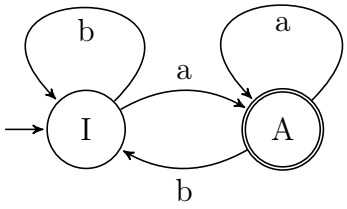
Exercices: Calculabilité

Lionel Vaux Auclair

Logique
HUGO@Centrale, 2021–2022

1 Automates finis

Exercice 1. Quels sont les mots acceptés par l'automate suivant ?



On précise que l'état initial est I (marqué par la flèche entrante) et que le seul état acceptant est A (le trait du cercle est doublé).

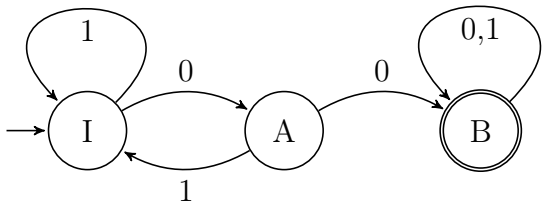
Exercice 2. Trouvez un automate qui reconnait les mots de la forme a^{2n} (c'est-à-dire constitués d'un nombre pair de a).

Exercice 3. Trouvez un automate qui accepte exactement les adresses IPv4 en notation décimale à points.

Exercice 4. Trouvez un automate qui accepte exactement les mots correspondant à l'expression régulière `[a-z][a-z0-9_-]{2,15}` (un identifiant de 3 à 16 caractères, commençant par une lettre en ne contenant que des lettres chiffres et tirets).

Exercice 5. Trouvez un automate qui accepte exactement les mots correspondant à l'expression régulière `#[a-fA-F0-9]{6}|[a-fA-F0-9]{3}` (les codes hexadécimaux des couleurs en CSS).

Exercice 6. Trouvez une expression régulière qui correspond aux mots acceptés par l'automate :



.

Les exemples précédents devraient vous convaincre : à chaque expression régulière (simples, sans références arrières) correspond un automate, et réciproquement. La preuve formelle est technique.

Exercice 7. Fixons un automate fini $\mathcal{A}$, avec n états. Montrez que si $a^n b^n$ est accepté par $\mathcal{A}$, alors il y a un $k < n$ tel que $a^k b^n$ soit aussi accepté. Déduisez-en qu'il n'y a pas d'automate fini qui reconnaisse exactement l'ensemble des mots de la forme $a^n b^n$.

Cette technique se généralise : c'est le lemme d'itération ou lemme de l'étoile.

2 Calculer avec une machine de Turing

On va utiliser un simulateur en ligne : `http://morphett.info/turing/turing.html`.

Le format utilisé pour définir une machine est simple, en codant chaque transition sur une ligne. Plus précisément, si $\delta(e, c) = (e', c', m)$ alors on met une ligne

```
e c c' m e'
```

(en utilisant `l` pour gauche et `r` pour droite).

Exercice 8. Implémentez dans ce simulateur la machine qui calcule le successeur en binaire. (Attention, le simulateur démarre sur le premier caractère, donc le bit de poids fort !)

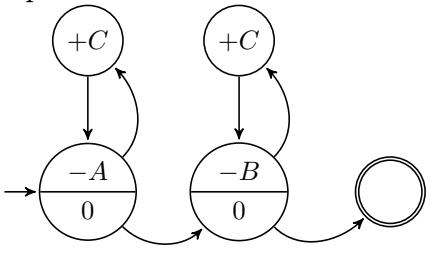
Exercice 9. Implémentez une machine qui accepte exactement le langage des mots de la forme $a^n b^n$.

3 Machines à compteurs

Les machines à compteurs sont un modèle de calcul extrêment simple, dû à Marvin Minsky. On peut les voir comme des automates agissant non pas sur des mots mais sur un nombre fini de compteurs A, B, C, ... Dans un état donné on peut :

- ou bien incrémenter un compteur et passer dans un état q' ;
- ou bien tester si un certain compteur est nul :
  - si oui, on change d'état ;
  - si non, on décrémente et on change d'état (possiblement différent de celui du cas nul).

On peut démontrer que les fonctions calculables sur les entiers par machines de Turing le sont par machines à compteurs (il faut au moins deux compteurs). Mais c'est surtout un modèle de calcul avec lequel on peut jouer, en les représentant graphiquement :



Exercice 10. Que calcule la machine à compteurs ci-dessus dans le compteur C, en fonction des valeurs initiales des compteurs A et B (en démarrant avec $C = 0$) ?

Exercice 11. Modifiez la machine précédente, pour que les valeurs données dans les compteurs A et B ne soient pas perdues (un compteur supplémentaire sera utile).

Exercice 12. Définissez une machine à compteurs pour la multiplication.

Exercice 13. Implémentez un simulateur de machines à compteurs dans votre langage préféré.

Exercice 14. On peut représenter une machine à compteurs par un listing comme celui-ci (qui correspond à la machine de la question 1) :

```
A_vers_C     -A  B_vers_C A_vers_C_bis
A_vers_C_bis +C  A_vers_C
B_vers_C     -B  fini B_vers_C_bis
B_vers_C_bis +C  B_vers_C
```

avec la convention que l'état initial est celui de la première ligne. Écrivez une fonction (ou une méthode) qui transforme un tel script en une machine à compteurs au sens de l'exercice précédent.

Vous avez écrit votre premier (ou pas) compilateur !

4 Problèmes indécidables : réduction

Fixez votre langage de programmation préféré, pourvu qu'il sache travailler avec des fonctions comme arguments (en les passant par pointeurs en C par exemple).

Étant donnés deux programmes `f` et `g` prenant chacun un argument, et étant donnée une valeur `x`, on dit que `f` et `g` coïncident en `x` si :

— ou bien les expressions `f(x)` et `g(x)` s'évaluent en le même résultat (c'est-à-dire que `f(x) == g(x)` s'évalue en `vrai`) ;

— ou bien ni `f(x)` ni `g(x)` ne produisent un résultat (ça boucle indéfiniment).

Exercice 15. Supposons qu'on vous donne une fonction `cestpareil` telle que `cestpareil(f,g,x)` renvoie `vrai` si `f` et `g` coïncident en `x` et `faux` sinon.

À l'aide de `cestpareil`, définissez une fonction qui résout le problème de l'arrêt. Déduisez-en que le problème consistant à savoir si deux fonctions coïncident en une certaine valeur n'est pas décidable.

Exercice 16. On dit que `f` est totale si `f(x)` s'évalue toujours en un résultat. Montrez qu'il n'y a pas de fonction qui décide si une fonction est totale.

Ces résultats se généralisent : toute propriété non triviale qui ne dépend que de la fonction (au sens mathématique) calculée par une machine de Turing (ou un programme) est indécidable. C'est le théorème de Rice.