math spé option info TP 3

Déterminisation des automates

Stéphane Legros Année scolaire 2024-2025

Nous travaillerons exclusivement sur l'alphabet à deux lettres $\{a, b\}$. Pour $n \in \mathbb{N}^*$, nous noterons $\mathbb{N}_n$ l'ensemble $\{0, 1, \ldots, n-1\}$ et nous identifierons les parties de $\mathbb{N}_n$ aux entiers de $[\![0, 2^n[\![$, par l'intermédiaire de leurs écritures en base 2. Par exemple, quand $n = 5$, l'entier $13 = \overline{1101}^2$ représente la partie $\{0, 2, 3\}$ de $\mathbb{N}_5$. Nous pourrons ainsi utiliser quelques fonctions du module `Int` :

- `Int.shift_left : int -> int -> int` (resp. `Int.shift_right : int -> int -> int`) décale les bits vers la gauche (resp. vers la droite) ;
- `Int.logand: int -> int -> int` applique un « et » logique bit-à-bit ;
- `Int.logor: int -> int -> int` applique un « ou » logique bit-à-bit.

Ainsi, `Int.shift_left 5 3` renvoie la valeur 40, `Int.shift_right 15 2` renvoie la valeur 3, `Int.logand 25 12` renvoie la valeur 8 et `Int.logand 25 12` renvoie 29 :

	1	1	0	0	1
`et`		1	1	0	0
	0	1	0	0	0

	1	1	0	0	1
`ou`		1	1	0	0
	1	1	1	0	1

Un graphe orienté G sur l'ensemble $\mathbb{N}_n$ est représenté par un tableau `g` d'entiers de longueur n : `g.(i)` est l'entier qui code l'ensemble des j tels qu'il y ait un arc de i à j.

Préliminaires

Écrire des fonctions `appartient: int -> int -> bool`, `intersecte: int -> int -> bool` et `ajoute : int -> int -> int` qui testent l'appartenance d'un élément à une partie, qui vérifie si deux parties ont un élément en commun et qui ajoute un élément à un ensemble.

Partie I : les automates déterministes

Un automate déterministe est défini en Caml par le biais du type :

```
type automate_det = {m: int; i0: int; fin: int; delta: int array array};;
```

Nous avons, pour un objet `ad` de type `automate_det` :

- `ad.m` est la taille m de l'automate (l'ensemble des états étant alors $\mathbb{N}_m$), `ad.i0` est l'état initial et `ad.fin` est l'ensemble des états finals ;
- la fonction de transition est représentée par `ad.delta` : `ad.delta.(i).(0)` (resp. `ad.delta.(i).(1)`) est l'état $i.a$ (resp. $i.b$). Par convention, nous poserons $i.a = m$ (resp. $i.b = m$) quand il n'y a pas de flèche d'étiquette a (resp. b) partant de l'état i.

1) Définir en OCaml les automates déterminisés Ad_1 et Ad_2 associés aux automates A_1 et A_2 (voir ci-dessous).

2) Écrire une fonction `deplacement` telle que l'instruction `deplacement ad q u` renvoie l'état atteint dans l'automate `ad` en lisant le mot `u` depuis l'état `q`. Tester votre procédure avec les automates Ad_1 et Ad_2.

3) Écrire une procédure `reconnu_det: automate_det -> string -> bool` qui teste si un mot est reconnu par un automate déterministe. Tester le fonctionnement de votre fonction.

Partie II : les automates non déterministes

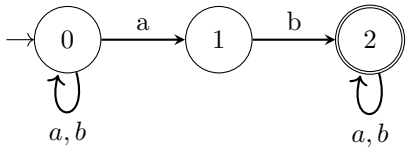
Un automate non déterministe est défini en Caml par le biais du type :

```
type automate = {n: int; ini: int; fin: int; a: int array; b: int array};;
```

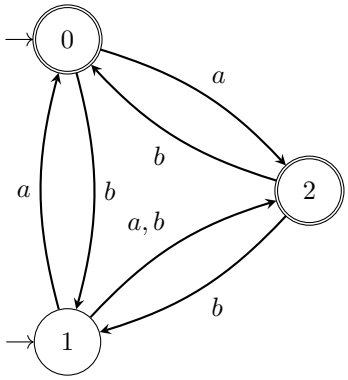
L'automate représenté par un objet `aut` de type `automate` est défini ainsi :

- `aut.n` est la taille n de l'automate, l'ensemble des états étant alors $\mathbb{N}_n$;
- les états initiaux et finals sont représentés par les entiers `aut.ini` et `aut.fin` ;
- `aut.a` et `aut.b` sont les graphes des transitions d'étiquettes a et b.

1) Définir en Caml les automates suivants :



Automate $\mathcal{A}_1$



Automate $\mathcal{A}_2$

2) Écrire une fonction `calcul: int array -> int -> int` qui s'applique à un graphe G sur $\mathbb{N}_n$ et à une partie q de $\mathbb{N}_n$ et renvoie la partie q' de $\mathbb{N}_n$ définie par $j \in q' \iff \exists\, i \in q,\ (i,j) \in G$.

3) Écrire une fonction `reconnu: automate -> string -> bool` qui teste si un mot (sur l'alphabet $\{a, b\}$) est reconnu par un automate. Vérifier le fonctionnement de votre fonction avec les automates A_1 et A_2.

4) Écrire deux fonctions `accessibles` et `coaccessibles` qui calculent respectivement les ensembles des états accessibles et co-accessibles d'un automate.

Partie III : automate des parties et détermination d'un automate

Si A est un automate, nous noterons A_p l'automate déterministe des parties de A. Si $n = A.n$, nous aurons $A_p.m = 2^n = m$. Les états de cet automate sont les parties de $\mathbb{N}_n$, c'est-à-dire les éléments de $\mathbb{N}_m$

1) Écrire une fonction `automate_parties: automate -> automate_det` qui calcule l'automate des parties d'un automate non déterministe. Tester votre procédure sur les automates A_1 et A_2.

2) Si A est un automate, nous noterons A_d le déterminisé de A, i.e. l'automate A_p restreint à ses états accessibles. Donner le code d'une fonction `determiniser: automate -> automate_det` qui calcule cet automate, sans calculer tout l'automate des parties.