NGUYEN Thi Minh Tuyen

Théorie des automates et langages formels

Décidabilité

Trois questions importantes

1. Est-ce que deux expressions régulières sont équivalentes (définissent le même langage)?
2. Est-ce que deux automates finis sont équivalents (définissent le même langage)?
3. Est-ce que le langage reconnu par un automate fini est **fini?** **infini?** **vide?**

Définition

Un problème est dit de décision si la solution à ce problème est de la forme “OUI ou NON”

Un problème est dit décidable s’il existe un algorithme qui le résout en un nombre fini d’étapes.

Problème 1 et 2

1. **Est-ce que deux expressions régulières sont équivalentes (définissent le même langage)?**
2. **Est-ce que deux automates finis sont équivalents (définissent le même langage)?**

Les problèmes 1 et 2 sont équivalents:

Il suffit de transformer les deux expressions régulières en automates finis et puis répondre à la deuxième question.

Ou bien transformer les deux automates finis en expressions régulières et puis répondre à la première question.

Problème 2: Méthode 1

Algorithme fini:

A partir des automates finis qui reconnaissent L_1 et L_2, on peut construire en un nombre fini d'étapes un automate fini qui reconnaît

$$(L_1 \cap L_2') + (L_2 \cap L_1')$$

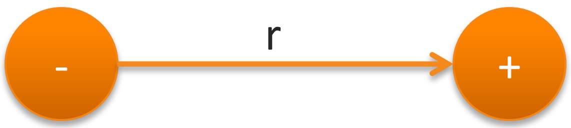
($L_1 = L_2$ si et seulement si le langage $(L_1 \cap L_2') + (L_2 \cap L_1')$ est vide.)

Donc on a besoin d'un algorithme fini pour décider si un automate fini accepte au moins un mot.

Méthode 1: On essaye de transformer cet automate en expression régulière (en utilisant l'algorithme fini dans la preuve du théorème de Kleene). Si on arrive à le faire, alors l'automate fini accepte au moins un mot, sinon il n'accepte aucun mot.

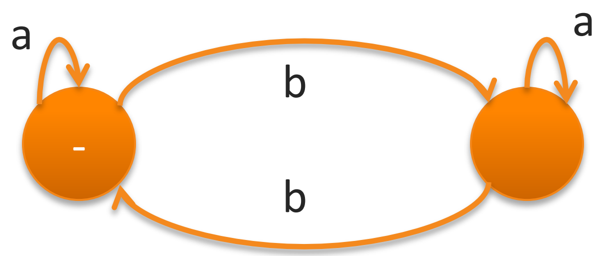
Exemple

Si l'algorithme mène à:



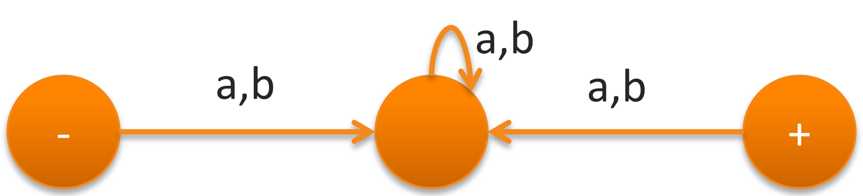
Alors l’automate fini accepte au moins un mot

Par contre si l'algorithme mène à :



OU

OU



Alors l'automate fini n'accepte aucun mot

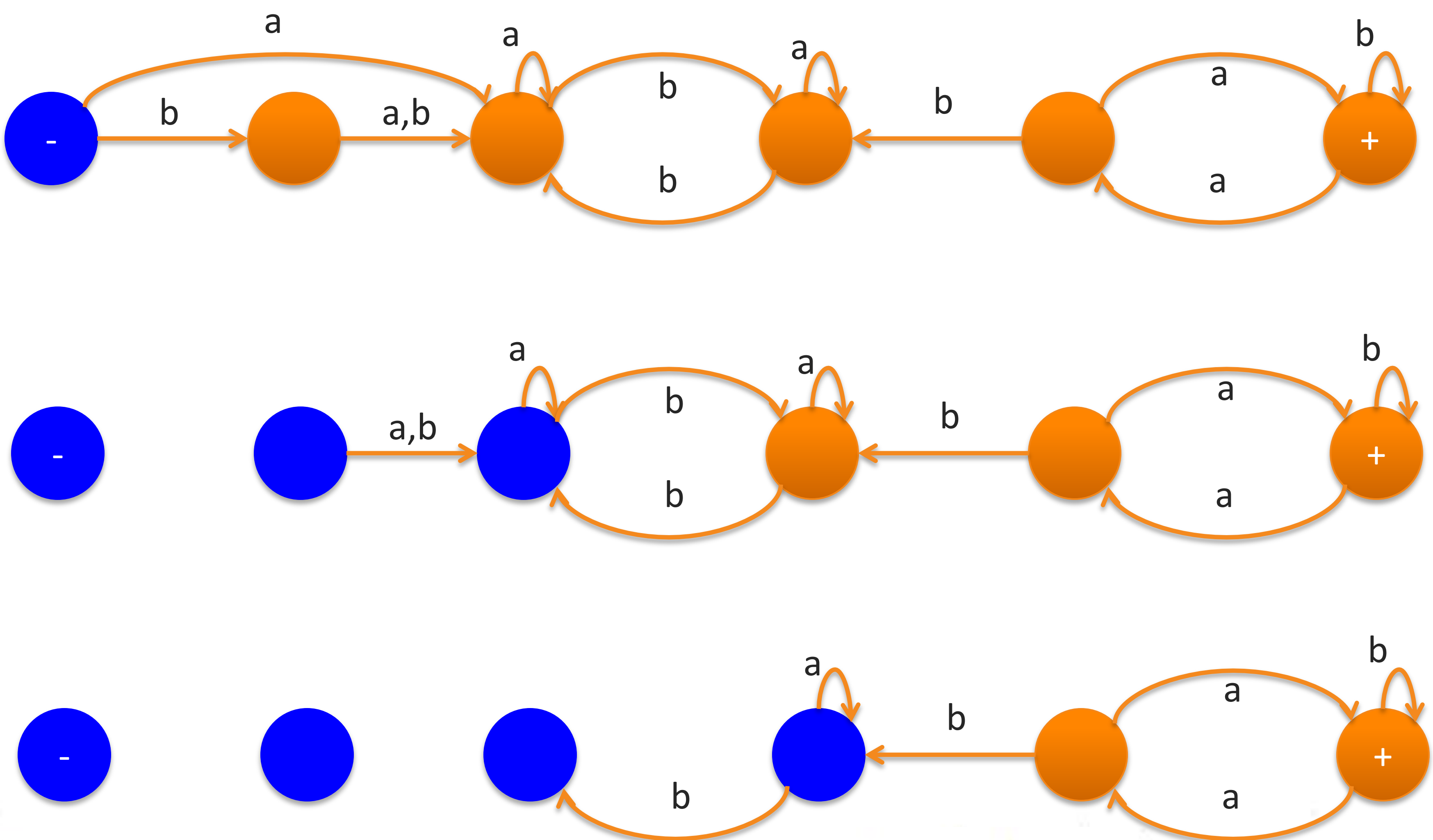
Problème 2: Méthode 2

L'automate fini accepte au moins un mot si et seulement si il existe un chemin de – vers +

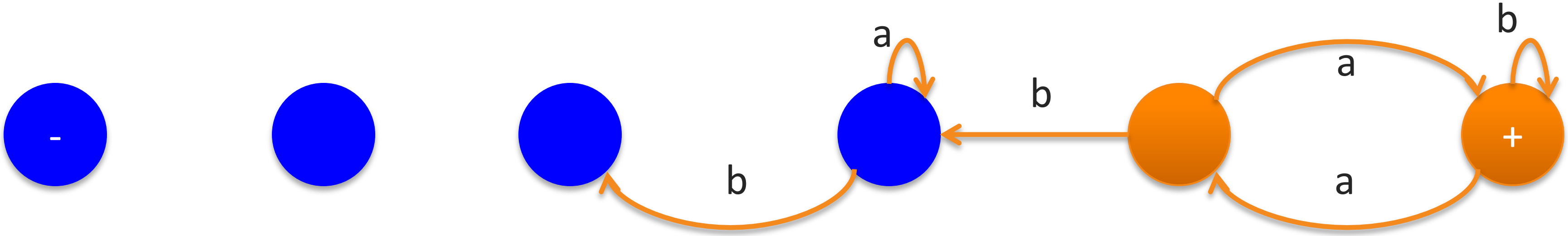
Algorithme:

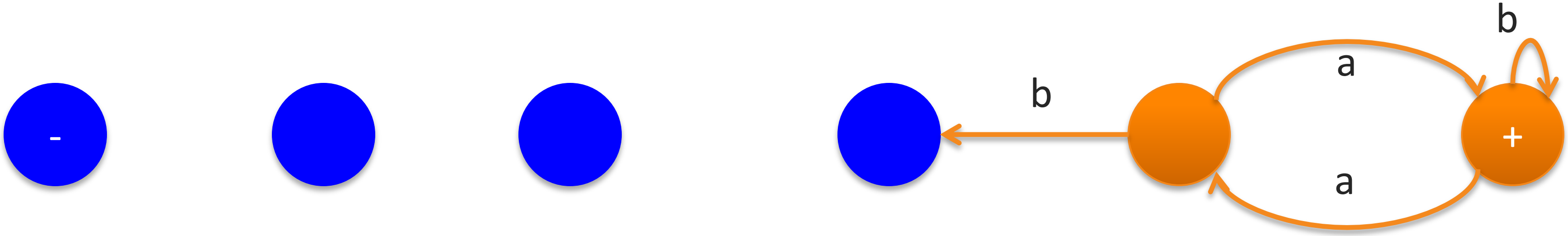
1. On marque l'état de départ. (On peint l'état en bleu.)
2. On suit chaque flèche qui sort de chaque état bleu. On peint chaque état qu'on atteint et efface la flèche.
3. On répète étape 2 jusqu'à il n'y a plus de nouveaux états bleus.
4. S'il y a un état final qui est bleu, alors il y a des mots dans le langage reconnu par cet automate. Sinon, le langage est vide.

Exemple [1]



Exemple [2]

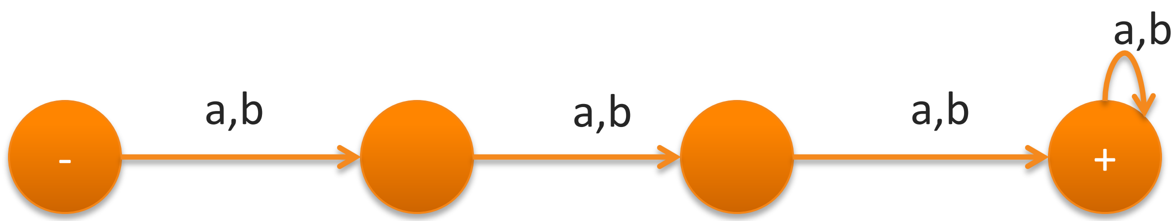




Le langage est vide.

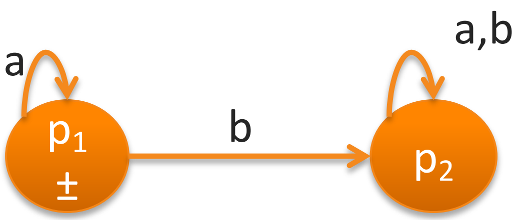
Problème 2: Méthode 3

- **Théorème**. Soit F un automate fini avec N états. Si le langage L reconnu par F n'est pas vide, alors L contient au moins un mot de longueur plus petite ou égale à N.
- **Méthode 3**. Il y a un nombre fini de mots de longueur plus petite ou égale à N. Il suffit d'essayer tous ces mots pour tester si le langage reconnu par F est vide ou pas.

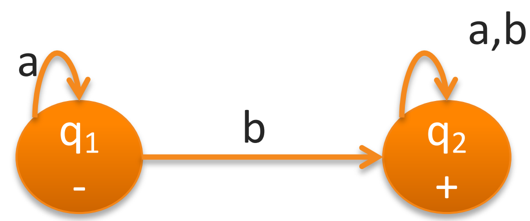


Exemple [1]

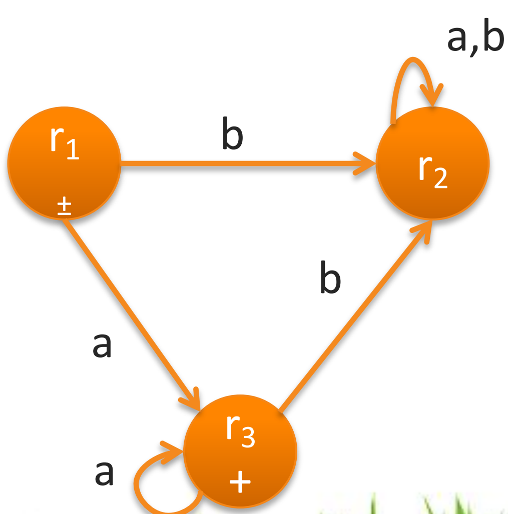
a^* FA_1



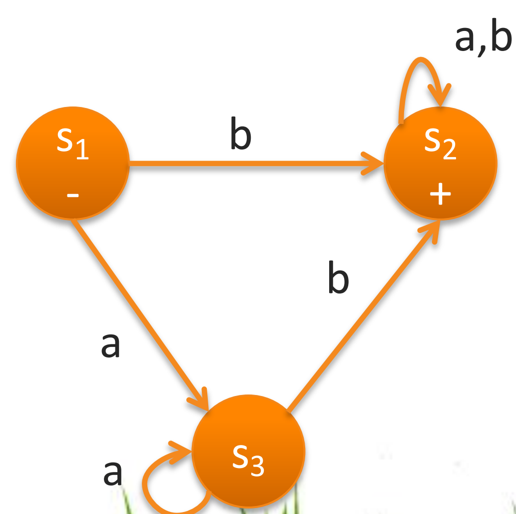
FA_1'



$\Lambda+aa^*$ FA_2

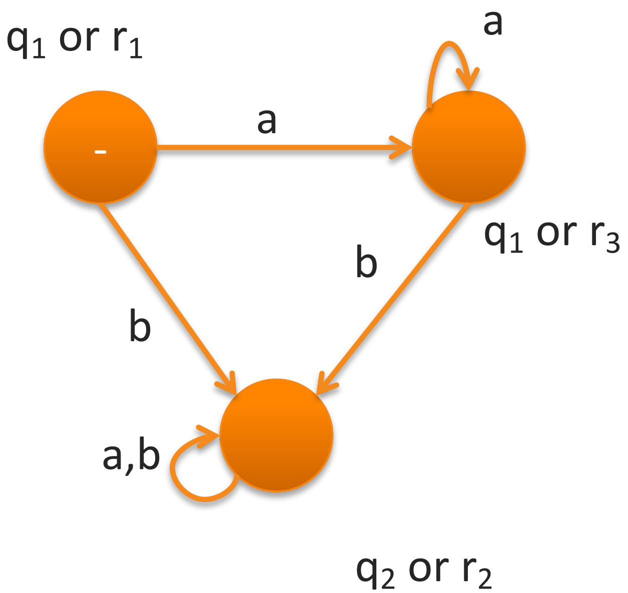


FA_2'

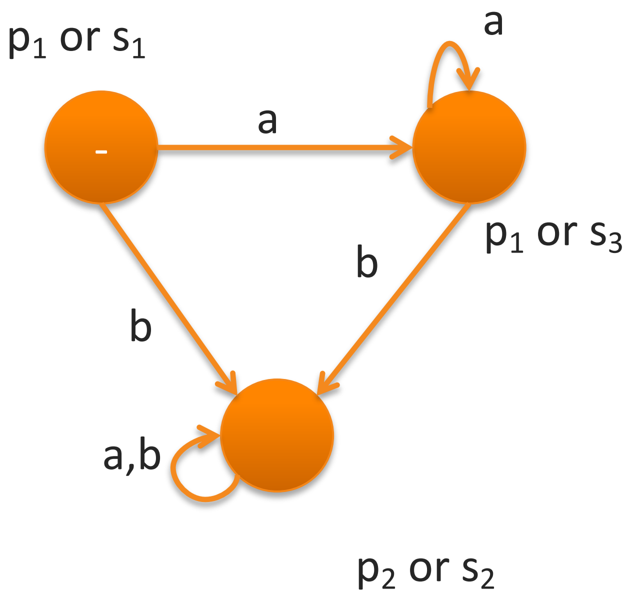


Exemple [2]

$(FA_1'+FA_2)'$



$(FA_2'+FA_1)'$



$(FA_1' + FA_2)' + (FA_2' + FA_1)'$?

FA_1 et FA_2 représentent le même langage

Problème 3

Est-ce que le langage reconnu par un automate fini est:
fini? **infini?** **vide?**

- **vide?** Le problème est décidable (on a décrit plusieurs algorithmes pour ce problème).

- **fini ou infini?**

Théorème. Soit F un automate fini qui contient N états.

1. Si F accepte un mot w tel que $N \leq \text{longueur}(w) < 2N$, alors le langage reconnu par F est infini.
2. Si le langage reconnu par F est infini, alors F accepte un mot w tel que $N \leq \text{longueur}(w) < 2N$.

Théorème

Il existe un algorithme qui peut décider si le langage reconnu par un automate fini est fini ou infini.

Algorithme:

Il suffit de tester tous les mots dont la longueur est entre N et 2N (il y a un nombre fini de testes et chaque teste demande un nombre fini d'étapes)

Question?