



L'air de rien N° 30

Les dernières nouvelles du LRDE

L'aléastriel du Laboratoire de Recherche et de Développement de l'EPITA¹

Numéro 30, Mars 2014

Édito

par *Olivier Ricou*

C'est le printemps et nous avons plein de choses à vous dire. Tellement qu'on va bientôt vous proposer un autre numéro.

L'événement important de ce début d'année est l'admission du LRDE à l'école doctorale informatique, télécommunications et électronique (EDITE) de Paris². C'est l'école doctorale de nos confrères de Paris VI, Télécom, le CNAM et l'INRIA entre autres. Cette admission nous permet désormais d'encadrer seuls nos doctorants. Elle nous permet également d'avoir des bourses CIFRE, bourses financées par les industriels dans le cadre d'un partenariat avec le laboratoire encadrant.

A propos de doctorants : Yongchao a soutenu sa thèse en décembre 2013, il est maintenant maître de conférence au LRDE. Le résumé de sa thèse se trouve dans la rubrique consacrée aux dernières publications. Nicolas Boutry a fini son stage et commence sa thèse avec Théo sur l'arbre des formes (une belle publication est déjà en cours!).

Autres évolutions : Myriam a été recrutée à la fin de son CDD. Son travail sur la reconnaissance de texte dans les vidéos donne des résultats intéressants (ainsi qu'une soumission d'article). Enfin, Luca Saiu et Alfred M. Szmidt ont rejoint l'équipe Vaucanson³ afin de travailler sur Vaucanson 2, projet ANR qu'Akim décrit en détail dans ce numéro.

Vaucanson 2, joindre les trois bouts

par *Akim Demaille*

Vaucanson est une plate-forme libre (GNU GPLv3+) dédiée aux automates et aux expressions rationnelles. Elle tente de réunir trois qualités qui peuvent facilement être contradictoires : pédagogie, généralité et efficacité.

Pédagogie

Les expressions rationnelles — bien connues aussi sous le nom d'expressions régulières, ou *regex* — fournissent un moyen simple, rigoureux et concis de définir un *langage* : un ensemble de mots. Par exemple⁴ $(a+b)^*a(a+b)(a+b)$ représente l'ensemble des mots qui commencent par un nombre quelconque de a ou de b ($(a+b)^*$ suivi d'un a puis de deux lettres à choisir entre a et b — autre-

ment dit les mots dont l'antépénultième lettre est un a . Voici comment définir cette expression rationnelle dans une session interactive de Vaucanson, ainsi que la réponse du système :

```
In [4]: r = b.ratexp('(a+b)*a(a+b)(a+b)'); r
```

```
Out[4]: (a+b)*a(a+b)(a+b)
```

Les expressions rationnelles sont extrêmement utilisées pour définir les langages de programmation, pour la fouille de texte par mots-clés, la recherche de motifs dans les brins d'ADN, la détection de traces d'intrusion ou d'événements anormaux, pour des *chercher-remplacer* puissants, etc.

Mais définir ne suffit pas, il faut pouvoir utiliser ! Les expressions rationnelles tirent leur succès de leurs indissociables compagnons calculatoires :

1. L'air de rien, <http://publis.lrde.epita.fr/LrdeBulletin>.

2. EDITE de Paris, <https://edite-de-paris.fr>.

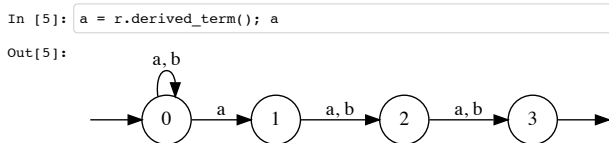
3. Ils partagent leur temps entre Télécom et le LRDE

4. On aurait pu écrire $[ab]^*a[ab]\{2\}$, plus familier aux greppeurs.

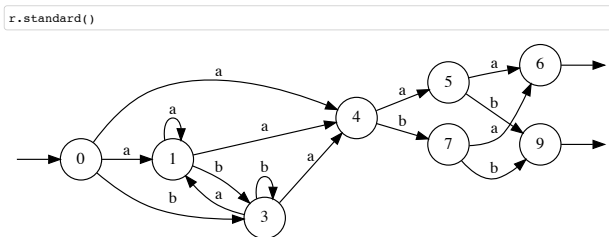
les automates. Ces derniers modélisent d'une façon remarquablement élémentaire une certaine famille de machines, quelque soit leur technologie : qu'il s'agisse d'une calculatrice électronique, d'une montre mécanique, d'engrenages mus par une force hydraulique, tous se représenteront par le fait que cette *machine* a un certain nombre d'*états* possibles (valeur affichée par la calculatrice, position des engrenages...) et qu'elle réagit aux événements (pression d'une touche, tic d'horloge...) en effectuant des *transitions*, c'est-à-dire qu'elle bascule d'un état à un autre.

L'idée clef qui permet aux automates d'unifier tous ces dispositifs sous une seule représentation est la métaphore du plan : les états sont des carrefours, les transitions des rues nommées, l'ensemble ayant des entrées et des sorties. Un mot est dit *accepté* s'il existe un chemin d'une entrée vers une sortie étiqueté par les lettres du mot.

Il est tout à fait remarquable qu'à partir de toute expression rationnelle on peut construire des automates équivalents, et réciproquement ! On se convainc facilement que l'automate suivant correspond bien à $(a + b)^*a(a + b)(a + b)$:



mais il en existe bien d'autres, moins jolis :



On peut aussi demander au système les premiers mots acceptés :

```
a.shortest(8)
```

$aaa + aab + aba + abb + aaaa + aaab + aaba + aabb$

ou au contraire les premiers rejetés :

```
a.determinize().complete().complement().shortest(8)
```

$\epsilon + a + b + aa + ab + ba + bb + baa$

ou encore ceux acceptés à l'endroit et à l'envers :

```
(a.transpose() & a).shortest(8)
```

$aaa + aba + aaaa + aaab + baaa + baab + aaaaa + aaaaab$

Vaucanson offre aux enseignants et aux étudiants un moyen simple, interactif, d'explorer la *théorie des langages rationnels*, nom donné à l'étude des auto-

mates et des expressions rationnelles.

Généralité

Mais il existe bien d'autres types d'automates ! En particulier chez les linguistes la notion de langage est plus floue (« après qu'il soit parti », est-ce vraiment du français ?) ce qui les conduit à donner un *score* à un énoncé, plutôt qu'un brutal correct/incorrect. Par ailleurs, ils s'intéressent à l'association entre mots, parfois de natures différentes (par exemple traduire la transcription de la voix sous forme de phonèmes en véritables mots). Et pourquoi ne pas faire tout cela en même temps ?

Vaucanson a l'ambition de supporter ces généralisations des automates et des expressions rationnelles. À titre d'exemple, l'expression rationnelle suivante n'accepte que les mots commençant par des *a* puis des *b*, et compte d'une part le nombre de *a* initiaux, et d'autre part le nombre de *b* finaux. En comparant les deux poids obtenus (ils doivent être égaux : (0, 0), (1, 1), etc.) on peut ainsi « reconnaître » $a^n b^n$, un langage bien connu des étudiants de THL/THLR :

```
r = zmin.ratexp('(<(1,0)>a)*(<(0,1)>b)*')
r.standard().shortest(6)
```

$\langle(0,0)\rangle\epsilon + \langle(1,0)\rangle a + \langle(0,1)\rangle b + \langle(2,0)\rangle aa + \langle(1,1)\rangle ab + \langle(0,2)\rangle bb$

Efficacité

Vaucanson offre aujourd'hui une interface graphique qui fournit un très grand confort d'utilisation, comparable à celui d'outils tels que Mathematica, Maple, SciLab, etc. Mais cette souplesse ne doit pas cacher la vraie nature du cœur de Vaucanson : une bibliothèque écrite en C++11, avec cette utilisation massive du mot-clé `template` qui est la patte du LRDE. Ce qui nous permet d'obtenir des performances comparables, parfois meilleures, aux outils les plus aboutis du domaine.

Vaucanson Reboot

Réécriture totale du projet initié autrefois par Jacques Sakarovitch et Sylvain Lombardy, auquel s'est ensuite joint le LRDE, Vaucanson 2 offre performance et généricité, rigueur et confort, en reposant sur différentes couches. Tout en bas règne le C++, statique et compilé, au-dessus duquel de nouvelles interfaces ajoutent confort et inter-activité, jusqu'au monde Python, dynamique et interprété. Comme on peut s'y attendre, la couche en charge de dynamiser ce qui est statique doit résoudre un certain nombre de problèmes, y compris la génération et compilation de code à la volée !

Utiliser Vaucanson n'a jamais été aussi simple ! Pour le télécharger, rendez-vous sur sa page⁵.

5. Projet Vaucanson, <https://vaucanson.lrde.epita.fr>.

Mini-bios

Luca Saiu



Luca Saiu, programmeur expérimenté et spécialiste de langages de programmation, a obtenu son master à l'Université de Pise et son doctorat à l'Université Paris 13. Dans sa thèse il a spécifié formellement et développé GNU epsilon⁶, un langage de programmation extensible, basé sur un noyau minimaliste. C'est resté son projet le plus ambitieux. Il a ensuite travaillé à l'Inria sur le support multi-cœur d'OCaml. Luca Saiu est également co-auteur de Marionnet, un simulateur de réseaux en OCaml, utilisé pour la pédagogie en France et à l'étranger. Il a rejoint l'équipe Vaucanson comme chercheur post-doctoral. Jusqu'à présent il a réalisé les algorithmes de minimisation, la normalisation de la numérotation des états et la refonte des expressions rationnelles.

Alfred M. Szmidt



Alfred M. Szmidt qui nous vient de la Suède, est un développeur de logiciels et un militant pour les logiciels libres. Il a travaillé activement sur le projet GNU⁷ depuis 1990, et en est un mainteneur depuis 1997. Entre 2007 et 2009, il a travaillé comme administrateur système pour Klarna AB⁸. En 2010, il a commencé à travailler en tant que consultant chez Purple Scout AB⁹ dans le secteur des télécommunications. Et depuis 2014, il collabore avec le LaBRI, l'ENST, et le LRDE sur le projet Vaucanson 2 pour développer une interface utilisateur graphique pour la manipulation d'automates pondérés.

En bref

Publications (disponibles sur publis.lrde.epita.fr)

XU, Y.. *Tree-based shape spaces : Definition and applications in image processing and computer vision*. PhD thesis, Université Paris-Est, Marne-la-Vallée, France

Le cadre classique des filtres connexes consiste à enlever d'un graphe certaines de ses composantes connexes. Pour appliquer ces filtres, il est souvent utile de transformer une image en un arbre de composantes, et on élague cet arbre pour simplifier l'image de départ. Les arbres ainsi formés ont des propriétés remarquables pour la vision par ordinateur. Une première illustration de leur intérêt est la définition d'un détecteur de zones d'intérêt, vraiment invariant aux changements de contraste, qui nous permet d'obtenir des résultats à l'état de l'art en recalage d'images et en reconstruction 3D à base d'images. Poursuivant dans l'utilisation de ces arbres, nous proposons d'élargir le cadre des filtres connexes. Pour cela, nous introduisons la notion d'espaces des formes basés sur des arbres : au lieu de filtrer des composantes connexes du graphe correspondant à l'image, nous proposons de filtrer des composantes connexes du graphe donné par l'arbre des composantes de l'image.

Ce cadre général, que nous appelons morphologie basée sur les formes, peut être utilisé pour la détection et la segmentation d'objets, l'obtention de segmentations hiérarchiques, et le filtrage d'images. De nombreuses applications et illustrations montrent l'intérêt de ce cadre.

LAZZARA, G., GÉRAUD, T., AND LEVILLAIN, R.. *Planting, growing and pruning trees: Connected filters applied to document image analysis*. In *Proceedings of the 11th IAPR International Workshop on Document Analysis Systems (DAS)*, Tours, France. IAPR

Cet article présente les filtres connexes, une branche de la morphologie mathématique moderne, et leur application à l'analyse d'images de documents. Une implémentation classique, efficace et simple de ces filtres repose sur une représentation arborescente du contenu d'une image. Un tel arbre peut être modifié selon divers critères pour simplifier, binariser, nettoyer, segmenter une image ou encore y reconnaître des formes. Nous montrons que les filtres connexes, peu connus et peu utilisés pour traiter des documents numérisés ou numériques dès le début (*born digital*), sont une

6. GNU epsilon, <http://www.gnu.org/projects/epsilon>.

7. Projet GNU, <http://www.gnu.org/>.

8. Klarna AB, <https://klarna.com/>.

9. Purple Scout AB, <http://purplescout.com/>.

piste très prometteuse dans ce domaine du traitement des images.

RENAULT, E., DURET-LUTZ, A., KORDON, F., AND POITRENAUD, D.. Three SCC-based emptiness checks for generalized Büchi automata. In McMillan, K., Middeldorp, A., and Voronkov, A., editors, *Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'13)*, volume 8312 of *Lecture Notes in Computer Science*, pages 668–682. Springer

Cet article passe en revue les algorithmes d'énumération des composantes fortement connexes au sein d'un graphe. Ces algorithmes sont essentiels à l'approche par automates du *model checking* puisqu'ils permettent de trouver des exécutions fautives rapidement. Pour détecter ces exécutions il est nécessaire de transformer ces algorithmes en *emptiness checks*. Cet article présente les différentes façons de le faire et suggère des optimisations et des pistes prometteuses pour utiliser ces algorithmes dans un contexte multi-coeurs.

BEN SALEM, A. E., DURET-LUTZ, A., KORDON, F., AND THIERRY-MIEG, Y.. Symbolic Model Checking of Stutter Invariant Properties Using Generalized Testing Automata. In *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, *Lecture Notes in Computer Science*, page to be published, Grenoble, France. Springer

Une propriété est dite *insensible au bégaiement* lorsque les répétitions des différentes configurations dans un comportement du modèle ne peuvent pas influencer sur la validité de celle-ci. Le cas est assez fréquent pour avoir conduit à la définition d'*automates testeurs*, spécialisés pour exprimer ce type de propriété : les transitions de ces automates observent les changements de variables et non leurs valeurs. Dans cet article nous étudions l'utilisation de tels automates dans l'*approche symbolique* du *model checking*, où les automates représentant le modèle et la propriété sont représentés par des diagrammes de décision binaires.

DURET-LUTZ, A.. LTL translation improvements in Spot 1.0. *International Journal on Critical Computer-Based Systems*, 5(1/2) :31–54

C'est une version étendue de l'article présenté à la conférence VECoS'11. Cette nouvelle version, publiée en mars 2014 dans le journal IJCCBS, donne plus de détails sur le module de traduction de formules de logique temporelle à temps linéaire (LTL) en automates de Büchi qu'offre notre bibliothèque Spot. Nous montrons qu'il rivalise avec d'autres outils de traduction, et expliquons comme il s'est amélioré au fil des ans.

Les logiciels **Spot 1.2** (2013-10-01) ajoute le support des automates de Rabin et Streett, des algorithmes de détermination d'automates de Büchi, ainsi que des algorithmes de minimisation d'automates de Büchi déterministes passant par l'utilisation d'un SAT-solver. Cette version a été suivie de trois versions mineures pour corriger de menus problèmes (**Spot 1.2.1** le 2013-12-11, **Spot 1.2.2** le 2014-01-24, et **Spot 1.2.3** le 2014-02-11).

Séminaire MeFoSyLoMa au LRDE Le prochain séminaire consacré aux Méthodes Formelles pour les Systèmes Logiciels et Matériels aura lieu au LRDE le 28 mars. Les laboratoires Cedric (Cnam), IBISC (Univ. Evry), LACL (Univ. Paris 12), LIP6 (UPMC), LIPN (Univ. Paris 13), LSV (École Normale Supérieure de Cachan), LTCI (TELECOM ParisTech) et le LRDE se réunissent tous les 2 mois pour exposer leurs travaux de recherche récents sur l'utilisation de méthodes formelles et pour les confronter à d'autres approches en invitant des chercheurs extérieurs. Au programme de la prochaine séance¹⁰ : **Laura Carnevali - Université de Florence**, « Quantitative evaluation of non-Markovian models through the approach of stochastic state classes : applications and future issues » et **Ridha Benosman - CNAM**, « Conception et Évaluation de Performances d'un Bus Applicatif Parallèle et Orienté Service ».

Contacter le LRDE

18, rue Pasteur
Paritalie, bâtiment X, aile Mistral
2e étage, droite droite Contact :
info@lrde.epita.fr
Site Web : <http://www.lrde.epita.fr>



10. MeFoSyLoMa, <http://www.mefosyloma.fr/j2014-03-28.html>.