Validation d'algorithmes
L3 MIAGE, UGA
2020/2021

Raphaël Charrondière
raphael.charrondiere@inria.fr
Semaine 2 - échéance S4

Feuille de TD 1: Preuves

Où l'on commence à manipuler les notions de variant, d'invariant, d'ordre bien fondé

Règles et fonctionnement des TD : Il vous est demandé de travailler en groupe de trois ou quatre personnes maximum. Chaque groupe devra rendre une correction à son responsable de TD. Elle devra inclure la solution de **tous** les exercices (même ceux non corrigés en TD), avec les justifications nécessaires. Vous devez déposer votre travail sur Moodle **avant le mercredi midi** de la semaine indiquée sur la fiche de TD. Chaque rendu sera noté, un non rendu recevra automatiquement un zéro. Si le correcteur se rend compte de plagiat entre les groupes, il mettra 0 à l'ensemble du rendu.

Pour la première séance : L'exercice 3 question 1 et 2, l'exercice 4 question 1 nécessitent d'avoir compris la notion d'invariant, si ce n'est pas le cas, vous pouvez attendre d'avoir fini le cours avant de les aborder. **Attention :** il y aura un autre TD la semaine prochaine, si l'échéance est en S4 c'est uniquement pour vous laisser le temps de bien assimiler le cours, je vous conseille fortement d'avoir traité toutes les questions que vous pouvez d'ici la semaine prochaine pour ne pas vous surcharger.

Exercice 1 — Ordre bien fondée

1. Rappelez la définition d'ordre, ordre total, ordre strict et ordre bien fondé.
2. Prouvez que l'ordre usuel $<$ sur $\mathbb{N}$ est bien fondé
3. Soit l'ordre $<_2$ sur $\mathbb{N}$ définit par $x <_2 y \Leftrightarrow \exists t \in \mathbb{N}^*\ y = x + 2t$. Prouvez que $<_2$ est un ordre strict, bien fondé mais qu'il n'est pas total.
4. Soit la relation $|$ (divise) sur $\mathbb{N}$ définie par $x|y \Leftrightarrow \exists t \in \mathbb{N},\ t > 1,\ y = xt$. Prouvez que $|$ est une relation d'ordre strict bien fondée mais pas totale. *Si la notation $|$ vous gêne, utilisez $|_<$*

Exercice 2 — Dictionnaires et ordre

D'ordinaire un dictionnaire donne des définitions aux mots, ce qui nous intéresse, c'est que les mots sont classés par ordre lexicographique (On va supposer qu'il s'agisse d'un dictionnaire standard, un petit malin aura toujours le plaisir de classer les mots autrement). Pour nous informaticien, un mot est juste un séquence de caractère, et «zabtrd» est un mot au même titre que «orange»

1. Montrez que l'ordre lexicographique n'est pas bien fondé. *Indice : Il suffit de montrer qu'il existe une séquence décroissante infinie de mots, la taille des mots de cette liste va certainement croitre avec la suite*
2. La relation «est préfixe de» est aussi un ordre, est-il total ? bien fondé ?
3. On pourrait remplacer les lettres par des entiers, cela ne changerait pas grand chose, si ce n'est qu'il y a une infinité d'entier. Reprendre les questions précédentes.
4. Fixons maintenant une limite de taille pour les mots, est-ce que les ordres précédents sont toujours des ordres ? bien fondés ? *Vous pouvez dire que les mots ont tous la même taille si on les complète avec un nouveau symbole ou -1*

Raphaël Charrondière
raphael.charrondiere@inria.fr


```
   Entrées : a, b ∈ ℕ
   Output : pgcd(a, b)
1  tant que a ≠ b faire
2  |  si a > b alors
3  |  |  a ← a − b;
4  |  sinon
5  |  |  b ← b − a;
6  |  fin
7  fin
8  retourner a
```

Algorithme 1 : L'algorithme d'Euclide

```
   Entrées : a, b ∈ ℕ
   Output : a × b
1  r ← 0;
2  tant que a > 0 faire
3  |  si a = 0  mod 2 alors
4  |  |  a ← a/2;
5  |  sinon
6  |  |  r ← r + b;
7  |  |  a ← (a − 1)/2;
8  |  fin
9  |  b ← 2b;
10 fin
11 retourner r
```

Algorithme 2 : Multiplication rapide

Exercice 3 L'histoire d'Euclide

On a vu en cours qu'un invariant devait être vérifié avant d'exécuter chaque bloc d'instructions (libre à nous de définir ces blocs), et qu'il ne portait pas forcément sur tout le code mais seulement un bloc, pas exemple un invariant pour l'algo 2 ne considérera pas la première ligne.

On rappelle aussi qu'un invariant est de la forme $P(\mathcal{M}_0, \mathcal{M})$ avec $\mathcal{M}_0$ l'état mémoire initial. P doit être vérifié au début de chaque bloc si $P(\mathcal{M}_0, \mathcal{M}_0)$ est vrai, c'est-à-dire si l'invariant est vrai en entrée.

1 L'algorithme d'Euclide permet de calculer le PGCD de deux entiers. On rappelle que le PGCD (plus grand diviseur commun) de deux nombres strictement positifs a et b est le plus grand entier qui divise a et b. La découpe du code en bloc, est ici très facile : on considère le sous-bloc lignes 2 à 6 uniquement. Il faut alors trouver l'invariant qui permet de prouver que le résultat retourné par l'algorithme est bien le PGCD des nombres donnés en entrée.

2 *(Cours)* Un invariant de boucle permet-il de prouver la terminaison d'une boucle ? Si oui, justifier, si non qu'est-ce qui permet de prouver la terminaison d'une boucle ?

3 Prouver alors la terminaison de l'algorithme.

Exercice 4 Un autre exemple simple

On continue avec un exercice très similaire avec la multiplication rapide qui calcule le produit de deux entiers en utilisant que des additions, soustractions, multiplications par 2, divisions par 2.

On pourrait discuter de l'intérêt de cet algorithme, ce n'est pas vraiment le sujet qui nous intéresse, mais en base 2, une multiplication par 2 est l'ajout d'un bit nul à droite (bit de poids faible), et la division entière est le retrait du bit à droite.

1 Prouver que l'algorithme 2 renvoie la bonne valeur.

2 Prouver que l'algorithme 2 termine.