Modèles de calcul et Validation d'algorithmes raphael.charrondiere@inria.fr
L3 MIAGE, UGA
2019/2020

DM : Testons, exploitons

---

*Ce devoir maison est **individuel**, doit être rendu sous forme manuscrite sur une copie au format A4. N'utilisez pas de couleur rouge, reservée à la correction. Il sera noté. En cas de non rendu la note sera automatiquement nulle, et chaque jour de retard entraînera une pénalité.*

Exercice 1 Sommer et multiplier les entiers

On considère le programme **P** suivant :
$x := 0$**;** $z := 1$**; while** z $\leqslant$ y **do** ($x := x + z$**;** $z := z + 1$)

1. Donner l'invariant et le variant de boucle du programme **P**, justifiez la réponse. *En Bonus Utiliser la logique de Hoare pour montrer* $\{y = n \wedge n \geqslant 0\}\mathbf{P}\{x = \frac{n\times(n+1)}{2}\}$. *(On annotera le programme entre les commandes comme dans l'exemple en cours.)*

Exercice 2 Hésiter entre deux instructions

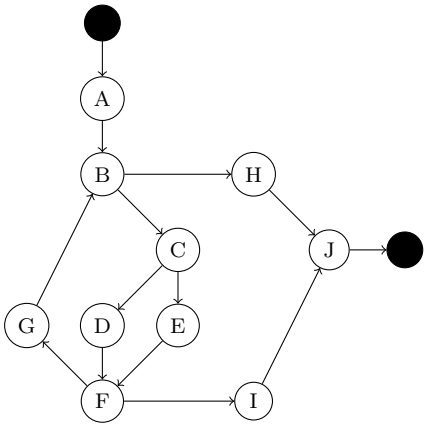
On se propose d'introduire du non-déterminisme dans IMP. On introduit ainsi la commande **maybe do** <commande> **otherwise do** <commande>.

1. Donner les deux règles d'inférence pour la sémantique à grands pas.
2. Donner les deux règles d'inférence pour la sémantique à petits pas.
3. Comment traduire cette commande dans un graphe de flot de contrôle ? (Faut-il créer un nouveau type de nœud, ou au cas contraire, comment intégrer cette commande dans le graphe ?)

Exercice 3 Parti indépendantiste

Cette partie porte sur les chemins indépendants.

1. Calcul des chemins indépendants



Donnez une base de chemin indépendants. Justifiez votre réponse.
Indice : Commencez par relire le cours, pour savoir combien de chemins faut-il trouver. Ensuite trouvez ces chemins et montrez qu'ils sont indépendants, au besoin, cherchez "famille libre" ou "vecteurs indépendants" dans un moteur de recherche.

2. Soit un graphe $\mathcal{G} = (N, A)$ où N est l'ensemble de ses $n = \#N$ nœuds et A l'ensemble de ses $a = \#A$ arcs. On rappelle que la complexité cyclomatique se définit par $v(\mathcal{G}) = 2 + a - n$. Montrez qu'en notant ϕ le nombre de nœuds de décision, alors $v(\mathcal{G}) = \phi + 1$.
Indice : vérifiez la propriété sur un exemple, et essayer de généraliser.

Exercice 4 Graphe de flot de contrôle

1. Nous avons vu en cours les nœuds de commande et les nœuds de condition, il peut être utile de définir un autre type de nœud pour traduire l'instruction `return c`.
   Proposez une définition pour ce type de nœud.
   Indice : Ai-je toujours le droit de rajouter un tel nœud s'il respecte la définition donnée en réponse ? Si non, alors la définition donnée est fausse/incomplète.
2. Dessiner le graphe de flot de contrôle de la fonction `ext4_convert_meta_bg` en annexe.
   Remarque : On considère que les appels de fonctions sont des instructions élementaires.
   Remarque : Pour simplifier la mise en page, dans les nœuds qui ne sont pas des nœuds de décision, on inscrira uniquement les numéros de lignes exécutées.
   Remarque : Il s'agit d'une fonction prise dans le noyau Linux, cette fonction participe au redimensionnement de partitions ext4.

Exercice 5 Enchaîner les ifs

Un magasin veut inciter les clients à acheter les produits alimentaire en fin de vie en premier pour éviter les pertes. Il baisse alors les prix suivants plusieurs critères, en considérant la date de péremption et le stock disponible. Si plusieurs critères sont applicables, le dernier critère de la liste est appliqué

C1 : Si la date de péremption est dans les trois jours, le produit reçoit une réduction de 10%
C2 : Si la date de péremption est dans les deux jours, le produit reçoit une réduction de 40%
C3 : Si la date de péremption est dans les dix jours et qu'il s'agisse d'un article à longue conservation, le produit reçoit une réduction de 50%
C4 : Si le stock est faible le produit n'est pas diminué
C5 : Si la date de péremption est le jour même, le produit reçoit une réduction de 70%

Le magasin veut coder un programme pour afficher la réduction à appliquer. Prenons la solution la plus simple :

```
productID ← readInput();
dureeVie ← getExpiration(productID)-clock();
dureeConservation ← getDureeConservation(productID);
stock ← getStock(productID);
reduction← 0;
si dureeVie<3 jours alors reduction ← 10;
si dureeVie<2 jours alors reduction ← 40;
si dureeVie<10 jours ET dureeConservation>30 jours alors reduction ← 50;
si stock<10 alors reduction ← 0;
si dureeVie<1 jour alors reduction ← 70;
retourner reduction;
```

Algorithme 1 : Réduction

1. Donner le graphe de flot de contrôle.
2. Donner des données de test pour couvrir le critère "tous les arcs"
3. Écrire l'expression algébrique des chemins.
4. Pour chaque chemin donner une donnée de test (on supposera que `clock()` renvoie le 1/1/2020).
   Indice : Regardez la question suivante.
5. Quel problème y a-t-il à décorréler la structure du code ?
6. Pour pouvoir exploiter un critère « tous les chemins », il faut réécrire le code. Proposez une nouvelle version du code. On demande à ce que l'instruction `retourner` soit utilisée une seule fois.
7. Donner le graphe de flot de contrôle du nouveau code, puis les données de test pour couvrir le critère « tous les chemins ».