Université de Montréal

IFT2105-A2016

Devoir #2

Philippe CARON

Travail remis à l'intention de
Alain TAPP

13 décembre 2016

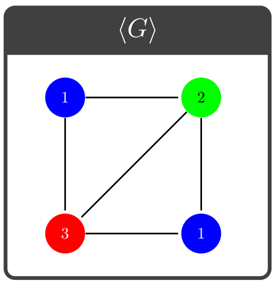
Table des matières

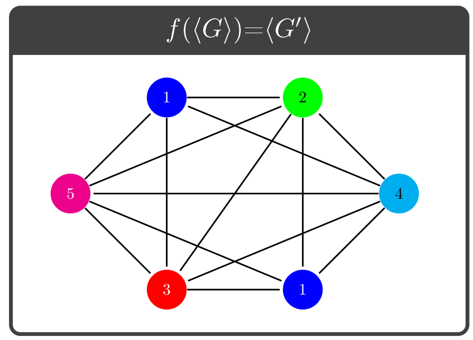
Question 1

On sait que *3-COL* est **NP**-complet. Afin de prouver que *5-COL* l'est aussi, nous allons prouver que *3-COL* peut être réduit à *5-COL* en un temps polynomial, donc que *3-COL* $\leq_{\mathbf{P}}$ *5-COL*.

1.1 Transformation

Afin de transformer un graphe dans *3-COL* en graphe dans *5-COL* équivalent, on ajoute tout simplement deux nouveaux sommets connectés ensemble et à tous les sommets du graphe original. Il est évident que l'un des sommets ajoutés prendra une couleur des 5 disponibles, l'autre en prendra une seconde, et il restera 3 couleurs pour colorier le graphe originial. Par exemple :





Cette transformation se fait de toute évidence en temps polynomial ; si ajouter une arrête se fait en temps constant et il y a n sommets, alors ajouter les 2 nouveaux sommets prend un temps $2n$.

1.2 Fonction de réduction

Soit la fonction f telle que définie en 1.1 avec les propriétés suivantes :

$$f : \Sigma^* \to \Sigma^*$$
$$f(\langle G \rangle) = \langle G' \rangle$$

On veut prouver que

$$\langle G \rangle \in \textit{3-COL} \Leftrightarrow f(\langle G \rangle) = \langle G' \rangle \in \textit{5-COL}$$

a) $\langle G \rangle \in \textit{3-COL} \Rightarrow f(\langle G \rangle) = \langle G' \rangle \in \textit{5-COL}$

Il est facile de se convaincre de cette implication ; si un graphe est 3-coloriable et qu'on lui ajoute 2 sommets, alors il est forcément (sans même regarder plus loin) 5-coloriable.

b) $\langle G \rangle \in \textit{3-COL} \Leftarrow f(\langle G \rangle) = \langle G' \rangle \in \textit{5-COL}$

De manière générale, si un sommet s est connecté à tous les autres d'un graphe $\langle G \rangle \in$ *k-COL*, alors il «réserve» une couleur. Le problème devient équivalent à calculer $\langle G$ sans le sommet $s \rangle \in$ *(k-1)-COL*. Ici deux sommets répondent à ce critère, donc cette implication est correcte.

Preuve de la question 1. Puisque :

$$\langle G \rangle \in \textit{3-COL} \Leftrightarrow f(\langle G \rangle) = \langle G' \rangle \in \textit{5-COL}$$

avec f en temps polynomial, et que *3-COL* $\in$ **NP**-complet alors :

$$\textit{3-COL} \leq_{\mathbf{P}} \textit{5-COL}$$

et *5-COL* $\in$ **NP**-complet □

Question 2

$$L = \{\langle C_1, C_2 \rangle \mid \text{les circuit } C_1 \text{ et } C_2 \text{ calculent une fonction différente}\}$$

[*] On sait que le langage *SAT-C* est **NP**-complet. Ce langage comprends les circuits booléens satisfaisable. Afin de démontrer que $L \in$ **NP**-complet, on veut prouver que $SAT\text{-}C \leq_{\mathbf{P}} L$.

Pour commencer, nous commencerons par prouver que pour que C_1 et C_2 ne calculent pas la même fonction, il suffit de trouver une affectation des variable c, appliquée sur C_1 et C_2 indépendament, telle que le résultat de C_1 n'égale pas celui de C_2.

2.1 Transformation

Afin de réduire les deux circuit à un seul circuit C_X, on cherche à être sûr que les deux réponses sont différentes, ce qui revient à obtenir le ou exclusif des deux réponse, donc :

$$\begin{aligned} C_X &= C_1 \oplus C_2 \\ &= C_1 \wedge \neg C_2 \end{aligned}$$

La transformation qu'on veut effectuer ici est la transformation inverse. Intuitivement, on choisira un des circuits C comme étant satisfaisable ($C \in SAT\text{-}C$). Ceci nous conduira à prendre comme circuit opposé un circuit toujours faux.

2.2 Fonction de réduction

À partir ce la, on peut explorer une définition de f. Soit C_f un circuit qui retourne toujours faux, on défini f comme suit :

$$f(\langle C \rangle) = \langle C, C_f \rangle$$

a) $\langle C \rangle \in SAT\text{-}C \Rightarrow f(\langle C \rangle) = \langle C, C_f \rangle \in L$

Si C est satisfaisable, alors C peut retourner vrai. Donc C ne calcule pas la même chose que C_f et $\langle C, C_f \rangle \in L$.

b) $\langle C \rangle \in SAT\text{-}C \Leftarrow f(\langle C \rangle) = \langle C, C_f \rangle \in L$

Si C ne calcule par la même chose que C_f, cela signifie que C peut ne pas être faux, donc peut-être vrai, donc satisfaisable.

Preuve de la question 2. Puisque :

$$\langle C \rangle \in SAT\text{-}C \Leftrightarrow f(\langle C \rangle) = \langle C, C_f \rangle \in L$$

qu'il est trivial de constater que f est exécutable en temps polynomial,
et que $SAT\text{-}C \in$ **NP**-complet alors :

$$SAT\text{-}C \leq_{\mathbf{P}} L$$

et donc L est **NP**-complet. □

*. En assumant que C_1 et C_2 sont des circuits booléens

Question 3

Soit *k-CLIQUE* la réponse à l'existence d'un sous-graphe complet de taille k d'un graphe G. On stipule que trouver cette réponse peut-être fait en temps polynomial, autrement dit que :

$$\forall k \in \mathbb{N}, k\text{-}CLIQUE = \{\langle G \rangle \mid \text{il existe un sous-graphe complet de } G \text{ de taille } k\} \in \mathbf{P}$$

3.1 Marche à suivre

Pour se faire on examine tous les ensembles de k sommets du graphe. En choisissant un sommet au hasard, puis un des sommets restant, puis encore et ainsi de suite jusqu'à obtenir un ensemble de k sommet. Une fois l'ensemble obtenu, on peut facilement tester en temps linéaire $O(k)$ si cet ensemble est une clique ou non. Ceci nous donne que pour un graphe de n sommets, le nombre d'ensemble à examiner correspond à trouver une permutation de la clique. (Même si cet algorithme n'est pas aussi efficace qu'avec une combinaison, il est beaucoup plus facile à implémenter). On trouve ainsi $t(n, k)$, soit la complexité temporelle de *k-CLIQUE* donnée par la formule suivante :

$$t(n, k) = \frac{n!}{(n-k)!} \cdot k = n \cdot (n-1) \cdot (n-2) \cdot ... \cdot (n-k+2) \cdot (n-k+1) \cdot k$$

3.2 Conclusion

Preuve de la question 3. On remarque que les termes à droite de l'équation sont tous des polynômes de degré 1, et qu'ils sont au nombre de k, on peut donc dire que

$$t(n, k) = P_k \in O(n^k)$$

où P_k est un polynôme de degré k.

Puisque $t(n, k) \in O(n^k)$, on peut dire que pour tout k fixe, *k-CLIQUE* est possible à trouver en temps polynomial et donc que

$$k\text{-}CLIQUE \in \mathbf{P}$$

. □

Question 4

4.1 Par réduction

Soit

$$L = \{\langle M \rangle \mid M \text{ est une MT telle que } L(M) \text{ est hors contexte}\}$$

Montrons que $A_{MT} \leq L$.
Soit la fonction calculable

$$f : \Sigma^* \to \Sigma^*$$

telle que

— si y n'est pas de la forme $\langle M, \omega \rangle$, alors $f(y) = \epsilon$;
— si $y = \langle M, \omega \rangle$, alors $f(y) = \langle M' \rangle$ où M' est la MT suivante :

`Prendre un mot` x `en entrée;`
`simuler` M `sur` ω`;`
`si` M `accepte` ω `alors`
`si` x `est de la forme` uv^ixy^iz`, alors accepter, sinon rejeter.`

On a :

$$\begin{aligned}
\langle M, \omega \rangle \in A_{MT} &\Rightarrow M \text{ accepte } \omega \\
&\Rightarrow L(M') = \{uv^ixy^iz | n \geq 0\} \subset GHC \\
&\Rightarrow \langle M' \rangle \in L \\
&\Rightarrow f(\langle M, \omega \rangle) \in L
\end{aligned}$$

$$\begin{aligned}
\langle M, \omega \rangle \notin A_{MT} &\Rightarrow M \text{ rejette ou boucle sur } \omega \\
&\Rightarrow L(M') = \emptyset \\
&\Rightarrow \langle M' \rangle \notin L \\
&\Rightarrow f(\langle M, \omega \rangle) \notin L
\end{aligned}$$

Donc L est indécidable

4.2 Par Rice

— il existe une MT M^* telle que $\langle M^* \rangle \in L$
— il existe une MT $M^\dagger$ telle que $\langle M^\dagger \rangle \notin L$

Soit M^* la MT qui engendre $L^* = \{uv^nxy^nz | n \in \mathbb{N}\}$
et $M^\dagger$ la MT qui engendre $L^\dagger = \{u^nv^nx^ny^nz^n | n \in \mathbb{N}\}$
On voit tout de suite que ces deux machine respectent le théorème et donc que L est indécidable.

Question 5

5.1 Par réduction

Soit

$$L = \{\langle M \rangle \mid M \text{ est une MT telle que } |L(M)| \text{ est paire}\}$$

Montrons que $A_{MT} \leq L$.
Soit la fonction calculable

$$f : \Sigma^* \to \Sigma^*$$

telle que
- si y n'est pas de la forme $\langle M, \omega \rangle$, alors $f(y) = \epsilon$;
- si $y = \langle M, \omega \rangle$, alors $f(y) = \langle M' \rangle$ où M' est la MT suivante :

  `Prendre un mot` x `en entrée;`
  `si` $x = 0$ `accepter;`
  `simuler` M `sur` ω`;`
  `si` M `accepte` ω `alors`
  `si` $x = 1$`, alors accepter, sinon rejeter.`

On a :

$$\begin{aligned} \langle M, \omega \rangle \in A_{MT} &\Rightarrow M \text{ accepte } \omega \\ &\Rightarrow |L(M')| = 2 \in \\ &\Rightarrow \langle M' \rangle \in L \\ &\Rightarrow f(\langle M, \omega \rangle) \in L \end{aligned}$$

$$\begin{aligned} \langle M, \omega \rangle \notin A_{MT} &\Rightarrow M \text{ rejette ou boucle sur } \omega \\ &\Rightarrow |L(M')| = 1 \\ &\Rightarrow \langle M' \rangle \notin L \\ &\Rightarrow f(\langle M, \omega \rangle) \notin L \end{aligned}$$

Donc L est indécidable

5.2 Par Rice

- il existe une MT M^* telle que $\langle M^* \rangle \in L$
- il existe une MT $M^\dagger$ telle que $\langle M^\dagger \rangle \notin L$

Soit M^* la MT qui engendre $L^* = \{a, b\}$
et $M^\dagger$ la MT qui engendre $L^\dagger = \{a\}$
On voit tout de suite que ces deux machine respectent le théorème et donc que L est indécidable.

Question 6

6.1 Énumérateur $\Leftarrow$ Reconnaissable

Si un langage est reconnaissable, alors il a tous ses éléments calculables par une machine de Turing. Il est donc possible pour une machine de calculer chacun des mots et de les imprimer successivement. Pour se faire, une MT peu boucler sur plusieurs «paliers». À chaque palier de k étape, la MT teste tous les mots possibles plus petits que k. Si la MT s'arrête à la k-ième étape, alors l'énumérateur imprime le mot, sinon, il continue avec le prochain mot. Quand tous les mots du palier sont testés, la MT augmente la valeur de k et recommence. Ainsi en commençant avec $k = 0$, on imprime chaque mot dans le langage en ordre d'étapes requises pour le reconnaître. Il est important de procéder ainsi car, si on ne borne pas chaque étape par k, la MT peut tomber dans une boucle infinie. On prouve par l'existence de l'énumérateur qu'un langage reconnaissable implique celui-là (l'énumérateur en question).

6.2 Énumérateur $\Rightarrow$ Reconnaissable

On sait que pour être reconnaissable, un langage doit pouvoir avoir chacun de ses éléments calculé par une machine de Turing (sans rejet, ou boucle infinie). L'énumérateur fait en fait le calcul sur chacun des mots potentiels afin de savoir si ceux-ci font partie du langage, vu qu'il imprime les mots, cela signifie qu'il les accepte et donc qu'il n'y a pas rejet ou boucle infinie ; il reconnaît donc langage.

6.3 Conclusion

En effet, on peut conclure qu'un langage est reconnaissable si et seulement si il existe un énumérateur capablede de l'énumérer.

Question 7

7.1 Énumérateur lexicographique $\Leftarrow$ Décidable

Contrairement aux langages reconnaissables les langages décidables se terminent toujours dans l'exécution. Ceci permet d'employer une technique différente pour les générer. Contrairement à la question 6 où l'on procédait par palier, l'énumérateur lexicographique procède justement par ordre lexicographique. Nous pouvons nous permettre de faire ainsi sans crainte car nous savons que tous les mots refusés ne seront pas acceptés. En parcourant l'ordre lexicographique, si un mot est accepté il est imprimé, et l'impression résultante est en ordre lexicographique. On prouve l'implication par l'existence du numérateur.

7.2 Énumérateur lexicographique $\Rightarrow$ Décidable

Pour qu'un langage soit décidable, il faut que chacun de ses mots puissent être acceptés, et chacun des mots de son complément refusé (autrement dit, pas de boucle infinie). Pour que les mots soient placés en ordre lexicographique, il faut forcément être certain qu'il n'y a aucun autre mot entre les deux qui est accepté, il faut donc calculer tous les mots entre et s'assurer qu'ils sont refusés et cette tâche n'est possible que si le langage est décidable. Nous savons d'on qu'un énumérateur lexicographique implique que le langage est décidable.

7.3 Conclusion

En effet, on peut conclure qu'un langage est décidable si et seulement si il existe un énumérateur lexicographique de l'énumérer.

Annexe

En se trompant sur la question 1, voici la preuve de *5-COL* $\leq_{\mathbf{P}}$ *3-COL*.

8.1 Fonction de réduction

L'idée est d'examiner un graphe 3-coloriable, il faut donc une façon de transformer le graphe 5-coloriable, en graphe 3-coloriable. On assume ici que chaque sommet est encodé par une structure contenant sa couleur et les autres sommets auquel il est lié. Soit la couleur d'un sommet s donnée par s_c, la transformation se fait de manière systématique et consiste à séparer le graphe en deux sous-graphes O et N, où O est le graphe original.

1. Choisir 3 couleurs au hazard, appelons cet ensemble C
2. Parcourir le graphe, pour tout sommet s : (Forme N)
   — Si $s_c \in C$
   (a) Ajouter s à N
   (b) $s_c \leftarrow N_c$
3. Parcourir le graphe, pour tout sommet s (Forme O)
   (a) Parcourir les sommets adjacents à s, pour tout sommet s' :
   — Si $s_c = N_c$ et si $s'_c = N_c$, alors fusionner s et s'

L'opération de fusion consiste à ajouter tous les liens de s' à s et de faire pointer tous les liens vers s' sur s. Pour un graphe complet de n sommets, l'opération de réduction a donc été effectuée en temps environ $n + n^2 = n(n+1)$. Ce qui est un polynôme, la réduction est donc polynomiale.

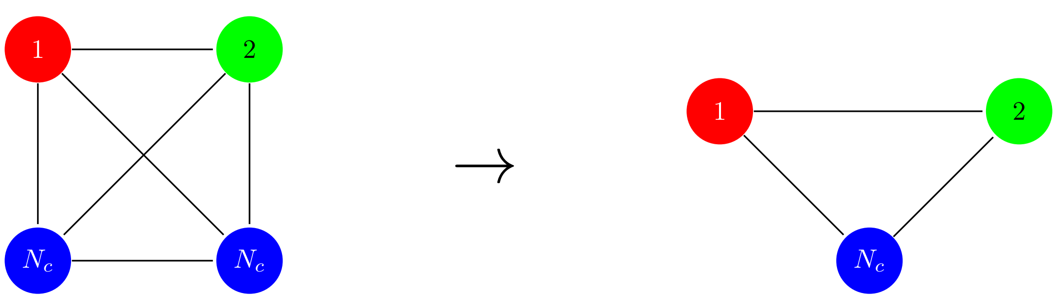
8.2 Équivalence

Nous voulons maintenant prouver que, soit un graphe G :

$$G \in \textit{5-COL} \Leftrightarrow O \in \textit{3-COL} \text{ et } N \in \textit{3-COL}$$

a) $G \in \textit{5-COL} \Rightarrow O \in \textit{3-COL}$ et $N \in \textit{3-COL}$

Commençons par le plus facile. Évidemment, si G est 5-coloriable, alors le graphe O où 3 de ses couleurs ont étés remplacées par une nouvelle couleur N_c, et où les arrête $N_c - N_c$ n'existent pas (car fusionnées) est 3-coloriable. Cela revient à constater qu'il est impossible qu'en fusionnant deux sommets de couleurs N_c on engendre un mauvais coloriage :



Encore plus trivialement, il est facile de réaliser que le graphe N obtenu suite à l'extraction des N_c est 3-coloriable, sinon cela signifirait que 2 nœuds adjacents du graphe N sont de la même couleur ce qui signifie que 2 nœuds adjacents du graphe G sont de la même couleur, ce qui signifie que G n'est pas 5-coloriable, G est 5-coloriable par hypothèse, donc N est 3-coloriable.

b) $G \in$ *5-COL* $\Leftarrow O \in$ *3-COL* et $N \in$ *3-COL*

Puisque O et N n'on aucune couleur en commun, on sait que $G \notin$ *5-COL* $\Rightarrow O \notin$ *3-COL* ou $N \notin$ *3-COL*. Autrement dit, si G n'est pas 5-coloriable, on sait que «la faute» pourra être détectée soit dans O, ou soit dans N (ou les deux), mais il est impossible qu'elle soit ni dans l'un ni dans l'autre. Ceci implique qu'il est impossible que G ne soit pas 5-coloriable et que O et N soient tous les deux 3-coloriables. Nous savons donc que l'équivalence est parfaite.

8.3 Conclusion

Preuve de l'annexe. Soit la fonction f expliquée en 8.1 formellement définie comme suit :

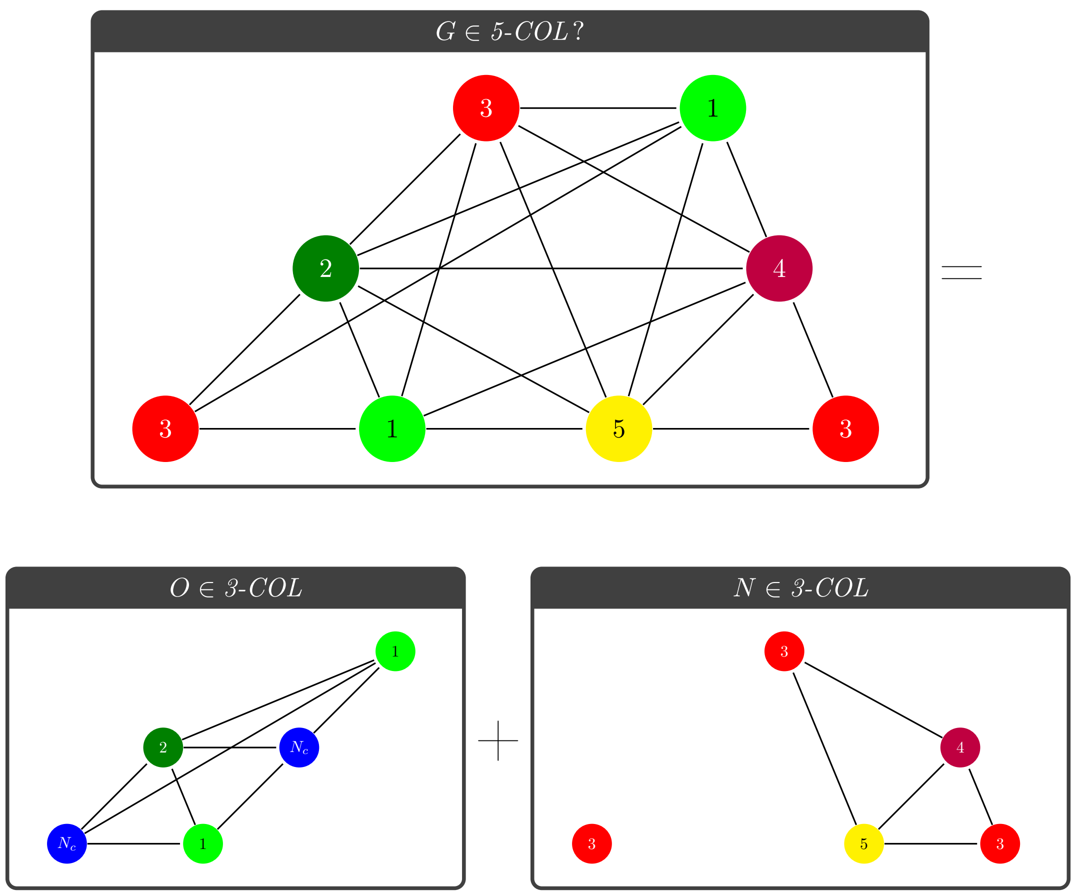
$$f : \Sigma^* \to \Sigma^*$$
$$f(\langle G \rangle) = \{\langle O \rangle, \langle N \rangle\}$$

On peut dire, étant donné l'équivalence expliquée en 1.2 que :

$$\langle G \rangle \in \textit{5-COL} \Leftrightarrow f(\langle G \rangle) = \{\langle O \rangle, \langle N \rangle\} \subset \textit{3-COL}$$

Ce qui nous permet de conclure que *5-COL* $\leq_{\mathbf{P}}$ *3-COL* $\square$

8.4 Exemple



Comme on peut le voir, O et N sont dans *3-COL* ce qui signifie que G est 5-coloriable. On remarque que le graphe N est en fait composé de deux parties, chaque partie correspond à un nœuds N_c qui a survécu à la fusion.