Automates Finis

Dominique Perrin

15 septembre 1999

Abstract

We present some recent applications using finite automata in the fields of text compression, coding, natural language processing and games.

1 Introduction

La théorie des automates a gardé depuis ses débuts un double aspect de théorie mathématique et de catalogue de méthodes pour divers problèmes d'informatique comme l'analyse lexicale, le traitement de textes, ou la spécification de processus. On pourra trouver en [9] une présentation à caractère historique de tous ces aspects remontant la chaîne des événements depuis les années 40 et en [8] une présentation générale des aspects contemporains.

Le texte ci-dessous contient quelques exemples de la variété des domaines d'applications des automates finis allant du traitement du langage naturel à la théorie des jeux. Il s'agit d'un choix personnel de quelques sujets parmi beaucoup d'autres possibles.

Du point de vue mathématique, la théorie des automates finis continue et, à mon avis, continuera à susciter des recherches intéressantes. Je pense que ceci est lié à la conjonction de plusieurs facteurs favorables parmi lesquels je citerai :

- L'existence de nombreuses conjectures non encore résolues ainsi que la solution de certaines d'entre elles dans le passé récent.
- Les liens avec d'autres théories mathématiques dont la dynamique symbolique, la théorie des nombres ou la combinatoire algébrique.
- L'existence enfin d'une grande variété d'applications, dont ce texte donne un aperçu.

2 Compression de textes

La transmission (ou l'archivage) de données utilise de façon maintenant très fréquente des méthodes de compression. Elles permettent de gagner un facteur important dans le débit des transmissions. Les méthodes les plus courantes pour la compression de textes sont le *codage de Huffman* (logiciels `compact` et `pack`) ou le codage de Ziv-Lempel (logiciels `compress` et `gzip`). Elles sont essentiellement basées sur l'utilisation d'automates finis transformant un texte en un texte plus court sans perte d'information.

Une nouvelle méthode basée elle aussi sur les automates a été proposée récemment par M. Crochemore, F. Mignosi, A. Restivo et S. Salemi [4]. Son principe est d'utiliser un *antidictionnaire*, c'est à dire une liste des mots qui n'apparaissent pas dans le texte. La méthode suppose un alphabet binaire, disons $\{a, b\}$. L'idée est très simple et consiste à exploiter systématiquement l'information fournie par l'antidictionnaire. Si, par exemple, l'antidictionnaire est $\{bb, aaa, babab\}$, on aura le codage

a	b	a	a	b	a	b	a	a	b	a	a	b	a	b	a	a	b	a	b	a
a	b		a			b					a			b					b	

de telle façon que le texte binaire original de longueur 21 est codé par le couple $(abababb, 21)$ (on doit garder la longueur de la source). La compression fonctionne ainsi : on part d'une lettre $x = a$ ou b du texte source et on continue tant qu'il n'y a qu'une possibilité pour la lettre suivante, compte tenu des mots interdits. On code tout le bloc par sa première lettre. On recommence à la lettre suivante. Le décodage utilise le même principe à l'envers. Le codage et le décodage peuvent être décrits par un transducteur comme sur la Figure 1.

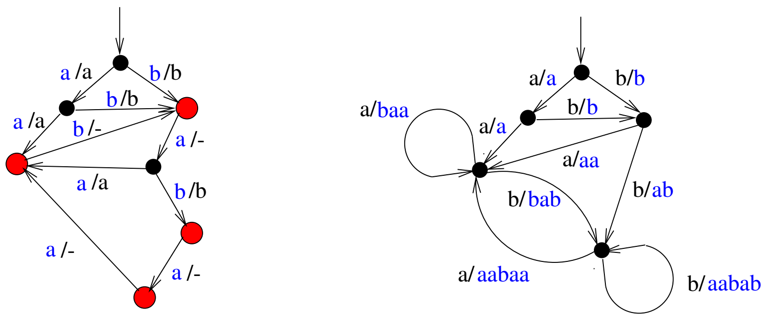

Figure 1: Codeur et décodeur

La méthode est asymptotiquement optimale en taux de compression. L'un de ses avantages par rapport aux autres méthodes connues est le fait que le codage est une fonction *locale*, i.e. sans propagation des erreurs. Cela permet de faire de la recherche de motifs directement sur le texte compressé.

3 Codage

Les systèmes de codage basés sur des automates finis sont nombreux dans le domaine du codage pour canaux contraints. Le livre de Marie-Pierre Béal présente nombre de ces méthodes ainsi que des exemples de codes utilisés dans divers contextes (dont le codage pour l'écriture sur support magnétique) [2].

Un résultat récent dans ce domaine est la réalisation d'une version sur automate fini du théorème de Kraft-McMillan. Ce dernier affirme que l'on peut réaliser un codage sur un alphabet a k lettres de mots ayant une distribution de longueurs $u = (u_n)_{n\geq 1}$ ssi la suite u satisfait l'inégalité bien connue

$$\sum_{n\geq 1} u_n k^{-n} \leq 1$$

c'est à dire encore si la série $u(z) = \sum_{n\geq 1} u_n z^n$ vérifie $u(1/2) \leq 1$. On montre dans [1] que l'on peut de plus réaliser le codage par un automate fini ssi la suite u est la distribution de longueurs d'un langage reconnaissable par automate fini. La preuve utilise une nouvelle notion appelée *automate des super-états*.

On trouvera sur la Figure 3 un exemple d'application de cette méthode. On part d'un automate repr'esenté à gauche et dont la distribution de longueurs a pour série $u(z) = 3z^2/(1-z^2)$ de sorte que $u(1/2) = 1$. On obtient d'abord l'arbre infini indiqué ́droite qui donne le code préfixe sur un alphabet $X = (aa)^*(ab + ba + bb)$.

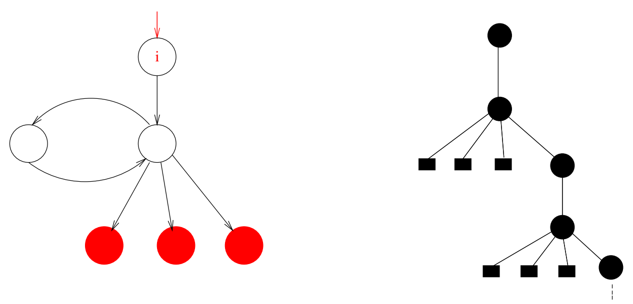

Figure 2: Codage par un code préfixe binaire.

Ce problème est lié au codage pour canaux contraints et, notamment au théorème de codage d'Adler, Hassner et Coppersmith (voir [6]).

4 Langage naturel

Les automates finis fournissent un cadre pour la description de nombreux phénomènes liés au langage naturel, aux niveaux phonétique, lexical ou même syntaxique. Un ouvrage récent, édité par Emmanuel Roche et Yves Schabes [10] donne un panorama de ce point de vue. De nouvelles applications des automates au traitement de langues naturelles sont motivées par l'amélioration des moteurs de recherche sur internet.

L'un des chapitres de [10] décrit le logiciel INTEX réalisé par Max Silberztein [11]. Il constitue une implémentation d'un grand nombre de méthodes d'analyse de textes basées sur des automates finis. Il utilise le système de dictionnaires du LADL qui répertorie dans plusieurs langues de façon exhaustive les formes de base ou fléchies de mots simples ou composés (voir [5]). Il contient des outils permettant d'utiliser ou d'éditer

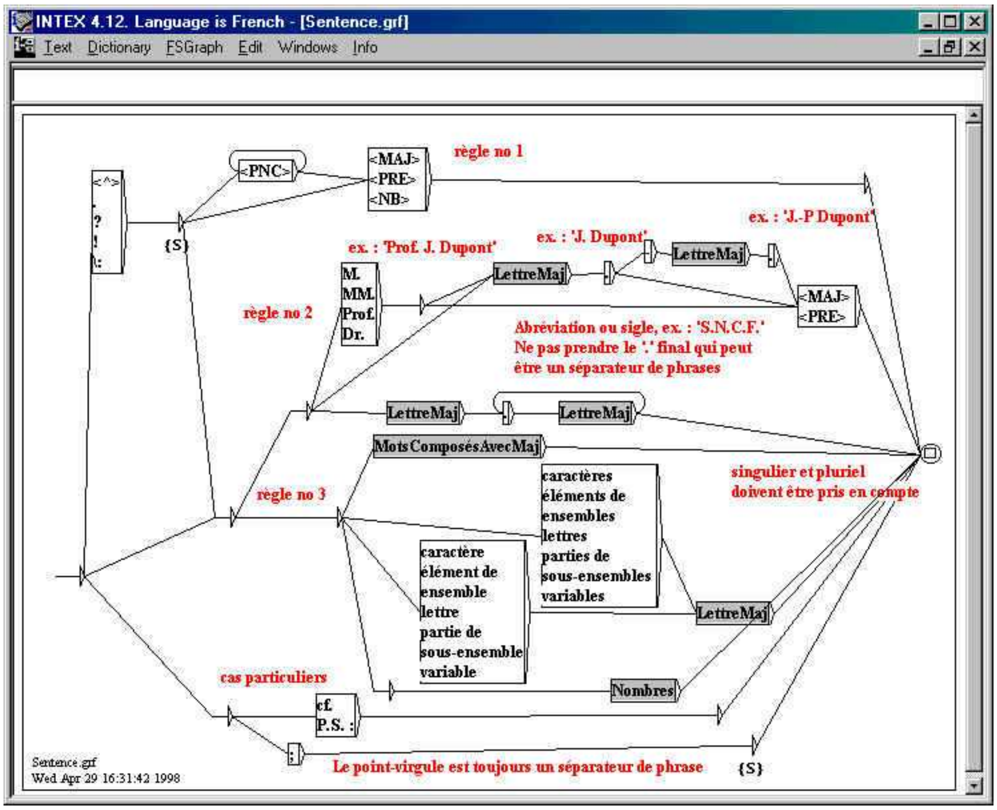

Figure 3: La fenêtre INTEX

un grand nombre de grammaires comme celle qui apparaît sur la Figure 4 et qui décrit les limites de phrases du point de vue de la forme de l'alphabet et de la ponctuation.

Les fonctions ainsi réalisées sont très nombreuses, depuis l'analyse syntaxique jusqu'à la recherche documentaire. Le logiciel est notamment utilisé pour l'enseignement des langues. Il sert aussi à l'indexation de corpus et a permis récemment d'indexer complètement l'ensemble des résumés d'articles constituant la base de données du NIH (de l'ordre du Go).

Un autre domaine d'application des automates finis est celui de la phonétique. Mehryar Mohri, Fernando Pereira et Michael Riley ont mis au point une librairie d'automates décrivant par des transducteurs, comme sur la Figure 4, un vocabulaire phonétisé [7].

5 Jeux et automates

Un des domaines très actifs de la recherche sur les automates durant les dernières années a été celui des jeux. Un compte-rendu assez complet a été réalisé par Wolfgang Thomas [12]. Les jeux considérés sont des jeux

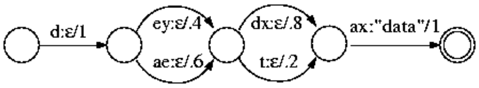

Figure 4: le mot *data*

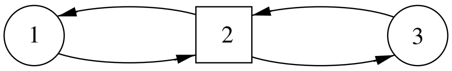

Figure 5: Jeu sur un graphe

à deux joueurs à information parfaite (ne faisant donc pas intervenir le hasard). Un grand nombre de ces jeux, comprenant bien entendu le célèbre jeu de Nim, sont décrits dans le livre de Guy, Conway et Berlekamp ([3]).

Des problèmes de décision très difficiles apparaissent lorsque l'on considère des jeux infinis (Conway distingue les jeux *indéfinis* des jeux *éternels*). Pourtant ces jeux apparaissent naturellement dans les modèles de comportement de processus. On définit dans ces jeux les parties gagnantes en spécifiant l'ensemble des états apparaîssant infiniment souvent. Cela permet d'exprimer des conditions sur les processus dits d'équité (`fairness`) : si l'évènement U se produit infiniment souvent, alors l'évènement V doit aussi se produire infiniment souvent.

Un exemple simple de tels jeux apparaît sur la Figure 5. Les deux joueurs jouent sur le graphe en choisissant successivement les sommets d'un chemin. La position de départ est le sommet 2. Le joueur I commence en choisissant le sommet 1 ou le sommet 3. Le joueur II n'a aucun choix puisqu'il doit revenir au sommet 2 et la même situation est reproduite. Le joueur I gagne si le chemin passe infiniment souvent par le sommet 1 et par le sommet 3. Ainsi, le joueur I gagne toujours à condition de ne pas toujours jouer le même sommet au bout d'un certain temps.

Le résultat le plus marquant de ce point de vue est le théorème de Gurevitch et Harrington qui dit essentiellement que pour un jeu joué sur un graphe fini, l'un des deux joueurs a une stratégie gagnante ne nécessitant qu'une mémoire finie, i.e. calculable par un automate fini.

Dans l'exemple précédent, le joueur I a une telle stratégie gagnante consistant à jouer une fois sur deux les sommets 1 ou 3.

Une série de travaux, dûs en particulier à R. McNaughton, W. Zielonka et N. Klarlund ont permis de dégager l'importance de la notion de *chaîne alternante*. Il s'agit d'ensembles de la forme

$$X = X_1 - X_2 + X_3 - \ldots$$

où les X_n forment une chaîne décroissante $X_1 \supset X_2 \supset X_3 \supset \ldots$. On a ainsi montré en particulier qu'un jeu admet pour l'un des joueurs une stratégie gagnante sans mémoire ssi l'ensemble des parties gagnantes pour lui peut être décrit par une chaîne alternante d'ensembles simples. Dans l'exemple précédent, le joueur I n'a pas de stratégie gagnante sans mémoire.

Ces progrès ont permis de mieux comprendre de nombreux problèmes difficiles, y compris le théorème de Rabin sur la décidabilité de la logique monadique du second ordre des deux successeurs (voir [12]).

References

[1] Frédérique Bassino, Marie-Pierre Béal, and Dominique Perrin. A finite-state version of the Kraft-McMillan theorem. 1999. (`www-igm.univ-mlv.fr/ beal/Recherche/Publications`).

[2] Marie-Pierre Béal. *Codage Symbolique.* Masson, 1993.

[3] Elwin R. Berlekamp, John H. Conway, and Richard K. Guy. *Winning Ways.* Academic Press, 1982.

[4] Maxime Crochemore, Filippo Mignosi, Antonio Restivo, and Sergio Salemi. Data compression using antidictionnaries. Technical Report IGM-98-10, Université de Marne la Vallée, 1998. (`www-igm.univ-mlv.fr/ mac/DCA.html`).

[5] Maurice Gross and Dominique Perrin, editors. *Electronic Dictionaries and Automata in Computational Linguistics.* Lecture Notes in Computer Science, 377. Springer, 1989.

[6] Brian H. Marcus, Ron M. Roth, and Paul H. Siegel. Constrained systems and coding for recording channels. In *Handbook of coding theory*, volume 2, pages 1635–1764. North Holland, 1998.

[7] Mehriar Mohri. Finite state transducers in language and speech processing. *Computational linguistics*, 23, 1997. (`www.research.att.com/sw/tools/fsm`).

[8] Dominique Perrin. Finite automata. In Jan Van Leeuwen, editor, *Handbook of Theoretical Computer Science*, pages 1–53. North Holland, 1990.

[9] Dominique Perrin. Les débuts de la théorie des automates. *Technique et Science Informatique*, 14:409–4033, 1995.

[10] Emmanuel Roche and Yves Schabes. *Finite-State Language Processing.* MIT Press, 1997.

[11] Max Silberztein. *Dictionnaires électroniques et analyse automatique de textes : le système INTEX.* Masson, 1993. (`www.ladl.jussieu.fr`).

[12] Wolfgang Thomas. Languages, automata and logic. In Gregorz Rosenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, pages 389–449. Springer-Verlag, 1997.