Composition d'Informatique n°2

Sujet 1 : Mots synchronisants

Proposition de corrigé

⋆ ⋆ ⋆

Remarques fréquentes

- 3A. Il faut bien lire l'énoncé, il est demandé un mot de 3 lettres, pas 4 !
- 4A. Une chaîne de caractères n'est pas une liste et ne peut pas être manipulée efficacement comme telle ! Cette structure s'apparente plutôt à celle d'un tableau de caractères (avec une syntaxe particulière toutefois).
- 5A. On peut ne faire qu'un nombre linéaire d'appels à `delta_etoile`, inutile de parcourir tous les couples d'états.
- 8A. Comme mentionné plusieurs fois en cours, lorsqu'on applique l'algorithme des parties, il est attendu de représenter la table de transitions avant de dessiner l'automate ! Par ailleurs, il faut commencer par l'ensemble des états initiaux, pour éviter d'avoir à représenter des parties non accessibles.
- 14A. Il fallait mentionner ici que la suite des cardinaux était ultimement égale à 1.
- 16A. On attend la construction d'une matrice d'entiers pour respecter le type imposé ! Une matrice de couples ne répondait pas à la question. Il fallait utiliser la bijection donnée par l'énoncé pour faire le lien entre couple d'états et entier associé.

1 Machines synchronisées

Question 1 Sur une machine à un seul état q, les transitions sont exactement les $\delta(q,a) = q$ pour $a \in \Sigma$, et on en déduit que tous les mots sont synchronisants.

Question 2 On montre par récurrence que pour $n \in \mathbb{N}$, $\delta^*(q_i, a^{2n}) = q_i$ (et de la même manière, $\delta^*(q_i, a^{2n+1}) = q_{1-i}$).

- si $n = 0$, $\delta^*(q_i, \varepsilon) = q_i$ par définition ;
- si le résultat est vrai pour $n \in \mathbb{N}$ fixé, alors comme $\delta^*(q_i, aa) = q_i$, on en déduit qu'il reste vrai pour $n+1$.

On en déduit que pour tout mot u, $\delta^*(q_0, u) \neq \delta^*(q_1, u)$, donc qu'il n'existe pas de mot synchronisant.

Question 3 On remarque que la lecture d'un a emmène à l'état q_1 ou l'état q_2. De plus, la lecture de da depuis l'un de ces deux états emmène nécessairement vers l'état q_1. On en déduit que ada est un mot synchronisant pour M_2. D'autres mots peuvent convenir, comme bca, bdb, aca, bcb, acb, adb.

Question 4 On calcule une transition tant pour chaque lettre du mot u, parcouru par une boucle. On utilise la fonction `code` pour passer d'un caractère à un entier dans le bon intervalle.

```
let delta_etoile delta q u =
    let rq = ref q in
    for i = 0 to String.length u - 1 do
        rq := delta.(!rq).(code u.[i])
    done;
    !rq
```

Question 5 Il faut tester pour chaque état que la lecture du mot emmène vers un unique état. Pour cela, on fait le calcul depuis l'état 0, puis on utilise une boucle pour vérifier que l'image depuis chaque autre état mène bien au même état. On vérifie que la condition de sortie de boucle est qu'on a fait la vérification pour tous les états.

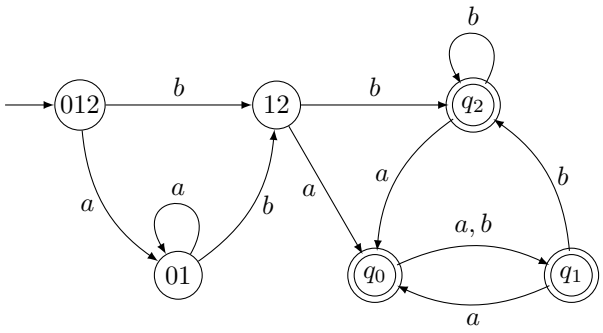
```
let synchronisant delta u =
    let p = delta_etoile delta 0 u in
    let i = ref 1 and n = Array.length delta in
    while !i < n && p = delta_etoile delta !i u do
        incr i
    done;
    !i = n
```

Question 6 On remarque que s'il existe un mot synchronisant u pour M, tel que $\forall q \in Q$, $\delta^*(q,u) = q_0$, alors $\widehat{\delta}^*(Q,u) = \{q_0\}$. On en déduit qu'il existe un mot synchronisant pour M si et seulement si un singleton est accessible depuis l'état $Q \in \widehat{Q}$ dans $\widehat{M}$.

Question 7 On définit l'automate déterministe $(\widehat{Q}, \Sigma, \widehat{\delta}, Q, \{\{q\} | q \in Q\})$. D'après la proposition précédente, cet automate reconnaît bien l'ensemble de tous les mots synchronisants de M. On en déduit que $LS(M)$ est reconnaissable.

Question 8 On détermine l'automate des parties sous forme de tableau, puis on le représente en enlevant les états non utiles (l'état initial étant Q) :

	a	b
$\{q_0, q_1, q_2\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_1, q_2\}$
$\{q_1, q_2\}$	$\{q_0\}$	$\{q_2\}$
$\{q_0\}$	$\{q_1\}$	$\{q_1\}$
$\{q_1\}$	$\{q_0\}$	$\{q_2\}$
$\{q_2\}$	$\{q_0\}$	$\{q_2\}$



Notons qu'on peut simplifier cet automate en fusionnant q_0, q_1 et q_2 d'une part et 012 et 01 d'autre part.

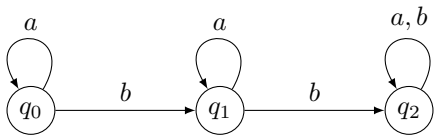
Question 9 Les mots synchronisants pour M_0 minimaux par sous-mot sont uniquement ba et bb.

Question 10 On remarque qu'un mot synchronisant u pour une machine $M = (Q, \Sigma, \delta)$ étiquette un chemin de $Q \in \widehat{Q}$ à un singleton, dans la machine des parties $\widehat{M}$. Par ailleurs, si ce chemin contient un cycle non vide, la suppression de cycle donne un chemin étiqueté par un mot v qui est sous-mot strict de u, et qui est toujours synchronisant.

On en déduit qu'une condition nécessaire (mais pas suffisante) pour qu'un mot synchronisant soit minimal par sous-mot est que le chemin correspondant dans $\widehat{M}$ ne contienne pas de cycle. Or, dans un graphe, il n'existe

qu'un nombre fini de chemins sans cycle, ce qui conclut.

Question 11 On considère la machine suivante.



Il est clair qu'un mot est synchronisant si et seulement s'il contient au moins deux b. En particulier, tous les mots de la forme ba^nb sont synchronisants. Pourtant, aucun facteur strict de ba^nb n'est synchronisant, car un facteur strict ne contiendrait qu'un seul b. Il existe donc bien une infinité de mots synchronisants minimaux par facteur.

Question 12 Raisonnons par l'absurde et supposons que $\widehat{\delta}(Q, a^m) = \widehat{\delta}(Q, a^n) = X$ pour $m < n$.

- sachant que $a^nb^na^n$ est synchronisant, $\widehat{\delta}(X, b^na^n) = \widehat{\delta}(Q, a^nb^na^n)$ est un singleton ;
- le mot $a^mb^na^n$ est un facteur strict de $a^nb^na^n$, donc par hypothèse, il n'est pas synchronisant. On en déduit que $\widehat{\delta}(X, b^na^n) = \widehat{\delta}(Q, a^mb^na^n)$ n'est pas un singleton.

Ainsi, la partie $\widehat{\delta}(X, b^na^n)$ doit être à la fois un singleton et un non-singleton. C'est absurde, donc on en déduit que $\widehat{\delta}(Q, a^m) \neq \widehat{\delta}(Q, a^n)$.

Dès lors, on en déduit que $n \mapsto \widehat{\delta}(Q, a^n)$ est une injection de $\mathbb{N}$ dans $\widehat{Q}$. C'est absurde, car $\widehat{Q}$ est fini. On en déduit qu'une telle machine M ne peut pas exister.

2 Existence de mot synchronisant

Question 13 Il s'agit d'implémenter un parcours de graphe. Ici, les étiquettes des transitions empruntées n'importent pas.

```
let accessibles delta q =
    let vus = Array.make (Array.length delta) false in
    let rec dfs p =
        if not vus.(p) then begin
            vus.(p) <- true;
            Array.iter dfs delta.(p)
        end
    in
    dfs q;
    vus
```

Pour la complexité, c'est celle d'un parcours de graphe, c'est-à-dire linéaire en le nombre d'arêtes et de sommets. Or, le nombre de sommets est $|Q|$, et le nombre d'arêtes est $|Q| \times |\Sigma|$, car chaque état possède exactement $|\Sigma|$ transitions sortantes, la machine étant déterministe et complète.

Question 14 On a nécessairement $|\delta^*(Q, u_0)| \geqslant |\delta^*(Q, u_1)| \geqslant \ldots \geqslant |\delta^*(Q, u_r)| = 1$. En effet la machine étant déterministe, en lisant un mot depuis un état, on ne peut atteindre qu'un seul état. En lisant un mot depuis un ensemble d'états, le nombre d'états atteignables ne peut que diminuer (en cas de collisions). De plus, comme $u = u_r$ est synchronisant, $|\delta^*(Q, u_r)| = 1$.

Question 15 Le sens direct est le plus simple, car $u_{p,q} = u$ convient (u est synchronisant).

Réciproquement, définissons les suites :

– $u_0 = v_0 = \varepsilon$ et $Q_0 = Q$;
– pour $i \in \mathbb{N}$, si $|Q_i| = 1$, alors on s'arrête, sinon, il existe deux états p et q de Q_i et un mot $u_{i+1} = u_{p,q}$ tel que $|Q_{i+1}| = |\delta^*(Q_i, u_{i+1})| < |Q_i|$. On pose alors $v_{i+1} = v_i u_{i+1}$.

La suite des $(|Q_i|)$ est strictement décroissante et minorée par 1, donc à partir d'un rang $j < n$, on a $|Q_j| = 1$, donc le mot v_j est synchronisant (car $Q_j = \delta^*(Q_0, v_j)$ par définition des suites et de δ^*).

Question 16 On suit la définition et on remplit la matrice par une triple boucle. On pense bien à convertir les couples en numéro.

```
let machine_couples delta =
    let n = Array.length delta and k = Array.length delta.(0) in
    let delta_tilde = Array.make_matrix (n * n) k 0 in
    for p = 0 to n - 1 do
        for q = 0 to n - 1 do
            for a = 0 to k - 1 do
                let pq = p * n + q in
                delta_tilde.(pq).(a) <- delta.(p).(a) * n + delta.(q).(a)
            done
        done
    done;
    delta_tilde
```

Question 17 On crée la machine des couples et on cherche les états accessibles depuis le couple (p, q). On vérifie ensuite s'il existe un singleton accessible (c'est-à-dire un couple (r, r), donc de numéro de la forme $r(n+1)$).

```
let couple_synchronise delta pq =
    let delta_tilde = machine_couples delta in
    let r = ref 0 and n = Array.length delta in
    let acces = accessibles delta_tilde pq in
    while !r < n && not acces.(!r * (n + 1)) do
        incr r
    done;
    !r < n
```

Question 18 On parcourt chaque couple

```
let synchronisee delta =
    let n = Array.length delta in
    let pq = ref 0 in
    while !pq < n * n && couple_synchronise delta !pq do
        incr pq
    done;
    !pq = n * n
```

Question 19 La fonction `couple_synchronisee` consiste à créer la machine des couples (en $\mathcal{O}(|Q|^2|\Sigma|)$), puis à faire un parcours en profondeur ((n $\mathcal{O}(|Q|^2|\Sigma|$ à nouveau, la machine des couples ayant $|Q|^2$ états), soit $\mathcal{O}(|Q|^2|\Sigma)$ au total. On l'appelle autant de fois qu'il y a de couples.

La complexité totale est donc en $\mathcal{O}(|Q|^4|\Sigma|)$.

Question 20 Au lieu de faire autant de parcours qu'il y a de couples, on peut ne faire qu'un seul parcours, mais dans l'automate transposé (où on a retourné toutes les transitions). L'idée est alors de faire un parcours

depuis tous les états singletons puis de vérifier qu'on a atteint tous les autres états. Comme il s'agit d'un seul parcours, on aurait bien une complexité en $\mathcal{O}(|Q|^2|\Sigma|)$.

Une autre solution est de faire des parcours dans la machine des couples, mais en gardant en mémoire les couples déjà explorés.

3 Bornes sur les mots synchronisants

Question 21 On commence par montrer qu'un couple d'états $(p, q) \subseteq Q$ peut toujours être synchronisé par un mot $u_{p,q}$ de longueur $\leqslant \binom{n}{2}$.

En effet, dans la machine $\widetilde{M}$, un plus court chemin de (p, q) à un état (r, r) comporte au plus $\binom{n}{2} + 1$ états. Ceci est dû au fait qu'un plus court chemin ne peut pas passer par deux états de la forme (s, t) et (t, s). Un tel chemin contient donc au plus $\binom{n}{2}$ paires et un singleton en dernier état. Le mot lu le long de ce chemin est bien un mot synchronisant (p, q).

De plus, en reprenant les notations de la question 15, chaque mot u_i pour $i > 0$ est un mot synchronisant un couple, et le mot synchronisant construit est la concaténation d'au plus $n - 1$ des u_i, donc est de taille $\leqslant (n-1)\binom{n}{2} = \frac{n(n-1)^2}{2}$.

Question 22 Montrons que $u = a(b^{n-1}a)^{n-2}$ est un mot synchronisant pour $\mathcal{C}_n$, en considérant les $\delta^*(Q, u_i)$, pour u_i certains préfixes de u bien choisis. Posons, pour $i \in [\![0, n-2]\!]$, $u_i = a(b^{n-1}a)^i$. Montrons par récurrence sur i que $\delta^*(Q, u_i) = \{1, 2, \ldots, n-i-1\}$.

- pour $i = 0$, par définition de δ, on a bien $\delta^*(Q, u_0) = \delta(Q, a) = \{1, \ldots, n-1\}$;
- supposons le résultat vrai pour $i \geqslant 0$ fixé. Sachant que pour $q \in Q$, $\delta(q, b) = (q + 1) \bmod n$, on en déduit par une récurrence rapide que $\delta^*(\{1, 2, \ldots, n-i-1\}, b^{n-1}) = \{0, 1, \ldots, n-i-2\}$. Dès lors, $\delta^*(Q, u_{i+1}) = \delta(\delta^*(\delta^*(Q, u_i), b^{n-1}), a) = \delta(\{0, 1, \ldots, n-i-2\}, a) = \{1, 2, \ldots, n-i-2\}$.

On conclut par récurrence. Finalement, $\delta^*(Q, u) = \delta^*(Q, u_{n-2}) = \{1\}$, donc u est synchronisant pour $\mathcal{C}_n$, et est bien de taille $1 + n \times (n-2) = (n-1)^2$.

Question 23 On commence par remarquer que pour $X \subseteq Q$, alors $\delta(X, b) = \{(q+1) \bmod n \mid q \in X\}$, donc $|\Delta(X)| = |\Delta(\delta(X, b))|$. Ainsi, $\alpha = a$.

De plus, $\delta(X, a) = X \setminus \{0\}$. On en déduit les propriétés suivantes :

- $0 \in X$ et $0 \in \Delta(\delta(X, a))$, sinon $\Delta(X) = \Delta(\delta(X, a))$;
- $1 \in \delta(X, a)$, car $\delta(0, a) = 1$, donc $1 \notin \Delta(\delta(X, a))$;
- en combinant les deux points précédents, $\Delta(\delta(X, a)) = \{n-j, n-j+1, \ldots, n-1, 0\}$.

Finalement, $\Delta(X) = \Delta(\delta(X, a)) \setminus \{0\} = \{n-j, n-j+1, \ldots, n-1\}$.

Question 24 Soit $u = a_1 a_2 \ldots a_k$ un mot synchronisant pour $\mathcal{C}_n$ de taille minimale. Pour $n = 1$, $u = \varepsilon$, de taille $0 = (1-1)^2$, est synchronisant. Pour $n = 2$, ε n'est pas synchronisant (car il y a deux états), mais $u = a$, de taille $1 = (2-1)^2$ est synchronisant. Pour la suite, supposons $n \geqslant 3$.

On commence par remarquer que $a_1 = a$, car $\delta(Q, b) = Q$, donc si bv est synchronisant, alors v aussi. Par minimalité de la taille de u, $a_1 = a$.

Posons, pour $i = 1, \ldots, k$, $X_i = \delta^*(Q, a_1 a_2 \ldots a_i)$. On a $X_1 = \{1, 2, \ldots, n-1\}$ (car $a_1 = a$) et $X_k = \{1\}$ (car u est synchronisant et que 1 est le seul état qui a deux transitions entrantes étiquetées par une même lettre).

Soit alors $i \in [\![2, k]\!]$ et $j \in [\![1, n-2]\!]$ tels que $|\Delta(X_{i-1})| = j$ et $|\Delta(X_i)| = j + 1$. Par la question précédente, $\Delta(X_{i-1}) = \{n-j, n-j+1, \ldots, n-1\}$ et $\Delta(X_i) = \{n-j, n-j+1, \ldots, n-1, 0\}$.

De plus, soit $\ell > 0$ tel que $|\Delta(X_{i+\ell})| = j + 2$. Alors $\ell \geqslant n$, car b^{n-1} est le mot le plus court qui peut transformer $\Delta(X_i)$ en $\{n-j-1, n-j, \ldots, n-1\}$ (on applique b $n-1$ fois pour faire « tourner » le « trou » autour de l'automate).

Finalement, $k \geqslant 1 + (n-2) \times n = (n-1)^2$ (il faut augmenter la taille de $\Delta(X_i)$ un total de $n-2$ fois, de 1 à $n-1$).

4 Réduction depuis SAT

Question 25 On se contente d'évaluer récursivement la formule.

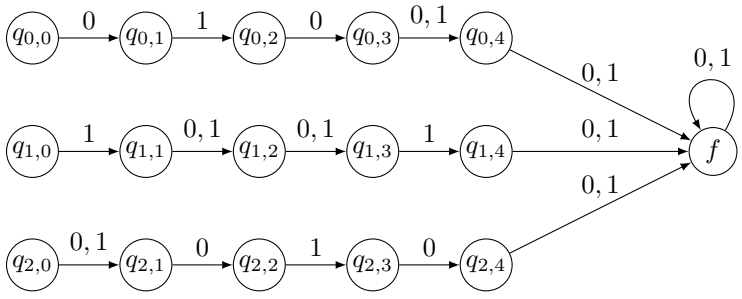
```
let rec interpretation mu = function
    | Var i -> mu.(i)
    | Non phi -> not (interpretation mu phi)
    | Ou (phi, psi) -> interpretation mu phi || interpretation mu psi
    | Et (phi, psi) -> interpretation mu phi && interpretation mu psi
```

Question 26 On crée un tableau correspondant à une valuation, et on utilise une fonction auxiliaire pour le remplir. Cette fonction prend en argument l'indice de la case en cours de modification. Si on est arrivé à la fin du tableau, on teste si c'est un modèle de la formule (si c'est le cas, on arrête les calculs). Sinon, on teste récursivement en y mettant un `false`, puis en y mettant un `true`.

```
let satisfiable phi n =
    let mu = Array.make n false in
    let rec backtracking i =
        if i = n && interpretation mu phi then
            raise Exit
        else if i < n then begin
            mu.(i) <- false; backtracking (i + 1);
            mu.(i) <- true; backtracking (i + 1)
        end
    in
    try backtracking 0; None
    with _ -> Some mu
```

Question 27 La fonction `interpretation` a une complexité linéaire en $|\varphi|$. Dans le pire cas, pour une formule non satisfiable, la fonction `satisfiable` appellera `interpretation` pour chaque valuation possible. Les autres opérations étant en temps constant, la complexité totale est en $\mathcal{O}(|\varphi|2^n)$.

Question 28 On obtient la machine suivante, à 16 états :



Pour alléger la figure, on a choisi de ne de pas représenter les transitions manquantes, qui doivent pointer vers f.

Question 29 On propose le modèle μ vérifiant $\mu(x_0) = \mu(x_1) = \mu(x_2) = 1$ et $\mu(x_3) = 0$. La première clause est satisfaite par x_0, la deuxième par x_3 et la troisième par x_1. On constate que le mot 1110 est bien

un mot synchronisant dans M_{φ_0}.

Question 30 On peut montrer que si u est un mot de taille $\geqslant n+1-\ell$, alors pour tout i, $\delta^*(q_{i,\ell}, u) = f$:

- c'est vrai si $\ell = n$ car $\delta(q_{i,n}, 0) = \delta(q_{i,n}, 1) = f$ et $\delta^*(f, v) = f$ pour tout mot v ;
- si on suppose que c'est vrai pour un certain ℓ, alors c'est vrai pour $\ell - 1$. En effet, si $u = av$, alors $\delta(q_{i,\ell-1}, a)$ vaut soit $q_{i,\ell}$, soit f. Dans les deux cas, par hypothèse, la lecture de v amène dans f.

On conclut par récurrence descendante qu'un mot de taille $n+1$ est synchronisant.

Un mot u de longueur n est synchronisant pour M_φ si et seulement si pour chaque $i \in [\![0, k-1]\!]$, $\delta^*(q_{i,0}, u) = f$ (car la lecture du mot u depuis n'importe quel autre état amène en f).

Question 31 Supposons que φ est satisfiable et soit μ un modèle de φ. Montrons que le mot $u = \mu(x_0)\mu(x_1)\ldots\mu(x_{n-1})$ est synchronisant.

Soient $i \in [\![0, k-1]\!]$ et $j \in [\![0, n-1]\!]$, c_i une clause de φ et ℓ_j un littéral x_j ou $\overline{x_j}$ satisfaisant c_i pour μ. On a donc $\mu(\ell_j) = 1$. Par définition de δ, on a $\delta(q_{i,j}, \mu(x_j)) = f$. En effet :

- si $\ell_j = x_j$, alors $\mu(x_j) = \mu(\ell_j) = 1$ et $\delta(q_{i,j}, 1) = f$ car x_j apparaît dans c_i ;
- si $\ell_j = \overline{x_j}$, alors $\mu(x_j) = 1 - \mu(\ell_j) = 0$ et $\delta(q_{i,j}, 0) = f$ car $\overline{x_j}$ apparaît dans c_i.

Si on suppose j minimal tel que ℓ_j est un littéral satisfaisant c_i, alors on a :

$$\delta^*(q_{i,0}, u) = \delta^*(\delta^*(q_{i,0}, \mu(x_0)\ldots\mu(x_{j-1})), \mu(x_j)\ldots\mu(x_{n-1})) = \delta^*(q_{i,j}, \mu(x_j)\ldots\mu(x_{n-1})) = \delta^*(f, \mu(x_{j+1})\ldots\mu(x_{n-1})) = f$$

Ce raisonnement pouvant être fait pour tout i (car φ est satisfaite par μ), on en déduit que u est bien synchronisant.

Question 32 Remarquons que, f étant un état puits, s'il existe un mot synchronisant u de longueur inférieure ou égale à n, alors pour tout $q \in Q$, $\delta^*(q, u) = f$. Sans perte de généralité, on peut supposer u de longueur exactement n, quitte à rajouter des lettres.

Posons $u = a_0\ldots a_{n-1}$ et définissons la valuation μ par $\mu(x_i) = a_i$. Montrons que μ est un modèle de φ. Soit $i \in [\![0, k-1]\!]$. Sachant que $\delta^*(q_{i,0}, u) = f$, il existe nécessairement $j \in [\![0, n-1]\!]$ tel que $\delta(q_{i,j}, a_j) = f$ (sinon $\delta^*(q_{i,0}, u) = q_{i,n} \neq f$). Par définition de δ, cela signifie l'une des deux choses :

- $a_j = 0$ donc $\overline{x_j}$ apparaît dans c_i, et on a $\mu(x_j) = a_j = 0$, donc c_i est satisfaite par μ ;
- $a_j = 1$ donc x_j apparaît dans c_i, et on a $\mu(x_j) = a_j = 1$, donc c_i est satisfaite par μ.

Dans tous les cas, c_i est satisfaite par μ, donc μ est un modèle de φ.

Question bonus La conjecture de Černý, énoncée en 1964 et toujours un problème ouvert à ce jour, affirme que toute machine synchronisée à n états possède un mot synchronisant de taille inférieure ou égale à $(n-1)^2$. Cette question n'a donc pas de preuve connue !

⋆ ⋆ ⋆