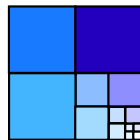

Thèse de doctorat

Automates WORM

et collages de mots et d'images

Oğuz Berke DURAK

`durak@liafa.jussieu.fr`

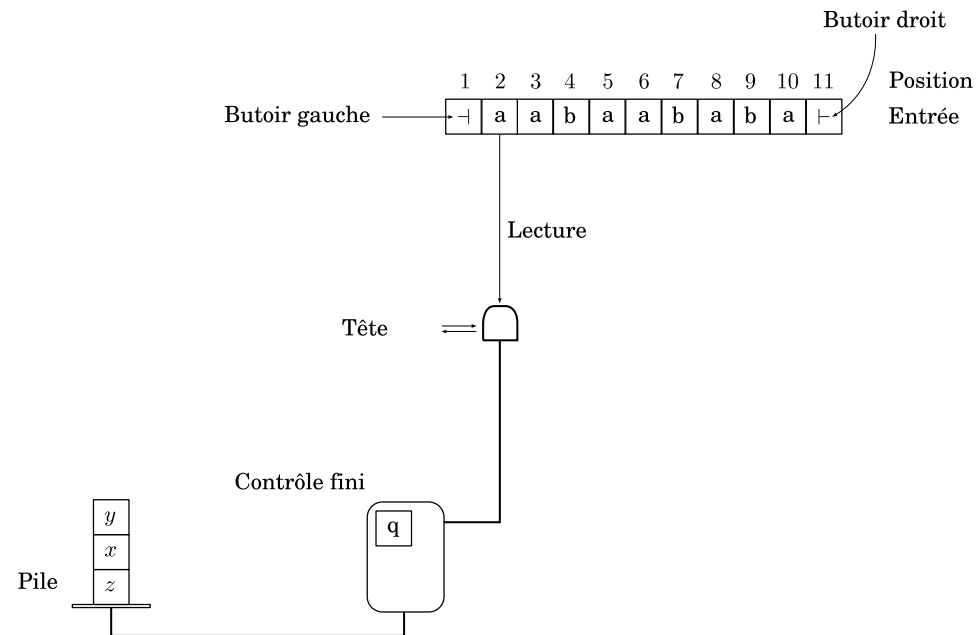
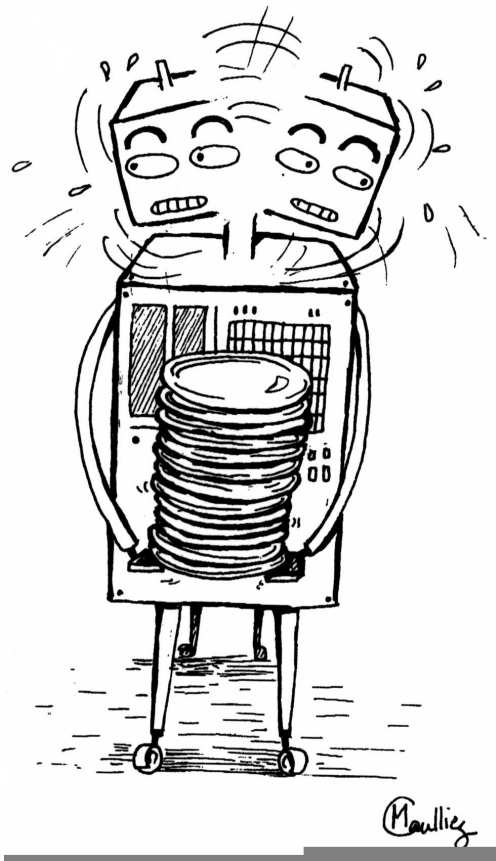


LIAFA, Université Paris VII



Il était une fois...

Il était une fois un automate déterministe bidirectionnel à pile (*two-way deterministic pushdown automaton* ou 2DPDA) (Gray, Harrison, Ibarra 1967).



Les 2DPDA

Ces machines peuvent reconnaître plein de langages intéressants :

- Palindromes : $\{uu^R \mid u \in \Sigma^*\}$
- Concaténations de palindromes : $\{uu^Rvv^R \mid u, v \in \Sigma^*\}$
- Problème de recherche de motif : $\{uv\#u \mid u, v \in \Sigma^*\}$
- Entiers carrés codés en unaire : $\{a^{n^2} \mid n \geq 0\}$

Simulation des 2DPDA

Soit un 2DPDA M

- avec n états,
- ayant un alphabet de pile à g symboles,
- sur une entrée u de longueur m .

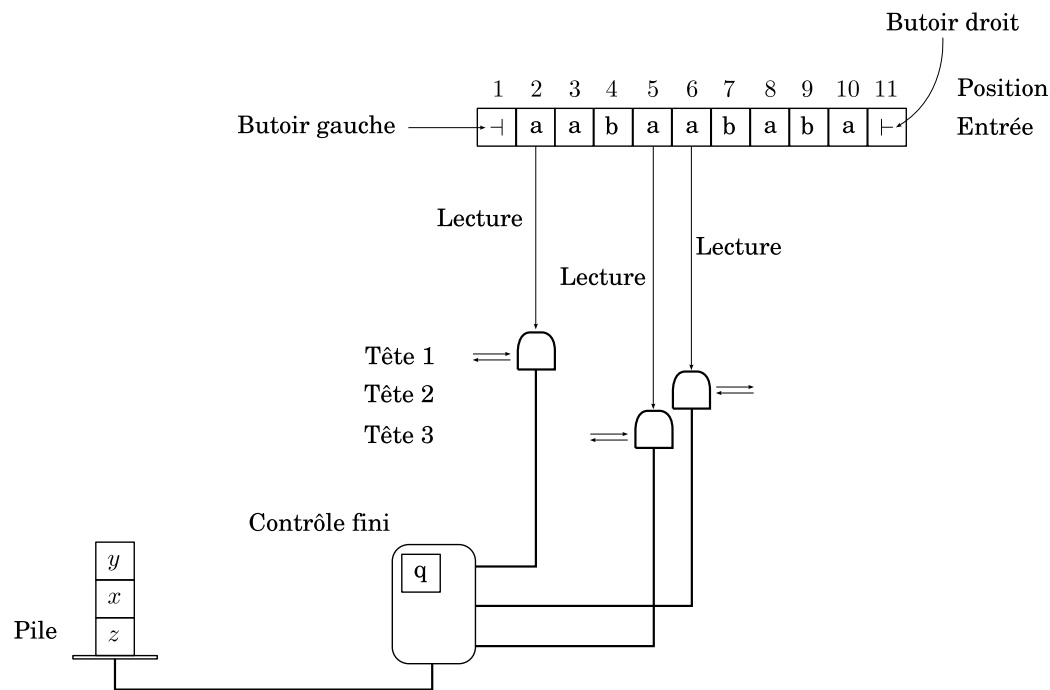
Théorème 1 (Cook, 1972). *Une machine RAM peut décider si M accepte u ou pas en temps*

$$\mathcal{O}(n \times g \times m).$$

L'algorithme de Cook est off-line. Andersen et Jones (1994) ont une version on-line et plus facile à comprendre.

Augmenter leur nombre de têtes est assez naturel

En rajoutant un nombre suffisant de têtes, on peut reconnaître tout langage reconnaissable en temps polynomial.



La simulation de cook se fait alors en temps (ngm^k) avec k têtes.

Les 2DPDAs et la classe \mathbf{P}

Les langages reconnus par 2DPDAs sont étroitement liés aux langages reconnaissables en temps polynomial :

$$\text{DTIME}(n^k) \sim k - \text{2DPDA}.$$

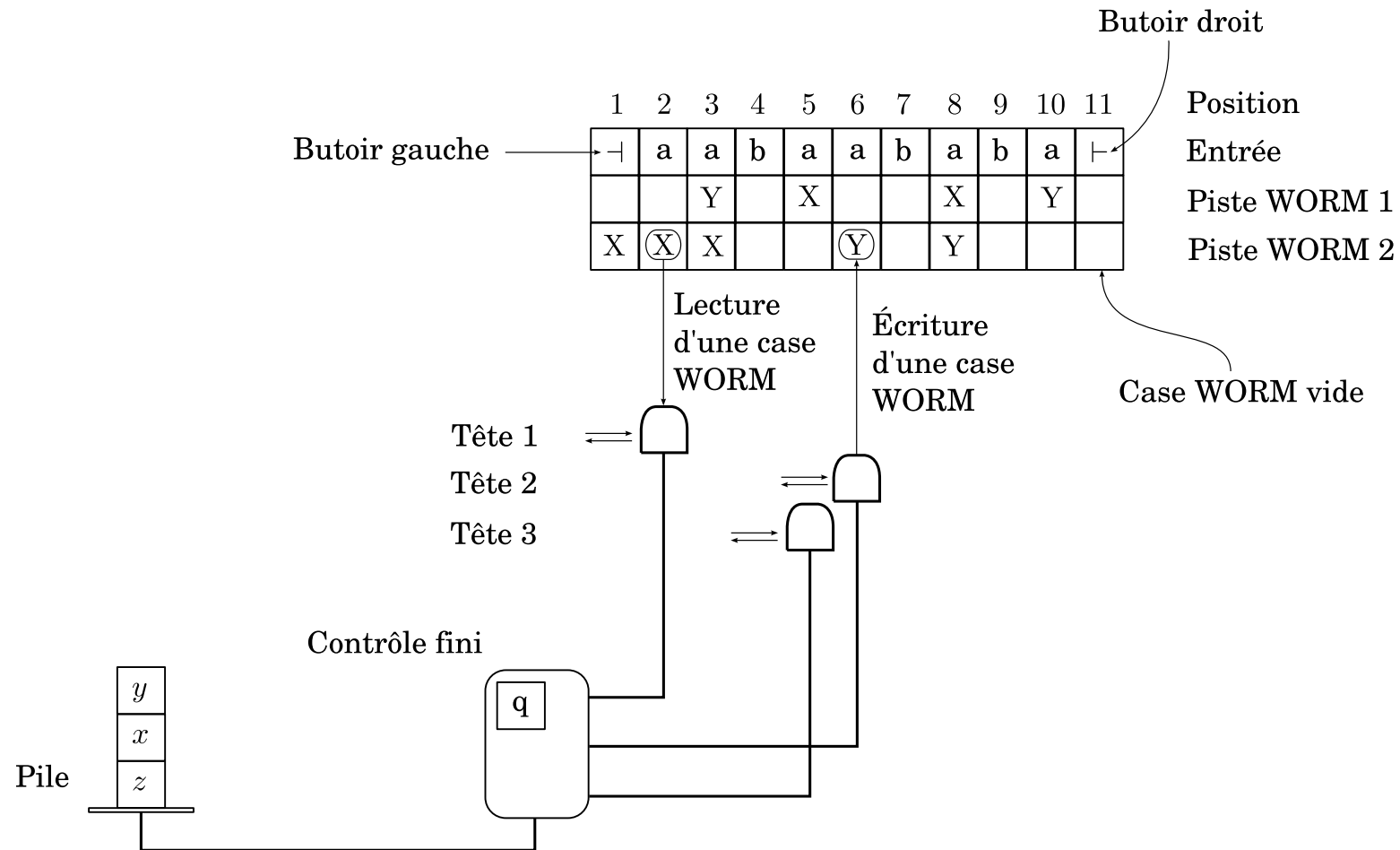
- Si $L \in \text{DTIME}(n^k)$ alors $L \in 2k - \text{2DPDA}$ (Cook, 1972)
- Si $L \in \text{DTIME}(n^k)$ alors $\text{pad}(L, n^k (\log n)^2) \in \text{2DPDA}$ où $\text{pad}(L, f) = \{u \# a^{f(|u|) - |u|} \mid u \in L\}$ (Monien, 1975)
- Si $L \in 2k - \text{2DPDA}$ alors $L \in \text{DTIME}(n^{2k})$ (Cook, 1971)

Ainsi d'importants problèmes de complexité peuvent se reformuler en termes d'acceptance par 2DPDAs.

Zvi Galil, "Some open problems in the theory of computation as questions about two-way deterministic pushdown automata languages.", Math. Syst. Th. 1977.

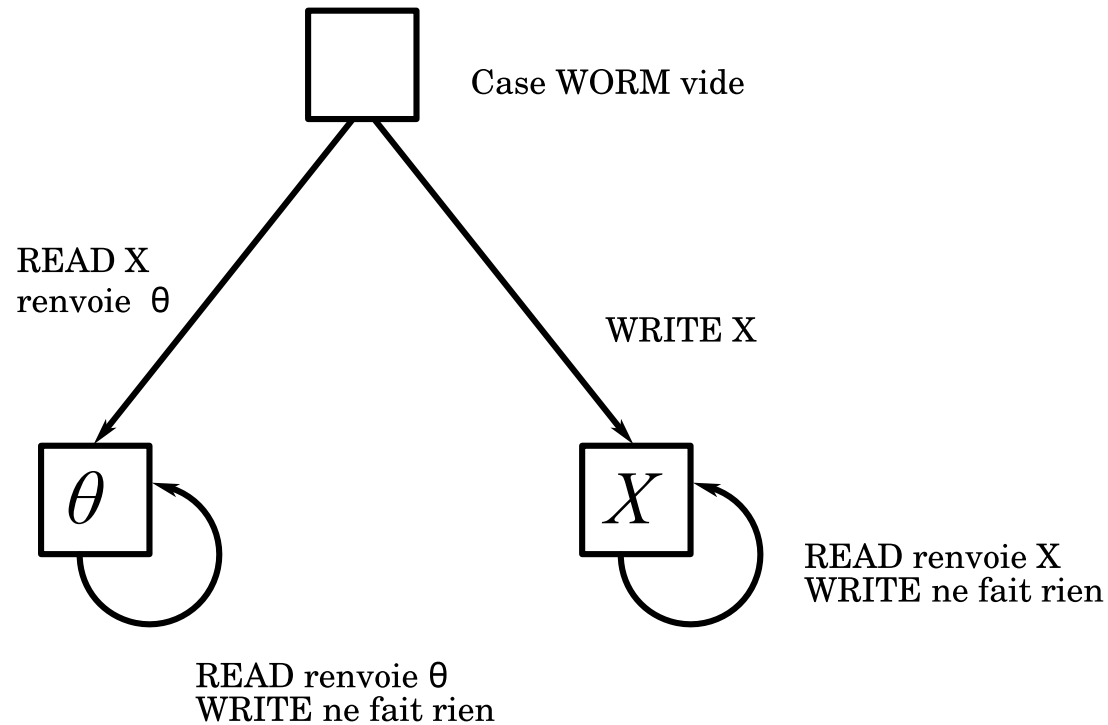
WORM-2DPDAs (Mogensen, 1994)

Étendre les 2DPDAs par l'ajout de pistes inscriptibles spéciales, dites pistes "WORM" :



Cases WORM

WORM = “Write once, read many”



Lire une case vide la pré-remplit avec θ . Ainsi, l'automate ne peut jamais savoir si une case est vide ou si elle contient θ .

Pistes WORM

Les pistes WORM :

- Sont constituées de cases WORM, en même nombre que les cases d'entrée.
- L'accès à ces pistes se fait avec les têtes de lecture.
- Sont utiles pour stocker des résultats intermédiaires.
- *Le théorème de Cook reste vrai*, mais avec la méthode on-line d'Andersen et Jones.

Application des 2DPDAs et des WORM-2DPDAs

Applications des 2DPDAs :

- Algorithme Knuth-Morris-Pratt (Knuth et al, 1977),
- Algorithme pour reconnaître $\text{PALSQUARE} = \{uu^Rvv^R \mid u, v \in \Sigma^*\}$ en temps linéaire (Galil, 1977).

Applications connues des WORM-2DPDAs :

- Un automate plus simple pour reconnaître PALSQUARE (Mogensen, 1992),
- Analyse lexicale en temps linéaire (Reps, 1998).

Le langage Diamond

Exprimer des 2DPDAs par leur table de transitions n'est pas commode. Les décrire en langage naturel reste flou et difficile à vérifier.

- Des langages pour (WORM-)2DPDAs existent déjà (Andersen & Jones 1994, Mogensen 1994, Christensen 2002).
- Ils sont impératifs et de bas niveau (`if-then-else`, `goto` ; ils n'ont pas de macro-instructions).

On propose Diamond: Un langage pour 2DPDAs et WORM-2DPDAs.

- La pile, la tête et les pistes WORM sont contrôlées explicitement.
- Des **procédures mutuellement récursives se passant des arguments finis** définissent le programme. Les récursions doivent être terminales.

Exemple de programme Diamond

```
(* Programme reconnaissant les palindromes de
 * longueur impaire *)

value a = "a"
value b = "b"

procedure rec is_parity kind =
  if home then
    if kind = "even" then
      push "yes"
    else
      push "no"
    else
      if kind = "even" then
        left_is_parity "odd"
      else
        left_is_parity "even"
  and left_is_parity kind =
    left;
    is_parity kind

procedure rewind =
  while not home do
    left
  done

procedure copy =
  while not eof do
    if see a then push a else push b;
    right
  done
```

```
procedure check =
  while not empty do
    if see a && top a or see b && top b then
      begin
        pop;
        right
      end
    else
      reject
  done;
  accept

procedure main =
  copy ();
  is_parity "odd";
  if top "yes" then
    begin
      pop;
      rewind ();
      check ()
    end
  else
    reject

start main
```

2DPDAs vs WORM-2DPDAs

Malgré les facilités fournies par Diamond, on n'a pas pu résoudre la question :

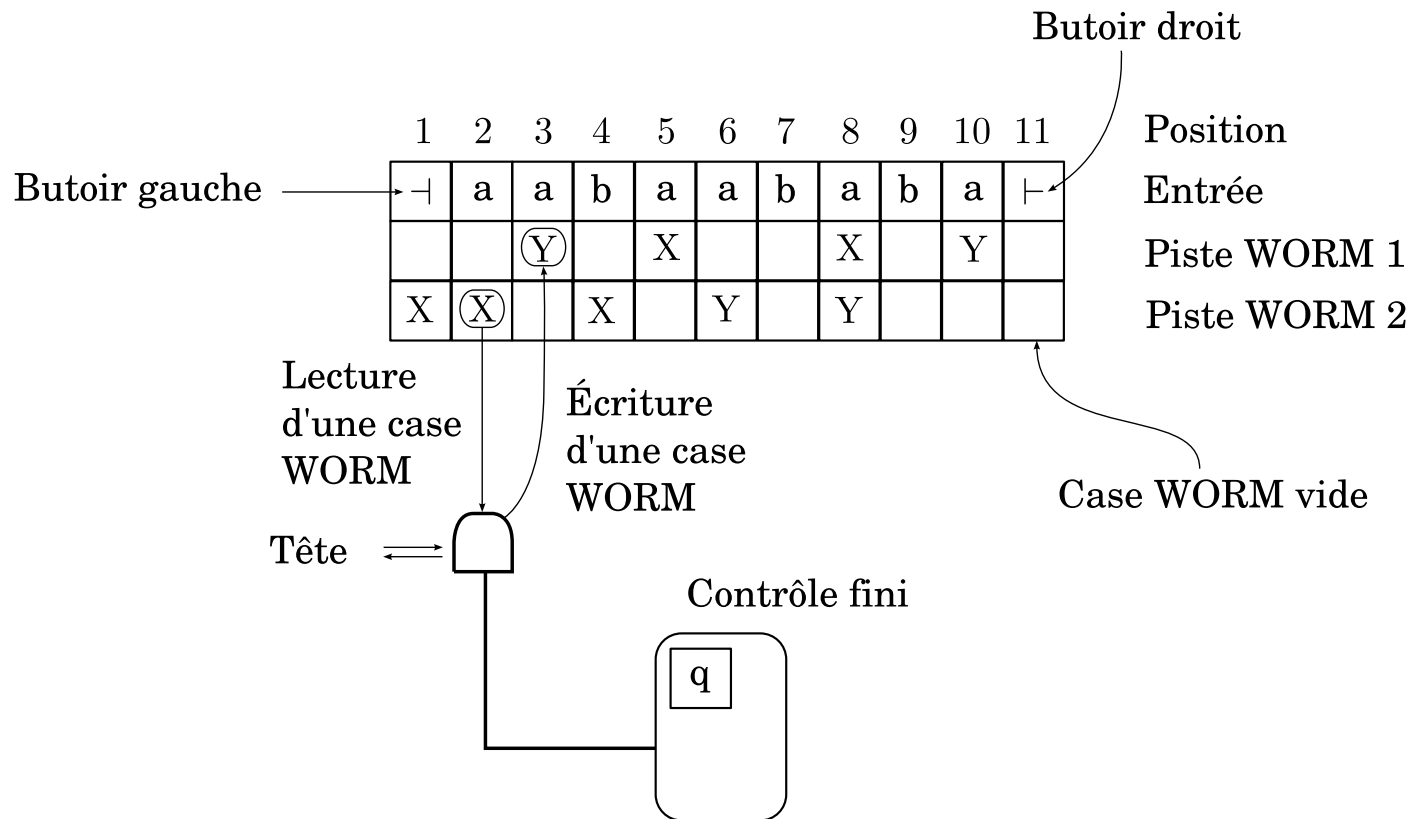
Les pistes WORM augmentent-ils la puissance (ou la concision) des 2DPDAs ?

On a donc étudié la question :

Les pistes WORM augmentent-ils la puissance (ou la concision) des automates finis (2DFAs, 2NFAs, etc.) ?

WORM-2DFAs

En enlevant la pile des WORM-2DPDAs (ou en rajoutant une piste WORM aux 2DFAs), on obtient les WORM-2DFAs : automates déterministes finis bidirectionnels munis d'une piste WORM.



Puissance des WORM-2DFAs

- Les langages des WORM-2DFAs sont réguliers (preuve conventionnelle et facile à trouver).
- Ils pourraient présenter un gain de concision (une cascade de k transducteurs lettre-à-lettre à n_1, \dots, n_k états peut se réaliser avec k pistes et $\mathcal{O}(n_1 + \dots + n_k)$ états) par rapport aux 2DFAs.

L'étape suivante est l'ajout du non-déterminisme.

- Contrairement aux WORM-2DFAs, un WORM-2NFA ne se répètera pas nécessairement s'il repasse deux fois dans un même état dans une position donnée.
- On notera que **la régularité des langages des WORM-2NFAs n'est pas facile à établir.**

Concision des WORM-2NFAs

Tranches finies de SAT :

- Des formules sur n variables,
- Codées sous forme de mots sur $\Sigma_n = \{x_1, \dots, x_n, \wedge, \vee, \neg, I\}$,
- S_n^+ : formules satisfaisables à n variables sur Σ_n .

Pour tout n , un WORM-2NFAs à $\mathcal{O}(n^2)$ états peut reconnaître S_n^+ .

Autres aspects :

- Composition de transducteurs (comme pour les 2DFAs, mais en non-déterministe),
- Simulation de 2AFAs (resp. 2AFAs sans boucle) à n états avec $\mathcal{O}(n^2)$ (resp. en $\mathcal{O}(n)$) états.

WORM-2NFAs vs l.p.hom-2NFAs

Soient :

- M un 2NFA sur Σ
- $\varphi : \Sigma \rightarrow \Delta$ un morphisme lettre-à-lettre

Alors (M, φ) est un *l.p.hom-2NFA* (length-preserving homomorphism-2NFA) dont le langage est $\varphi(\mathcal{L}(M))$ (Birget, 1996).

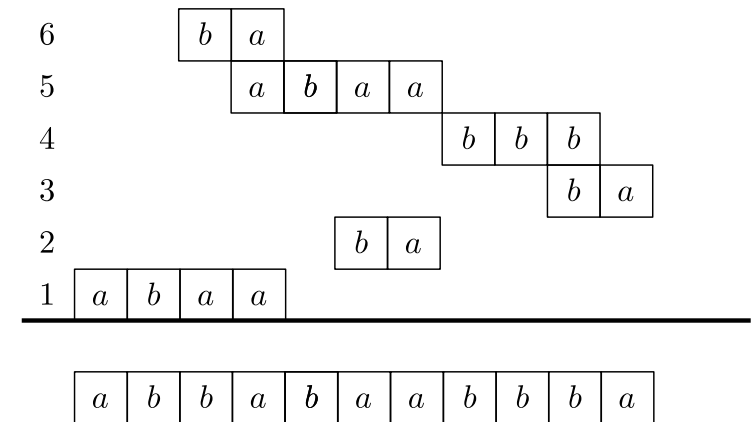
Un l.p.hom-2NFA à $\mathcal{O}(n^2)$ états peut aussi reconnaître S_n^+ ! **Les pistes WORM apportent-elles quelque chose ?**

On cherche donc un langage facile pour les WORM-2NFAs mais difficile pour les l.p.hom-2NFAs.

Collages

Soit K un langage rationnel sur Σ^* . Soit $\square \notin \Sigma$. On engendre des mots sur Σ comme suit :

- On choisit $m \geq 0$ et on commence avec un mot de m blancs.
- Tant qu'on a envie ou que le mot contient des blancs :
 - Choisir au hasard une position $1 \leq i \leq m$ et un mot $v \in K$,
 - Écrire v dans u à la position i **en remplaçant les anciens caractères.**



On obtient ainsi un nouveau langage $\text{Collage}(K)$.

Rationalité des collages

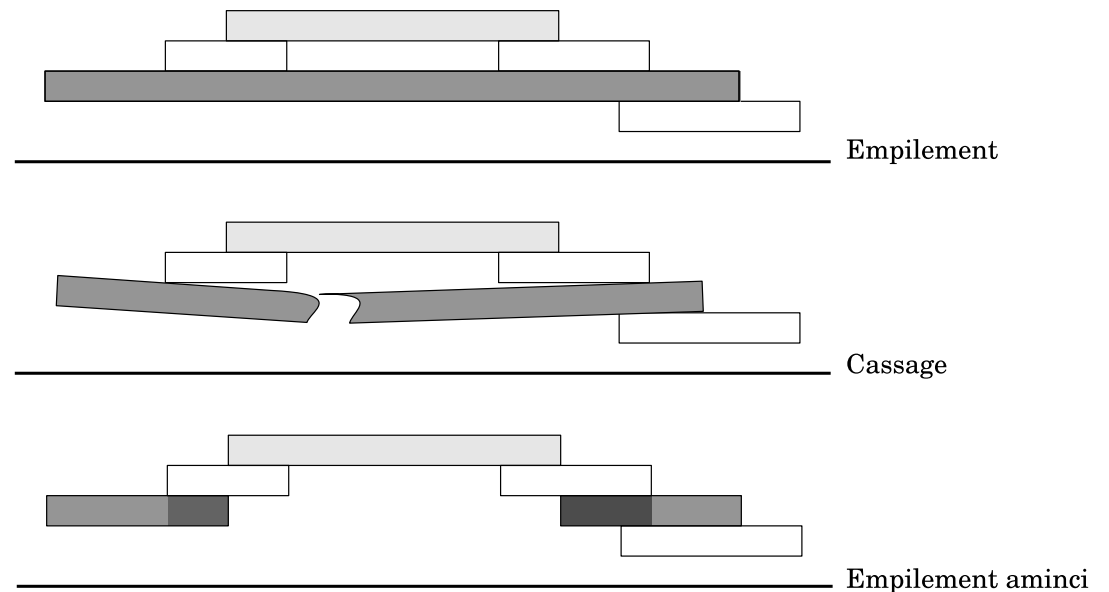
Théorème 2 (Choffrut & Durak, 2005).

$$K \in \text{Rat}(\Sigma^*) \implies \text{Collage}(K) \in \text{Rec}(\Sigma^*)$$

Les WORM-2NFAs
peuvent facilement
accepter $\text{Collage}(K)$.

Question
ouverte : les l.p.hom-2NFA
peuvent-ils facilement
reconnaître $\text{Collage}(K)$?

(Remarque : l'automaticité
de $\text{Collage}(K)$
peut être exponentielle
en celle de K .)



Quelques propriétés des WORM-2NFAs

Proposition 1. *La taille des calculs peut être supposée quadratique.*

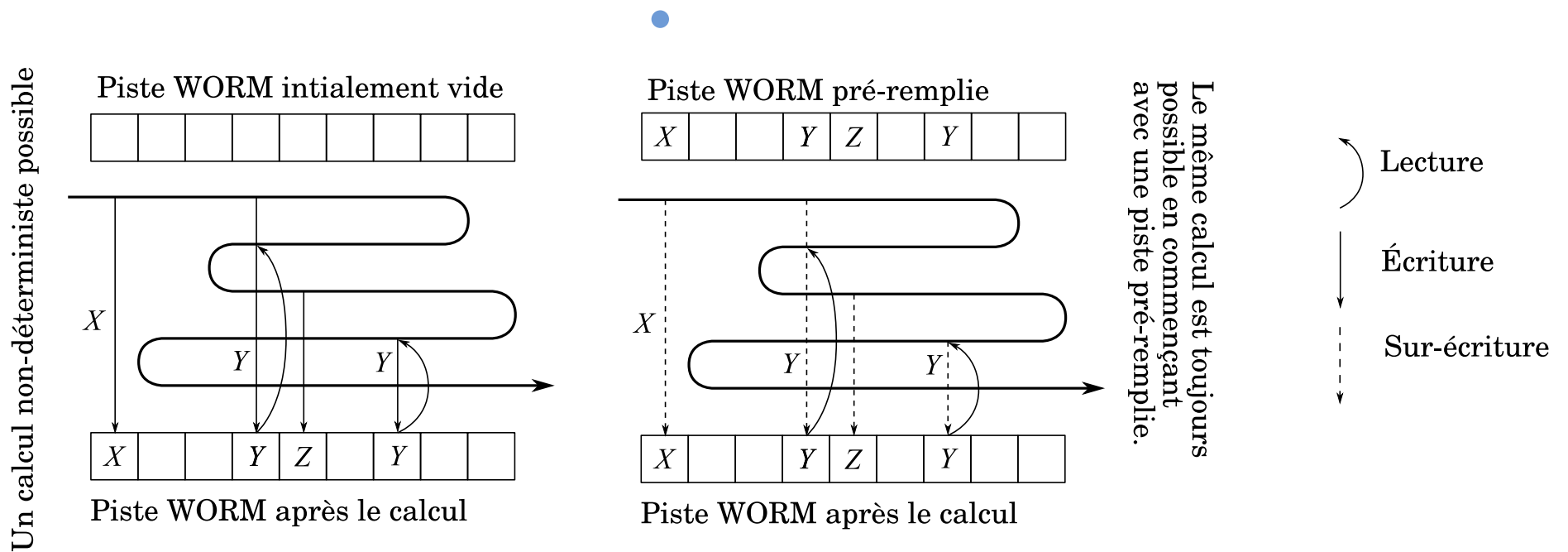
Corollaire 1. *Le problème d'acceptance pour les WORM-2NFAs est dans NP, et donc NP-complet.*

De plus, la longueur d'une "crossing sequence" peut être linéaire : **les techniques habituelles de "crossing sequences" ne marchent pas.**

Résultat principal

Théorème 3 (Durak, 2005). *Les langages des WORM-2NFAs sont réguliers.*

La preuve utilise une propriété spécifique aux WORM-2NFAs :
la répétabilité.



Extensions des WORM-2NFAs

Toutes les extensions que nous avons pu imaginer donnent des modèles reconnaissant des langages non réguliers.

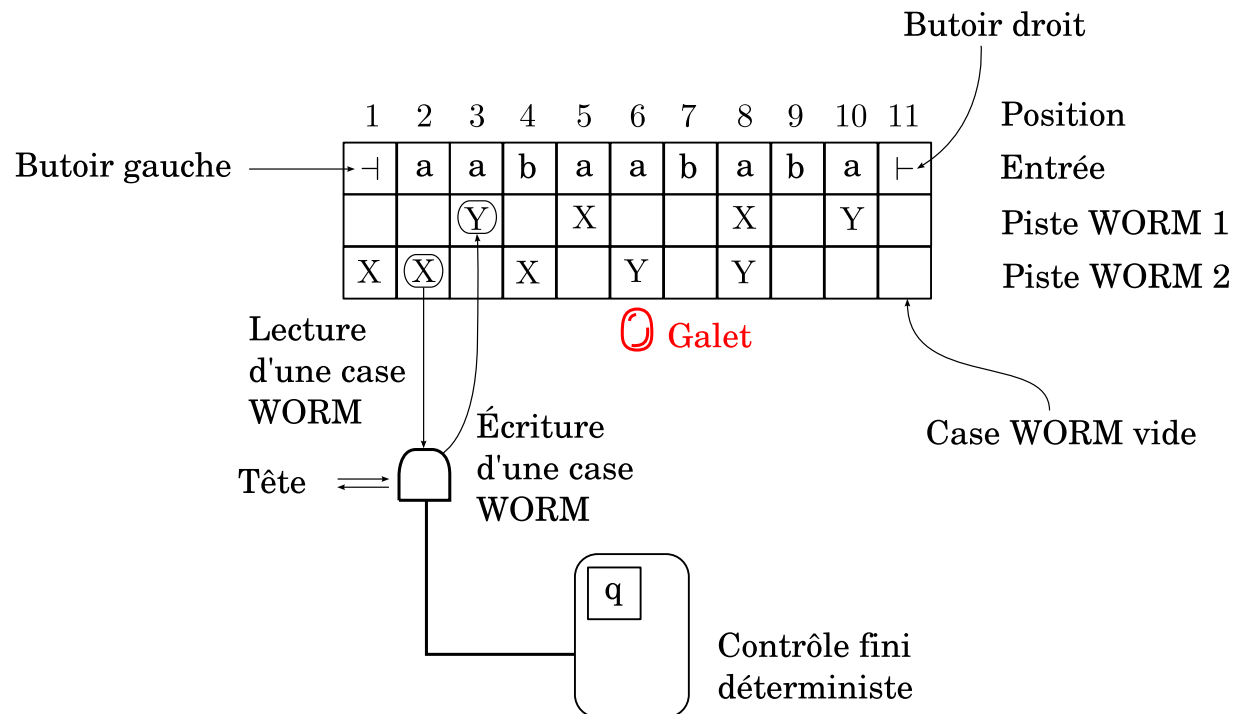
- L'interdiction des écritures multiples permet d'accepter $Z = \{a^n b^{\geq n} \mid n \geq 0\}$.
- L'alternation permet de simuler l'interdiction des écritures multiples.
- L'ajout d'un galet permet aussi d'accepter Z .

P-WORM-2DFAs

Une extension qui reste régulière : les pebble-WORM-2DFAs.

- Un WORM-2DFA (donc **déterministe**)
- Muni d'un seul galet qui peut être déplacé, déposé, détecté.

Théorème 4 (Durak, à paraître). *Les langages de P-WORM-2DFAs sont réguliers.*



Retour aux collages

On a vu que les collages de langages rationnels sont rationnels. Il existe des langages algébriques dont les collages ne sont pas algébriques :

$X = \{ca^n b^m d \mid n > m\}$ Collage(X) n'est pas algébrique.

L'opération de collage peut aussi se définir sur les images.

- Image = mot bidimensionnel = matrice de pixels, chaque pixel ayant une couleur.
- Couleur = lettre, palette = alphabet.
- Ensemble des images sur Σ : $\Sigma^{*\times*}$
- Ensemble des images à m lignes et n colonnes : $\Sigma^{m\times n}$.

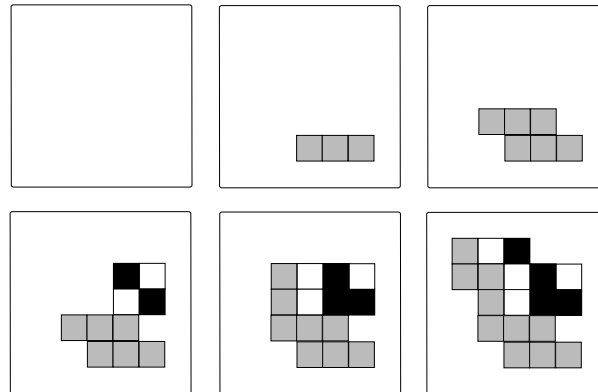
Collages d'images

Soit un langage (fini) d'images $P = \{p_1, p_2, p_3\}$ avec

$$p_1 = \begin{array}{|c|c|c|} \hline \text{gris} & \text{gris} & \text{gris} \\ \hline \end{array} \quad p_2 = \begin{array}{|c|c|} \hline \text{noir} & \text{blanc} \\ \hline \text{blanc} & \text{noir} \\ \hline \end{array} \quad p_3 = \begin{array}{|c|c|c|} \hline \text{gris} & \text{blanc} & \text{noir} \\ \hline \text{gris} & \text{blanc} & \text{noir} \\ \hline \end{array} \quad p_4 = \begin{array}{|c|c|c|} \hline \text{gris} & \text{gris} & \text{blanc} \\ \hline \text{gris} & \text{blanc} & \text{noir} \\ \hline \end{array}$$

Collons ces morceaux suivant la suite de positions :

$$s = (6, 4, p_1) \cdot (5, 3, p_1) \cdot (3, 5, p_2) \cdot (3, 3, p_3) \cdot (2, 2, p_4)$$

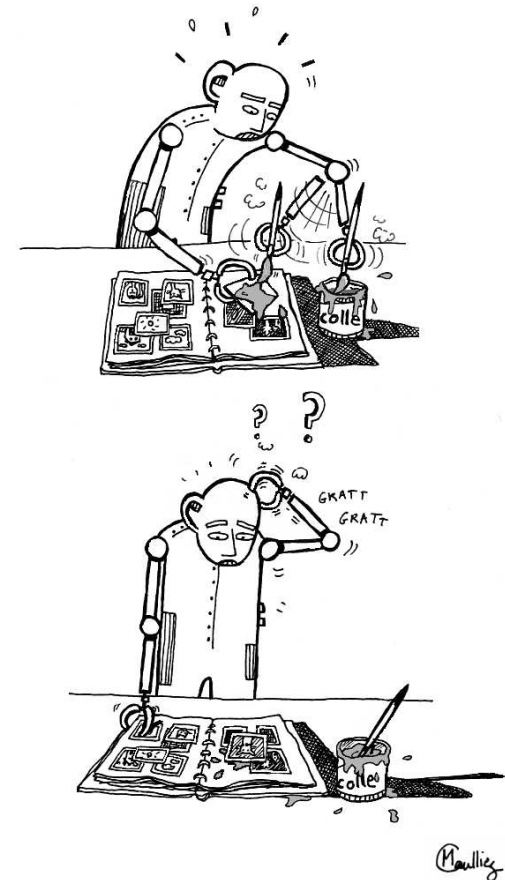


Cela définit un langage $\text{Collage}(P)$ comme pour les mots, **sauf que le symbole vide est autorisé.**

Langages d'images reconnaissables

Il existe une notion de reconnaissabilité, moins élégante, pour les images, avec quatre caractérisations : (cf. par exemple Giammaressi, 1997)

- PCFRE : expressions régulières sans complémentation et avec projection
- EMSO : fragment existentiel d'une logique monadique du second ordre
- 2OTA : automates à carrelage
- **TS : systèmes de pavage**



Systemes de pavage

Un systeme de pavage est un tuple $(\Sigma, \Theta, \Delta, \pi)$ avec :

- Σ, Δ : deux alphabets (palettes)
- $\Theta \subseteq (\Sigma \cup \{\#\})^{2 \times 2}$: ensemble de **sous-rectangles 2×2 autorisés**
- $\pi : \Sigma \rightarrow \Delta$: projection

Une image $u \subseteq \Delta^{* \times *}$ appartient au langage defini par un s.p. $(\Sigma, \Theta, \Delta, \pi)$ s'il existe $v \subseteq \Sigma^{* \times *}$ tel que :

- Chaque sous-rectangle 2×2 de v entouré de $\#$ est dans Θ ,
- Et $\pi(v) = u$.

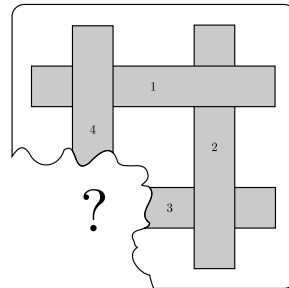
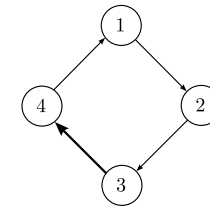
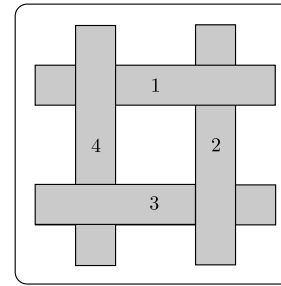
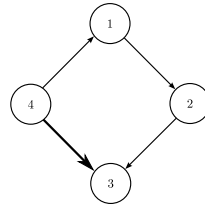
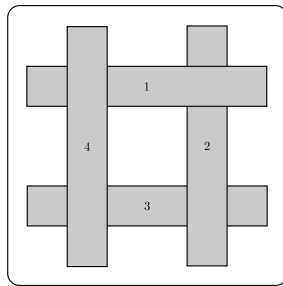
Exemple 1. *Systeme de pavage definiissant les images carrées blanches.*

$$\Sigma = \{\square, \blacksquare\} \quad \Delta = \{\square\} \quad \pi(\blacksquare) = \pi(\square) = \square$$

$$\Theta = \left\{ \begin{array}{|c|c|} \hline \# \# \\ \hline \# \blacksquare \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# \# \\ \hline \square \square \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# \# \\ \hline \square \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \# \square \\ \hline \# \square \\ \hline \end{array}, \begin{array}{|c|c|} \hline \blacksquare \square \\ \hline \square \blacksquare \\ \hline \end{array}, \begin{array}{|c|c|} \hline \square \square \\ \hline \blacksquare \square \\ \hline \end{array}, \begin{array}{|c|c|} \hline \square \blacksquare \\ \hline \square \square \\ \hline \end{array}, \begin{array}{|c|c|} \hline \blacksquare \# \\ \hline \# \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \square \square \\ \hline \# \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \square \# \\ \hline \square \# \\ \hline \end{array}, \begin{array}{|c|c|} \hline \square \square \\ \hline \square \square \\ \hline \end{array} \right\}.$$

Reconnaissabilité des collages

En général, $\text{Collage}(P)$ n'est pas reconnaissable même si P l'est.



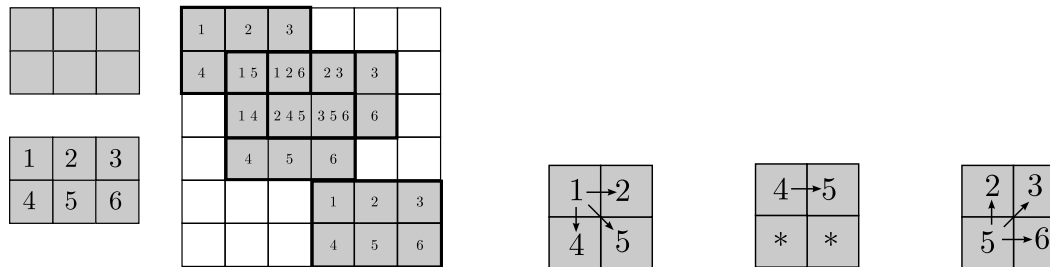
Théorème 5 (Choffrut & Durak, 2005). Si $P = \left\{ \begin{array}{|c|c|c|} \hline b & a & e \\ \hline \end{array} \right\}, \left. \begin{array}{|c|} \hline b \\ \hline a \\ \hline e. \\ \hline \end{array} \right\}$ alors

$\text{Collage}(P)$ n'est pas reconnaissable.

Condition de reconnaissabilité : monochromaticité

Lorsque P est “monochrome” (avec P quelconque, y compris indécidable), $\text{Collage}(P)$ est un langage reconnaissable à 2 couleurs (lemme de Dickson et OU logique pixel-à-pixel).

Reconnaissance des collages d'un rectangle 2×3 par un système de pavage étiquetant chaque pixel avec des ensembles d'hypothèses :



(a) Étiquetage

(b) Propagation

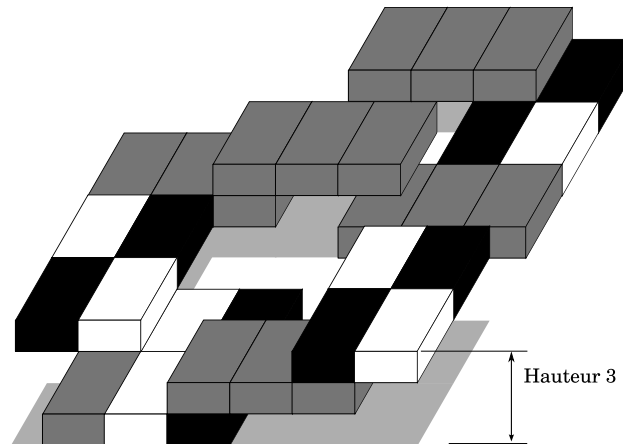
Autres conditions de reconnaissabilité

Dimension unitaire

Si $P \subset \Sigma^{1 \times *}$, c'est-à-dire si ce sont des lignes, alors $\text{Collage}(P)$ est constitué des images dont les lignes sont des collages de $P \cup \{\square\}$, et est donc reconnaissable.

Hauteur bornées

Si P est reconnaissable, mais qu'on ne considère que les collages de **hauteur** $\leq k$, alors cet ensemble est aussi reconnaissable.



Remerciements

- Christian Choffrut
- Giovanni Pighizzini
- Jean Berstel
- Les examinateurs :
 - Géraud Sénizergues
 - Sylvain Lombardy
- Roberto Di Cosmo & Xavier Leroy
- Les contribuables (allocation de recherche)
- Marie !

N'oubliez pas... le pot dans le sous-marin !