Calculabilité

TP6

Y. Deville
C-H. Bertrand Van Ouytsel - V. Coppé - A. Gerniers
N. Golenvaux - M. Parmentier

Mars 2021

Question 1 du test

Question : L'affirmation suivante est-elle vraie : l'ensemble des fonctions calculées par le langage BLOOP est exactement l'ensemble des fonctions totales calculables.

Réponse : FAUX
Le théorème d'Hoare-Allison nous dit qu'un langage de programmation qui ne permet de calculer que des fonctions totales ne permet pas de calculer sa fonction interpréteur (qui est une fonction totale).

Question 2 du test

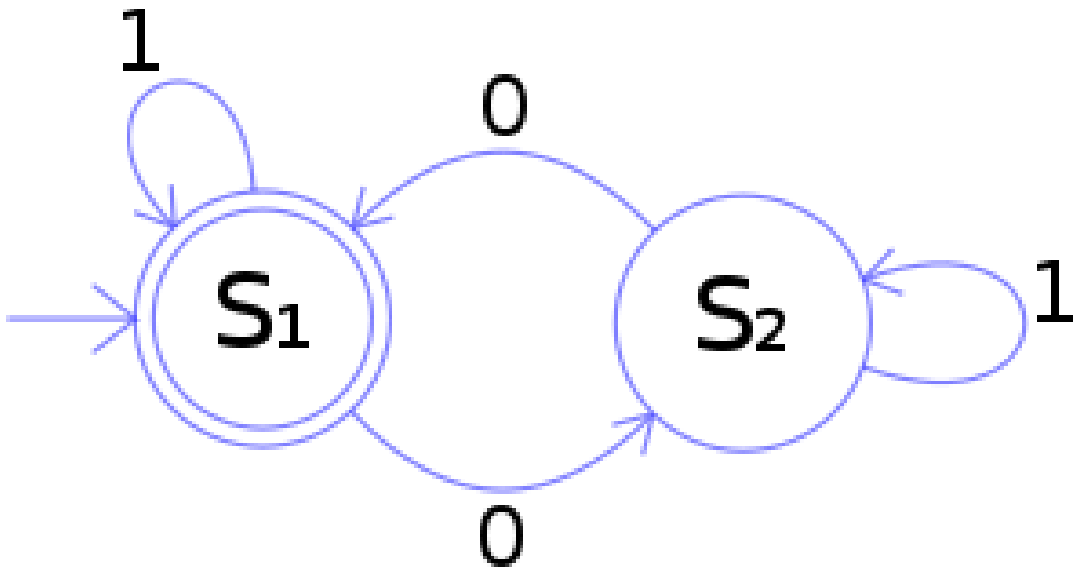
k Question : L'affirmation suivante est vraie : $\overline{K}$ est ND-récursivement énumérable.

Réponse : FAUX
Si un ensemble est ND-récursivement énumérable, il est récursivement énumérable. Or, $\overline{K}$ n'est pas récursivement énumérable.

Sous-question : comment justifier qu'un ensemble ND-récursivement énumérable est automatiquement récursivement énumérable ?

Question 3 du test

Question : Que permet de décider l'automate fini déterministe ci-dessous ?



Réponse : si l'entrée contient un nombre pair de 0.

Question 4 du test

Question : Quel ensemble de mots l'automate suivant décide-t-il ? Alphabet : $\Sigma = \{x, y, z\}$. Ensemble des états : $S = \{s_0, s_1\}$. État initial : s_0. Ensemble des états acceptants : $A = \{s_1\}$. Table de la fonction de transition :

	x	y	z
s_0	s_1	s_0	s_0
s_1	s_1	s_1	s_1

Réponse : L'ensemble des mots formés à partir des lettres x, y, z qui contiennent au moins un x.

Question 5 du test

Question : L'affirmation suivante est-elle vraie ? L'interpréteur des automates finis peut être représenté à l'aide d'un automate fini.

Réponse : FAUX
Le formalisme des automates finis ne permet de calculer que des fonctions totales. Comme avec le langage BLOOP de la question 1, on en déduit qu'il ne permet pas de calculer sa fonction interpréteur (qui est une fonction totale).

Question 6 du test

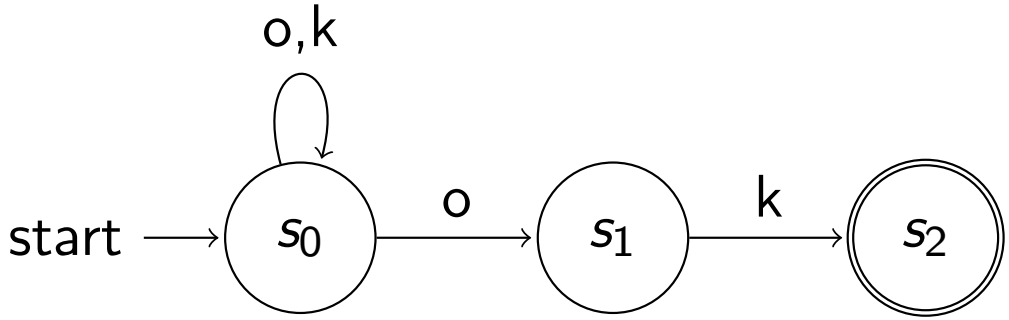
Question : L'affirmation suivante est-elle vraie ?
Tout sous-ensemble d'un langage accepté par un automate fini est récursif.

Réponse : FAUX
Il existe un automate qui accepte l'ensemble de tous les mots formé à partir de l'alphabet $\Sigma = \{a\}$ (qui est certainement récursif). Mais le sous-ensemble de cet ensemble des mots formés à partir de cet alphabet dont la longueur est dans K (pour un langage de programmation choisi) n'est pas récursif.

Question 7 du test

Question : L'affirmation suivante est-elle vraie ?
Il existe un automate fini non déterministe qui permet de déterminer le langage des mots construits à partir de l'alphabet $\Sigma = \{k, o\}$ qui terminent avec 'ok'.

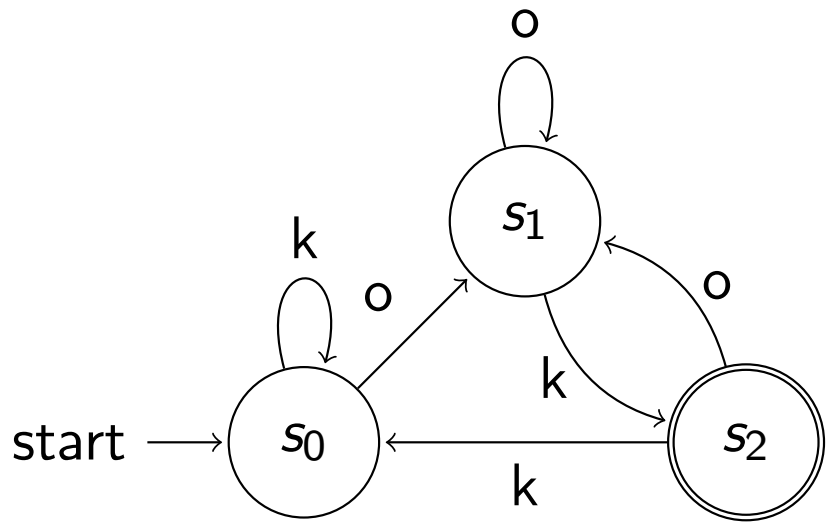
Réponse : VRAI



Question 8 du test

Question : L'affirmation suivante est-elle vraie ?
Il existe un automate fini déterministe qui permet de déterminer le langage des mots construits à partir de l'alphabet $\Sigma = \{k, o\}$ qui terminent avec 'ok'.

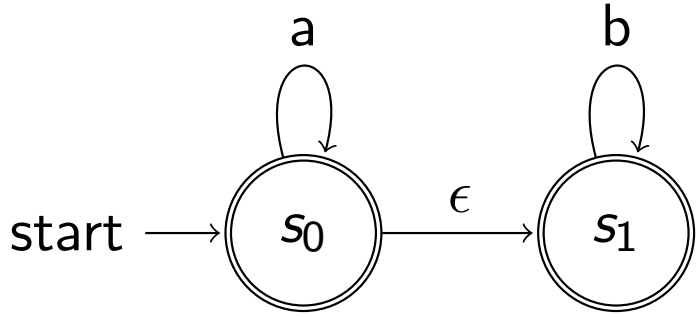
Réponse : VRAI



Question 9 du test

Question : L'affirmation suivante est-elle vraie ?
Il existe un automate fini avec transition vide qui permet de déterminer le langage $\{a^m b^n \mid n, m \in \mathbb{N}\}$.

Réponse : VRAI



Sous-question : la réponse change-t-elle si on fixe m et n ?

Question 10 du test

Question : L'affirmation suivante est-elle vraie ?
Il existe un automate fini déterministe qui permet de déterminer le langage $\{a^m b^n \mid n, m \in \mathbb{N}\}$.

Réponse : VRAI
Les automates avec ϵ-transition ont la même puissance que les automates finis déterministes (voir slides du cours).

Sous-question : la réponse change-t-elle si on fixe m et n ?

Question 1 du TP

Question : Let $A = \{a \in \mathbb{N} \mid \exists x \in \mathbb{N} : -x^5 + x + a = 0\}$. Show that A is ND-recursive. Is A also recursive ?

Question 1 du TP

Question : Let $A = \{a \in \mathbb{N} \mid \exists x \in \mathbb{N} : -x^5 + x + a = 0\}$. Show that A is ND-recursive. Is A also recursive ?

Réponse : notons d'abord que s'il est ND-récursif il est automatiquement récursif. Pour justifier qu'il est ND-récursif, il suffit de considérer le programme qui reçoit en entrée le coefficient a, attribue au hasard une valeur à la variable x et renvoie 1 si la formule est alors égale à 0, et 0 sinon. Cependant, la fonction `choose` nécessite une borne supérieure pour pouvoir tirer une valeur au hasard.
On remarque que :

- si $a = 0$, le polynôme admet comme racines 0 et 1
- si $a \neq 0$, alors $\forall x \in \mathbb{N}$ avec $x > a$, on a $-x^5 + x + a < 0$.

Question 1 du TP

Dès lors, le programme suivant prouve que A est ND-récursif.

$$P_A(a) \equiv \left[\begin{array}{l} x = \texttt{choose}(a+1) \\ \texttt{if} \;\; -x^5 + x + a = 0 : \\ \quad \texttt{return 1} \\ \texttt{else} : \\ \quad \texttt{return 0} \end{array}\right.$$

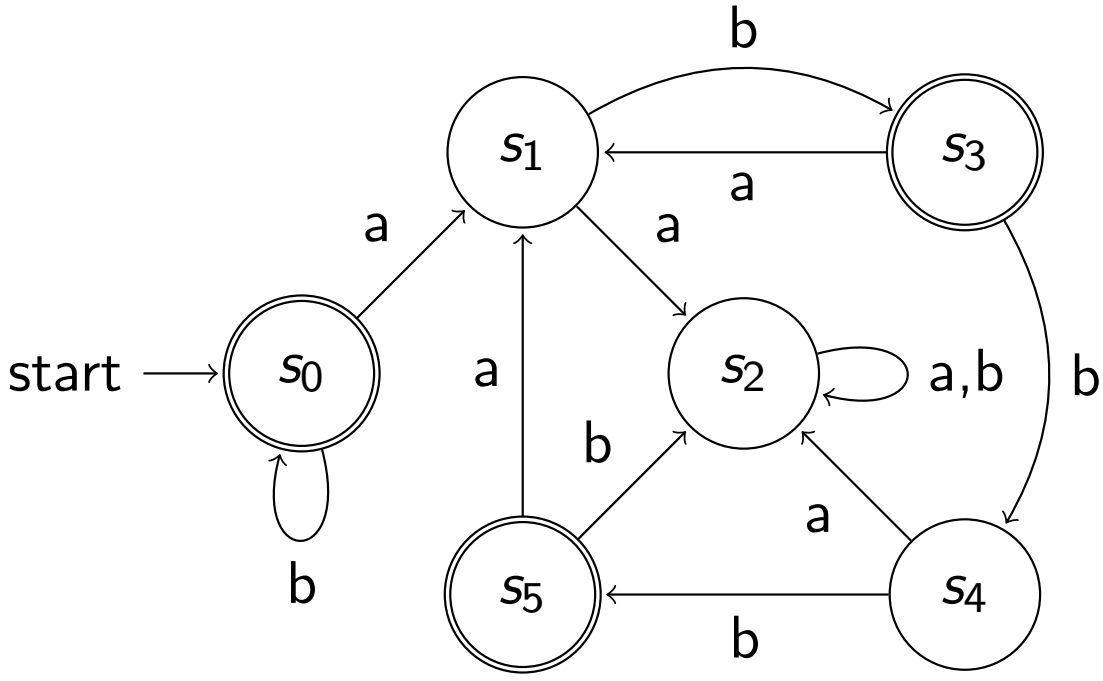
Question 2a du TP

Question : For the following language, construct a deterministic finite automaton that accepts this language.
Alphabet : $\Sigma = \{a, b\}$. Language :
$\{w \in \Sigma^* \mid \text{each } a \text{ in } w \text{ is followed by exactly one or three } b\}$.

Question 2a du TP

Question : For the following language, construct a deterministic finite automaton that accepts this language.
Alphabet : $\Sigma = \{a, b\}$. Language :
$\{w \in \Sigma^* \mid \text{each } a \text{ in } w \text{ is followed by exactly one or three } b\}$.

Réponse :



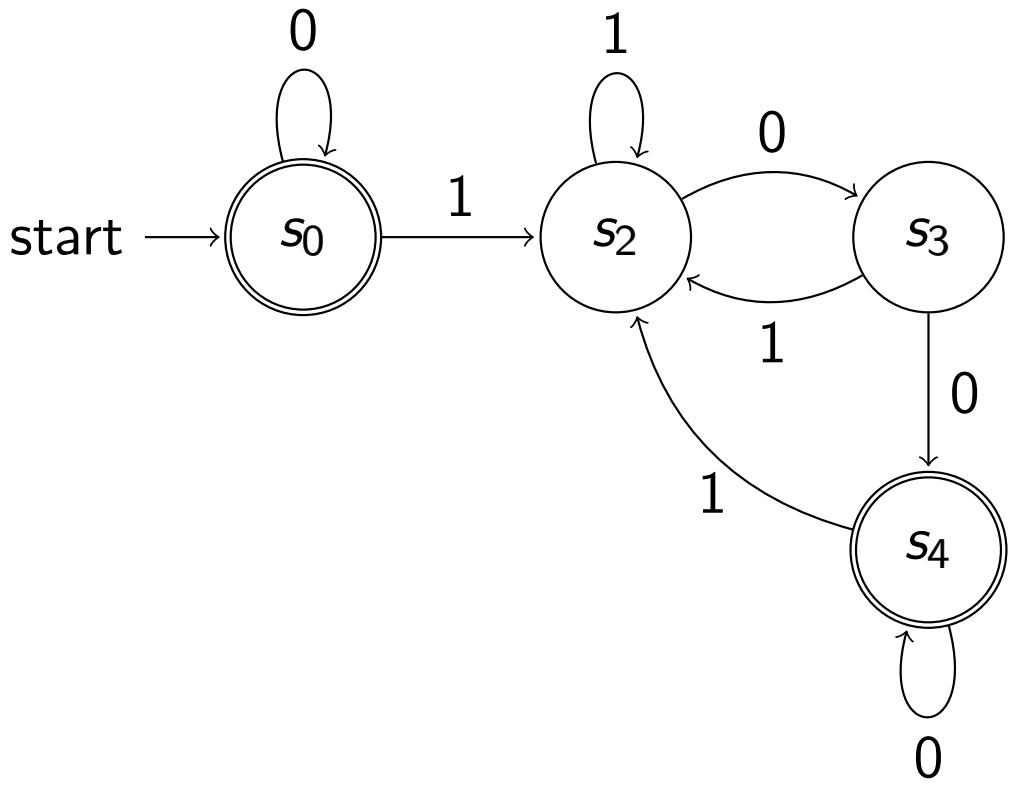
Question 2b du TP

Question : For the following language, construct a deterministic finite automaton that accepts this language.
Alphabet : $\Sigma = \{0, 1\}$.
Language : $\{w \in \Sigma^* \mid \text{the decimal value of } w \text{ is divisible by } 4\}$.

Question 2b du TP

Alphabet : $\Sigma = \{0, 1\}$.
Language : $\{w \in \Sigma^* \mid \text{the decimal value of } w \text{ is divisible by } 4\}$.

Réponse :



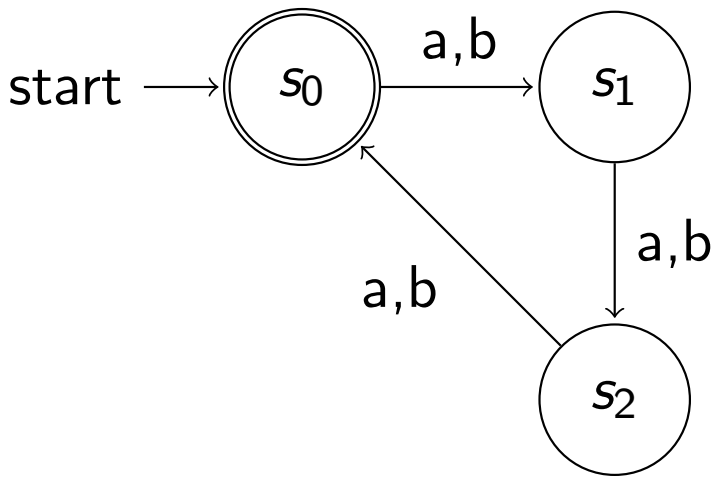
Question 2c du TP

Question : For the following language, construct a deterministic finite automaton that accepts this language.
Alphabet : $\Sigma = \{a, b\}$.
Language : $\{w \in \Sigma^* \mid \text{the length of } w \text{ is divisible by } 3\}$.

Question 2c du TP

Question : For the following language, construct a deterministic finite automaton that accepts this language.
Alphabet : $\Sigma = \{a, b\}$.
Language : $\{w \in \Sigma^* \mid \text{the length of } w \text{ is divisible by } 3\}$.

Réponse :



Question 3 du TP

Question : Suppose we have two languages M and L accepted by some finite automata and their corresponding diagrams. Show by construction that the following languages are also accepted by a finite automaton :

1. $\bar{L}$
2. $L \cdot M$
3. $L \cup M$

Question 3 du TP

Question : Suppose we have two languages M and L accepted by some finite automata and their corresponding diagrams. Show by construction that the following languages are also accepted by a finite automaton :

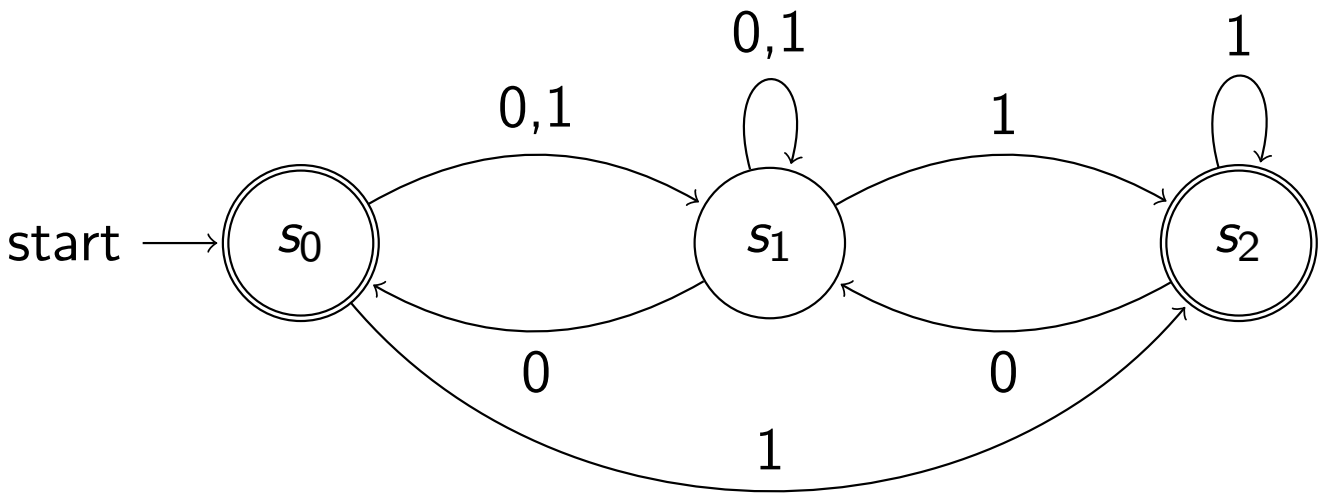
1. $\bar{L}$
2. $L \cdot M$
3. $L \cup M$

Réponse :

1. On transforme les états acceptants en états non acceptants et inversement.
2. On enlève le statut d'état acceptant aux états acceptants de l'automate qui décide L. On enlève le statut d'état initial à l'état initial de l'automate qui décide M. On crée des ϵ-transitions de tous les anciens états acceptants de l'automate qui décide L vers l'ancien état initial de M.
3. On crée un nouvel état initial avec une ϵ-transition vers chacun des états initials des deux automates fournis.

Question 4a du TP

Question : Transform the following non-deterministic automaton in a finite deterministic automaton accepting the same language.



Question 4a du TP

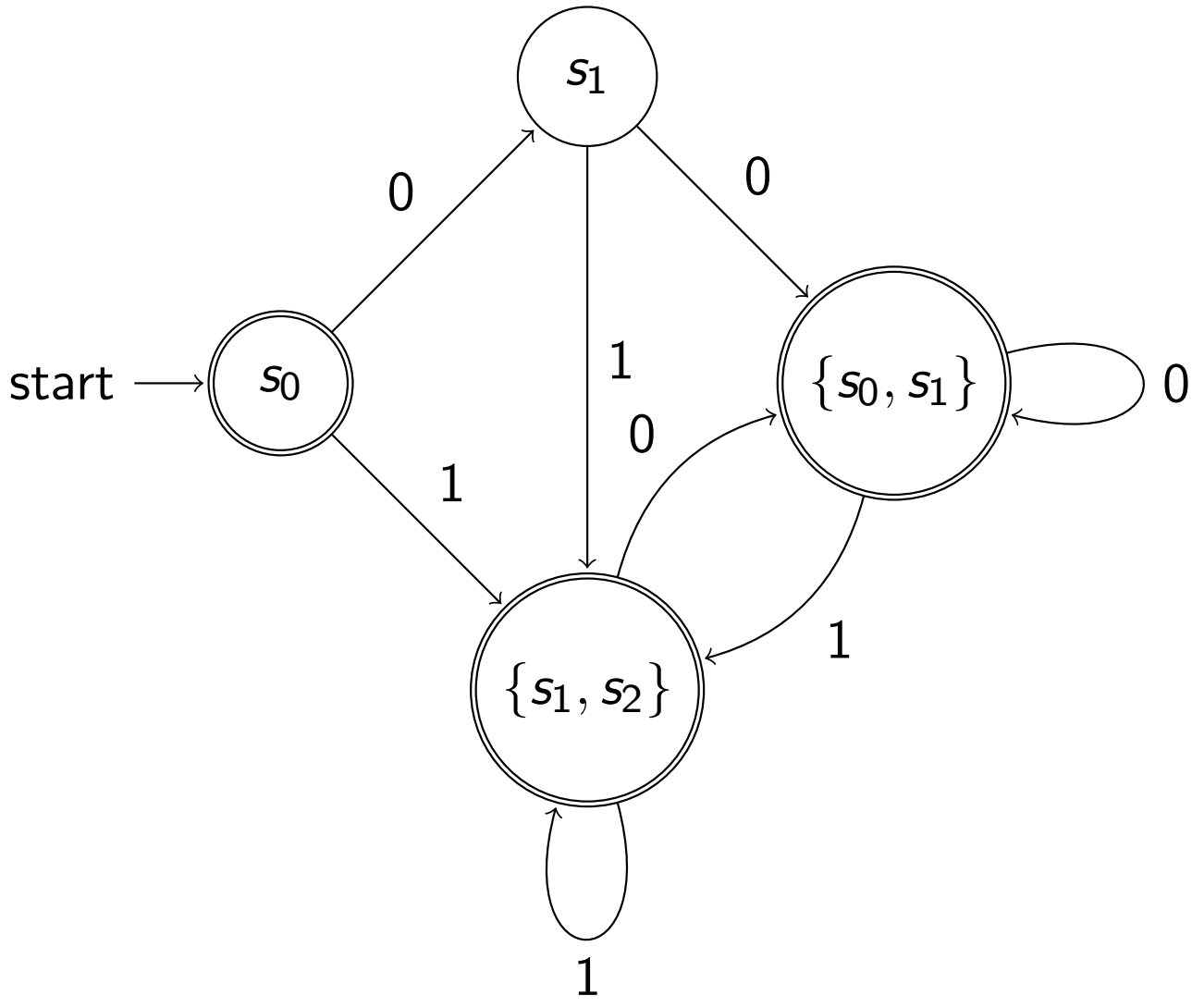
Réponse : on transforme la table de transition de la manière suivante :

	0	1
s_0	s_1	s_1, s_2
s_1	s_0, s_1	s_1, s_2
s_2	s_1	s_2

	0	1
s_0	s_1	$\{s_1, s_2\}$
s_1	$\{s_0, s_1\}$	$\{s_1, s_2\}$
s_2	s_1	s_2
$\{s_0, s_1\}$	$\{s_0, s_1\}$	$\{s_1, s_2\}$
$\{s_1, s_2\}$	$\{s_0, s_1\}$	$\{s_1, s_2\}$

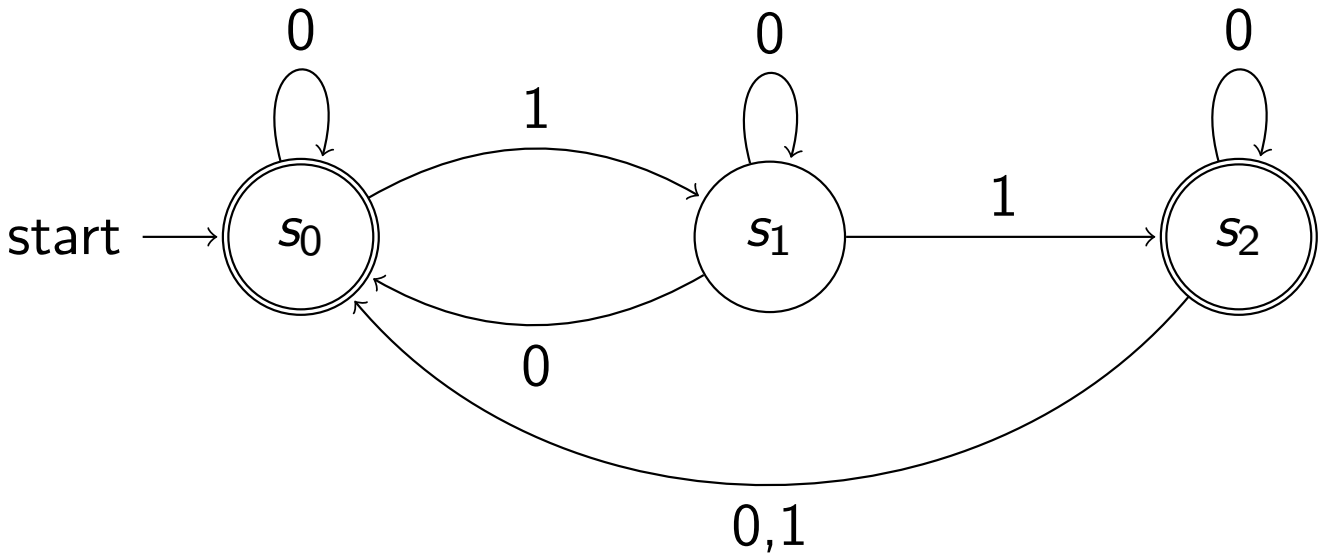
On remarque que l'état s_2 est inaccessible, on peut donc l'ignorer dans le diagramme.

Question 4a du TP



Question 4b du TP

Question : Transform the following non-deterministic automaton in a finite deterministic automaton accepting the same language.



Question 4b du TP

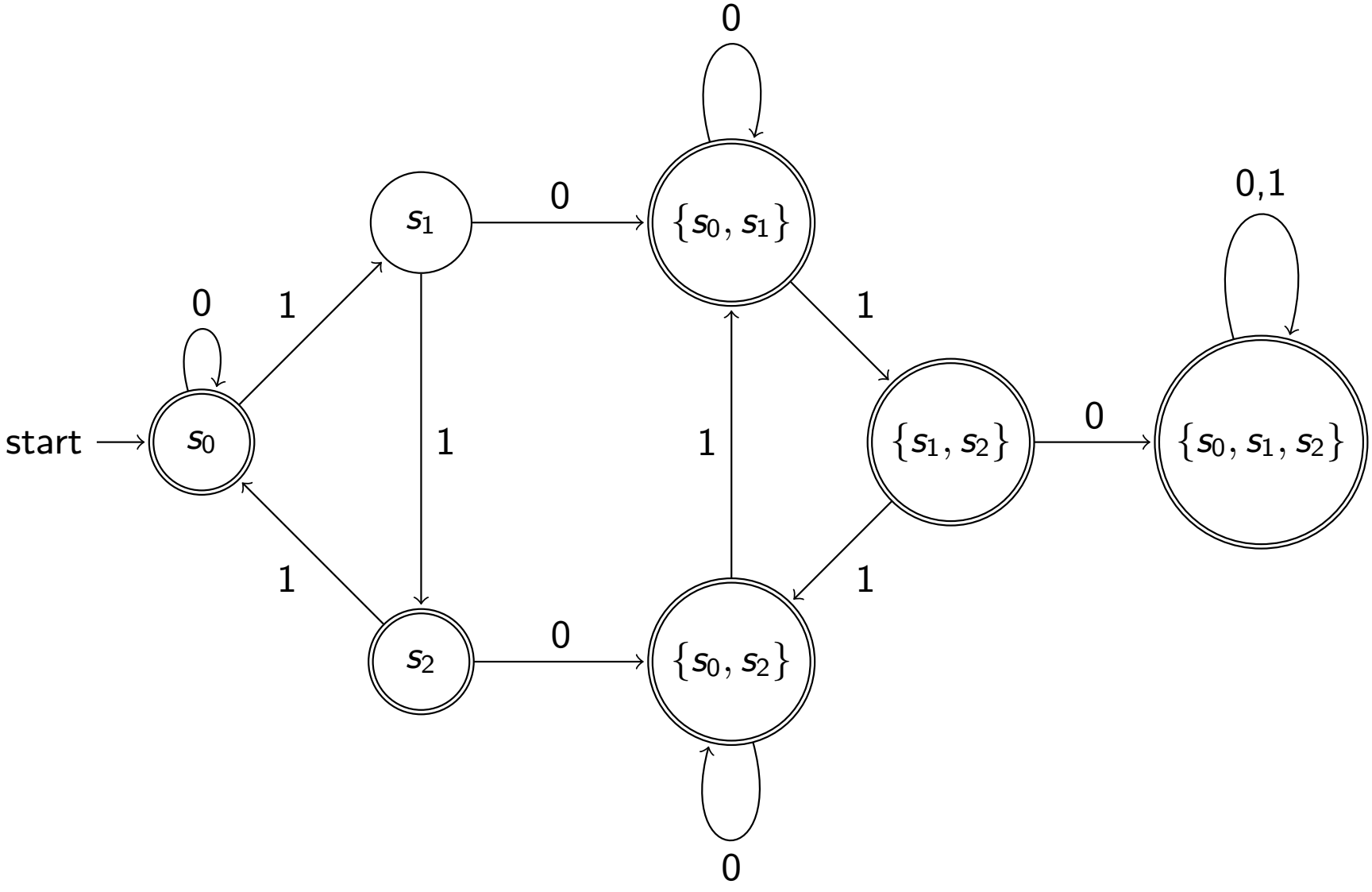
Réponse : on transforme la table de transition de la manière suivante :

	0	1
s_0	s_0	s_1
s_1	s_0, s_1	s_2
s_2	s_0, s_2	s_0

	0	1
s_0	s_0	s_1
s_1	$\{s_0, s_1\}$	s_2
s_2	$\{s_0, s_2\}$	s_0
$\{s_0, s_1\}$	$\{s_0, s_1\}$	$\{s_1, s_2\}$
$\{s_0, s_2\}$	$\{s_0, s_2\}$	$\{s_0, s_1\}$
$\{s_1, s_2\}$	$\{s_0, s_1, s_2\}$	$\{s_0, s_2\}$
$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_2\}$	$\{s_0, s_1, s_2\}$

On remarque que cette fois-ci, la nouvelle table de transition a beaucoup plus de rangées. Elle comprend même tous les sous-ensembles d'états de l'automate non-déterministe. Pour un automate non-déterministe à n états, l'automate déterministe correspondant peut donc avoir jusqu'à 2^n états.

Question 4b du TP



Challenge

Question : during the lectures, you were told that ϵ-moves do not improve the computability power of finite automata. In practice, give a finite automata with ϵ-moves which decides a language L, how do you build a finite automata without ϵ-moves that will decide the same language ?

Réponse : voir lien ci-dessous.
Élimination des ϵ-transitions dans un automate fini