Rapport de jury de l'épreuve orale d'informatique fondamentale LCR

Banque MP Inter-ENS – Session 2019

Membres du jury : Benoît Delahaye, Vincent Jugé et Frédéric Prost

Coefficients (Concours MP) : Lyon MI : 10.8 % – Paris-Saclay MPI : 23.1 % – Rennes MPI : 23.1 %
Coefficients (Concours Info) : Lyon M : 12.7 %[1] – Lyon P : 12.7 %[1] – Paris-Saclay : 13.2 % – Rennes 8.6 %

L'épreuve orale d'informatique fondamentale concerne les candidats aux ENS de Lyon, Paris-Saclay, et Rennes des concours MPI et Informatique.

Cette année, 232 candidates et candidats ont participé à l'épreuve. Leurs notes sont comprises entre 3 et 19, avec une moyenne de 12 et un écart-type de 3,01. L'histogramme de la figure 1 présente la distribution des notes. Les membres du jury tiennent par ailleurs à souligner qu'ils ont été choqués et peinés de constater le très faible nombre de candidates – à peine 19 !

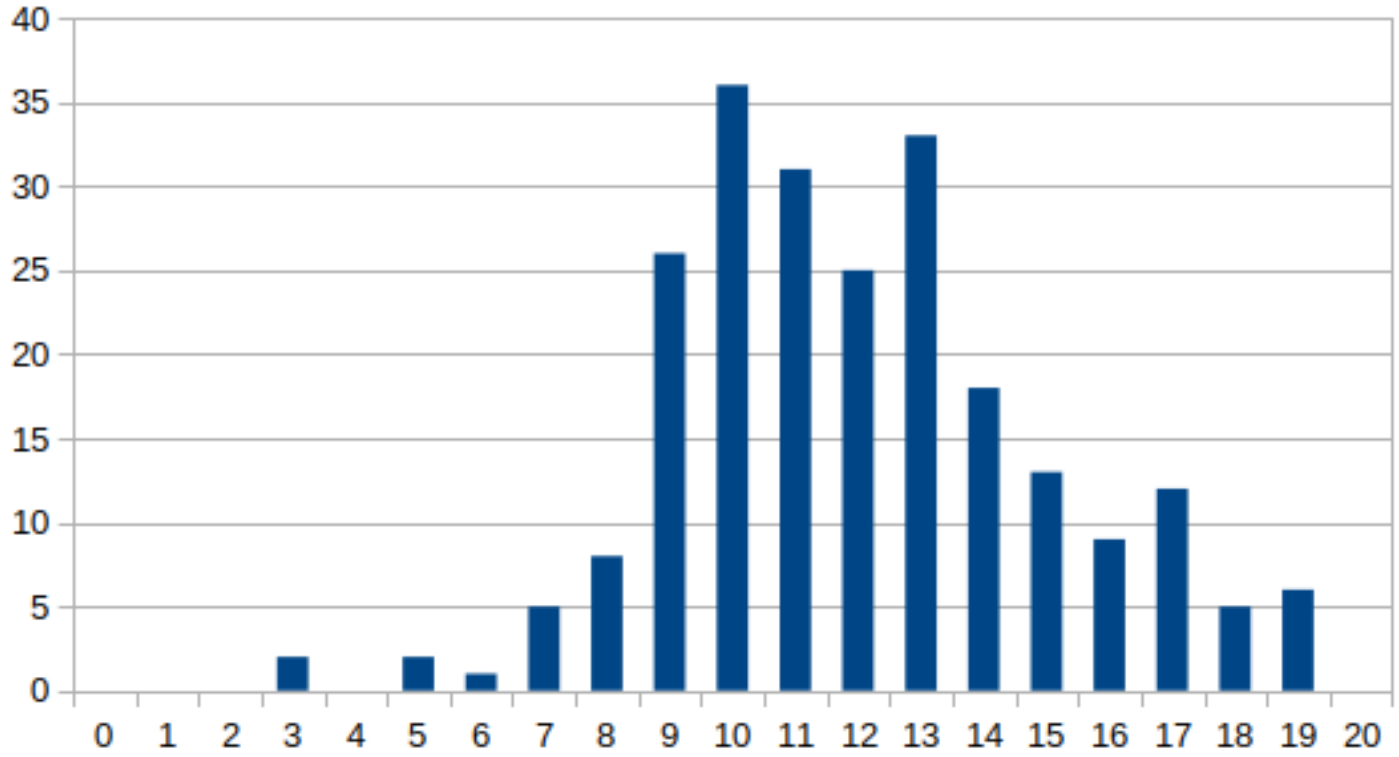


FIGURE 1 – Histogramme des notes de l'épreuve

Après avoir reçu un sujet, les candidats disposaient de 30 minutes de préparation, suivi de 28 minutes d'interrogation devant un des examinateurs. En effet, 2 minutes sont utilisées par l'examinateur pour rappeler au candidat les conditions de l'évaluation : cette évaluation se base principalement sur la progression des candidats dans les questions, mais aussi sur la pédagogie de leur présentation et l'autonomie dont ils ont fait preuve pendant l'oral. À ce sujet, le jury a été favorablement impressionné par la pédagogie des exposés de nombreux candidats.

Le jury a proposé 22 sujets originaux, dont trois exemples représentatifs sont présentés en annexe. Cela représente une moyenne de plus de 10 candidats sur chaque sujet, ce qui permet aux membres du jury d'effectuer une meilleure harmonisation des évaluations, notamment via une pondération des questions selon leur difficulté.

1. Ces coefficients passeront à 14.1 % pour l'année 2020.

Chaque sujet débute par un énoncé qui présente un problème d'informatique et introduit ses notations, puis comporte entre six et dix questions de difficulté globalement croissante. Une ou deux premières questions relativement faciles d'accès permettent aux candidats de vérifier leur compréhension de l'énoncé et de « se lancer » dans l'interrogation orale. Les questions suivantes, progressivement plus difficiles, occupent la plus majeure partie du temps de préparation et d'interrogation, et visent à permettre de départager les candidats. Enfin, la plupart des sujets terminent par une ou plusieurs questions considérées comme très difficiles. En général, il n'est pas attendu des candidats qu'ils traitent l'ensemble des questions dans le temps imparti.

Les sujets font appel aux différentes compétences nécessaires en science informatique : comprendre des concepts nouveaux, démontrer des résultats théoriques, et construire des solutions techniques telles que des algorithmes. Si l'épreuve d'informatique *fondamentale* met l'accent sur les concepts théoriques de l'informatique, on veille néanmoins à garder à l'esprit le sens des objets que l'on étudie.

Le jury a constaté avec satisfaction que la plupart des candidats sont extrêmement bien préparés aux questions techniques portant sur les automates et langages. Cependant, d'autres domaines du programme des classes préparatoires ne sont pas toujours aussi bien maîtrisés. En particulier, beaucoup de candidats ne pensent pas à employer d'autres structures de données que les tableaux et les listes simplement chaînées, alors que les dictionnaires et files de priorité sont explicitement au programme. Beaucoup de candidats ont également éprouvé d'importantes difficultés à utiliser des notions basiques de probabilités pourtant au programme de mathématiques, ce qui les a lourdement pénalisés dès lors que le sujet portait sur l'étude de systèmes probabilistes.

Le jury adresse les conseils suivants aux futurs candidats :

— Si le principe même d'un concours nous impose de faire un classement et que l'épreuve orale peut sembler intimidante, nous tenons à rappeler que *tous* les candidats admissibles ont un excellent niveau et peuvent aborder l'oral avec confiance. Les candidats ne doivent en aucun cas se dévaloriser ou s'auto-censurer sur la base de leur parcours, leurs notes aux classes préparatoires, ou de ce qu'ils estiment être leur position dans le classement !

— L'objectif du jury est d'amener chaque candidat à réussir à traiter au mieux le maximum de questions. Il est donc dans l'intérêt des candidats d'écouter les demandes et les indications de l'examinateur et d'en tenir compte.

— Lors de la préparation, penser à lire l'ensemble du sujet pour comprendre la direction générale de l'exercice et identifier des questions éventuellement plus simples à traiter. Lors de l'interrogation, ne pas hésiter à proposer à l'examinateur de sauter ou remettre à plus tard des questions pour avoir le temps de présenter l'ensemble des résultats préparés. Plusieurs candidats cette année sont arrivés au terme des 28 minutes imparties avant d'avoir pu présenter toutes leurs solutions.

— Au contraire, certains candidats, de peur d'être déstabilisés par le fait de devoir réfléchir en direct devant l'examinateur, ont fait le choix de passer un maximum de temps sur les questions qu'ils avaient traitées durant la préparation, au détriment de leur avancement dans le problème lors de l'oral. Une partie significative de la note étant basée sur le nombre et la difficulté des questions traitées, cette stratégie s'est avérée particulièrement contre-productive.

— Deux écueils à éviter sont donc, d'une part, de traiter de manière trop superficielle les démonstrations et les algorithmes, en passant à côté de points techniques importants, et inversement, de détailler excessivement les réponses de manière trop formelle et manquer de temps pour traiter suffisamment de questions. Nous conseillons aux candidats de soigner la forme des réponses aux premières questions, plus faciles, quitte à prendre progressivement de la hauteur pour se concentrer sur le fond dans les questions difficiles. Dans tous les cas, on prendra soin de présenter brièvement la démarche ou l'intuition suivie avant de se lancer dans une démonstration, un calcul ou un algorithme.

En annexe, nous présentons trois exemples de sujet représentatifs de cette année :

— Algorithme Union-Find

— Algorithmes randomisés

— Chaînes de Markov décisives

Algorithme Union-Find

L'algorithme *Union-Find*, inventé en 1964 par M. Fischer et B. Galler, a pour but de gérer dynamiquement une partition d'un ensemble $\{1, 2, \ldots, n\}$ fixé. Initialement, on part de la partition maximale $\{\{1\}, \{2\}, \ldots, \{n\}\}$. Puis on peut faire deux types de requêtes :

1. **Test**(k, ℓ) : on demande si deux entiers k et ℓ appartiennent au même sous-ensemble ;
2. **Union**(k, ℓ) : on demande de réunir les deux sous-ensembles auxquels appartiennent les entiers k et ℓ.

L'algorithme Union-Find se base sur la réprésentation suivante des partitions de $\{1, \ldots, n\}$:

— chaque entier k possède un (unique) *père* $p(k)$, qui est un entier de $\{1, \ldots, n\}$ (on peut avoir $k = p(k)$, et on notera $a \to b$ le fait que $b = p(a)$), ainsi qu'un *poids*, w_k, qui est un entier naturel (sans propriété spécifique a priori) ;

— deux entiers distincts ne peuvent être ancêtres l'un de l'autre ;

— deux entiers k et ℓ appartiennent au même sous-ensemble si et seulement s'ils ont un ancêtre commun ;

— si l'entier k appartient à un singleton, alors $w_k = 0$.

Question 1. Représenter la partition $\{\{1, 2, 4\}, \{3, 5\}, \{6\}\}$ de l'ensemble $\{1, 2, 3, 4, 5, 6\}$ comme indiqué ci-dessus. *On n'indiquera pas les poids des entiers, qui n'ont pas encore de sens a priori.*

Question 2. On dit que m est le *chef* de k, ce que l'on note $m = c(k)$, si m est un ancêtre de k tel que $p(m) = m$. Démontrer que tout entier a un unique chef, et deux entiers k et ℓ appartiennent au même sous-ensemble si et seulement s'ils ont le même chef.

Les poids introduits plus haut jouent un rôle au cours de l'algorithme Union-Find, qui utilise également une troisième requête de type **Chef**(k), et peut être codé comme suit :

Algorithme 1 Requête **Chef**(k)

```
1: si k = p(k) alors
2:     renvoyer k
3: sinon
4:     ℓ ← Chef(p(k))
5:     p(k) ← ℓ
6:     renvoyer ℓ
```

Algorithme 2 Requête **Test**(k, ℓ)

```
1: renvoyer Chef(k) = Chef(ℓ)
```

Algorithme 3 Requête **Union**(k, ℓ)

```
1: c ← Chef(k)
2: c' ← Chef(k)
3: si c ≠ c' alors
4:     si w_c < w_c' alors
5:         p(c) ← c'
6:     sinon
7:         p(c') ← c
8:         si w_c = w_c' alors
9:             w_c ← w_c + 1
```

Par commodité, on pourra noter $c' \Rightarrow c$ le fait que c devienne le père de c' et voie son poids augmenter (en ligne 9 de l'algorithme 3).

Question 3. En partant de l'ensemble $\{1, 2, 3, 4\}$, on effectue successivement les requêtes **Union**$(1, 2)$, **Union**$(3, 4)$, **Union**$(2, 4)$ et **Test**$(2, 4)$. Indiquer le poids w_k de chaque entier $k \leqslant 4$, et dessiner les graphes induits par les relations $\to$ et $\Rightarrow$.

Question 4. Démontrer que l'algorithme Union-Find donne toujours un résultat correct.

Question 5. Démontrer que, quelque soit le poids w considéré, chaque entier k aura eu, au cours de l'algorithme, au plus un ancêtre dont le poids valait w.

Question 6. Démontrer que, pour tout $r \geqslant 0$, il y a au plus $n/2^r$ entiers k dont le poids final vaut $w_k = r$.

Question 7. Démontrer que si, au cours de l'algorithme, on a effectué m requêtes, alors l'algorithme a nécessité $\mathcal{O}(\min\{m \log_2(m), m + n \log_2(n)\})$ opérations de base.

Algorithmes randomisés

Question 1. Supposons qu'on ait accès à une fonction `foo()` qui retourne uniformément au hasard des entiers entre 1 et 5 compris. Écrire un algorithme qui retourne un entier entre 1 et 7 de manière équiprobable en n'utilisant que `foo()` comme source d'aléa. On se limitera à utiliser des opérations sur les entiers.

Question 2. Supposons maintenant que nous ayons une fonction `foo()` qui implante le comportement d'une pièce biaisée qui répond 40% du temps « pile » et 60% du temps « face ». Ecrire un algorithme en pseudo-code qui implante une fonction qui retourne 0 et 1 avec une probabilité de 50% et qui n'utilise que `foo()`.

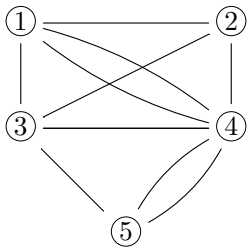
Question 3. Donnez un algorithme qui mélange complètement un tableau de n éléments en $\mathcal{O}(n)$; « mélanger complètement » signifiant que chaque permutation sur $\{0, \ldots, n-1\}$ est équiprobable.

Question 4. On cherche à choisir k éléments de manière équiprobable dans un tableau de n éléments, mais n est tellement grand qu'on ne peut avoir tout le tableau en mémoire (on peut penser à une liste de recherches sur Google) : on pourra considérer que les n éléments arrivent un par un comme un flux de valeur. Donnez un algorithme permettant de résoudre ce problème en temps $\mathcal{O}(n)$.

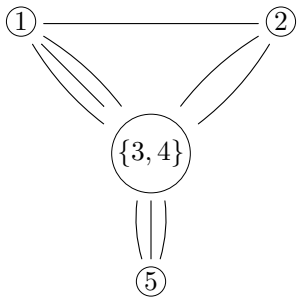
Nous allons étudier un algorithme randomisé sur les multigraphes (un multigraphe est la donnée d'un ensemble de sommets S et d'un multi-ensemble d'arêtes A, c'est à dire qu'il peut y avoir plus d'une arête entre deux sommets) non orientées. Une coupure est une partition des sommets en deux ensembles S_1 et S_2. Une coupure $\{S_1, S_2\}$ est dite minimale si elle est telle que, pour toute autre coupure $\{S'_1, S'_2\}$, il y a plus d'arêtes entre S'_1 et S'_2 qu'il n'y en a entre S_1 et S_2.

Initialement, les sommets d'un graphe sont étiquetés par des entiers ; dans la suite, pour plus de commodité, on pourra identifier chaque entier k au singleton $\{k\}$.

L'opération de **contraction** consiste à prendre deux sommets reliés par au moins une arête et à les confondre : les arêtes entre les deux sommets contractés disparaissent, le sommet issu de la contraction a comme voisins l'ensemble des voisins des sommets dont il est issu, et son étiquette est identifiée à l'union des étiquettes des sommets dont il est issu. Par exemple si on a le multigraphe G suivant :



Le multigraphe issu de la contraction des sommets 3 et 4 est noté $G/\{3,4\}$, ce qui sur notre exemple donne :



Le but est, étant donné un multigraphe connexe, de trouver une coupure minimale. On se propose d'étudier l'algorithme suivant :

```
Contract(G = (S, A)) :
   Tant que |S| > 2
     choisir une arête (x, y) ∈ A de manière aléatoire uniforme (dans l'exemple
         ci-dessus, on avait 2 chances sur 10 de choisir l'une des deux arêtes (1,4))
     G ← G/{x, y}
   Renvoyer S
```

Question 5. Quelle est la complexité de l'opération de contraction ?

Question 6. Montrer que cet algorithme termine bien avec une coupure.

Question 7. Quelle est la probabilité de succès de l'algorithme donné ci-dessus de trouver une coupure minimale donnée ? On pourra se contenter de donner une borne inférieure.

Question 8. Si on tire simplement au hasard une coupure, quelle est la probabilité de trouver a coupure minimale ?

On remarque que les premières contractions ont beaucoup moins de chance de faire disparaître une arête qui serait dans toutes les coupures minimales que les dernières contractions. On se propose d'améliorer l'algorithme précédent en se basant sur cette idée et sur le concept de diviser pour régner. On considère l'algorithme suivant :

```
Min_Cut(G = (S, A)) :
   n ← |S|
   Si n = 2 alors renvoyer S
   sinon
     G' = (S', A') ← copie de G = (S, A)
     Tant que |S'| > 1 + n/√2
       choisir une arête (x, y) ∈ A' de manière aléatoire uniforme
       G' ← G'/{x, y}
     {S_1, S̄_1} = Min_Cut(G')
     {S_2, S̄_2} = Min_Cut(G')
     Renvoyer la coupure {S_i, S̄_i}, avec i ∈ {1, 2}, qui minimise le nombre d'arêtes
         entre S_i et S̄_i
```

Question 9. Montrez que la complexité de l'algorithme `Min_Cut` est bien en $\mathcal{O}(n^2 \log(n))$, où n est le nombre de sommets du graphe.

Question 10. Quelle est la probabilité de répondre une coupure minimale ?

Chaînes de Markov décisives

En informatique, une chaîne de Markov est un système de transitions dont les transitions sont probabilistes. Formellement, une chaîne de Markov est une structure $\mathcal{M} = (Q, q_0, T)$, où

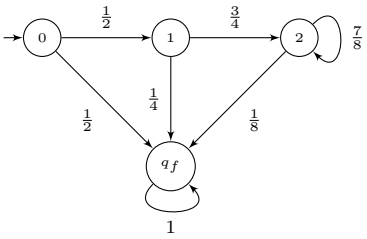
— Q est un ensemble d'états et $q_0 \in Q$ est l'état initial, et

— $T : (Q \times Q) \to [0, 1]$ est la fonction de transitions, telle que $\sum_{q' \in Q} T(q, q') = 1$ pour tout $q \in Q$.

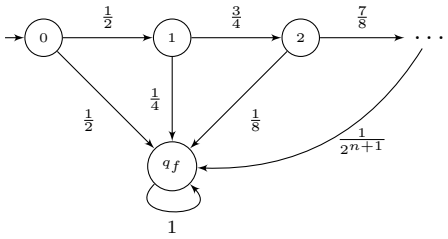
Une *exécution* d'une chaîne de Markov est une suite, finie ou infinie, d'états $w = q_1 q_2 \ldots$. La longueur d'une exécution finie $w = q_1 \ldots q_n$, notée $|w|$ est le nombre de transitions parcourues le long de l'exécution. Ici, $|w| = n - 1$.

Pour chaque état $q \in Q$ de la chaîne, il est possible de définir une mesure de probabilité des exécutions finies d'une chaîne de Markov : $\mathbb{P}_{\mathcal{M},q}(q_1 \ldots q_n) = 0$ si $q_1 \neq q$, $\mathbb{P}_{\mathcal{M},q}(q) = 1$ et $\mathbb{P}_{\mathcal{M},q}(q_1 \ldots q_n) = \Pi_{i=1}^{n-1} T(q_i, q_{i+1})$ pour $n \geq 2$ et $q_1 = q$. On dit d'un état q' qu'il est *accessible* dans $\mathcal{M}$ depuis q s'il existe une exécution fine $w = q \ldots q_n$ telle que $q_n = q'$ et $\mathbb{P}_{\mathcal{M},q}(w) > 0$.

Question 1 *Soit $n \in \mathbb{N}$. Démontrer que, pour tout état q, $\mathbb{P}_{\mathcal{M},q}$ définit bien une mesure de probabilité sur l'ensemble des exécutions de longueur n.*



MC $\mathcal{M}_1$



MC $\mathcal{M}_2$

Question 2 *Soit la chaîne de Markov $\mathcal{M}_1$ ci-dessus. Calculer la probabilité de l'ensemble des exécutions de taille 4 partant de 0 et dont le dernier état est q_f.*

Les mesures de probabilité précédemment définies sur les exécutions finies des chaînes de Markov peuvent être étendues aux exécutions infinies. On définit alors ces mesures de probabilité sur l'ensemble des *cylindres*, qui permettent de définir une σ-algèbre pour chaque état de départ. Un cylindre est l'ensemble des exécutions infinies qui "prolongent" une même exécution finie. Soit $w = q_1 \ldots q_n$ une exécution finie. Le cylindre associé est alors $\mathsf{Cyl}(w) = \{w' = q'_1 q'_2 \ldots \mid \forall 1 \leq i \leq n, q'_i = q_i\}$. La mesure d'un cylindre à partir d'un état q est la probabilité du préfixe fini associé w $\mathbb{P}_{\mathcal{M},q}(w)$.

Question 3 *Dans la chaîne de Markov $\mathcal{M}_1$ de la question 2, quelle est la mesure de l'ensemble des exécutions infinies qui partent de 0 et passent par q_f ?*

Dans la suite, étant donné une chaîne de Markov $\mathcal{M} = (Q, q_0, T)$ et un ensemble d'états $F \subseteq Q$, on notera $(\Diamond_q F)$ l'ensemble des exécutions infinies qui partent de q et passent par un état de F. On notera de plus $\widetilde{F}$ l'ensemble des états à partir desquels F n'est pas accessible.

Etant donné une chaîne de Markov $\mathcal{M} = (Q, q_0, T)$ et un ensemble d'états $F \subseteq Q$, on dit que $\mathcal{M}$ est **décisive** par rapport à F si et seulement si, pour tout $q \in Q$, on a $\mathbb{P}_{\mathcal{M},q}(\Diamond_q F \cup \Diamond_q \widetilde{F}) = 1$ (à partir de n'importe quel état, on peut atteindre soit F soit $\widetilde{F}$ avec probabilité 1).

Dans la suite de l'exercice, on s'intéresse à trouver des conditions suffisantes pour qu'une chaîne de Markov soit décisive.

Question 4 *Montrer que toute chaîne de Markov $\mathcal{M} = (Q, q_0, T)$ où Q est fini est décisive pour n'importe quel ensemble d'états $F \subseteq Q$.*

Question 5 *Montrer que la chaîne $\mathcal{M}_2$ ci-dessus n'est pas décisive pour l'ensemble $\{q_f\}$. Que peut-on en conclure ?*

Soit $\mathcal{M} = (Q, q_0, T)$ une chaîne de Markov. L'ensemble d'états $A \subseteq Q$ est un **attracteur** pour $\mathcal{M}$ si et seulement si, pour tout état $q \in Q$, on a $\mathbb{P}_{\mathcal{M},q}(\Diamond_q A) = 1$.

Question 6 *Montrer qu'une chaîne $\mathcal{M} = (Q, q_0, T)$ qui a un attracteur A* **fini** *est nécessairement décisive pour tout ensemble $F \subseteq Q$.*

On propose maintenant une conditions plus fine pour garantir qu'une chaîne est décisive. Soit $\mathcal{M} = (Q, q_0, T)$ une chaîne de Markov, $q \in Q$ un état et $\beta > 0$ un réel. On dit que l'état q de $\mathcal{M}$ est **de granularité** β si et seulement si, pour tout $q' \in Q$ on a $T(q, q') > 0 \Rightarrow T(q, q') \geq \beta$. La chaîne $\mathcal{M}$ est de granularité β si tous ses états le sont, et on dira que $\mathcal{M}$ est *granulaire* s'il existe un tel β. Finalement, on dit que $\mathcal{M}$ est *globalement granulaire* pour un ensemble $F \subseteq Q$ s'il existe $\alpha > 0$ tel que, pour tout $q \in Q$, si F est accessible depuis q, alors $\mathbb{P}_{\mathcal{M},q}(\Diamond_q F) \geq \alpha$.

Question 7 *Montrer qu'une chaîne $\mathcal{M} = (Q, q_0, T)$ qui est granulaire et de branchement fini vers $F \subseteq Q$ (il existe une constante $N \geq 0$ telle que pour tout $q \in Q$, si F est accessible depuis q, alors il existe une execution $w = qq_1 \dots q'$ de longueur inférieure à N avec $q' \in F$) est nécessairement globalement granulaire pour F.*

Question 8 *Montrer qu'une chaîne $\mathcal{M}$ qui est globalement granulaire pour un ensemble F est nécessairement décisive pour F.*

On s'intéresse finalement à une autre propriété : la décisivité forte. Pour cette propriété, étant donné un ensemble d'états F et un état q, on notera $(\Box_q \Diamond F)$ l'ensemble des exécutions infinies partant de q et visitant infiniment souvent F. Formellement, $(\Box_q \Diamond F) = \{w = qq_2 \dots \mid \forall i, w_i = q_i q_{i+1} \dots \in \Diamond_{q_i} F\}$.
On dit alors que $\mathcal{M}$ est **fortement décisive** par rapport à F si et seulement si, pour tout $q \in Q$, on a $\mathbb{P}_{\mathcal{M},q}(\Diamond_q \widetilde{F} \cup \Box_q \Diamond F) = 1$ (à partir de n'importe quel état, la mesure des exécutions pour lesquelles F est toujours accessible mais qui ne le visitent qu'un nombre fini de fois est nulle).

Question 9 *Montrer que, pour toute chaîne $\mathcal{M} = (Q, q_0, T)$ et pour tout ensemble d'état F, $\mathcal{M}$ est décisive par rapport à F si et seulement si elle est fortement décisive par rapport à F.*