Lycée Claude Fauriel Le théorème d'Arrow : corrigé

A la preuve d'Arrow

1 Par U, (x,y) étant dans tous les s(v) est dans L(v)

2 Par I, le scrutin s_0 est le pire de ceux lors desquels P a préféré y à x, dans tout autre scrutin l'issue devra être au moins aussi bonne

3 Puisque P est décisif pour y contre x, dans L(u) on aura $y > x$. Comme il y a eu unanimité pour préférer x à z, dans L(u) on aura $x > z$. L(u) est donc $y > x > z$. Ce scrutin u a donc placé y devant z alors que tout $V\backslash P$ voulait le contraire, P est donc décisif pour y contre z

4 On considère le scrutin w unanime par blocs : pour les v de P $z > y > x$, pour les v de $V\backslash P$ $x > z > y$. Par U dans L(w) on aura $z > y$ et par décisivité de P pour y contre x on aura $y > x$. On a donc finalement $z > x$ selon les opinions de P et contre les opinions des autres : P est donc décisif pour z contre x

On considère maintenant deux quelconques candidats s et t mais $\{s, t\} \neq \{x, y\}$

5 parmi s et t l'un (au moins) est un $z \notin \{x, y\}$. d'après la question précédente, P est décisif pour z contre x et pour y contre z. En utilisant de nouveau la question précédente à partir de P décisif pour z contre x on en tire P décisif pour z contre $\{s, t\}\backslash z$ et symétriquement P décisif pour $\{s, t\}\backslash z$ contre z

6 Si m n'est ni x ni y, on utilise la question précédente pour $\{s, t\} = \{x, m\}$ pour montrer que P est décisif pour x contre m, on la réutilise avec $\{s, t\} = \{x, y\}$ par rapport à la paire $\{x, m\}$ et on a P décisif pour x contre y

7 $\boxed{\text{SI}}$ D" est non vide

on peut en effet considérer le scrutin indiqué. Comme D est décisif, on aura $x < y$, si on avait $z < y$ nous autions un scrutin rendant D" décisif pour y contre z, donc décisif (par Q1..Q6), on aura donc $y < z$: d' vient donc d'obtenir $x < z$ contre d" et e, donc (par Q1..Q6) D' est decisif et par notre hypothèse de cardinal minimum D" est vide

$\boxed{\text{AGAGA}}$ D" est donc vide, et v_1 est dictateur

B l'informatisation

Les problèmes à implémenter sont :

- une loi vérifiant U et I est donnée, trouver d : On organise dichotomiquement des scrutins unanimes pour la moitié des votants, la moitié de la moitié qui vient de gagner ... après ln(card(V)) opération on a le dictateur

- une loi vérifiant U et ¬D étant donnée fournir un contre exemple à I

  Je fixe ici n=3, il y a donc 6 opinions possibles, un scrutin est une liste de p entiers de 1..6 (codant les opinions). L'ensemble des scrutins est naturellement ordonné (lexicographiquement), le plus petit élément est le vote unanime 1 (pour dire a < b < c) et le dernier sera le vote unanime 6 (pour dire c < b < a)

  Pour un couple (c,c') donné la partie hypothèse de la "règle d'indépendance" est une relation de préordre (R et T), on veut vérifier que L est croissante de cet ensemble vers les booléens (en fait on veut vérifier que L n'est pas croissante et avoir un contre exemple)

  Comme il ne faut surtout pas parcourir l'ensemble des paires de scrutins (pour raison de complexité), on va partir de tous les scrutins minimaux pour (c,c') : tous les électeurs ont déclaré c s(v) c' faux. A partir de chacun de ces points de départ on va avancer en modifiant à chaque fois unne opinion, l'expression avancer signifiant qu'une fois qu'un v a déclaré c s(v) c' vrai il ne revient pas en arrière

  S'il est évident que l'on va bien tout parcourir de cette façon (preuve par exemple par récurrence sur le nombre d'électeurs préférant c à c') ce qui est mauvais c'est que l'on n'a pas un arbre mais un GNOAC et qu'un programmation récursive naïve va tout faire exploser. On peut y remédier au moyen d'une ... énorme mémoire dans laquelle on enregistrera les noeuds du graphe déja visités

  Il reste à faire cela pour tous les couples (c,c')

  Un moyen de se convaincre que la complexité de cette horreur est bien minimale est de voir comment on fabriquerait des tests pour cet algorithme : on prend un mode de scrutin "courant" (il faut donc traiter tous les cas d'égalité) ... mais on peut aussi prendre une loi Dictatoriale ... sauf UN cas dans lequel on retourne la préférence du Dictateur pour une paire de candidats contigüs : pour trouver ce cas il faudra bien avoir testé tous les scrutins vis à vis de tous les candidats

  Dans l'algorithme précédent je teste pour les (c,c'), dans ma "minoration" ci-dessus je minore par des tests sur des paires contigües : il y a un n^2 qui devient n. Mais si on se limite à un contexte de type "électoral" ce sera p (et p!) et non n qui sera grand

  Je n'ai pas programmé (ni rédigé de preuve de programme pour) ce qui précède, il y a donc la possibilité d'une grossière erreur