Mots synchronisants (en C)

30 septembre 2024

Une machine sera représentée par un objet de type `machine` défini par :

```
struct Machine{
    int tQ;
    int tSigma;
    int** delta;
};

typedef struct Machine machine;
```

tel que si $M = (Q, \Sigma, \delta)$ est une machine représenté par un objet `M` de type `machine*`, alors :

— `M->tQ` représente $|Q|$, on suppose $Q = [\![0, |Q| - 1]\!]$;

— `M->tSigma` représente $|\Sigma|$, on suppose $\Sigma = [\![0, |\Sigma| - 1]\!]$;

— `M->delta` correspond à un tableau des transitions, c'est-à-dire tel que si $q \in Q$ et $a \in \Sigma$, alors `M->delta[q][a]` vaut $\delta(q, a)$.

1. On alloue la mémoire comme il faut. Il faut penser à créer, allouer et initialiser chaque sous-tableau (c'est ce que font les deux boucles `for`).

```
machine* init_machine(int tQ, int tSigma){
    machine* M = malloc(sizeof(*M));
    int** delta = malloc(sizeof(*delta));
    for (int q=0; q<tQ; q++){
        int* delta_q = malloc(sizeof(*delta_q));
        for (int a=0; a<tSigma; a++){
            delta_q[a] = q;
        }
        delta[q] = delta_q;
    }
    M->tQ = tQ; M->tSigma = tSigma;
    M->delta = delta;
    return M;
}
```

2. Il faut bien libérer chaque sous-tableau avant le libérer le tableau `M->delta`. Ensuite, on peut libérer `M`.

```
void liberer_machine(machine* M){
    for (int q=0; q<M->tQ; q++){
        free(M->delta[q]);
    }
    free(M->delta);
    free(M);
}
```

3. Sur une machine à un seul état q, on a toutes les transitions $\delta(q, a) = q$ pour $a \in \Sigma$, et on en déduit que tous les mots sont synchronisants.
   Dans toute la suite du problème, on supposera que les machines ont au moins deux états.

4. On remarque que si $|u| \equiv 0[2]$, alors $\delta^*(q_i, u) = q_i$ et si $|u| \equiv 1[2]$, alors $\delta^*(q_i, u) = q_{1-i}$. 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.

5. 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.
6. On calcule une transition tant qu'on n'est pas arrivé sur le caractère de fin du mot `u`. On pense à faire le décalage de 97 pour passer d'un caractère à un entier dans le bon intervalle.

```
int delta_etoile(machine* M, int q, char* u){
    int p = q, i = 0;
    while (u[i] != '\0'){
        p = M->delta[p][u[i]-97];
        i++;
    }
    return p;
}
```

7. 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.

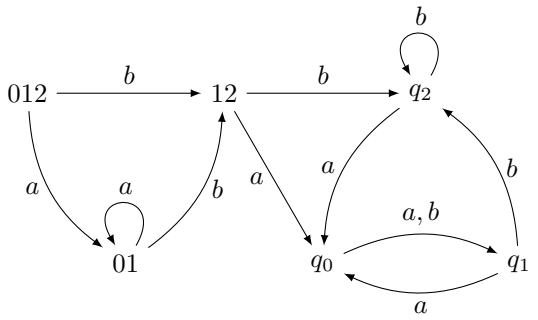
```
bool synchonisant(machine* M, char* u){
    int p = delta_etoile(M, 0, u);
    for (int q=1; q<M->tQ; q++){
        if (delta_etoile(M, q, u) != p){
            return false;
        }
    }
    return true;
}
```

8. Soit M une machine à au moins deux états admettant un mot synchronisant $u = u_1 \dots u_k$ où les $u_i \in \Sigma$. Soient $q \neq q'$ deux états de M. Alors $\delta^*(q,u) = \delta^*(q',u)$. Notons $q = q_0 \overset{u_1}{\to} q_1 \overset{u_2}{\to} \dots \overset{u_k}{\to} q_k = \delta^*(q,u)$ et $q' = q'_0 \overset{u_1}{\to} q'_1 \overset{u_2}{\to} \dots \overset{u_k}{\to} q'_k = q_k$ des calculs de u depuis les états q et q'. Comme $q_0 \neq q'_0$ et $q_k = q'_k$, on en déduit qu'il existe $i \in [\![0, k-1]\!]$ minimal tel que $q_i \neq q'_i$ et $q_{i+1} = q'_{i+1}$. Dès lors, on a $\delta(q_i, u_{i+1}) = \delta(q'_i, u_{i+1})$.
   Soit $LS(M)$ le langage des mots synchronisants d'une machine $M = (Q, \Sigma, \delta)$. On introduit la machine des parties $\widehat{M} = (\widehat{Q}, \Sigma, \widehat{\delta})$ où $\widehat{Q} = \mathcal{P}(Q)$ et $\widehat{\delta}$ est définie par :

$$\forall P \subset Q, \forall a \in \Sigma, \widehat{\delta}(P,a) = \{\delta(p,a), p \in P\}$$

9. 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 $\widehat{M}$.
10. 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.
11. 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.

12. Soit $M = (Q, \Sigma, \delta)$ une machine et q_0 un état de M. On pose $M' = (Q, \Sigma, \delta')$ telle que :
    - $\forall q \in Q \setminus \{q_0\}$, $\forall a \in \Sigma$, $\delta'(q, a) = \delta(q, a)$;
    - $\forall a \in \Sigma$, $\delta'(q_0, a) = q_0$.

    On remarque qu'avec cette construction, si u est un mot synchronisant, alors $\forall q \in Q$, $\delta'^*(q, u) = q_0$. On en déduit que le chemin menant de q à $\delta^*(q, u)$ passe forcément par q_0. Réciproquement, si le chemin menant de q à $\delta^*(q, u)$ passe forcément par q_0, alors u est synchronisant.