Outils logiques et algorithmiques – TD 12 – Composantes connexes

Exercise 1 (Vrai ou faux ?)

1. Soit G un graphe non-orienté, la relation $\mathsf{path}(x, y)$ qui est vraie s'il existe un chemin de x à y dans G est une relation d'équivalence.
2. Soit G un graphe orienté, la relation $\mathsf{path}(x, y)$ qui est vraie s'il existe un chemin de x à y dans G est une relation d'équivalence.

□

Exercise 2 (Relations d'équivalence)

1. Soit $f : A \longrightarrow B$ une fonction. On définit une relation binaire R sur l'ensemble A par : xRy si et seulement si $f(x) = f(y)$. Montrer que R est une relation d'équivalence. Comment caractériser ses classes d'équivalence ?
2. Montrer que la relation $\subseteq$ n'est pas une relation d'équivalence.

□

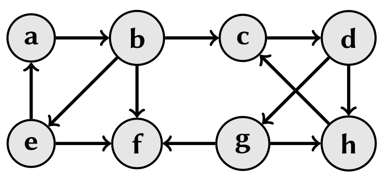
Exercise 3 (Composantes connexes) La composante connexe d'un sommet a dans un graphe non orienté G est définie comme l'ensemble des points b tels qu'il existe un chemin (éventuellement vide) de a à b. Elle est notée $CC_G(a)$.

1. Montrer que l'ensemble des composantes connexes forme une partition de l'ensemble des sommets :
   - chaque composante est non vide
   - tout sommet appartient à une composante
   - deux composantes sont soit disjointes, soit égales
2. Quel est le nombre maximal de composantes connexes dans un graphe en fonction du nombre de sommets, le nombre minimal ?
3. On suppose que le graphe G est acyclique et a k composantes connexes. Soient deux sommets a et b de G et le graphe G' qui est le même que G avec une arête de plus qui a pour extrémités $\{a, b\}$. Montrer que soit G' contient un cycle, soit G' a $k - 1$ composantes connexes.
4. En déduire qu'un graphe acyclique à n sommets a au plus $n - 1$ arêtes.

□

Exercise 4 (Composantes fortement connexes) Étant donné un graphe G *orienté*, la *composante fortement connexe* $C(s)$ d'un sommet s de G est l'ensemble des sommets s' de G tels qu'il existe dans G un chemin de s à s' *et* un chemin de s' à s.

1. Identifier les composantes fortements connexes du graphe suivant.



2. Démontrer que si s_1 et s_2 appartiennent tous deux à la composante fortement connexe d'un même sommet s, alors il existe un chemin de s_1 vers s_2.

On définit le *graphe des composantes* $C(G)$ d'un graphe orienté G comme suit :

- les sommets de $C(G)$ sont les composantes fortement connexes de G,
- il y a un arc de $C(G)$ du sommet C_1 vers le sommet C_2 si et seulement s'il existe dans G : un sommet s_1 dans la composante C_1, un sommet s_2 dans la composante C_2, un arc de s_1 à s_2.

3. Donner les graphe des composantes du graphe donné en exemple à la question 1.

4. Soit un graphe G et son graphe des composantes $C(G)$. Démontrer par récurrence que pour tout k, s'il existe un chemin de longueur k dans $C(G)$ d'un sommet C_1 à un sommet C_2, alors pour tous sommets s_1, s_2 de G tels que $s_1 \in C_1$ et $s_2 \in C_2$, il existe dans G un chemin de s_1 à s_2 de longueur supérieure ou égale à k.
5. Démontrer qu'un graphe des composantes est toujours acyclique.

□

Exercise 5 (Ordre et équivalence) On se donne un ensemble A et une relation R sur A qui est réflexive et transitive. On introduit la relation E définie par $E(x, y) \quad \equiv \quad R(x, y) \wedge R(y, x)$

1. Montrer que E est une relation d'équivalence. On appelle $A_{/E}$ l'ensemble des classes d'équivalence de A pour la relation E. Montrer que pour tout $X \in A_{/E}$ on a $\forall x\ y \in X,\ R(x, y)$.
2. Soit $A = \{a, b, c, d\}$ et R la relation telle que $R(a, a)$, $R(b, b)$, $R(c, c)$, $R(d, d)$, $R(a, b)$, $R(b, a)$, $R(b, c)$, $R(a, c)$. Définir la relation E et donner les classes d'équivalence correspondantes.
3. On introduit la relation R_E sur $A_{/E}$ par $R_E(X, Y) \quad \equiv \quad \exists x \in X,\ \exists y \in Y,\ R(x, y)$.
   (a) Montrer que $R_E(X, Y) \quad \Longleftrightarrow \quad \forall x \in X,\ \forall y \in Y,\ R(x, y)$.
   (b) Montrer que R_E est une relation d'ordre sur $A_{/E}$.
   (c) Construire la relation R_E sur l'exemple de relation de la question précédente.

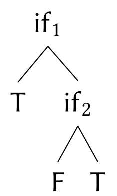
□

Exercise 6 (Arbres de décision) On utilise des arbres binaires pour représenter des formules booléennes sur des variables $x_1, \ldots, x_n$. Pour cela on introduit deux constantes T et F et pour chaque variable x_i, on introduit un constructeur if_i binaire. On appelle bdt l'ensemble des termes ainsi construits.

Chaque terme représente une formule booléenne. La constante T représente la formule $\top$ (vrai), la constante F représente la formule $\perp$ (faux), et si le terme t représente la formule P et le terme u représente la formule Q, alors le terme $\mathsf{if}_i(t, u)$ représente la formule $(x_i \wedge P) \vee (\neg x_i \wedge Q)$. On introduit une fonction val qui associe une formule booléenne à chaque terme de bdt et qui est définie de manière récursive par les équations suivantes :

$$\mathsf{val}(\mathsf{T}) = \top \qquad \mathsf{val}(\mathsf{F}) = \perp \qquad \mathsf{val}(\mathsf{if}_i(t, u)) = (x_i \wedge \mathsf{val}(t)) \vee (\neg x_i \wedge \mathsf{val}(u))$$

L'arbre ci dessous représente donc la formule $(x_1 \wedge \top) \vee (\neg x_1 \wedge ((x_2 \wedge \perp) \vee (\neg x_2 \wedge \top)))$ (qui est équivalente à $x_1 \vee (\neg x_1 \wedge \neg x_2)$)



1. Donner des arbres représentant des formules équivalentes respectivement à x_i et à $\neg x_i$.
2. Montrer que l'arbre $\mathsf{if}_i(t, t)$ représente une formule équivalente à la formule représentée par t.
3. Montrer que les formules $\neg((x \wedge P) \vee (\neg x \wedge Q))$ et $(x \wedge \neg P) \vee (\neg x \wedge \neg Q)$ ont la même valeur de vérité quelles que soient les valeurs de x, P et Q.
4. Déduire de la question précédente des équations récursives pour définir une fonction not qui transforme un arbre t qui représente la formule P en un arbre $\mathsf{not}(t)$ qui représente une formule équivalente à $\neg P$.
5. A quelle condition sur l'arbre reconnait-on que la formule associée est une tautologie ? Donner les équations récursives pour définir une fonction tauto qui prend en argument un arbre t et renvoie vrai si la formule associée à t est une tautologie et faux sinon.

□