

MMIM Modèles mathématiques  
en informatique musicale

Marc Chemillier

Master M2 Atiam (Ircam), 2013-2014

Algorithme de Morris & Pratt

Oracle des facteurs (logiciels d'improvisation OMax, ImproteK)

- Construction de l'oracle
  - o Liens suffixiels
  - o Utilisation musicale de l'oracle

## 1. Algorithme de Morris & Pratt

### 1.1 Problème du pattern matching

Exemple : Rechercher le motif  $x = aaaab$  dans le texte  $t = aaaacaaaab$

$a a a a c a a a a b$

$a a a a \underline{b}$  échec !!!

$a a a \underline{a}$  échec !!!

$a a \underline{a}$  échec !!!

on se décale à chaque fois *d'une lettre dans t*, car chaque  $a$  est potentiellement le début du motif  $x$

**problème :** on n'utilise pas toute l'information obtenue par la lecture de  $t$  conduisant à l'échec

$a a a a \underline{c} a a a a b$

$a a a a \underline{b}$  échec !!! **information :** à cause de la lettre  $c$  dans le texte  $t$

or le motif  $x$  ne contient pas la lettre  $c$

=> on peut se décaler *de la longueur du motif x*

$a a a a \underline{b}$  motif trouvé !!!

algorithme de Morris & Pratt : utiliser toute l'information contenue dans le motif  $x$  pour optimiser les décalages

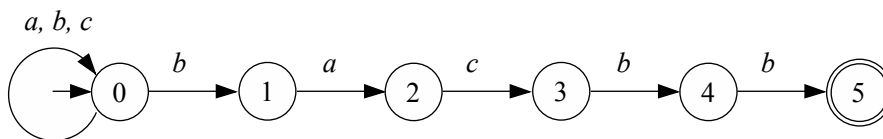
c'est un automate qui fait ça!...

utilisation musicale :

- pattern matching sur des grilles de jazz (ImproteK) : on adapte des fragments de solos joués sur une grille à la grille d'un autre morceau **variante de Morris & Pratt** (on ne recherche pas un motifs fixe, mais des facteurs communs)
- recombinaison de solos en mode free (OMax) : basé sur une variante de l'algorithme de Morris & Pratt appelée **oracle des facteurs**

## 1.2 Construction d'un AFN pour l'ensemble $\Sigma^*x$ des mots se terminant par $x$

Rappelons que l'écorché d'un mot  $x$  de longueur  $n$  est l'AFD de  $n + 1$  états avec  $n$  flèches correspondant aux lettres successives du mot. Pour reconnaître  $\Sigma^*x$ , il suffit d'ajouter une boucle sur l'état initial avec toutes les lettres de l'alphabet. Exemple :  $x = bacbb$



-> Pourquoi est-ce un AFN ?

il y a deux flèches avec  $b$  partant de l'état initial.

Application à la recherche d'un motif  $x$  dans un texte  $t$  :

- on détermine l'AFN pour obtenir un AFD reconnaissant  $\Sigma^*x$
- on lit le texte  $t$  dans cet automate. Que se passe-t-il si on arrive à l'état terminal ?

-> On a trouvé le motif  $x$  dans le texte  $t$ .

## 1.3 Utilisation d'une fonction d'échec

**Algorithme de Morris & Pratt** : calcul direct de l'AFD pour  $\Sigma^*x$ , **sans passer par la détermination de l'AFN**. On utilise une fonction d'échec définie sur les états  $p$  de l'écorché de  $x$ . Soit  $w$  le mot lu de l'état 0 à l'état  $p$  :

$f(p) =$  état d'arrivée du plus long suffixe propre de  $w$  qui est aussi préfixe de  $w$  (donc de  $x$ )

Exemple : Fonction d'échec pour l'écorché de  $x = bacbb$  :

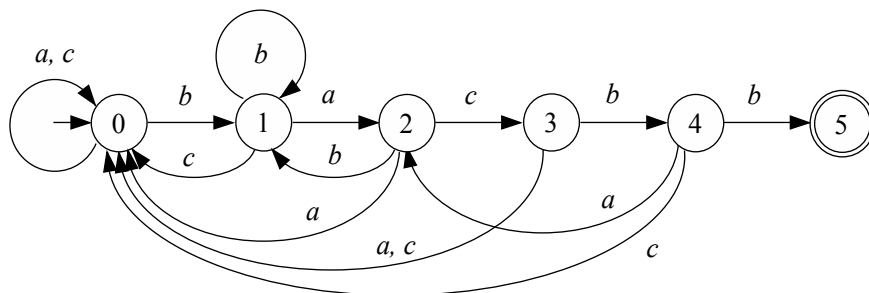
État $p$	0	1	2	3	4	5
$f(p)$		0	0	0	1	1

- pour  $p = 2$ , le mot lu est  $ba$ , il n'a pas de suffixe propre qui soit préfixe, donc  $f(2) = 0$
- pour  $p = 4$ , le mot lu est  $bacb$ , le plus long suffixe propre qui soit préfixe est  $b$ , donc  $f(4) = 1$
- pour  $p = 5$ , le mot lu est  $bacbb$ , le plus long suffixe propre qui soit préfixe est  $b$ , donc  $f(5) = 1$

On ajoute de nouvelles transitions dans l'écorché de  $x$  de la manière suivante :

pour toute lettre  $a$  telle que  $\delta(p, a)$  non défini, on pose

- $\delta(p, a) = \delta(f(p), a)$  si  $p \neq 0$ ,
- $\delta(0, a) = 0$ .



Exemple : Lecture de l'AFD pour chercher le motif  $x = bacbb$  dans le texte  $t = abacbcbba$

$a \ b \ a \ c \ \underline{b \ a \ c \ b \ b} \ a$   
 0 1 2 3 4 2 3 4 5 motif trouvé !!!

#### 1.4 Algorithme de calcul de la fonction d'échec

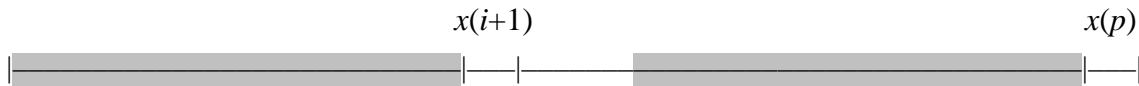
Pour calculer facilement  $f(p)$ , il faut voir si on peut le déduire de  $f(p-1)$ .

-> Il se trouve que ça marche.

On note le motif  $x = x(1) \dots x(n)$

On suppose construits  $f(1), f(2) \dots f(p-1)$  et on cherche  $f(p)$ .

$$i = f(p-1)$$



- si la lettre  $x(i+1) = x(p)$   
alors  $x(1) \dots x(i+1)$  est à la fois préfixe et suffixe,  
donc on peut prolonger le suffixe précédent :  $f(p) = i + 1 = f(p-1) + 1$

- si la lettre est différente, il faut essayer un suffixe plus court, lequel ?

$$i = ?$$



=> ça doit être un suffixe du suffixe précédent :

donc on réapplique f



$$i = f(f(p-1))$$

et on recommence la comparaison entre  $x(i+1)$  et  $x(p)$

calcul de la fonction d'échec f

```

f(0) ← -1
i ← -1
pour p = 1 à n
    tant que i ≥ 0 et x(i+1) ≠ x(p)
        i ← f(i)
    i ← i+1
    f(p) ← i
    
```

En fait, l'algorithme de Morris & Pratt ne construit pas explicitement l'AFD de  $\Sigma^*x$ . Il se contente de lire le texte  $t$  en avançant dans l'écorché de  $x$  (AFD), et en effectuant les retours arrière définis par la fonction d'échec.

```

 $p \leftarrow 0$  (état initial)
tant que  $t$  non vide
avancer dans  $t$ ,  $a \leftarrow$  1ère lettre de  $t$ 
    tant que  $p = 0$  et  $\delta(p, a)$  non défini
         $p \leftarrow f(p)$ 
    si  $\delta(p, a)$  défini alors avancer lecture  $p \leftarrow \delta(p, a)$ 
    si  $p \in F$  alors signaler occurrence de  $x$  dans  $t$ 

```

L'algorithme naïf de détection de motif consiste à essayer de lire  $x$  dans le texte à partir de la 1<sup>ère</sup> lettre, puis de la 2<sup>ème</sup>, puis de la 3<sup>ème</sup>, etc. **en se décalant à chaque fois d'une seule lettre.**

Dans l'algorithme de Morris & Pratt, si on a un échec de la lecture à la lettre en position  $p$  du motif, on se décale de  $|x| - f(p)$  lettres. **Cas optimal** : le motif  $x$  ne comporte aucune répétition, c'est-à-dire  $f(p) = 0$  pour tout  $p$ , donc **on se décale à chaque fois de la longueur du motif.**

-> ça accélère considérablement le calcul

## 2. Construction de l'oracle

### 2.1 Liens suffixiels

L'ensemble des facteurs d'un mot  $x$  est fini, donc reconnaissable par automate. On obtient facilement un AFN reconnaissant tous les facteurs en prenant l'écorché de  $x$  et en rendant tous les états initiaux et terminaux.

Problème : **il n'existe pas de construction simple pour l'AFD des facteurs de  $x$**  (contrairement à l'automate des mots se terminant par  $x$ ).

L'« oracle des facteurs » a été introduit pour pallier cette difficulté.

L'*oracle des facteurs* est un automate qui reconnaît tous les facteurs d'un mot  $x$ , mais avec quelques mots en plus. Il est utilisé dans les applications de bio-informatique de façon **négative** : si un motif n'est pas reconnu par l'oracle d'une séquence ADN, on sait qu'il n'apparaît pas dans cette séquence.

On peut donner directement l'AFD correspondant (sans déterminisation) par une technique analogue à l'algorithme de Morris & Pratt.

Exemple : écorché de la séquence  $x = abcadbcd$



Construction de l'oracle : on part de l'écorché de  $x$ , et on ajoute des transitions à l'aide d'une fonction  $f$  dite de « lien suffixiel » entre états, en construisant simultanément  $f$  et les transitions.

Au départ, on place un lien suffixiel de 1 vers 0. Puis on suppose l'oracle construit avec ses liens suffixiels jusqu'à l'état  $p$  compris. La lettre suivante  $a$  donne un nouvel état  $\delta(p, a) = p+1$ .

Pour ajouter les transitions, on suit les liens suffixiels déjà existants  $p' = f(p)$ , puis  $p' = f(f(p))$ , etc.

- si  $\delta(p', a)$  est non défini, on ajoute une transition  $\delta(p', a) = p+1$ 
  - si  $p' \neq 0$ , on continue à suivre les liens,
  - si  $p' = 0$ , on stoppe et on crée un lien suffixiel en posant  $f(p+1) = 0$
- si  $\delta(p', a)$  est défini, on stoppe (pas de nouvelle transition), et on crée un lien suffixiel en posant  $f(p+1) = \delta(p', a)$  (= état d'arrivée de la transition existante)

Remarque : toutes les transitions arrivant dans un même état ont la même lettre.

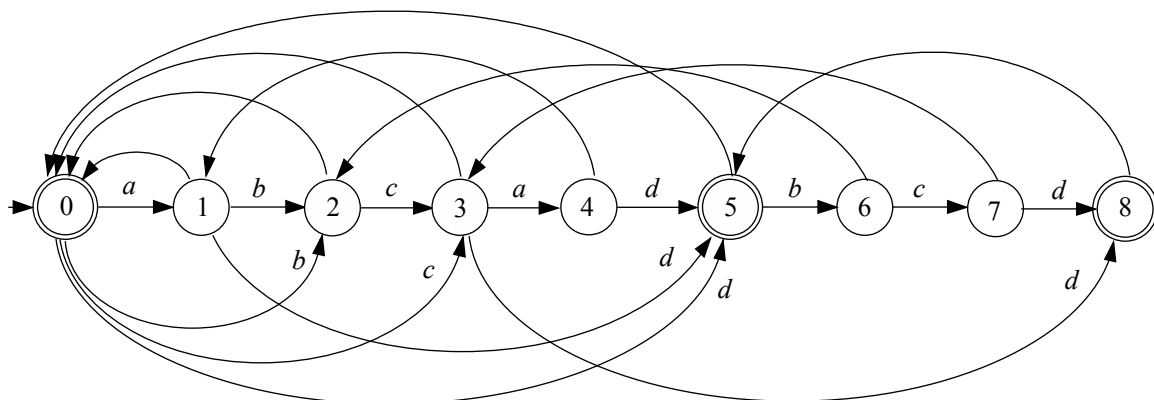
Pour la séquence  $abcdabcd$ , l'oracle des suffixes donne l'AFD suivant (les liens suffixiels sont indiqués par des flèches en pointillé au-dessus).

- Oracle des facteurs : tous les états sont terminaux
- Oracle des suffixes : les états terminaux sont ceux du chemin suffixiel partant du dernier état, soit 8 (suffixe  $cd$ ), 5 (suffixe  $d$ ), 0 (suffixe vide)

Les flèches du dessus *sans étiquette* sont les liens suffixiels.

Les flèches du dessous avec étiquette sont les transitions additionnelles (ajoutées à l'écorché).

Tant qu'on lit des lettres nouvelles ( $a, b, c, \dots$ ), les liens suffixiels vont vers 0. Quand il y a une répétition (par ex.  $a$ ), les liens suffixiels vont vers des états plus grands.





-> cela permet d'enchaîner des motifs avec des chevauchements plus grands

## Références

- oracle des suffixes (simulation stylistique)

Allauzen, Cyril & Maxime Crochemore, Mathieu Raffinot, Factor oracle : A new structure for pattern matching, *SOFSEM '99: Proceedings of the 26th Conference on Current Trends in Theory and Practice of Informatics*, Lecture Notes in Computer Science, Springer-Verlag, 1999, p. 291-306 ([biblio](#)).

Assayag, Gérard, Shlomo Dubnov, Olivier Delerue, Guessing the Composer's Mind: Applying Universal Prediction to Musical Style, *Proceedings of the ICMC (Int. Computer Music Conf.)*, 1999, p. 496-499 ([biblio](#)).

Assayag, Gérard, Shlomo Dubnov, Using factor oracles for machine improvisation, *Soft Computing*, special issue on Formal Systems and Music, G. Assayag, V. Cafagna, M. Chemillier (eds.), 8 (9) (2004) 604–610 ([biblio](#)).

Assayag, Gérard, Georges Bloch, Navigating the oracle, *Proceedings of the ICMC (Int. Computer Music Conf.)*, 2007 ([biblio](#)).