Modélisation de systèmes par automates finis

Table des matières

1 Introduction : modélisation par automates finis **1**

2 Systèmes de transitions et automates finis **2**
2.1 Définitions générales 2
2.2 Equivalence de langages et déterminisation 2
2.3 Modèles de calcul 3

3 Opérations sur les automates finis **4**
3.1 Union, intersection 4
3.2 Produit synchronisé 5

4 Exemples de modélisation **6**

1 Introduction : modélisation par automates finis

Exemple de problèmes pour lesquels on cherche à construire un modèle :

1. Comportement d'un digicode comportant 3 touches A, B, C, et qui ouvre la porte dès que l'utilisateur a tapé le code ABA.
2. Traversée d'une rivière pour un loup, une chèvre et une salade, qui disposent d'un passeur avec une barque, avec les contraintes suivantes :
   - le passeur ne peut prendre qu'un passager dans sa barque,
   - pour des raisons de survie, le loup ne peut pas rester seul avec la chèvre, et la chèvre ne peut pas rester seule avec la salade.
3. Stratégie gagnante d'un barman aveugle (avec des gants de boxe) pour retourner des verres sur un plateau et réussir à ce que les 4 verres soient dans le même sens.
4. Comportement d'un couple producteur/consommateur (a) avec tampon quelconque, ou (b) avec tampon borné.
5. Dominos de Wang : pavage d'un plan (a), ou d'un carré 5×5 (b), avec des dominos à 4 couleurs, de façon que deux couleurs adjacentes soient les mêmes.

Exercice 1.
Dessiner :
- un modèle de comportement du digicode,
- un modèle pour la traversée,
- des modèles pour les problèmes de producteur/consommateur.

2 Systèmes de transitions et automates finis

2.1 Définitions générales

Etant donné un alphabet Σ, on rappelle que l'ensemble des mots sur cet alphabet est noté Σ^* et qu'un langage est un sous-ensemble de Σ^*. Par exemple, si $\Sigma = \{a, b\}$ contient deux lettres, $\Sigma^* = \{\varepsilon, a, b, a^2, b^2, ab, ba, \ldots\}$.

L'opération de concaténation de deux mots, notée $w_1 \cdot w_2$ ou simplement $w_1 w_2$ pour deux mots w_1 et w_2 consiste à mettre w_2 tout de suite après w_1. Par exemple, la concaténation des deux mots ab et ba produit le mot $abba$. Le mot vide ε n'a aucune lettre et vérifie $w\varepsilon = \varepsilon w = w$ pour tout mot w de Σ^*.

Exercice 2. Décrire le langage du producteur/consommateur avec tampon non borné sur l'alphabet $\Sigma = \{a, b\}$, la lettre a représentant une opération de production et la lettre b une opération de consommation.

Définition 2.1.
Un système de transitions sur l'alphabet (d'actions) Σ *est un triplet* $\mathcal{T} = (S, S_0, T)$ *où* S *est l'ensemble des configurations,* S_0 *est le sous-ensemble de* S *des configurations initiales et* $T \subseteq S \times (\Sigma \cup \{\varepsilon\}) \times S$ *est l'ensemble des transitions.*

On note généralement $s \xrightarrow{a} s'$ une transition (s, a, s') de T. Une transition étiquetée par le mot vide ε correspond à l'exécution d'une action non observable, dite *action interne*.

Une exécution de $\mathcal{T}$ est un chemin dans ce graphe, c'est-à-dire une séquence de transitions $(s_0, a_1, s_1)(s_1, a_2, s_2)\ldots$ qui s'écrit aussi $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \ldots$, commençant dans une configuration initiale. Le mot $w = a_1 a_2 \ldots$ est l'*étiquette* du chemin (ou de l'exécution).

Définition 2.2. *Un automate fini est un système de transitions*
- *dont l'ensemble des configurations est fini (les éléments sont alors appelés des états),*
- *auquel on adjoint un sous-ensemble* F *de* S *correspondant à des états finals.*

Si $\mathcal{A} = (Q, Q_0, T, F)$ est un automate fini sur Σ, on peut lui associer le langage des mots acceptés, noté $L(\mathcal{A})$ et défini par :

un mot w de Σ^* appartient à $L(\mathcal{A})$ s'il existe une exécution finie $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \ldots \xrightarrow{a_n} q_n$ commençant dans un état initial ($q_0 \in Q_0$), se terminant dans un état final ($q_n \in F$) et ayant w comme étiquette.

2.2 Equivalence de langages et déterminisation

Deux automates finis $\mathcal{A}_1$ et $\mathcal{A}_2$ sur un même alphabet Σ sont langage-équivalents si $L(\mathcal{A}_1) = L(\mathcal{A}_2)$. Pour tester cette équivalence, on peut associer à chaque automate fini un unique (aux noms près) automate canonique et comparer les automates canoniques associés aux deux automates de départ $\mathcal{A}_1$ et $\mathcal{A}_2$. Cet automate canonique est obtenu à partir d'un automate $\mathcal{A}$ en construisant d'abord un automate déterministe équivalent à $\mathcal{A}$, puis en minimisant cet automate déterministe.

Définition 2.3. *Un automate fini* $\mathcal{A} = (Q, Q_0, T, F)$ *sur* Σ *est déterministe si :*
- *il a un unique état initial* $Q_0 = \{q_0\}$,
- *aucune transition n'est étiquetée par* ε,

– *pour tous les états* q, q_1, q_2 *de* Q *et pour toute lettre* $a \in \Sigma$*, si* $q \xrightarrow{a} q_1$ *et* $q \xrightarrow{a} q_2$ *alors* $q_1 = q_2$.

On a alors :

Proposition 2.4. *Pour tout automate fini* $\mathcal{A}$*, on peut construire un automate fini déterministe* $\mathcal{D}$ *qui soit langage-équivalent à* $\mathcal{A}$.

Partant de $\mathcal{A} = (Q, Q_0, T, F)$, la construction est la suivante :
– L'ensemble des états de $\mathcal{D}$ est l'ensemble des parties de Q, noté 2^Q ou $\mathcal{P}(Q)$.
– L'unique état initial de $\mathcal{D}$ est la partie

$$I = \{q \in Q \mid \text{ il existe un état } q_0 \in Q_0 \text{ et un chemin d'étiquette } \varepsilon \text{ de } q_0 \text{ à } q\}.$$

– Une partie P de Q est un état final de $\mathcal{D}$ ssi $P \cap F \neq \emptyset$.
– Pour toute partie P de Q et toute lettre a de Σ, la transition associée de $\mathcal{D}$ est $P \xrightarrow{a} P'$ avec :

$$P' = \{p' \in Q \mid \text{il existe des états } q \in Q \text{ et } p \in P \text{ tels que } p \xrightarrow{a} q \text{ est une transition dans } T \text{ et il existe un chemin d'étiquette } \varepsilon \text{ de } q \text{ à } p'\}.$$

Exercice 3.
Quel est le nombre maximal d'états de l'automate déterminisé ?
Comparer l'automate fini du digicode avec un automate acceptant le langage $\Sigma^* ABA$ des mots se terminant par ABA.

Exercice 4. Résoudre le problème de la stratégie du barman.

2.3 Modèles de calcul

Les machines à registres sont des modèles de calcul qui comportent un ensemble fini de registres manipulés par des opérations. Un compteur est un registre particulier qui peut contenir seulement une valeur entière et, à part l'initialisation, admet comme seules opérations : l'incrémentation et la décrémentation avec test à 0.

Définition 2.5.
Une machine $\mathcal{M}$ *à deux compteurs* C *et* D *est une suite finie d'instructions étiquetées. Chaque instruction peut être soit l'instruction particulière* $STOP$*, soit une opération sur un des deux compteurs* $X \in \{C, D\}$ *:*

1. $\ell : X := X + 1;$ *goto* ℓ';
2. $\ell :$ *si* $X > 0$ *alors* $X := X - 1;$ *goto* ℓ';
   sinon goto ℓ'';

La sémantique d'une machine à deux compteurs $\mathcal{M}$ est définie par un système de transitions $\mathcal{T} = (S, S_0, T)$ où :
- une configuration $s \in S$ est de la forme (ℓ, n, m) où ℓ est une étiquette, et n et m sont les valeurs respectives des compteurs C et D juste avant l'instruction ℓ.
- Il y a une seule configuration initiale : $S_0 = \{s_0\}$ avec $s_0 = (\ell_0, 0, 0)$, où ℓ_0 est l'étiquette de la première instruction.

- Pour une étiquette ℓ ne correspondant pas à l'instruction STP, une transition de T fait passer de la configuration (ℓ, n, m) à la configuration :
  - $(\ell', n+1, m)$ si l'instruction d'étiquette ℓ est de type 1. sur le compteur C,
  - $(\ell', n-1, m)$ si l'instruction d'étiquette ℓ est de type 2. sur le compteur C et que $n > 0$, ou $(\ell'', 0, m)$ si $n = 0$,
  - ou des configurations similaires pour des opérations sur le compteur D.

Exercice 5.
Construire une machine à deux compteurs qui réalise l'addition $3 + 2$.
En déduire un automate fini dont le langage soit une abstraction du comportement de cette machine.

Exercice 6. Par analogie, définir une machine à un seul compteur et montrer comment modéliser le comportement d'un producteur/consommateur avec tampon non borné avec une telle machine.

Exercice 7. Montrer que le langage $\{a^n b^n \mid n \geq 0\}$ ne peut pas être accepté par un automate fini.

Le problème de l'arrêt d'une machine à deux compteurs est défini par :
Donnée : une machine à deux compteurs
Question : existe-t-il une exécution qui atteint l'instruction $STOP$?

Théorème 2.6 (Minski, 1967)**.** *Le problème de l'arrêt d'une machine à deux compteurs est indécidable.*

Cela signifie qu'il n'existe pas d'algorithme qui, prenant en entrée une machine à deux compteurs, répond à la question par oui ou par non.

Exercice 8. Etudier la terminaison du problème de Collatz (appelé aussi conjecture de Syracuse), qui prend en entrée un entier naturel strictement positif :

```
Tant que n ≠ 1 faire
    si n est pair n := n/2
    sinon n := 3n + 1
```

3 Opérations sur les automates finis

3.1 Union, intersection

On considère deux automates finis $\mathcal{A}_1 = (Q_1, I_1, T_1, F_1)$ et $\mathcal{A}_2 = (Q_2, I_2, T_2, F_2)$ sur un même alphabet Σ et on note $L_1 = L(\mathcal{A}_1)$ et $L_2 = L(\mathcal{A}_2)$ les langages acceptés respectivement par ces deux automates.

En supposant les ensembles d'états Q_1 et Q_2 disjoints, un automate fini acceptant $L_1 \cup L_2$ est défini par $\mathcal{A} = (Q_1 \cup Q_2, I_1 \cup I_2, T_1 \cup T_2, F_1 \cup F_2)$. Autrement dit, il suffit de considérer ces deux automates comme un seul, et il n'y a pas d'ambiguïté sur les transitions du fait que $Q_1 \cap Q_2 = \emptyset$.

Pour l'intersection, **on suppose qu'il n'y a pas de transitions étiquetées par ε** et on réalise une synchronisation entre les deux automates, en simulant des exécutions parallèles dans $\mathcal{A}_1$ et dans $\mathcal{A}_2$. L'automate $\mathcal{A}_1 \times \mathcal{A}_2$ accepte $L_1 \cap L_2$, avec $\mathcal{A}_1 \times \mathcal{A}_2 = (Q_1 \times Q_2, I_1 \times I_2, T, F_1 \times F_2)$, l'ensemble T des transitions étant défini par :

une transition $(q_1, q_2) \xrightarrow{a} (q_1', q_2')$ est dans T si et seulement si $q_1 \xrightarrow{a} q_1'$ est dans T_1 et $q_2 \xrightarrow{a} q_2'$ est dans T_2.

Exercice 9.
Sur l'alphabet $\{0, 1\}$, construire un automate $\mathcal{A}_1$ qui accepte les multiples de 3 (écrits en binaire) et un automate $\mathcal{A}_2$ qui accepte les multiples de 4 (également écrits en binaire).
En déduire un automate qui accepte les multiples 12.

Exercice 10. Dans le cas général, donner la construction d'un automate fini qui accepte le produit L_1L_2. On rappelle que ce produit qui étend la concaténation contient les mots de la forme w_1w_2, où w_1 appartient à L_1 et w_2 appartient à L_2.

Donner aussi la construction d'un automate fini qui accepte le produit L_1^*. On rappelle que ce langage contient les produits d'un nombre quelconque de mots de L_1.

Exercice 11. Montrer que le langage du producteur/consommateur avec tampon non borné ne peut pas être accepté par un automate fini.

3.2 Produit synchronisé

On considère trois alphabets Σ_1, Σ_2 et Σ et deux automates finis $\mathcal{A}_1 = (Q_1, I_1, T_1, F_1)$ sur Σ_1 et $\mathcal{A}_2 = (Q_2, I_2, T_2, F_2)$ sur Σ_2, et une fonction de synchronisation $f : (\Sigma_1 \cup \{-\}) \times (\Sigma_2 \cup \{-\}) \mapsto \Sigma \cup \{\varepsilon\}$ (la fonction est partielle, en particulier $f(-,-)$ n'est pas défini). Le produit synchronisé de $\mathcal{A}_1$ et de $\mathcal{A}_2$ par f, noté $\mathcal{A}_1 \otimes_f \mathcal{A}_2$, généralise la construction de l'intersection :

$$\mathcal{A}_1 \otimes_f \mathcal{A}_2 = (Q_1 \times Q_2, I_1 \times I_2, T, F_1 \times F_2),$$

l'ensemble T des transitions étant défini par :

une transition $(q_1, q_2) \xrightarrow{f(a,b)} (q_1', q_2')$ est dans T si et seulement si
- $q_1 \xrightarrow{a} q_1'$ est dans T_1 et $q_2 \xrightarrow{b} q_2'$ est dans T_2, si $a \neq -$ et $b \neq -$,
- $q_1' = q_1$ et $q_2 \xrightarrow{b} q_2'$ est dans T_2, si $a = -$ et $b \neq -$,
- $q_1 \xrightarrow{a} q_1'$ est dans T_1 et $q_2' = q_2$ si $a \neq -$ et $b = -$.

Exercice 12. Modéliser un passage à niveau avec deux trains, une barrière et un contrôleur. Le train émet des signaux !*app* (resp. !*exit*) en entrant dans (resp. en sortant de) la zone du passage à niveau. Lorsque ces signaux sont reçus par le contrôleur (?*app* et ?*exit* respectivement), celui-ci émet des commandes (!*godown* et !*goup* respectivement) à destination de la barrière. La réception des commandes par la barrière (?*godown* et ?*goup* respectivement) déclenche un changement d'état visant à fermer ou ouvrir la barrière. Ces opérations de fermeture et d'ouverture ne sont pas instantanées. Les synchronisations seront représentées dans une table, avec par exemple :

train 1	*train* 2	*contr.*	*bar.*	*produit*
!*app*	–	?*app*	–	*app*
–	!*app*	?*app*	–	*app*
–	–	!*godown*	?*godown*	*godown*
–	–	!*goup*	?*goup*	*goup*

les autres actions étant des actions internes aux composants.

4 Exemples de modélisation

Voici une liste d'exemples pour lesquels on peut obtenir une modélisation à l'aide d'automates finis.

1. Comportement d'une mémoire : bascule RS avec portes NON-OU.
2. Dictionnaires électroniques.
3. Digicode avec compteur d'erreurs : construire une machine à un compteur conduisant à un état d'erreur lorsque l'utilisateur s'est trompé 3 fois de suite dans la composition du code. En déduire un automate fini par dépliage.
4. Exclusion mutuelle pour deux processus qui partagent une variable entière T (valant 1 ou 2) : représenter d'abord le comportement d'un processus P_1 qui exécute

   `Tant que` *vrai* `faire`
   `SR1`
   `Tant que` $T = 2$ `faire rien ;`
   `SC1`
   $T := 2$;

   puis le comportement de P_2 et celui de la variable T. En déduire le comportement de l'ensemble avec une synchronisation sur T.
5. Comportement d'un système réparti.