C'est le cas de ceux qui classent des pages Web.

Classement de pages Web

par *Johan Oudinet* ⏲ *24 mai, 14h00, amphi P004*

Les moteurs de recherche sont devenus les carrefours incontournables de l'Internet. Cette révolution culturelle pose deux nouvelles problématiques : d'un côté, les moteurs de recherche doivent trouver les pages qui satisferont l'internaute à partir seulement de quelques mots clefs ; et de l'autre, les créateurs de sites Web se livrent à une lutte acharnée pour améliorer leur classement et faire croître le nombre de leurs visiteurs.

Pour caractériser la pertinence d'une page en fonction d'une requête, on mesure différents critères comme le nombre d'occurrences de chaque mot contenu dans la requête, leurs emplacements, ou encore leurs proximités. En ce qui concerne la popularité d'une page, on étudie les liens de chaque page en partant du principe que plus une page est populaire, plus elle reçoit de liens. L'algorithme le plus connu et le plus utilisé pour évaluer la popularité d'une page est celui du PageRank.

Au cours de ce séminaire, nous expliquerons le fonctionnement du PageRank, et nous introduirons des algorithmes plus performants et peut-être même plus efficaces.

OLENA

OLENA[10] est une bibliothèque de traitement d'images générique et performante. Un algorithme est écrit une fois et peut s'appliquer sur tout type d'images (binaire, en couleur, 2D, 3D, etc). OLENA sert également de cadre aux recherches sur le traitement d'images.

Taxonomie des images dans Olena

par *Christophe Berger* ⏲ *24 mai, 14h30, amphi P004*

Actuellement un nouveau paradigme de programmation est adopté pour OLENA, ce qui est l'occasion d'étudier la modélisation actuelle et de l'améliorer avant de passer à la phase de programmation. La taxonomie des types d'images est la classification de ces types et leur organisation.

Ce paradigme, *SCOOP 2* (qui fait suite à SCOOP[11]) fonctionne sur le principe de l'héritage par propriétés. Nous disposons de hiérarchies abstraites qui définissent les types d'images, et nous ne souhaitons pas spécifier explicitement l'héritage, mais plutôt que ceci soit fait implicitement grâce aux propriétés.

C'est dans ce contexte que nous allons insérer certains types de base qui permettront de proposer toutes les propriétés fondamentales autorisant, selon les combinaisons, à répondre à tous les besoins des utilisateurs. Nous présenterons ces types, leurs propriétés et les outils, appelés « morpheurs », permettant d'étendre ces types d'images.

Segmentation temps réel

par *Nicolas Widynski* ⏲ *24 mai, 15h00, amphi P004*

La quantité d'information dans une image naturelle (paysage, visage, etc.) est telle qu'il est difficile d'en extraire automatiquement les objets.

En guise de prétraitement, la segmentation a pour but de simplifier, partitionner et d'extraire les éléments caractéristiques d'une image. Le domaine d'application est vaste : médical (extraction de tumeur, segmentation du cerveau, ...), reconnaissance humaine (main, iris, ...), suivi d'objets (bien souvent de personnes). Dans la littérature, il existe beaucoup de méthodes vouées à la segmentation d'images. Quelques-unes, dédiées le plus souvent au suivi d'objets dans les séquences d'images, fonctionnent en temps réel.

Nous nous proposons de comparer deux approches. L'une d'elles est basée sur la reconnaissance et l'évolution des contours (*watershed*, *watersnake*, *snake*). La seconde manipule les objets de l'image, en extrayant ses caractéristiques (opérateurs connectés).

En bref

Les publications (disponibles sur publis.lrde.epita.fr)

- On a Polynomial Vector Field Model for Shape Representation par Mickael Chekroun, Jérôme Darbon et Igor Ciril accepté à *International Conference on Image Analysis and Recognition (ICIAR)*.
- How to make LISP go faster than C – Tuning LISP for performance par Didier Verna accepté à *International MultiConference of Engineers and Computer Scientists (IMECS)*.

[10] OLENA, `http://olena.lrde.epita.fr`.

[11] Nicolas Burrus, Alexandre Duret-Lutz, Thierry Géraud, David Lesage and Raphaël Poss, *A Static C++ Object-Oriented Programming (SCOOP) Paradigm Mixing Benefits of Traditional OOP and Generic Programming*, `http://publis.lrde.epita.fr/200310-MPOOL`.

L'air de rien N° 4

Séminaires CSI de mai

L'aléastriel du **L**aboratoire de **R**echerche et de **D**éveloppement de l'**E**PITA[1] Numéro 4.1, Mai 2006

Edito

par *Roland Levillain*

Dans ce numéro et le suivant, les étudiants CSI présentent leurs séminaires, dans lesquels ils exposent leurs travaux. Les deux premiers auront lieu les 17 mai[2] et 24 mai[3] 2006 à l'EPITA. Au programme, des exposés sur le projet Vaucanson et les automates finis, les avancées sur la transformation de programmes et le projet Transformers, les techniques de classement de pages Web, et Olena et le traitement des images. Les séminaires suivants auront lieu les mercredi après-midi 7 et 24 juin 2006. Venez nombreux !

VAUCANSON

Le projet VAUCANSON [4] s'attache à développer une bibliothèque générique de manipulation d'automates. L'équipe d'étudiants, intéressée aussi bien par la théorie mathématique des automates que par les enjeux des implémentations, vous propose ses présentations cuvée 2006.

Remodélisation de VAUCANSON

par *Robert Bigaignon* ⏲ *17 mai, 14h00, amphi P004*

Le but de VAUCANSON est de permettre la manipulation efficace de n'importe quel type d'automate tout en restant fidèle au cadre algébrique établi par la théorie des automates. Ainsi les algorithmes généraux que nous fournissons peuvent aussi bien traiter des automates à multiplicité dans un semi-anneau numérique que tropical – souvent au prix d'une écriture lourde.

Aujourd'hui nous envisageons la refonte du cœur de la bibliothèque afin de tirer parti des dernières avancées notamment en termes de programmation C++. A l'instar du projet OLENA, nous souhaitons utiliser dans VAUCANSON des paradigmes de programmation statiques évolués, permettant d'exprimer nos abstractions tout en gardant de bonnes performances de calcul.

Ainsi nous présenterons lors de cet exposé une comparaison de techniques de modélisations appliquées à un sous-ensemble représentatif de VAUCANSON.

On m'a dit que 275 604 541 était premier

Les derniers seront les premiers (modulo *n*)

par *Michaël Cadilhac* ⏲ *17 mai, 14h30, amphi P004*

Si vous cherchez des nombres premiers ou que vous voulez prouver leur réelle primalité, vous avez à disposition des vingtaines de théorèmes et autres algorithmes. Votre choix dépendra de deux critères : la taille des nombres et la probabilité désirée qu'ils soient réellement premiers.

Du crible d'Ératosthène aux algorithmes qui ne requièrent pas même de factoriser, nous verrons des méthodes plus ou moins artisanales, plus ou moins mathématiques pour montrer la primalité. Nous aboutiront au fameux algorithme qui prouve l'appartenance de ce test à la classe de complexité

[1] L'air de rien, `http://publis.lrde.epita.fr/LrdeBulletin`.

[2] Séminaire CSI du 17 mai 2006, `http://publis.lrde.epita.fr/Seminar-2006-05-17`.

[3] Séminaire CSI du 24 mai 2006, `http://publis.lrde.epita.fr/Seminar-2006-05-24`.

[4] VAUCANSON, `http://vaucanson.lrde.epita.fr`.

[5] Agrawal, 2002, `http://www.cse.iitk.ac.in/users/manindra/primality_v6.pdf`.

$\mathcal{P}$(Agrawal, 2002)[5].

Durant cette présentation, des tests probabilistes aussi bien que déterministes seront revus. Nous étudierons en particulier ceux qui ont permis de vérifier la primalité des plus grands nombres premiers connus. Nous aurons l'occasion de faire des comparatifs de performance de certains de ces algorithmes codés en LISP, dans lesquels nous pourrons constater que les plus simples sont souvent les plus efficaces pour des nombres de moins de trente chiffres.

Extension du format XML

par *Florent Terrones* ⏲ *17 mai, 15h00, amphi P004*

Proposé lors des différentes conférences CIAA (*Conference on Implementation and Application of Automata*), le format XML de description d'automates présenté par l'équipe Vaucanson a pour but de permettre et faciliter le transfert des informations d'un automate entre les différents logiciels qui les manipulent.

On peut en effet charger en mémoire un automate stocké dans un fichier XML et le modifier dans VAUCANSON, ou encore sauvegarder un automate en format XML. Le but est à terme de proposer un format universel qui permette à quiconque d'effectuer des travaux sur des automates de taille conséquente, sur différents logiciels, sans problème de format.

Mais un détail gêne encore la parfaite portabilité de ce format : les étiquettes des transitions sont pour l'instant de simples chaînes de caractères. Elles sont donc dépendantes des syntaxes utilisées par chaque logiciel, à l'instar de VAUCANSON.

Le but de ce séminaire est de faire le point sur les propriétés que cette extension doit satisfaire, puis d'écrire cette dernière. Enfin, certaines parties du contenu de VAUCANSON seront modifiées afin de supporter ces changements.

Automates et performance

par *Guillaume Lazzara* ⏲ *24 mai, 15h45, amphi P004*

Une implémentation naïve, trop proche de la théorie, est rarement performante. Optimiser les implémentation des automates finis et leurs algorithmes associés est une réelle nécessité.

Pour utiliser des automates, il est nécessaire de pouvoir représenter des états ainsi que des transitions orientées. C'est donc tout naturellement les graphes qui sont les plus adaptés. Quelle implémentation de graphes choisir ? Deux candidats se distinguent : les matrices d'adjacence et les listes d'adjacence. Il a été décidé d'implémenter nous-mêmes ces structures afin d'en comparer les performances et ne retenir que la meilleure. Boost[6], proposant une implémentation de graphes, a également été retenue pour les tests.

Les tests se basent essentiellement sur deux algorithmes de la théorie des automates : la déterminisation et la minimisation. Ces algorithmes sont intéressants pour leur complexité aussi bien du point de vue du temps de calcul que de la place en mémoire.

Resynchronisation des transducteurs

par *Guillaume Leroi* ⏲ *24 mai, 16h15, amphi P004*

Les transducteurs sont des automates qui permettent de réaliser des relations rationnelles. Certaines de ces relations sont calculables par des transducteurs « lettre à lettre », qui sont des automates dont les transitions sont étiquetées par des couples de lettres. Il existe des transducteurs dont les transitions sont étiquetées par des couples de mots. Cependant, sur ce type d'automates nous ne disposons pas des outils habituels comme l'intersection ou la composition.

C'est pourquoi nous désirons resynchroniser ces transducteurs et obtenir des transducteurs « lettre à lettre » quand cela est possible. Un second avantage des transducteurs « lettre à lettre » est qu'ils peuvent être déterminisés (et donc minimisés). Ceci permettrait l'utilisation d'algorithmes plus efficaces pour le calcul des relations que ces transducteurs réalisent. Nous présenterons donc deux algorithmes, l'un permettant de déterminer si un transducteur est synchronisable, l'autre synchronisant effectivement notre transducteur.

Algorithme de fermeture d'un automate

par *Matthieu Varin* ⏲ *24 mai, 16h45, amphi P004*

Les -transitions (également connues sous le nom de *transitions spontanées*) sont très utiles dans le cadre de la manipulation d'automates. Rappelons qu'une -transition d'un état s_1 vers un état s_2 représente le fait que s_1 possède les mêmes propriétés que s_2.

En pratique, ceci pose beaucoup de problèmes. Par exemple, on ne peut pas évaluer un automate sur $\mathbb{Z}$ contenant des -transitions. D'une manière plus générale, beaucoup d'algorithmes fonctionnent uniquement sur des automates déterministes. Or, par définition, ne pas contenir d' -transitions est une condition nécessaire pour qu'un automate soit déterministe. Nous cherchons donc à obtenir ce que l'on appelle la *fermeture de l'automate*, nécessaire, entre autres, pour sa déterminisation.

Une implémentation de la fermeture existe déjà dans VAUCANSON. Cette implémentation est basée sur la méthode décrite par Jacques SAKAROVITCH[7]. Ce séminaire étudie une autre fermeture, plus optimisée, mais faisant apparaître des problèmes dans le cas d'automates sur $\mathbb{Z}$.

[6] Boost, `http://www.boost.org/`.

[7] Jacques SAKAROVITCH, *Éléments de théorie des automates*, Vuibert, 2003, `http://www.infres.enst.fr/~jsaka/ETA/eta.html`.

TRANSFORMERS

TRANSFORMERS [8] est une plateforme de transformation de programmes dédiée au C et au C++. Le projet repose sur Stratego/XT[9] et sur de nombreux outils développés en interne, tels qu'un moteur de grammaire attribuée utilisé lors de la phase de désambiguïsation.

Preprocessing & Unpreprocessing du C et C++

par *Thomas Largillier* ⏲ *17 mai, 15h45, amphi P004*

La transformation de source à source se doit d'être la plus fidèle possible. Ainsi tout prétraitement fait sur le fichier reçu doit pouvoir être inversé. Le premier traitement fait sur les sources C & C++ est évidemment le preprocessing. Bien qu'il existe de nombreux logiciels réalisant ce travail, aucun d'entre eux ne permet de faire le travail inverse. Pour rendre un fichier le plus semblable possible à celui fourni par l'utilisateur, il est impératif de pouvoir inverser cette passe de preprocessing. Pour pouvoir « dé-preprocesser » un fichier C ou C++, il faut donc laisser de l'information dans le fichier lors du preprocessing pour retrouver les endroits préalablement modifiés.

Les problèmes rencontrés lors du développement des deux outils vous seront exposés, ainsi qu'une implémentation réalisée à l'aide des outils présents dans la plate-forme Stratego/XT.

Grammaires hors-contexte et désambiguïsation

par *Renaud Durlin* ⏲ *17 mai, 16h15, amphi P004*

Autrefois rejetées au profit des classes LL ou LR, les grammaires hors-contexte générales sont de plus en plus utilisées car ces technologies permettent d'analyser des langages réels en utilisant un formalisme simple et naturel. De manière générale, les grammaires hors-contexte permettent de spécifier des langages ambigus.

En utilisant de telles grammaires, un analyseur syntaxique généralisé produit non pas un seul arbre de *parse* mais une forêt (un arbre par interprétation possible). La désambiguïsation consiste alors à analyser cette forêt pour obtenir l'unique arbre correspondant à l'entrée en prenant en considération les règles sémantiques contextuelles.

Cette présentation décrira trois méthodes différentes pour effectuer la désambiguïsation : la réécriture de termes guidée par des spécifications algébriques (ASF+SDF), ou en utilisant des stratégies (Stratego/XT), ou alors en utilisant le formalisme des grammaires attribuées (TRANSFORMERS). Cette comparaison sera faite en utilisant une grammaire simplifiée volontairement ambiguë. Nous discuterons des points forts et des points faibles de chacune de ces approches.

Vectorisation automatique grâce à la transformation de programme

par *Alexandre Borghi* ⏲ *17 mai, 16h45, amphi P004*

La vectorisation est née dans les années 70 avec les super-ordinateurs dédiés aux calculs scientifiques. Aujourd'hui, la majorité des processeurs sur le marché permet de tirer partie de la vectorisation. Cependant, les langages et le code existants n'y sont pas particulièrement adaptés. Les derniers compilateurs essaient d'auto-vectoriser mais les résultats sont rarement satisfaisants.

Le C étant intrinsèquement séquentiel, il exprime un grand nombre de dépendances qui n'ont pas lieu d'être. Ceci rend difficile la vectorisation automatique. Les outils d'auto-vectorisation doivent en particulier considérer les boucles pour en extraire un maximum d'information suivant des analyses de dépendances parfois très élaborées.

Il s'agit ici de transformer un programme C pour le rendre plus facilement vectorisable par les derniers compilateurs à l'aide de TRANSFORMERS et de ses outils de transformation de programmes.

MARKOV

Les modèles probabilistes sont très en vogue dans le milieu de la recherche et de l'industrie. Beaucoup d'algorithmes s'appuient sur les hypothèses de Markov.

[8] TRANSFORMERS, `http://transformers.lrde.epita.fr`.

[9] Stratego/XT, `http://www.stratego-language.org`.