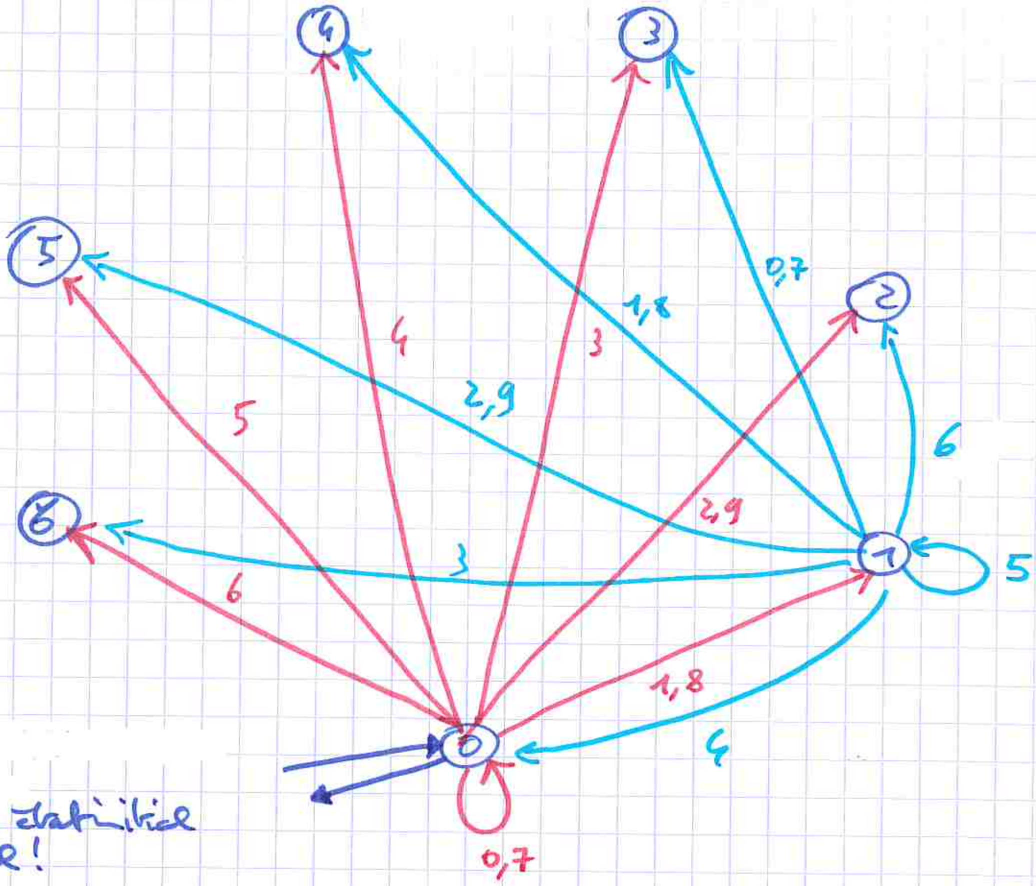


⚠ 0 état initial et final unique !

On "code" tout d'un coup !

* Idée 2 (voir "Lang. formels..." de O. CARTON chez Vuibert, page 173) : l'idée est simple. On pose $p \xrightarrow{d} q$ si $10p + d \equiv q\ [7]$ où $(p,d) \in [\![0;9]\!]^2$ et $q \in [\![0;6]\!]$

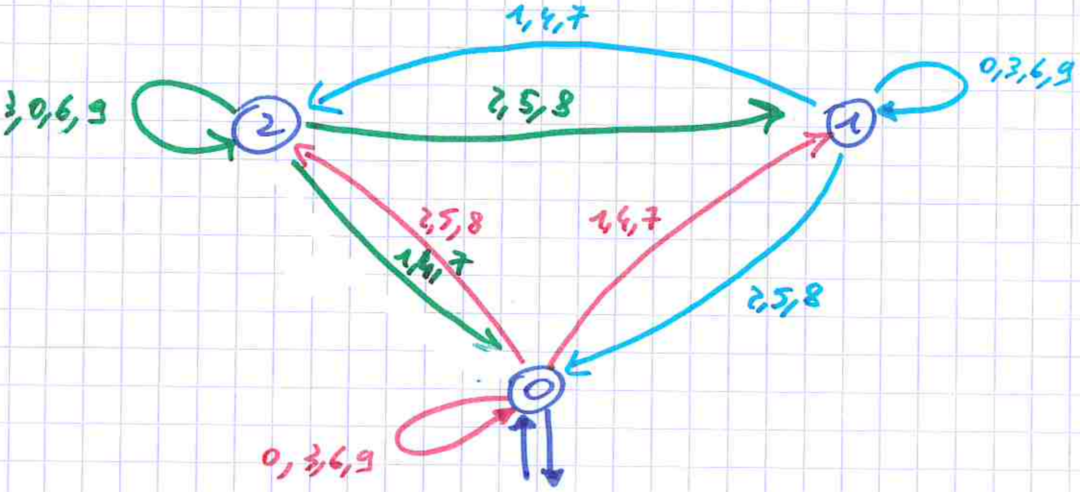
Voici le début de l'auto. (seuls les états 0 et 1 ont ttes leurs flèches sortantes !)



⚠ 0 unique état initial et final !

Exp. RAT.

Qui dit auto. fini, dit exp. rat. Trouver à la main celle pour critère de % par 3 (pour une taille raisonnable d'auto.), et ne gardons que l'idée 2 qui ne modifie rien en entrée !



⚠ $10 \equiv 1\ [3]$ crée une grosse sym.

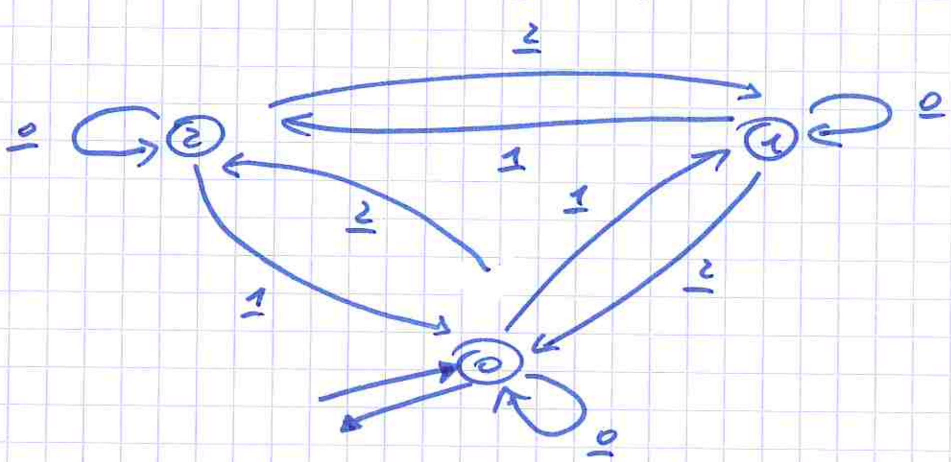
⇓

Faire avec 4 après !

On va utiliser un algo. donnant exp. rat. facilement à la main ! C'est l'algo.

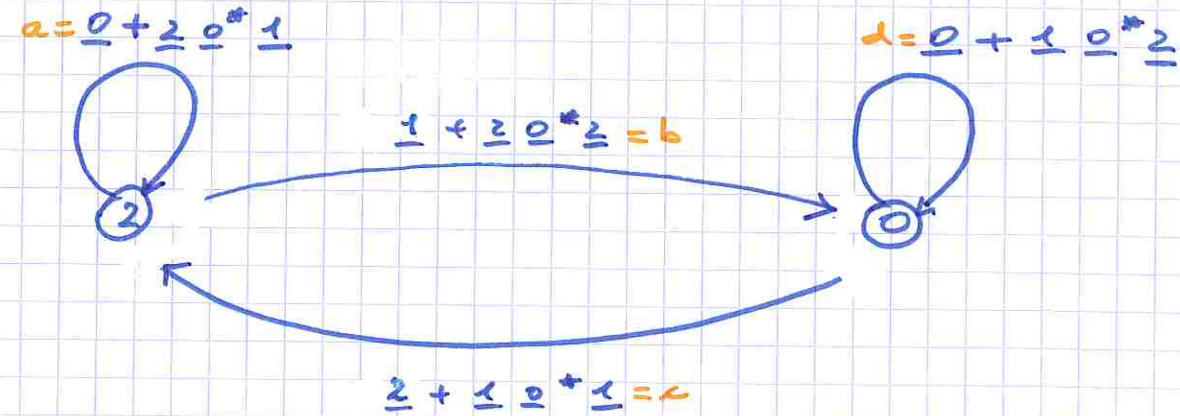
ce n'est pas l'addition mais le "OU" des exp. rat.

Pour simplifier, on note $\underline{0} = 0+3+6+9$ (alternatives), $\underline{1} = 1+4+7$ et $\underline{2} = 2+5+8$. L'automate devient :



On applique l'algo. d'élimination présenté page 39 de "Lang. formels..." d'O. CARTON chez Vuibert. ⚠ On part d'un auto. normalisé, càd avec un seul état initial et un seul final. C'est le cas ici.

On commence par éliminer l'état ① :



On élimine l'état ② (pas besoin de dessiner l'auto !) :

$$(d + c\,a^*\,b)^* = \Big(\underline{0} + \underline{1}\,\underline{0}^*\,\underline{2} + (\underline{2} + \underline{1}\,\underline{0}^*\,\underline{1}) \cdot (\underline{0} + \underline{2}\,\underline{0}^*\,\underline{1})^* \cdot (\underline{1} + \underline{2}\,\underline{0}^*\,\underline{2})\Big)^*$$

Il ne reste plus qu'à vérifier que cela fonctionne informatiquement (exemple sur les regexp).

Par sym. des rôles de 2 et 1, on a aussi :

$$\left(\underline{0} + \underline{2}\,\underline{0}^*\underline{1} + (\underline{1} + \underline{2}\,\underline{0}^*\underline{2})\cdot(\underline{0} + \underline{1}\,\underline{0}^*\underline{2})^* \cdot (\underline{2} + \underline{1}\,\underline{0}^*\underline{1})\right)^*$$

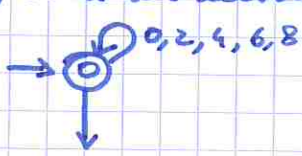
Voyons comment cette exp. rat. fonctionne sur 111, 102, ... etc.

- **102** capté via $\underline{1}\cdot\underline{0}\cdot\underline{2}$
- **111** ——— $\underline{1}\cdot\underline{1}\,\underline{0}^*\underline{1}$

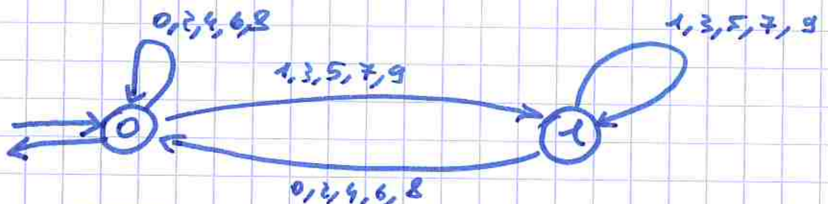
... etc

Il semble difficile de faire mieux.

Que donne notre technique pour divisibilité par 2 ? Si on passe par le mot miroir, on a un automate basique, et de m̂ pour la regex associée, suivant :



Pour une lecture de gauche à dte, cela semble moins immédiat.



Notant $\underline{P} = 0+2+4+6+8$ et $\underline{I} =$ 1+3+5+7+9, on obtient :

$$(\underline{P} + \underline{I}\cdot\underline{I}^*\cdot\underline{P})^* = \underbrace{(\underline{P} + \underline{I}^+\cdot\underline{P})^*}$$

Facile à trouver directt mais moins efficace que $\underline{C}^*\cdot\underline{P}$ où $\underline{C} = 0+1+2+3+4+5+6+7+8+9$