Traitement de graphes PROLOG

Mohammed EL KHAIRA

2018 / 2019

Table des matières

Introduction

Notre objectif est de concevoir un logiciel de manipulation de graphes étiquetés non orientés. Il permettra d'entrer des requêtes sous forme de motifs et de construire les sous-graphes associés. De plus, le logiciel permettra de construire les abstractions de ces sous-graphes, associées à différentes propriétés d'abstraction.

Pour cela, nous allons utiliser le langage de programmation Prolog. Plus précisément, nous avons écrit notre code en utilisant l'implémentation libre SWI-Prolog.

Les résultats seront stockés dans un fichier .dot. Ce rapport contient donc une partie expérimentale illustrant les possibilités de notre logiciel.

Les fonctionnalités attendues sont :

1. Pouvoir répondre à des requêtes de la forme : "Quel est le sous-graphe GS induit par les sommets de G dont les étiquettes contiennent un motif donné M" ?
2. Pouvoir construire l'abstraction GSP d'un sous-graphe GS de G, associée à une propriété basée sur les sommets.

Parmi les abstractions possibles, nous avons implémenté :

- $deg(x) \geq k$,
- $star(x) \geq k$: x a un degré au moins k OU est voisin d'un sommet de degré au moins k.

I. Présentation

Tout d'abord, nous avons décidé de diviser notre travail de la manière suivante :

- **dotToList.pl** : prédicats pour lire les fichiers dot,
- **motif.pl** : prédicats pour extraire le sous-graphe correspondant à un motif donné,
- **degre.pl** : prédicats pour extraire le sous-graphe correspondant à l'abstraction deg(x) ≥ k,
- **star.pl** : prédicats pour extraire le sous-graphe correspondant à l'abstraction star(x) ≥ k,
- **listToDot.pl** : prédicats pour écrire les fichiers dot,
- **test.pl** : prédicats pour tester le logiciel,
- **final.pl** : prédicats graphe_induit() et point_fixe().

Cela facilitera grandement la lecture et la mise à jour du code. Les fichiers **dotToList.pl** (initialement **es.pl**) et **test.pl** nous ont été fournis. Nous avons cependant réaliser quelques modifications sur le fichier **es.pl**.

En effet, nous avons modifié le prédicat **read_dot_file(NomF, G)** en ajoutant 'G' afin que nous puissions récupérer le graphe lu. Aussi, nous avons ajouté une variable dynamique **listeMotifs** et l'instruction **assertz(listeMotifs(L))** (ligne 48). Cela nous permet de sauvegarder le titre du graphe dans la base de données : lors de l'écriture des sous-graphes, récupérer le titre sera alors facile.

Enfin, le fichier **final.pl** est le fichier qui doit être chargé dans SWI-Prolog. En effet, il inclut tous les autres fichiers grâce au prédicat **:-include.**

II. Explications

Tous les prédicats de nos fichiers source sont commentés : les variables qu'ils utilisent sont ainsi détaillés, tout comme le mode d'appel. Par exemple :

/* get_motif_sommet(+M, +LS, -Res)
M : liste représentant le motif à trouver
LS : liste des sommets de départ
Res : liste de tous les sommets qui correspondent avec le motif M */

motif.pl :

Ce fichier contient les prédicats suivant :
- motif_sommet()
- get_motif_sommet()
- get_motif_sommet_nom()
- liste_arete()
- get_liste_arete()

Les prédicats essentiels sont **get_motif_sommet()** et **get_liste_arete()**. En effet, ceux sont eux qui vont permettre de récupérer la liste des sommets et des arêtes du sous-graphe, grâce à **findall**. Cependant, **get_liste_arete()** a besoin de connaître le nom des sommets dont elle doit conserver les arêtes : c'est pourquoi nous avons implémenter **get_motif_sommet_nom()** qui fait la même chose que **get_motif_sommet()** mais seulement pour les noms des sommets.

Aussi, il est important de noter que pour conserver les arêtes qui nous intéressent, nous commençons par récupérer la liste des arêtes contenant un sommet indésirable (grâce à **findall** et **liste_arete()**), puis nous retranchons cette liste à la liste d'arêtes de départ : cela explique les **length(I, 0) ; length(II, 0)** dans le **if** de **liste_arete()** (intersection vide = sommet indésirable).

Enfin, nous pouvons dire que notre logique est simple. D'abord nous écrivons un prédicat permettant de récupérer les sommets ou les arêtes qui nous intéressent. Ce prédicat utilise un **accumulateur** afin d'obtenir la réponse par **backtrack**. Ensuite, nous écrivons un prédicat utilisant **findall** et le prédicat précédent : de ce fait, la réponse est obtenue directement dans une liste et non plus par **backtrack**.

degre.pl :

Ce fichier contient les prédicats suivant :
- degre()
- degre_sommet()
- get_degre_sommet()
- get_degre_sommet_nom()
- get_degre()

La logique est exactement la même que pour **motif.pl**. Cependant, nous utilisons le prédicat degre() comme intermédiaire à degre_sommet : il permet de calculer le degré d'un sommet. Aussi, nous réutilisons **get_liste_arete()** pour récupérer les bonnes arêtes.

Enfin, le prédicat get_degre() se comporte comme un algorithme de point fixe : il permet d'obtenir l'abstraction désirée.

star.pl :

Ce fichier contient les prédicats suivant :
- voisin_star()
- get_voisin_star()
- is_star()
- star_sommet
- get_star_sommet()
- get_star_sommet_nom()
- get_star()

De même que précédemment. En revanche, cette fois ci nous avons besoin de plus de prédicat intermédiaire. En effet, pour savoir si un sommet fait partie d'une étoile, nous devons d'abord récupérer ses voisins qui font parties d'une étoile.

listToDot.pl :

Ce fichier contient les prédicats suivant :
- write_sommet()
- write_arete()
- write_dot_file()

Ces prédicats sont relativement simples à écrire. Le plus difficile a été de récupérer le titre du graphe : grâce à la BDD, cela a pu être fait. La mise en forme du titre n'est pas exactement la même que l'originale (il y a des "" en plus) mais cela ne pose aucun problème.

final.pl :

Ce fichier contient les prédicats suivant :
- graphe_induit()
- point_fixe()

Ces deux prédicats sont ceux permettant de réaliser le travail demandé. Ils appellent les prédicats **get** des autres fichiers.

III. Tests

Tous les prédicats de nos fichiers source sont accompagnés d'un exemple de test que nous avons réalisé. Par exemple :

/* get_motif_sommet([rock, blues], [sommet(a,[rock,folk,jazz]),sommet(b, [rock,folk,blues,jazz]),sommet(c,[rock,folk,jazz]),sommet(d,[rock,folk,jazz]),sommet(e, [rock,folk,blues]),sommet(f,[folk,blues]),sommet(g,[rock,folk,jazz]),sommet(h, [rock,folk]),sommet(i,[folk,blues]),sommet(j,[blues,jazz]),sommet(k,[rock]),sommet(l, [jazz,pop]),sommet(m,[rock,folk]),sommet(n,[rock,folk,pop,blues]),sommet(o, [rock,folk,pop]),sommet(p,[rock,folk,pop,jazz]),sommet(q,[rock,folk,pop]),sommet(r, [rock,folk,pop,blues])], Res), writeln(''), writeln(Res). */

Cela nous a permis de s'assurer que tous nos prédicats fonctionnaient individuellement. Ensuite, nous avons fait les tests fournis dans le fichier **test.pl** :

- test(1, 'mougel_bis.dot', 'test1.dot').
- test(2, 'mougel_bis.dot', 'test2.dot').
- test('s50_an1_redDevoir.dot', 'test4.dot', [tobacco_occasional_regular], deg(2)).
- test('s50_an1_redDevoir.dot', 'test5.dot', [tobacco_Non], star(7)).

Les résultats sont dans le dossiers **tests** ; tout semble bien fonctionner.

Cependant, les tests suivant n'ont pas fonctionné :

- test('brauer_elati_3_7Devoir.dot', 'test.dot', [S_1_2], deg(1)).
- test('galbrun_dblpE2_wegonet_GeorgeKarypis2Devoir.dot', 'test4.dot', [multithread], deg(1)).

Nous pensons que cela est dû au fait que les noms des sommets sont en majuscule pour le fichier 'brauer_elati_3_7Devoir.dot' : SWI-Prolog les traitent donc comme des variables.

Pour 'galbrun_dblpE2_wegonet_GeorgeKarypis2Devoir.dot', le **read_dot_file()** échoue :

```
[?- read_dot_file('galbrun_dblpE2_wegonet_GeorgeKarypis2Devoir.dot', G).
false.
```