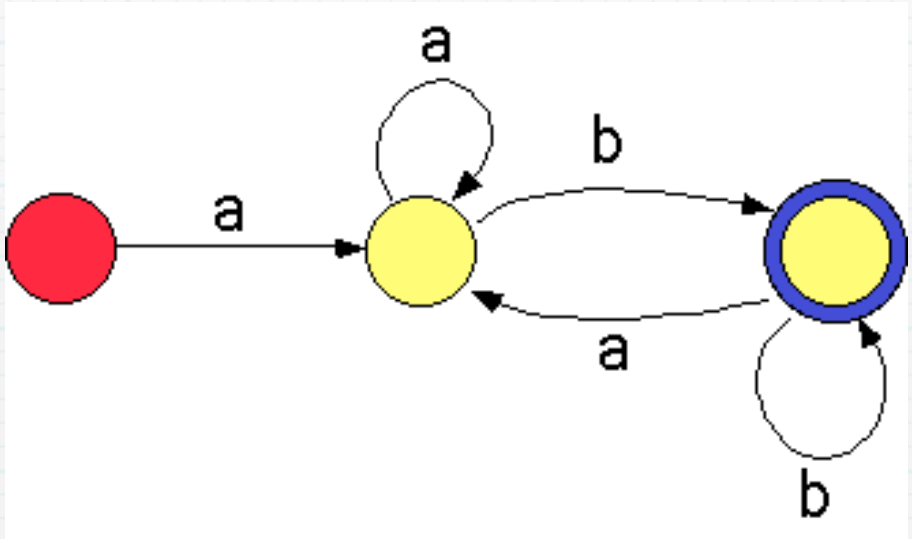
8. Automates finis

Automates finis

* Les automates finis sont des « machines abstraites » qui savent reconnaître l'appartenance ou la non-appartenance d'un mot à un langage régulier donné.
* Ces machines abstraites constituent un modèle théorique de référence.
  * Dans la pratique, nombreuses sont les applications qui implémentent la notion d'automates finis ou ses variantes (cela va du compilateur ... à la machine à café).
  * Un automate « lit » un mot écrit sur son ruban d'entrée.
  * Il part d'un état initial et à chaque lettre lue, il change d'état. Si, à la fin du mot, il est dans un état final, on dit qu'il reconnaît le mot lu.

Exemple d'automate fini



* Les mots reconnus par cet automate sont
  * ab
  * aab,abb
  * aaab,aabb,abab,abbb
  * ... et plus généralement tous les mots commençant par a et finissant par b.

Automate fini

* Un automate fini A est la donnée d'un quintuplet
* $(\Sigma, Q, \delta, q_0, F)$ tel que :
  Σ est un alphabet
  * Q est un ensemble fini d'états
  * δ est un ensemble de règles de transition
    $\delta \subseteq (Q \times (\Sigma \cup \{\varepsilon\}) \times Q)$
  * q_0 est l'état initial
  * F est un sous-ensemble de Q appelé l'ensemble des états finaux

Automates finis déterministes

* Un automate fini est déterministe si et seulement si δ est une fonction de transition telle que :

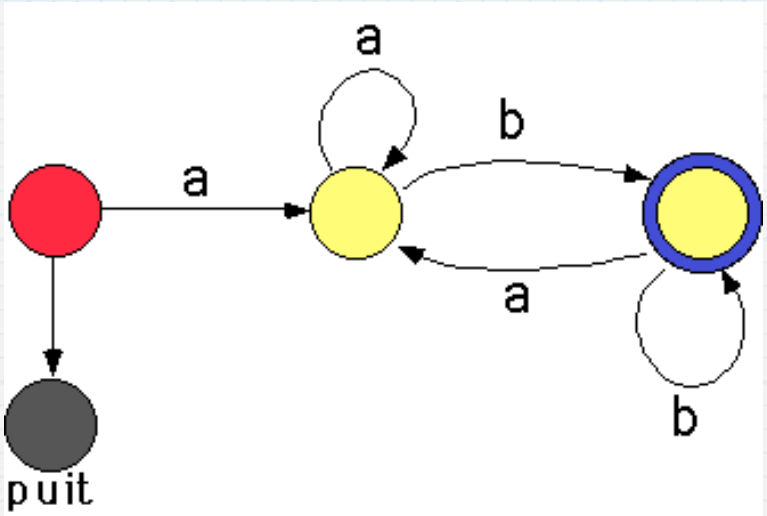
$$\delta : Q \times \Sigma \rightarrow Q$$

* On ne peut plus effectuer de transition sur ε.
* D'un état donné, il part au plus un seul arc étiqueté par une lettre donnée.
* En informatique, le déterminisme est le fait de ne pas avoir le choix entre plusieurs exécutions.

Automates finis complets

* Un automate fini et déterministe est complet
  * si et seulement si δ est une application de $Q \times \Sigma$ sur Q.
* De chaque état, il part alors exactement un arc étiqueté par chacune des lettres de l'alphabet Σ.
* Quand la fonction n'est pas une application, l'automate fini peut se trouver bloqué.
  * Un automate fini complet ne sera jamais bloqué, quitte à contenir des états surperflus : les états-poubelles ou le puits

Automate complet

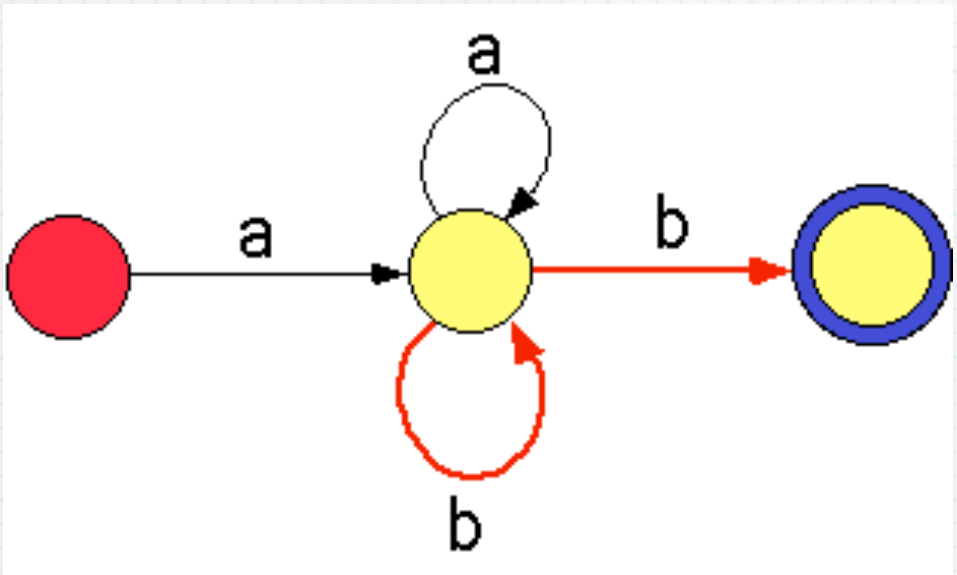


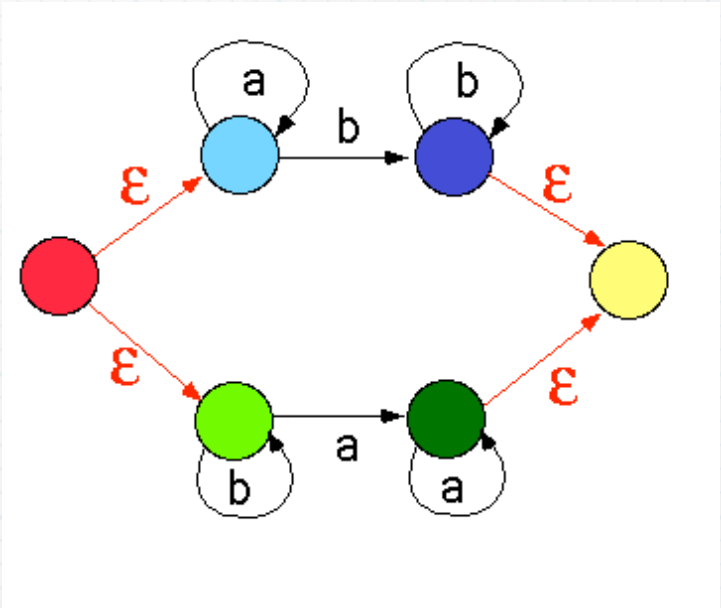
Reconnaissance

* Un mot w de longueur n est **reconnu** ou **accepté** par un automate fini s'il existe un chemin menant de q_0 à un état final de F étiqueté par la suite de lettres du mot w.
* Le chemin est alors une suite d'arcs étiquetés, pour i de 1 à n.

* Le mot w est égal à la concaténation des étiquettes: $a_1 \ldots a_n$.
  * k_0 est l'état initial q0.
  * k_n est un état final.
* L'ensemble des mots acceptés par un automate fini A forme le langage reconnu par cet automate. On le note : L(A)

Automates non déterministes





Non-déterminisme

* Dans un automate fini non-déterministe, il peut y avoir le choix entre plusieurs chemins lors de la lecture d'un mot.
* Pour qu'un mot soit accepté, il suffit que ses lettres étiquettent un chemin d'un état initial à un état final Cependant :
  * il peut y avoir d'autres chemins ne menant pas à un état final et étiqueté par le mot accepté par ailleurs, ou même des lectures du mot s'arrêtant en cours de route.
* Notez que les transitions peuvent être étiquetées par le mot vide ε, ce qui n'est pas le cas dans un automate fini déterministe.

Déterminisme / non-déterminisme

* Théorème si un langage est reconnu par un automate fini, alors il est également reconnu par un automate fini déterministe.
  * si l'automate fini du départ A est déterministe, c'est évident
  * si l'automate de départ n'est pas déterministe, on se propose de construire un automate fini déterministe B qui intègre tous les choix existant dans l'automate de départ (cf. algorithme de déterminisation)
  * il restera à prouver formellement que le nouvel automate B accepte exactement les mots acceptés par A.

Algorithme de déterminisation

* Soit $A = (\Sigma, Q, \delta, q0, F)$ un automate fini non-déterministe, on construit l'automate fini déterministe $B = (\Sigma, Q', \delta', q0', F')$ où Q' sera alors un sous-ensemble de $P(Q)$

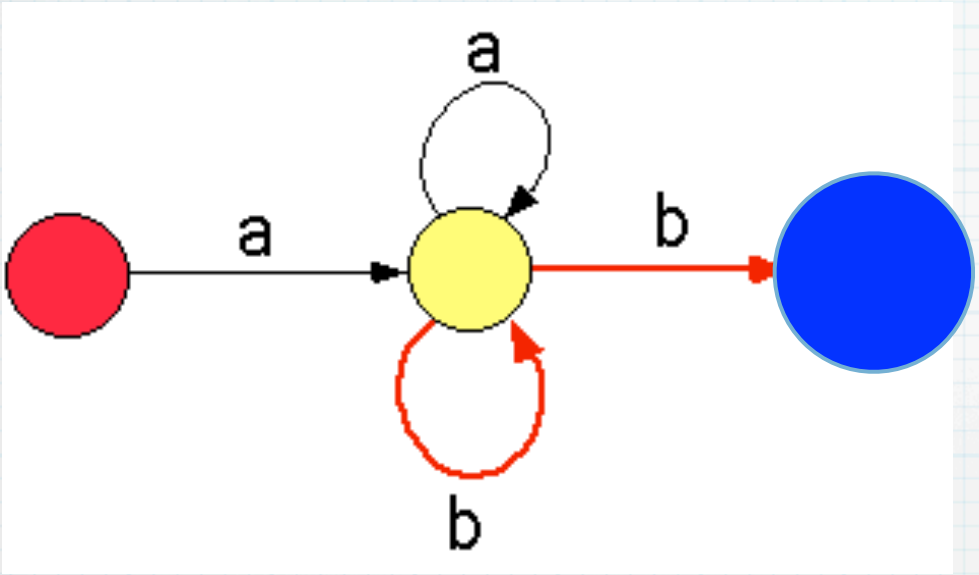
* $\delta' \leftarrow \varnothing$

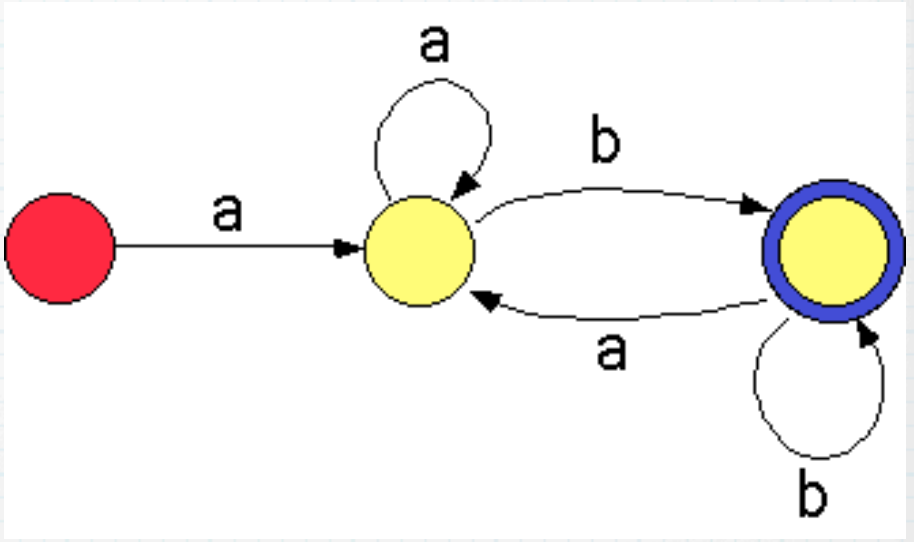
  $q0' \leftarrow \{q0\} \cup \{$les états q tels que $(q0,\varepsilon,q) \in \delta \}$

* pour tout $q' \in Q'$ non encore considéré faire
  pour tout $\sigma \in \Sigma$ faire
  $q'' \leftarrow \{ y$ / il existe $x \in q'$, $y \in Q$ tels que
  $(x,\sigma,y) \in \delta \}$
  si $q'' \neq \varnothing$ alors
  $q'' \leftarrow q'' \cup \{y$ /il existe $y \in q''$ et $z \in Q$ tels que $(y,\varepsilon,z) \in \delta \}$
  $\delta' \leftarrow \delta' \cup \{ (q', \sigma, q'')$
  $Q' \leftarrow Q' \cup \{q''\}$

* $F' \leftarrow \{q'$ tels que $q' \cap F \neq \varnothing \}$

Déterminisation d'un automate

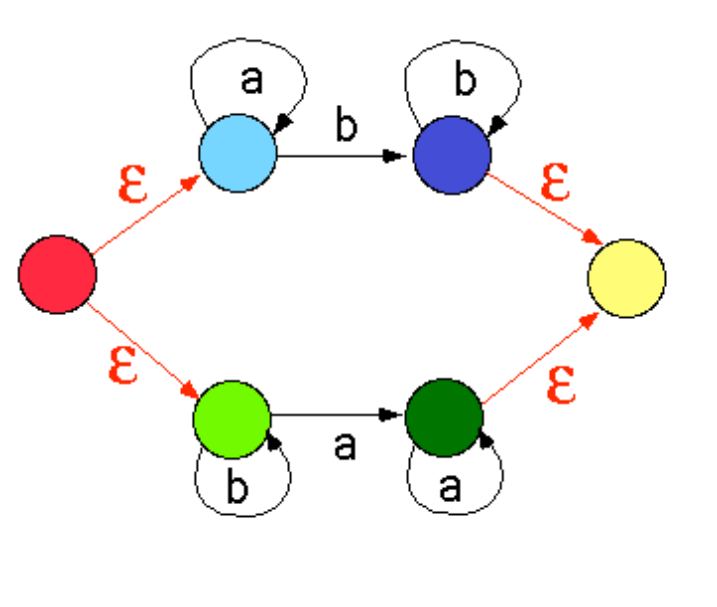


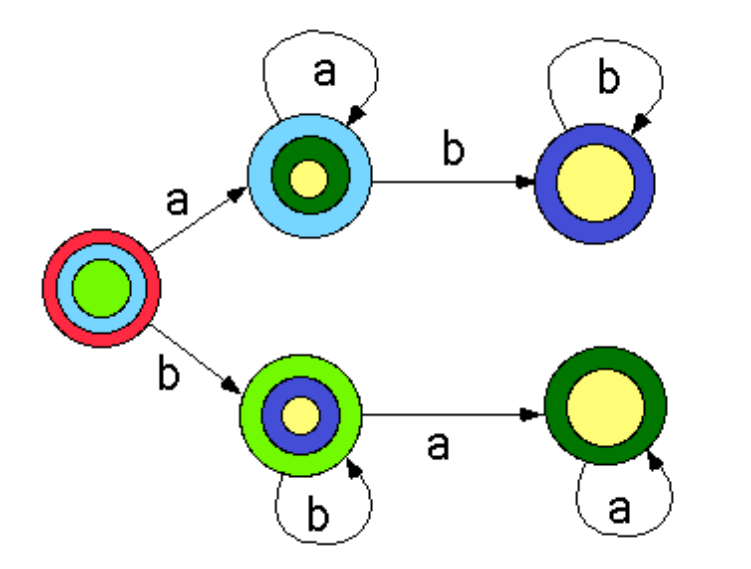


	a	b
1	2	x
2	2	2,3
3	x	x

	a	b
1	2	x
2	2	2,3
3	x	x
2,3	2	2,3

Déterminisation d'un automate





	a	b
1	2, 5, 6	3, 4, 6
2	2	3, 6
3	x	3, 6
4	5, 6	4
5	5, 6	x
6	x	x

	a	b
1	2, 5, 6	3, 4, 6
2, 5, 6	2, 5, 6	3, 6
3, 4, 6	5, 6	3, 4, 6
3, 6	x	3, 6
5, 6	5, 6	x

Théorème de Kleene

Un langage sur un alphabet Σ est régulier si et seulement si il est reconnu par un automate fini.

Démonstration

* On sait construire un automate pour chaque élément de la base. Pour chacune des 3 opérations, on sait construire un automate lui correspondant à partir des automates reconnaissant les langages de départ.
* Réciproquement, on peut passer d'un automate fini à un langage régulier (admis).

Rappel définition inductive

(B) $\emptyset \in R$

$\{\varepsilon\} \in R$

$\{a\} \in R$, pour tout $a \in \Sigma$

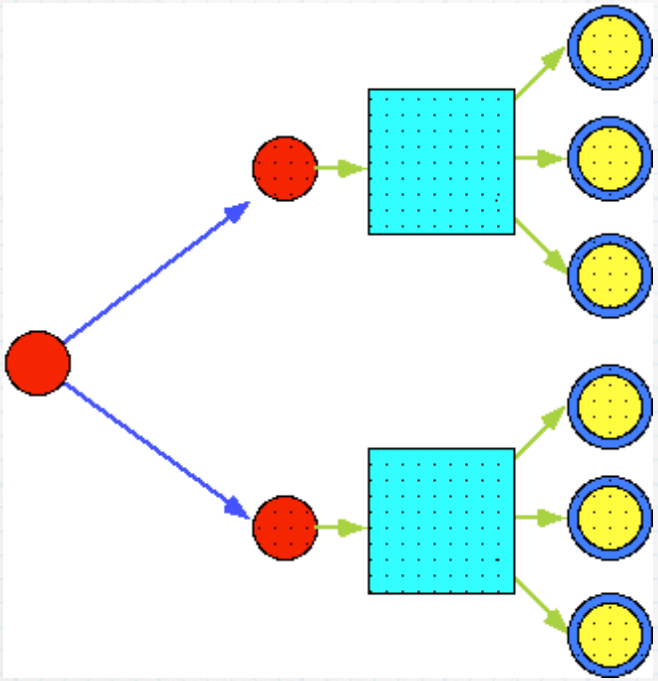
(I) si $L, M \in R$ alors

$L \cup M \in R$

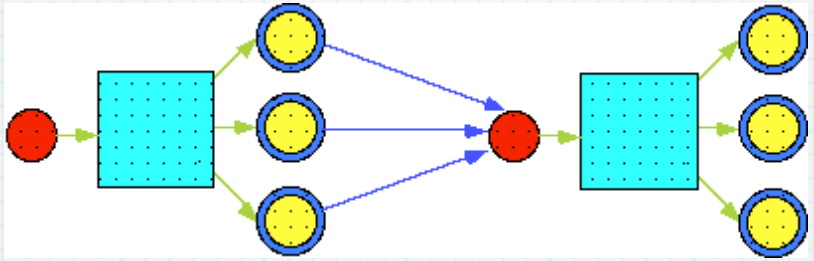
$L.M \in R$

$L^* \in R$

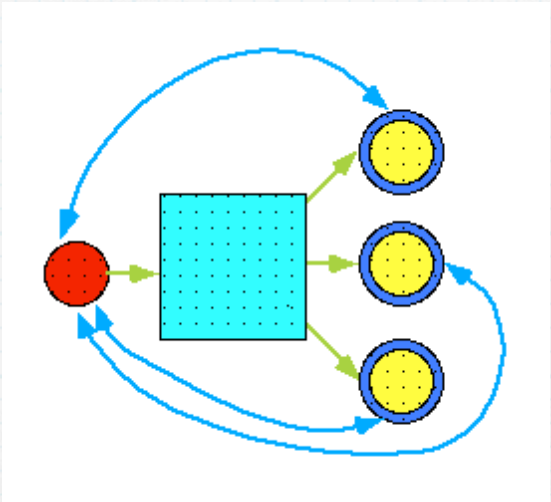
Union de langages



Concaténation de langages

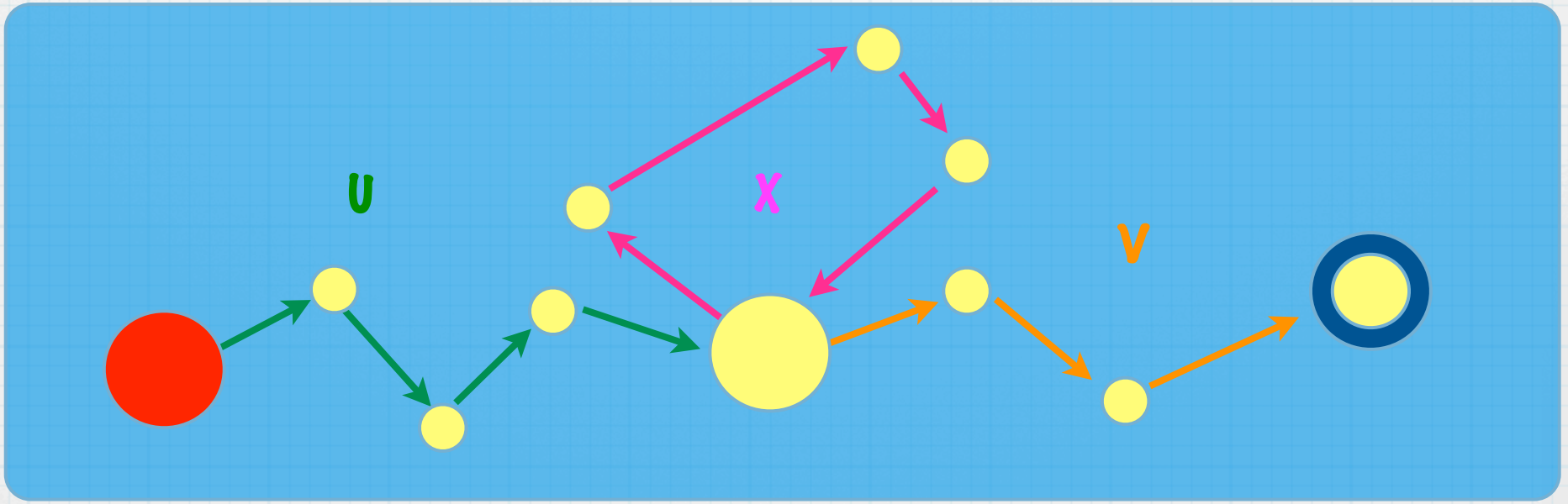


Etoile d'un langage



Lemme de l'étoile

* Soit L un langage reconnu par un automate fini déterministe. On note n le nombre de ses états.
  * A tout mot de longueur k >n correspond un chemin de longueur k dans l'automate qui passe donc par k+1 >n+1 états, et donc au moins deux fois par le même état ... d'où le résultat :



* Le mot w se factorise alors w=uxv
* Et, pour tout entier i, ux^iv est reconnu par l'automate

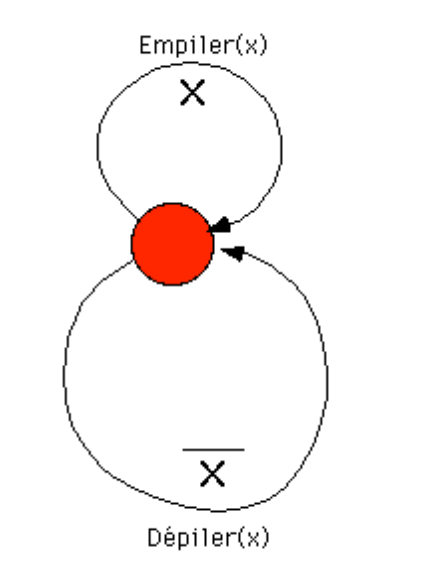
Langages

Un langage peut être défini de multiples façons

* Par propriétés Lp est l'ensemble des mots contenant un nombre pair de 1 : $L_p = \{ \varepsilon, 0, 00, 11, 000, 011, 101, 110, \ldots \}$
* Par induction
  * $\varepsilon \in L_i$
  * Si $u \in L_i$ alors $0u$, $u0$ et $1u1 \in L_i$
* A l'aide d'une expression régulière : $L_r = (0^*10^*1)^* 0^*$
* A l'aide d'un automate fini
* On peut aussi s'intéresser au nombre de mots de longueur n de ce langage et à sa série génératrice

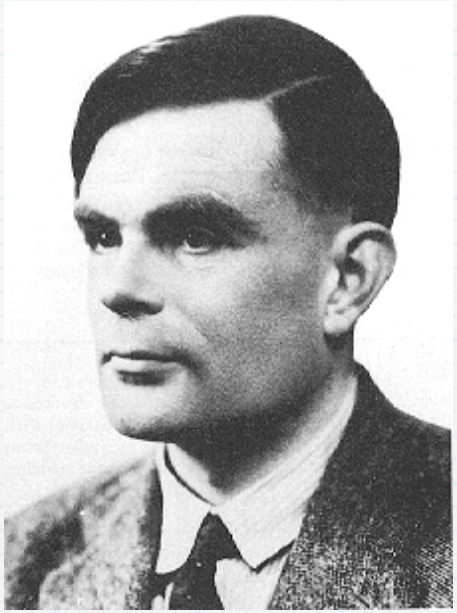
Langages non rationnels

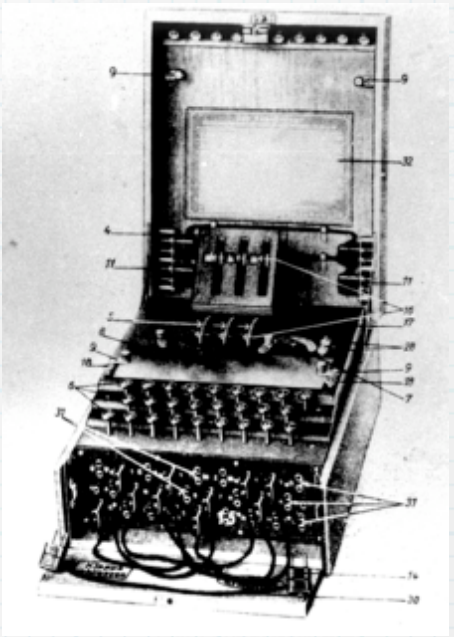
* L'ensemble des mots M contenant autant de a que de b n'est pas régulier donc n'est pas reconnu par un automate déterministe
  * Application du lemme de l'étoile
  * Pourtant, ce langage peut être défini par induction
    * $\varepsilon \in M$
    * Si $u,v \in M$ alors $aubv$ et $buav \in M$
  * Il existe une autre forme d'automates, les automates à pile qui reconnaissent une classe plus grande de langages.



Machine de Turing (1912-1954)

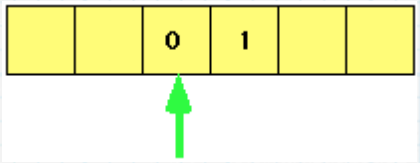
* Une bande infinie
* Un jeu de caractères A,B...
* Un ensemble fini d'états S,T ...
* Des opérations :
  * Lecture
  * Ecriture
  * Déplacement à gauche (1 case)
  * Déplacement à droite (1 case)
  * Un ensemble de règles qui précise, pour chaque état, le comportement de la machine à l'étape suivant. Plus précisément,
    - La lettre à écrire dans la case en cours
    - Le nouvel état de la machine
    - Le sens de déplacement de la tête de lecture



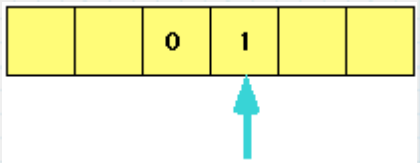


Exemple

* Une machine de Turing qui boucle sur deux états

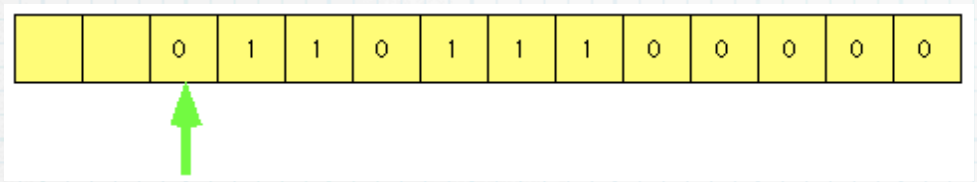


	0	1
E1	0,E2,D	0,E2,D
E2	1,E1,G	1,E1,G

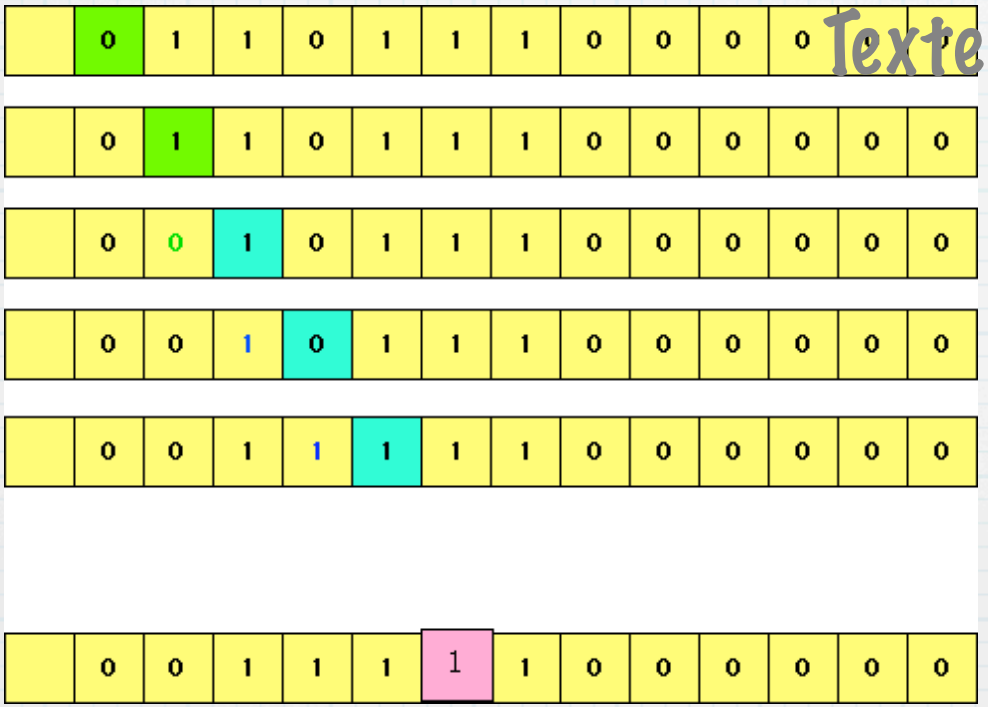


Exemple

* L'addition de deux nombres
(cf p 158 "Les mathématiques d'aujourd'hui" chez Belin)



	0	1
E1	0,E1,D	0,E2,D
E2	1,E3,D	1,E2,D
E3	ARRET	ARRET



Machine de Turing

- La machine de Turing a permis de classifier les problèmes
  - Les problèmes calculables sont les problèmes résolus par une machine de Turing.
  - Il existe des problèmes non calculables : par exemple le problème de l'arrêt de la machine de Turing.
  - Les problèmes de complexité P : ceux qui sont résolubles par une machine de Turing en temps polynomial. De fait ce sont les seuls problèmes effectivement résolubles par ordinateur.
  - Les problèmes N-P (Non déterministe en temps polynomial) qui sont résolus à l'aide d'une machine de Turing non déterministe. Ceux sont les problèmes qui "reconnaissent" une solution en temps polynomial.
    - Problème du sac à dos
    - Problème du voyageur de commerce
- Les problèmes intrinsèquement difficiles, comme celui de l'arrêt d'une machine de Türing.