Compilation – TP 7 : Coloriage de graphe

Université Paris Diderot – Master 1

(2016-2017)

L'objectif de ce TP est d'implémenter l'algorithme de coloriage de graphes.

Si vous n'avez pas fini le TP précédent, vous pouvez tout de même travailler sur ce TP car l'algorithme de coloriage de graphe est générique et testable indépendamment du reste du compilateur. Pour cela, il faut avoir compilé le compilateur avec la commande :

```
# make byte
```

Puis à la racine du projet, il faut lancer :

```
# ./flap.top
```

et vous obtiendrez un *toplevel* OCaml avec tous les modules de `flap` chargés par défaut. Dès lors, vous pourrez lancer les fonctions de tests directement depuis ce *toplevel* :

```
# GraphColoring.test ();;
```

Pour visualiser les graphes, il faut installer le paquet `graphviz` sur votre machine.

1 Coloriage simple

Pour le moment, on prend uniquement en compte les relations de conflit (en ignorant les relations de préférence). De plus, le graphe ne contient pour le moment que des nœuds variables et aucun registre physique. La section suivante introduira le raffinement de l'algorithme permettant de fusionner les nœuds et de partir d'un graphe précolorié par des registres physiques.

Exercice 1 *Coloriage de graphe (à la main)*

1. *Rappelez-vous l'algorithme de coloration de graphe vu en cours en coloriant manuellement le graphe suivant à l'aide de 2 couleurs :*

   ```
   a - b, a - c, a - d
   b - c, b - d
   c - d
   ```

2. *Écrire l'algorithme sur papier en pseudo-code.*

□

Exercice 2 *Implémentation du coloriage simple de graphe*

1. *Pourquoi peut-on dire que la structure de données de graphe est persistente (ou, dit autrement, purement fonctionnelle) ? En quoi est-ce intéressant pour notre algorithme de coloriage de graphe ?*
2. *Observez la fonction* `GraphColoring.test`*. Comment s'assurer que l'on ne teste que des graphes sans relation de préférences ?*
3. *Qu'est-ce qu'un coloriage correct ?*
4. *Implémentez la fonction* `check_coloring`*.*
5. *Implémentez la fonction* `pick`*.*
6. *Implémentez la fonction* `colorize`*.*

□

2 Coloriage avec fusion des nœuds

L'objectif de cette section est d'implémenter une nouvelle version de l'algorithme de coloriage de graphe qui prend en compte les relations de préférences.

On rappelle les deux heuristiques permettant de décider de la fusion de deux nœuds :

Briggs Si en fusionnant a et b dans G, le nombre de nœuds qui sont des voisins du nouveau nœud et qui ne sont pas simplifiables est strictement inférieur au nombre de couleurs disponibles.

George Si les nœuds non simplifiables et en conflit avec a sont des nœuds qui étaient aussi en conflit avec b (ou inversement) alors on peut fusionner a et b.

Exercice 3

1. *Rappelez en pseudo-code l'algorithme de coloriage de graphe avec relation de préférence.*
2. *Exécutez l'algorithme de coloriage sur le graphe suivant :*

   ```
   a - b, a - c, a - h
   b - c, b = h
   c = d, c - e
   d - f
   e = f, e - i, e - j, e - k, e - l
   i - j, i - k, i - l
   j - k, j - l
   k - l
   ```

   En supposant que les tirets sont des relations de conflits, que les égalités sont des relations de préférence et qu'il y a 3 couleurs. Vous traiterez les relations de préférences dans l'ordre suivant : `c = d`, `e = f` *puis* `h = b`.
3. *Dans l'algorithme de coloriage raffiné, à quoi peut servir le constructeur* `PreferenceNodes` *de données du type* `pick_result` *? Mettez à jour la fonction* `pick` *pour qu'elle produise ce nouveau type de résultat.*
4. *Implémentez les deux heuristiques en complétant les fonctions* `briggs` *et* `george`.
5. *Mettez à jour l'algorithme de coloriage en complétant le cas manquant de* `colorize`.

□