Expressions régulières

Philippe Quéinnec

3 janvier 2011

Plan

Exemple d'utilisation

```
> ls monrepertoire/
memoire.aux memoire.tex picture004.jpg rapsody.jpg
memoire.dvi picture001.jpg presentation.tex raw.jpg
memoire.old picture002.jpg price-list.txt
memoire.log picture003.jpg taches.txt
```

Afficher uniquement les images :

```
ls *.jpg
```

Effacer les fichiers relatifs à `memoire` :

```
rm mem*
```

Effacer les images commençant par `pic`, de 1 à 3 :

```
rm pic*[1-3].jpg
```

Expressions Régulières

Expressions régulières

Soit X un alphabet fini, et $Y = \{(,), *, +, \bullet, \Lambda, \emptyset\}$ un alphabet disjoint. Un mot m de $(X \cup Y)$ est une *expression régulière* sur X ssi :

- soit m est $\emptyset$ ou Λ ou un symbole de X,
- soit m est de la forme $(x + y)$ ou $(x \bullet y)$ ou x^*, où x et y sont des expressions régulières sur X.

Langage associé

Une expression régulière m sur X définit un langage $L(m)$ sur X d'après les règles suivantes :

- $L(\emptyset)$ est le langage vide ;
- $L(\Lambda) = \{\Lambda\}$;
- Si $a \in X$, alors $L(a) = \{a\}$;
- Pour tout expression régulière u et v sur X,

$$\begin{aligned} L(u+v) &= L(u) \cup L(v) \\ L(u \bullet v) &= L(u)L(v) \\ L(u^*) &= L(u)^* \end{aligned}$$

Règles de calcul

$$\begin{array}{rcl|rcl|rcl} \Lambda \bullet e & = & e \bullet \Lambda = e & e^* & = & e^* + \Lambda & \Lambda^* & = & \Lambda \\ \emptyset \bullet e & = & e \bullet \emptyset = \emptyset & e + \emptyset & = & \emptyset + e = e & \emptyset^* & = & \Lambda \end{array}$$

$$\begin{array}{rcl|rcl} (e_1 + e_2) + e_3 & = & e_1 + (e_2 + e_3) & e + e & = & e \\ (e_1 + e_2)e_3 & = & e_1 e_3 + e_2 e_3 & e^* \bullet e^* & = & e^* \\ e_1(e_2 + e_3) & = & e_1 e_2 + e_1 e_3 & (e^*)^* & = & e^* \\ (e_1 \bullet e_2) \bullet e_3 & = & e_1 \bullet (e_2 \bullet e_3) & ee^* & = & e^* e \\ e^* & = & \Lambda + ee^* & ee^* & = & e^* \text{ ssi } \Lambda \in e \\ e & = & e^* \text{ ssi } e = e^2 & & & \end{array}$$

$$\begin{array}{rcl} (e_1^* e_2)^* e_1^* & = & (e_1 + e_2)^* = e_1^*(e_2 e_1^*)^* \\ (e_1^* + e_2^*)^* & = & (e_1 + e_2)^* = (e_1^* e_2^*)^* \end{array}$$

Plan

Théorème d'Arden

Théorème d'Arden

Soient e_1 et e_2 deux expressions régulières. L'équation en $x : x = e_1 x + e_2$ admet $e_1^* e_2$ pour solution. Si $\Lambda \notin e_1$, cette solution est unique.

Tout système de n équations à n inconnues du type $x_i = e_{i,1} x_1 + \cdots + e_{i,n} x_n + e_{i,n+1}$ avec $\Lambda \notin e_{i,j}$ $\forall i, j \leq n$, admet une solution unique.

Expression régulière correspondante à un automate

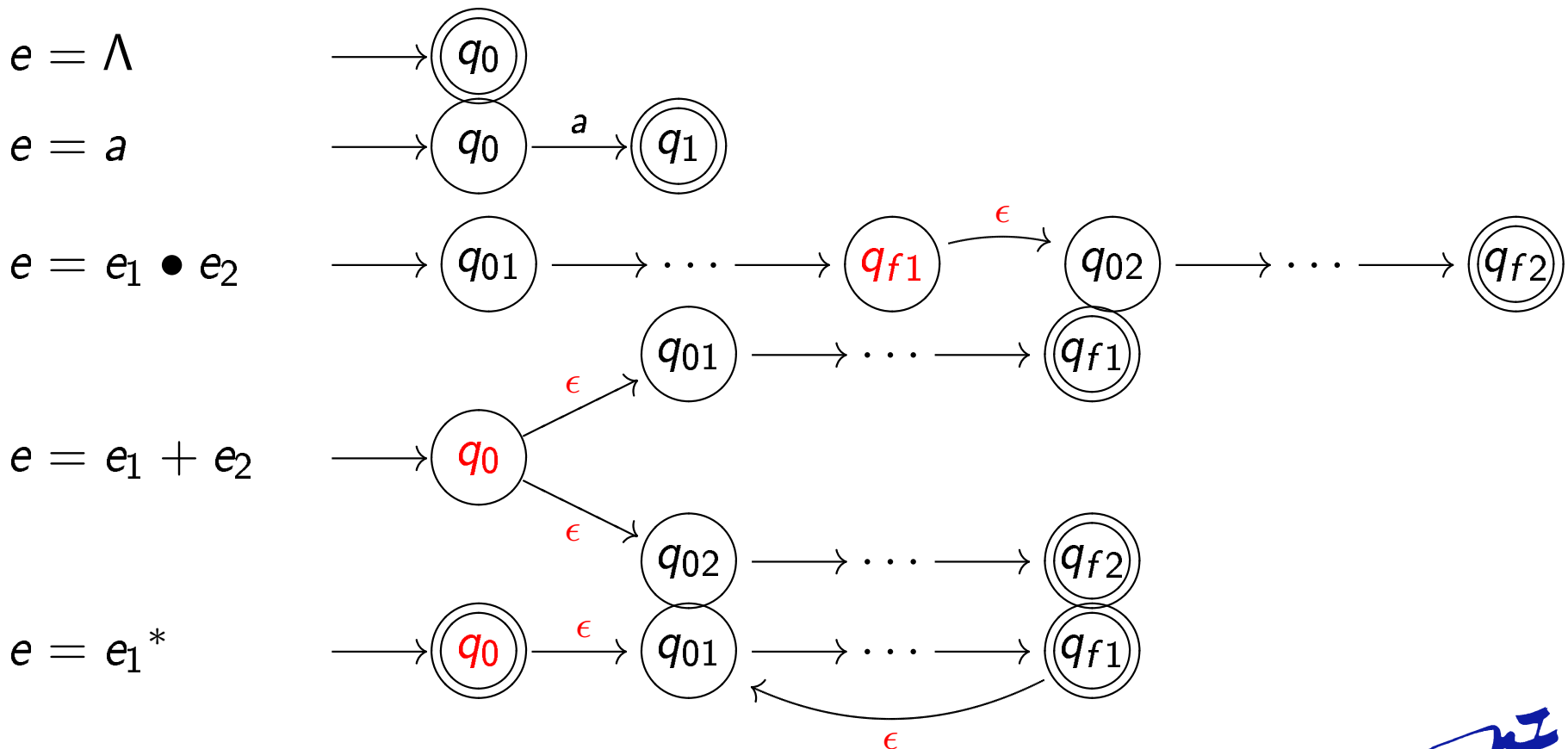
Soit $A = (Q, X, \delta, q_0, F)$ un automate fini déterministe, avec $Q = \{q_0, \ldots, q_n\}$.
Pour tout état $q_i \in Q$, soit e_i l'expression régulière qui représente le langage $\{m \mid \hat{\delta}(q_i, m) \in F\}$.
e_i contient Λ si $q_i \in F$, et l'ensemble des mots de la forme aw, $a \in X$ et $w \in \{v \mid \hat{\delta}(q_{i_a}, v) \in F\}$, où $q_{i_a} = \delta(q_i, a)$.
On a donc un système :

$$e_i = \sum_{a \in X} a e_{i_a} + \{\Lambda \text{ si } q_i \in F\}$$

La solution $e_0 = \{m \in X^* \mid \hat{\delta}(q_0, m) \in F\}$ est $L(A)$.
Rq : la méthode fonctionne avec un automate non-déterministe et/ou non complet.

Plan

Automate associé à une expression régulière - Méthode morphologique



Automate associé – Méthode des dérivées

Soit e une expression régulière sur X et a un symbole de X. La dérivée (à gauche) de e par rapport à a est $D_a(e) = \{m \mid am \in e\}$.

Calcul des dérivées : soit $\Delta(e) = \begin{cases} \Lambda & \text{si } \Lambda \in e \\ \emptyset & \text{si } \Lambda \notin e \end{cases}$

On a alors : $\begin{cases} D_a(a) = \Lambda \quad \text{et} \quad D_a(b) = D_a(\Lambda) = D_a(\emptyset) = \emptyset \\ D_a(e_1 + e_2) = D_a(e_1) + D_a(e_2) \\ D_a(e_1 \bullet e_2) = D_a(e_1) \bullet e_2 + \Delta(e_1) \bullet D_a(e_2) \\ D_a(e_1^*) = D_a(e_1) \bullet e_1^* \end{cases}$

Extension aux mots :

$\begin{cases} D_\Lambda(e) = e \\ D_{yx}(e) = D_x(D_y(e)) \quad \forall x \in X,\ y \in X^* \end{cases}$

Automate associé

Soit e une expression régulière. Le nombre de ses dérivées successives est fini, et ces dérivées sont liées par le système :

$$D_y(e) = \sum_{a \in X} aD_{ya}(e) + \Delta(D_y(e))$$

Soient $e_0, e_1, \ldots, e_n$ les différentes dérivées de e avec $e_0 = e = D_\Lambda(e)$. Soit $A = (\{q_0 \ldots q_n\}, X, \delta, q_0, F)$ où q_i est associé à e_i, $q_i \in F$ si $\Lambda \in e_i$ et $\delta(q_i, a) = q_j$ si $e_j = D_a(e_i)$.
Alors A reconnaît le langage e_0.
L'automate A est déterministe et minimal.

Plan

Syntaxe « shell »

shell	sens	syntaxe théorique
*	un nombre quelconque de caractères	X^*
?	un caractère quelconque	X
[123]	un caractère parmi ...	$\{1,2,3\}$ ou $1+2+3$
[1-5]	un caractère parmi intervalle	$\{1,2,3,4,5\}$ ou ...
[^1-5]	complémentaire	$X \setminus \{1,2,3,4,5\}$

Exemple : `ls sound0[^5-7]*.mp?`

Syntaxe classique

(éditeurs vi/emacs, Perl/PHP, générateur d'analyseur lexical...)

	sens	syntaxe théorique
`.`	un caractère (quasi) quelconque	X
`[123]`	un caractère parmi ...	$\{1,2,3\}$ ou $1+2+3$
`[1-5]`	un caractère parmi intervalle	$\{1,2,3,4,5\}$ ou ...
`[^1-5]`	complémentaire	$X \setminus \{1,2,3,4,5\}$
`e*`	répétition	e^*
`e+`	répétition au moins une fois	ee^*
`e?`	0 ou 1 fois	$e + \Lambda$
`e{n}`	n fois	e^n
`e{n,m}`	de n à m fois	$e^n + e^{n+1} + \dots e^m$
e_1`|`e_2	alternative	$e_1 + e_2$
`()`	groupement p.e. `(a|b)*`	

Conclusion

Il y a équivalence entre :

- les langages rationnels (Rat) ;
- les langages reconnus par les AFN ;
- les langages reconnus par les AFD ;
- les langages définis par les expressions régulières.