Implémentation d'algorithmes standards sur les graphes

Olivier Michel

LaMI[1], Équipe SPÉCIF, UMR 8042 CNRS, Université d'Évry val d'Essonne, GENOPOLE
Tour Évry-2, 523 Place des terrasses de l'agora, 91000 Évry Cedex

2 décembre 2003

Nombre d'étudiants : 1 binôme

Mots-clés : graphe, algorithme sur les graphes.
Public visé : stage IIE, TER de maîtrise, stage Polytechnique.

Contexte de l'étude

Le projet `MGS` développe un langage de programmation original dédié à la modélisation et la simulation de processus biologiques à structure dynamique. Pour ce faire, `MGS` permet la représentation d'organisations complexes entre des entités variables et hétérogènes, ainsi que leur transformation par des règles locales. Ces travaux trouvent leurs inspirations dans les travaux de J. Von Neuman sur les automates cellulaires, A. Lindenmayer sur les L systèmes, G. Paun sur les P systèmes, G. Berry *et al.* sur la CHAM et la réécriture de multi-ensembles.

La structure de données fondamentale en `MGS` est la *collection topologique*. Une collection topologique est un ensemble d'éléments organisés par une relation de voisinage. Une *transformation* permet de spécifier de nouvelles fonctions sur les collections par des cas filtrant des *sous-collections*. Ces notions permettent d'unifier dans le même cadre formel les différents modèles de calculs cités plus haut. Pour chacun des modèles il suffit de choisir le bon voisinage pour la collection utilisée. Un point remarquable est l'existence d'un langage de filtres, utilisé pour écrire les règles d'une transformation, qui est commun à tous les types de collection. Ce langage de filtres se fonde sur la notion de voisinage et de chemin.

Sujet du stage

Les graphes sont sans doute le type de collection topologique le plus naturel. Un graphe en `MGS` correspond à un graphe orienté dont les sommets portent des valeurs. Les transformations en `MGS` permettent d'écrire de manière très concise certains algorithmes sur les graphes. Par exemple la recherche H d'un chemin hamiltonien (un chemin qui passe une fois et une seule par tous les sommets du graphe) s'écrit simplement par la transformation suivante :

$$\texttt{transformation}\ H(g) = \{\ x* / \texttt{size}(x*) = \texttt{size}(g)\ \Longrightarrow\ \texttt{Print}(x*)\ \}$$

Cette transformation ne contient qu'une seule règle. L'expression à gauche du signe $\Rightarrow$ est un filtre qui sélectionne un chemin $x*$ (i.e. : une suite de sommets adjacents) dont la longeur est égale au nombre de sommets du graphe g (qui est argument de la transformation H).

La simplicité de programmation en `MGS` de cet algorithme repose sur la possibilité de spécifier des chemins vérifiant des propriétés grâce à des filtres. **L'objectif de ce stage** est d'étoffer l'implémentation de la structure de données des graphes en intégrant une librairie implémentant les fonctions classiques de la théorie des graphes (à la manière du *package* `Combinatorica` pour `Mathematica`[2]). Il faudra implémenter par exemple les algorithmes classiques de plus court chemin, recherche en largeur/profondeur, problèmes de flots, etc.

---

[1] *Contacts :* par courier électronique : `michel @ReMoVeMeFIRST.lami.univ-evry.fr`. Des informations supplémentaires sont disponibles à partir de la page : `http://mgs.lami.univ-evry.fr`

[2] `http://www.cs.sunysb.edu/~algorith/implement/combinatorica/implement.shtml`