MITRO210: Automates et données structurées

Feuille d'exercices 6 – Corrigé

Antoine Amarilli

1 Exemples d'automates d'arbres

Question 0. Le DbTA proposé a trois états : $q_\bot$ et $q_\top$. Sa fonction initiale associe les nœuds étiquetés a à $q_\top$ et les autres à $q_\bot$. L'unique état final est $q_\top$. L'intuition est que $q_\top$ indique qu'on a déjà vu un a dans le sous-arbre lu jusqu'à présent, et $q_\bot$ qu'on en a pas encore vu. Formellement, la fonction de transition est définie comme suit sur un tuple (q_1, q_2, x) :

— Si l'un de q_1 ou q_2 vaut $q_\top$ alors on renvoie $q_\top$
— Sinon, si x vaut a alors on renvoie $q_\top$
— Sinon, cela veut dire que $q_1 = q_2 = q_\bot$ et que $x \neq a$, auquel cas on renvoie $q_\bot$

Question 1. C'est similaire à la question précédente, mais attention aux pièges. Le DbTA a des états $q_\emptyset, q_\top, q_b, q_c, q_{bc}$. La fonction initiale associe les nœuds b à q_b et les nœuds c à q_c et les autres à $q_\emptyset$. L'unique état final est $q_\top$. L'intuition est encore que $q_\top$ stocke si on a vu un témoin, q_b et q_c respectivement si on a vu un b ou un c, q_{bc} si on a vu les deux (important !), et $q_\emptyset$ si on a rien vu. La fonction de transition est définie comme suit sur (q_1, q_2, x) :

— Si l'un de q_1 ou q_2 vaut $q_\top$ alors on renvoie $q_\top$
— Sinon, si x vaut a et que q_1 vaut q_b ou q_{bc} et que q_2 vaut q_c ou q_{bc}, on renvoie $q_\top$
— Sinon, associons l'étiquette x à un ensemble S_x qui vaut $\{b\}$ ou $\{c\}$ ou $\emptyset$ selon que c'est b ou c ou autre chose, associons q_1 à un sous-ensemble S_1 de $\{b, c\}$ selon que c'est $q_\emptyset$ ou q_b ou q_c ou q_{bc}, associons q_2 à un sous-ensemble de S_2 de la même manière, et on renvoie l'état associé au sous-ensemble $S_x \cup S_1 \cup S_2$. En particulier on renvoie $q_\emptyset$ précisément quand $q_1 = q_2 = \emptyset$ et quand $x = a$.

Question 2. On caractérise d'abord les expressions arithmétiques bien formées : il faut et il suffit que les nœuds internes soient étiquetés par $+$ et $\times$ et que les feuilles soient étiquetées par 0 et 1.

On remarque ensuite un point crucial : dans l'évaluation d'une opération arithmétique avec $+$ et $\times$ sur les entiers naturels, pour savoir si le résultat est nul ou non, il suffit de se souvenir pour chaque sous-expression si le résultat est nul ou non. En effet, une somme est non-nulle ssi l'une de ses opérandes est non-nulles, et un produit est non-nul ssi toutes ses opérandes sont non-nulles. Autrement dit, on peut interpréter 0 et 1 comme des Booléens et $+$ et $\times$ comme $\vee$ et $\wedge$ respectivement, et obtenir le bon résultat. (Remarquer que ce ne serait pas vrai avec les entiers relatifs par exemple.)

On construit un DbTA avec trois états, q_0, q_1, et $q_\bot$. On va intuitivement garantir l'invariant suivant : un nœud n est envoyé vers $q_\bot$ si le sous-arbre enraciné en n a une erreur, sinon il est

envoyé vers q_0 ou q_1 selon que l'évaluation de l'expression arithmétique du sous-arbre enraciné en n est nulle ou non. L'état final sera bien sûr q_1.

La fonction d'initialisation envoie $+$ et $\times$ vers $q_\perp$ (erreur de syntaxe) et 0 vers q_0 et 1 vers q_1.

La fonction de transition δ envoie (q_1, q_2, x) vers :
- $q_\perp$ si $x = 0$ ou $x = 1$ (erreur de syntaxe sur le nœud)
- $q_\perp$ si $q_1 = q_\perp$ ou $q_2 = q_\perp$ (erreur de syntaxe dans un sous-arbre)
- sinon, on a $q_1 = q_a$ et $q_2 = q_b$ pour $a, b \in \{0, 1\}$, et $x \in \{+, \times\}$. On renvoie alors $q_{\min(1,axb)}$, autrement dit q_0 si $axb = 0$ et q_1 si $axb > 0$.

Il est clair que l'automate obtenu satisfait l'invariant mentionné plus haut, et donc il est correct.

Question 3. On peut là encore évaluer l'expression arithmétique en tronquant les entiers au-delà de 42. En effet, dès qu'un entier strictement supérieur à 42 est appliqué à un opérateur $+$, alors le résultat sera aussi strictement supérieur à 42, et s'il est appliqué à un opérateur $\times$ le résultat sera aussi strictement supérieur à 42 sauf si l'autre argument vaut 0.

On peut donc construire un automate similaire à la question précédente. Remarquer que le nombre d'états est proportionnel à 42 ; ici on en utiliserait 45, à savoir les 43 entiers entre 0 et 42 inclus, un état pour les entiers strictement supérieurs à 42, et un état pour les erreurs de syntaxe.

Question 4. La difficulté est que plusieurs feuilles peuvent porter la même étiquette, ou que le même sous-arbre peut intervenir à différents endroits. Une façon de procéder est d'utiliser le non-déterminisme, c'est ce qui est présenté ci-dessous. Une autre façon consiste à utiliser un automate d'arbres déterministe de haut en bas, pour lequel la construction est plus aisée. Une autre façon encore consiste à quotienter les nœuds de l'arbre T par la relation d'équivalence "engendrer le même sous-arbre" et de travailler avec des états correspondant à ces classes d'équivalence. D'une manière simplifiée ceci peut se faire avec la notion de *sérialisation* : on présente ceci à la fin du corrigé de cette question.

Présentons d'abord la construction avec nondéterminisme. On définit un NbTA dont l'ensemble d'états est l'ensemble des nœuds de T. L'unique état final est l'état correspondant à la racine de T. La relation initiale ι est définie comme suit : pour chaque étiquette $x \in \Sigma$, pour chaque feuille n de T étiquetée x, on a un couple $(x, n) \in \iota$. La relation de transition δ est définie comme suit : pour chaque nœud interne n de T étiqueté $x \in \Sigma$ ayant pour enfants n_1 et n_2, on a un tuple $(q_1, q_2, x, q) \in \delta$.

On montre que cet automate est correct. Étant donné un arbre T' accepté par l'automate, pour ρ un run acceptant, on voit que $\rho(r') = r$ pour r' la racine de T' et r la racine de T. Ensuite, la définition d'un run nous assure que la racine de T' a la même étiquette que celle de T, et que (si ce n'est pas une feuille) ses enfants r'_1 et r'_2 sont associés par ρ aux états correspondant aux enfants r_1 et r_2 de la racine r de T. En poursuivant le raisonnement, on voit que ρ définit en réalité un isomorphisme entre T et T', donc T' est bien l'arbre T. En particulier on peut remarquer que le run acceptant est nécessairement unique, aussi ce NbTA est en réalité inambigu, c'est un UbTA.

Réciproquement, pour T' un arbre qui est égal à T, on construit le run acceptant sans difficulté en envoyant chaque nœud de T sur l'état correspondant au même nœud dans T. Il est clair que la racine de T est envoyée sur l'état final de l'automate, et que la relation initiale et la relation de transition sont respectées. Ceci conclut la démonstration.

Présentons maintenant la contsruction avec sérialisation. Étant donné l'alphabet Σ, on pose un alphabet Σ' constitué des symboles `<a>` et `</a>` pour chaque $a \in \Sigma$. À tout Σ-arbre on associe le mot de $(\Sigma')^*$ obtenu, intuitivement, comme en HTML. Formellement, à chaque feuille étiquetée a on associe le mot `<a></a>`, et à chaque nœud interne étiqueté a ayant pour enfants n_1 et n_2 on associe le mot `<a>` w_1 w_2 `</a>` où w_1 et w_2 sont les mots respectivement associés à n_1 et à n_2. On peut remarquer que, pour un nœud n étiqueté a, la position de la balise ouvrante `<a>` et de la balise fermante `</a>` pour ce nœud dans la sérialisation correspond à la date de visite du nœud dans l'ordre préfixe et postfixe respectivement. Il est en particulier clair que deux Σ-arbres sont égaux si et seulement si leur sérialisation est identique.

(On note bien sûr que la sérialisation n'est pas un mot arbitraire de $(\Sigma')^*$: en particulier il doit être bien parenthésé, et on peut montrer sans difficulté que le langage des sérialisations possibles n'est pas un langage régulier sur $(\Sigma')^*$. On peut définir des notions d'automates à pile appelés *visibly pushdown* dont l'usage de la pile consiste à empiler sur les symboles ouvrants et dépiler sur les symboles fermants, et de tels automates ont alors le même pouvoir d'expression que des automates d'arbres sur l'arbre représenté, mais on n'en parlera pas davantage.)

Considérons le Σ-arbre T que l'on souhaite obtenir, et soit T' un Σ-arbre dont on veut tester s'il est égal à T. Une condition nécessaire pour cela est que, pour chaque nœud n' de T', la sérialisation w'_n du sous-arbre de T' enraciné en n' soit identique (comme mot de $(\Sigma')^*$) à la sérialisation d'un certain sous-arbre enraciné de T. On va donc poser les états d'un DbTA A reconnaissant uniquement T comme l'ensemble des sérialisations des sous-arbres enracinés de T : c'est un nombre linéaire d'états. La fonction initiale associe chaque feuille étiquetée a à la sérialisation `<a></a>` (s'il y a une feuille a dans T), et la fonction de transition associe chaque couple d'états q_1 et q_2 sur une étiquette a à la sérialisation `<a>` q_1 q_2 `</a>`, où on identifie les états à des sérialisations et où on les concatène, à condition que la sérialisation ainsi obtenue soit effectivement un état de l'automate c'est-à-dire la sérialisation d'un arbre enraciné de T. Enfin, on choisit pour état final celui qui correspond à la sérialisation de T tout entier.

Il est clair inductivement que, lorsque A s'exécute sur un arbre T', son unique run associe chaque nœud à sa sérialisation. En particulier si on peut définir un run acceptant alors la sérialisation de la racine de T' est celle de la racine de T et ainsi T et T' sont identiques. À l'inverse étant donné un arbre T' identique à T on pourra bien définir un run accepant. Ceci conclut la preuve. À noter que travailler avec les sérialisations revient clairement à travailler avec les différents sous-arbres de T quotientés par la relation d'équivalence "être le même arbre", puisque deux sous-arbres sont égaux si et seulement s'ils ont la même sérialisation.

Question 5. Pour chaque couple $(x, y) \in \Sigma^2$, on aura un état $q_(x, y)$ qui dénotera intuitivement que, pour le sous-arbre enraciné en n, on a que x et y sont respectivement l'étiquette de la première feuille et de la dernière feuille du sous-arbre. (Noter que ces feuilles peuvent être identiques si et seulement si n est une feuille, auquel cas n est lui-même sa première feuille et sa dernière feuille). Les états finaux seront les $q_{x,x}$ pour $x \in \Sigma$.

La fonction d'initialisation envoie $a \in \Sigma$ vers $q_{a,a}$.

La fonction de transition envoie $(q_{x_1,y_1}, q_{x_2,y_2}, x)$ vers q_{x_1,y_2}. Noter que l'étiquette des nœuds internes est ignorée.

Une induction immédiate montre que l'automate satisfait l'invariant et donc qu'il est correct.

2 Propriétés des automates d'arbres

Question 0. Soit A un DbTA, qui est complet (c'est-à-dire que les fonctions d'initialisation et de transition sont totales). Ainsi, A a exactement un run sur chaque arbre T. Construisons A' en échangeant les états initiaux et finaux. L'automate A' est également un DbTA complet, donc il a un unique run sur chaque arbre T. Or, le run de A sur T est acceptant ssi celui de A' sur T ne l'est pas. Ainsi, A' reconnaît bien le langage formé des arbres qui sont rejetés par A.

On en déduit donc bien que les langages reconnaissables d'arbres sont clos par complémentation.

Question 1. On va utiliser une construction analogue à l'automate produit. Donnons-nous $A_1 = (Q_1, \Sigma, \iota_1, F_1, \delta_1)$ et $A_2 = (Q_2, \Sigma, \iota_2, F_2, \delta_2)$ deux DbTA complets. On pose le DbTA complet $A = (Q, \Sigma, \iota, F, \delta)$ obtenu comme le produit de A_1 et A_2 c'est-à-dire :

- $Q = Q_1 \times Q_2$,
- $\iota(x) := (\iota_1(x), \iota_2(x))$ pour $x \in \Sigma$,
- $\delta(((q_1, q_2), (q_1', q_2'), x)) := (\delta_1(q_1, q_1', x), \delta_2(q_2, q_2', x))$ pour $(q_1, q_2), (q_1', q_2') \in Q_1 \times Q_2$ et $x \in \Sigma$.
- Pour F on a le choix : soit on prend $F_1 \times F_2$, soit on prend $Q_1 \times F_2 \cup F_1 \times Q_2$.

On voit alors que, pour tout Σ-arbre T, en notant ρ_1 et ρ_2 les runs respectifs de A_1 et A_2 sur T, et en notant ρ le run de A sur T, on a la condition suivante : pour chaque nœud n de T, on a $\rho(n) = (\rho_1(n), \rho_2(n))$. En effet, cela se démontre par induction :

- Si n est une feuille d'étiquette x, alors on a $\rho(n) = (\iota_1(x), \iota_2(x)) = (\rho_1(x), \rho_2(x))$.
- Si m est un nœud interne d'étiquette x et d'enfants n et n', alors on a $\rho(m) = \delta(\rho(n), \rho(n'), x)$. Par hypothèse d'induction ceci vaut $\delta((\rho_1(n), \rho_2(n)), (\rho_1(n'), \rho_2(n')), x)$. Par définition de δ ceci vaut $(\delta_1(\rho_1(n), \rho_1(n'), x), \delta_2(\rho_2(n), \rho_2(n'), x))$. On retrouve bien $(\rho_1(m), \rho_2(m))$.

Ainsi, la racine de T sera envoyée vers (q_1, q_2) où q_1 et q_2 sont les états respectivement obtenus dans ρ_1 et ρ_2 pour la racine. Selon le choix de F effectué, on obtient bien un automate qui reconnaît l'intersection des langages de A_1 et A_2, ou leur union.

(On peut aussi se contenter de démontrer une seule de ces deux affirmations, et utiliser la loi de De Morgan grâce à la question précédente.)

Question 2. Les langages finis d'arbres sont des unions finies de langages singletons qui sont reconnaissables d'après la question 4 de l'exercice précédent. Ainsi, d'après la question précédente, ces langages sont reconnaissables.

Question 3. On applique une construction analogue à celle de l'automate des parties. Soit $A = (Q, \Sigma, \iota, F, \delta)$ un NbTA. On construit $A' = (Q', \Sigma, \iota', F', \delta')$ un DbTA défini comme suit :

- $Q' := 2^Q$ l'ensemble des parties de Q
- $\iota'(x) := \{q \in Q \mid (x, q) \in \iota\}$ pour chaque $x \in \Sigma$
- $\delta'(X_1, X_2, x) := \{q \in Q \mid \exists q_1 \in X_1, q_2 \in X_2, (q_1, q_2, x, q) \in \delta\}$ pour chaque $x \in \Sigma$ et $X_1, X_2 \in 2^Q$
- $F = \{X \subseteq Q \mid X \cap F \neq \emptyset\}$

Soit T un arbre quelconque. On montre par induction que, dans le run ρ de A' sur T, chaque nœud n est envoyé vers l'ensemble $X \subseteq Q$ des états q pour lesquels il existe un run de A sur le sous-arbre T_n de T enraciné en n où la racine est envoyée vers q. En effet :

— Si n est une feuille, c'est immédiat : en posant $x \in \Sigma$ l'étiquette de n, les runs de A sur l'arbre singleton n peuvent envoyer n précisément vers les états q tels que $(x, q) \in \iota$, or c'est exactement l'ensemble $\iota'(x)$
— Si n est un nœud interne, posons x son étiquette et n_1 et n_2 ses deux enfants. Par hypothèse d'induction, $\rho(n_1)$ est précisément l'ensemble X_1 des états q tels qu'il existe un run $\rho_{1,q}$ de A sur le sous-arbre T_{n_1} qui envoie la racine n_1 vers q ; et l'affirmation analogue vaut pour $\rho(n_2)$ et un ensemble X_2. Maintenant, observons qu'un run de A sur T_n revient à choisir un état q pour n, une transition $(q_1, q_2, x) \in \delta$, et deux sous-runs sur T_{n_1} et T_{n_2} envoyant respectivement la racine vers q_1 et vers q_2. Ainsi, l'ensemble X des états q tel qu'il existe un run ρ_q de A sur T_n envoyant vers q est précisément donné par l'ensemble $\{q \in Q \mid \exists q_1 \in X_1, q_2 \in X_2, (q_1, q_2, x, q) \in \delta\}$, et on retrouve la définition de δ'.

Ainsi, in fine, ρ envoie la racine de T vers l'ensemble des états q de A pour lesquels A a un run sur T envoyant la racine vers q. Or, A accepte T si et seulement s'il existe un état de F satisfaisant cette condition. Par notre choix d'états finaux F', on conclut que la construction est correcte.

Question 4. Pour un arbre T et deux nœuds internes n et n' tel que n soit un ancêtre strict de n', on définit l'arbre $T_{n,n',i}$ pour $i \in \mathbb{N}$ obtenu en itérant i fois la partie de l'arbre située entre n et n'. (Cette construction peut être formalisée de manière plus commode avec les contextes, mais on n'entrera pas dans ces détails ici.)

Le lemme de pompage est le suivant : pour tout langage reconnaissable de Σ-arbres L, il existe un entier $k \in \mathbb{N}$ (la constante de pompage, intuitivement un entier exponentiel devant le nombre d'états d'un DbTA qui accepte L) tel que, pour tout arbre T de L ayant plus de k nœuds, il existe deux nœuds internes n et n' de T avec n ancêtre strict de n' tel que $T_{n,n',i}$ soit dans L pour tout $i \in \mathbb{N}$. (Si on rapporte cela au lemme de pompage habituel, la condition imposant que le mot du milieu ne soit pas vide revient à imposer que n soit un ancêtre *strict* de n', et la condition sur la longueur des deux premiers mots peut se ramener à une borne sur la profondeur de n et sur la distance entre n et n' que l'on peut borner par k.)

Esquissons à présent la démonstration du lemme de pompage. Soit L un langage reconnaissable de Σ-arbres, et soit A un DbTA reconnaissant L. Soit k' son nombre d'états. Prenons $k := 2^{k'+3}$. Soit à présent un arbre T de L ayant plus que k nœuds. Comme T est binaire, sa hauteur est strictement supérieure à $k + 2$, donc il a une feuille n tel que le chemin C de la racine à n passe par $k' + 1$ nœuds internes. Considérons le run acceptant ρ de A sur T, et les états obtenus sur les nœuds de C. Par le principe des tiroirs, il y a deux nœuds internes distincts de C, qu'on appellera n et n' avec n ancêtre strict de n', où $\rho(n) = \rho(n')$. On peut alors considérer $T_{n,n',i}$ pour un $i \in \mathbb{N}$ quelconque, et construire un run acceptant ρ_i de A sur $T_{n,n',i}$ à partir de ρ. Intuitivement, on répète le run dans la partie répétée, et le fait que $\rho(n) = \rho(n')$ garantit qu'on peut recoller les parties en continuant à respecter δ.

Pour en déduire ce qui est demandé : considérons le langage L de la question et soit k sa constante de pompage. Soit T un arbre de L avec plus de k nœuds. En appliquant le lemme de pompage, soient n et n' deux nœuds internes avec n ancêtre strict de n'. Soit n' est descendant de l'enfant gauche de la racine, soit de l'enfant droit. On traite le premier cas, le second est symétrique. On constate alors que, dans $T_{n,n',2}$, le nombre de feuilles descendantes de l'enfant droit de la racine est inchangé relativement à T, alors que le nombre de feuilles descendantes de l'enfant gauche a augmenté : en effet on a remplacé le sous-arbre de T enraciné en n' par une copie du sous-arbre de T enraciné en n, et comme n est ancêtre strict de n' celui-ci a plus de feuilles que celui-là. On

en conclut que $T_{n,n',2}$ appartient au langage d'après le lemme de pompage alors qu'il ne satisfait pas les conditions du langage ; c'est absurde.

3 3-coloriage avec contraintes

Question 0. Il est manifeste que l'unique coloriage à considérer est celui déjà donné par la couleur des nœuds, et qu'il est valide si et seulement si l'arbre donné ne contient pas deux nœuds adjacents de la même couleur.

Pour construire un DbTA reconnaissant ce langage, on prend simplement les états $Q = \{r, g, b\}$ qui sont tous finaux. La fonction initiale associe chaque nœud étiqueté $x \in \{R, G, B\}$ à l'état correspondant. Pour la fonction de transition, étant donné les états q et q' et le nœud x, si on constate que la couleur de x est la même que celle de q ou celle de q' alors on ne définit pas de transition et on rejette ; sinon on transitionne vers l'état x. (Bien sûr on ne définit pas de transition pour un nœud étiqueté $\perp$ pour rejeter les arbres qui ne satisfont pas l'hypothèse de la question.)

Question 1. On transforme le DbTA précédent en un NbTA comme suit : on ajoute des transitions $(\perp, q)$ pour chaque $q \in \{r, g, b\}$. Du reste, sur les nœuds internes étiquetés $x = \perp$, pour chaque paire d'états (q, q'), on a une transition $(q, q', \perp, q'')$ pour chaque $q'' \notin \{q, q'\}$.

L'intuition est que l'automate résultant a un run sur un arbre si et seulement si on peut choisir une couleur pour chaque nœud $\perp$ de sorte à obtenir un coloriage qui soit valide (noter que le run correspondant est aussi un run de l'automate de la question précédente sur l'arbre où les nœuds $\perp$ sont coloriés comme indiqué par le run).

Question 2. Étant donné un Σ-arbre, on peut calculer de bas en haut l'ensemble des états que l'on peut obtenir en chaque nœud. (On peut aussi déterminiser le NbTA pour que l'évaluation soit plus simple, mais ce n'est pas nécessaire.) Cet algorithme est en temps linéaire en l'arbre à automate fixé (comme c'est le cas ici). On en déduit donc que l'on peut déterminer en temps linéaire si un Σ-arbre est coloriable ou non. Noter que ce n'est pas évident a priori !

(Une compréhension intuitive des Σ-arbres coloriables : tant qu'il y a deux nœuds de même parent qui portent des couleurs différentes alors on colorie le parent avec l'unique couleur possible, et si au terme de ce processus on a deux nœuds adjacents de même couleur on rejette. Sinon, on a maintenant l'invariant (*) : aucun nœud incolore a deux enfants coloriés de même couleur. On peut toujours itérativement compléter pour obtenir un coloriage valide : pour chaque nœud non colorié le plus haut possible, on lui attribue une couleur compatible avec son parent et ses enfants (noter qu'il en existe forcément une car ses enfants ne peuvent être tous deux coloriés de la même couleur). Comme le parent du nœud nouvellement colorié n'était pas incolore, l'invariant (*) est toujours respecté.)

Question 3. On souhaite compter le nombre de runs acceptants du NbTA A de la question 1, car il est manifeste que ces runs sont en bijection avec les coloriages valides. On procède par programmation dynamique : sur un Σ-arbre T, pour chaque nœud n de T, pour chaque état q de A, on stocke dans la case $T[n][q]$ du tableau le nombre de runs ρ de A sur le sous-arbre T_n de T où $\rho(n) = q$.

Le cas de base pour une feuille n étiquetée x est que $T[n][q]$ vaut 1 ou 0 selon que $(x, q) \in \iota$ ou non.

Le cas d'induction est le suivant. Pour un nœud interne n étiqueté x et ayant pour enfants n_1 et n_2, pour chaque état q de A, on souhaite calculer le nombre de runs ρ de A sur T_n tels que $\rho(n) = q$. Or, on sait qu'un run ρ de A sur T_n satisfaisant cette condition se définit en choisissant des états q_1 et q_2 tels que $(q_1, q_2, x, q) \in \delta$ ainsi qu'un run ρ_1 de A sur T_{n_1} tel que $\rho_1(n_1) = q_1$ et un run ρ_2 de A sur T_{n_2} tel que $\rho_2(n_1) = q_2$. À noter que le choix de q_1 et q_2 partitionne les runs ρ sur T_n tels que $\rho(n) = q$, donc on peut compter ces derniers en sommant sur les choix de q_1 et q_2. Du reste, par hypothèse d'induction, le nombre de runs ρ_1 et ρ_2 satisfaisant les conditions est donné par $T[n_1][q_1]$ et $T[n_2][q_2]$ respectivement. En définitive, pour chaque $q \in Q$, on définit :

$$T[n][q] := \sum_{(q_1,q_2,x,q)\in\delta} T[n_1][q_1] \times T[n_2][q_2]$$

(À noter que si l'un des facteurs du produit vaut 0 cela élimine les q_1 et q_2 correspondants.)

On veut obtenir à la fin le nombre total de runs, donc on veut $\sum_q T[r][q]$ pour r la racine de T.

Ainsi, le nombre de coloriages valides de T peut être calculé en temps linéaire.