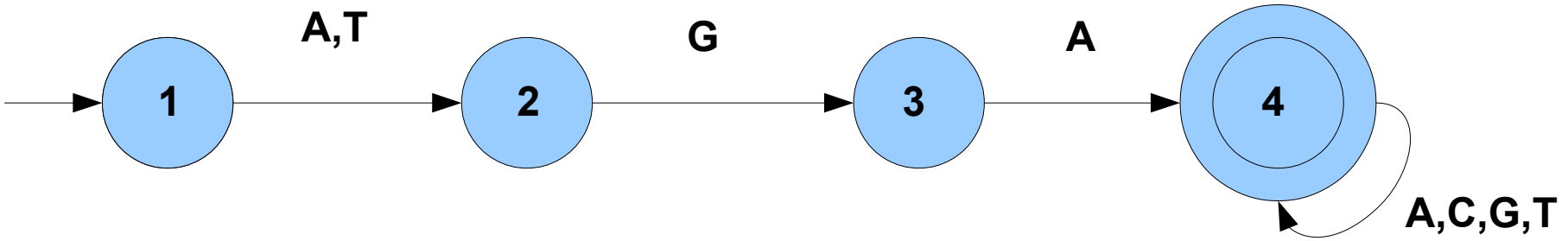
LES AUTOMATES



Martine.Leonard@univ-rouen.fr
Master BioInforMatique

Mots et langages

lettres 

Mots et langages

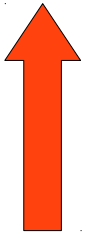
lettres  mots

Mots et langages

lettres  mots  langages

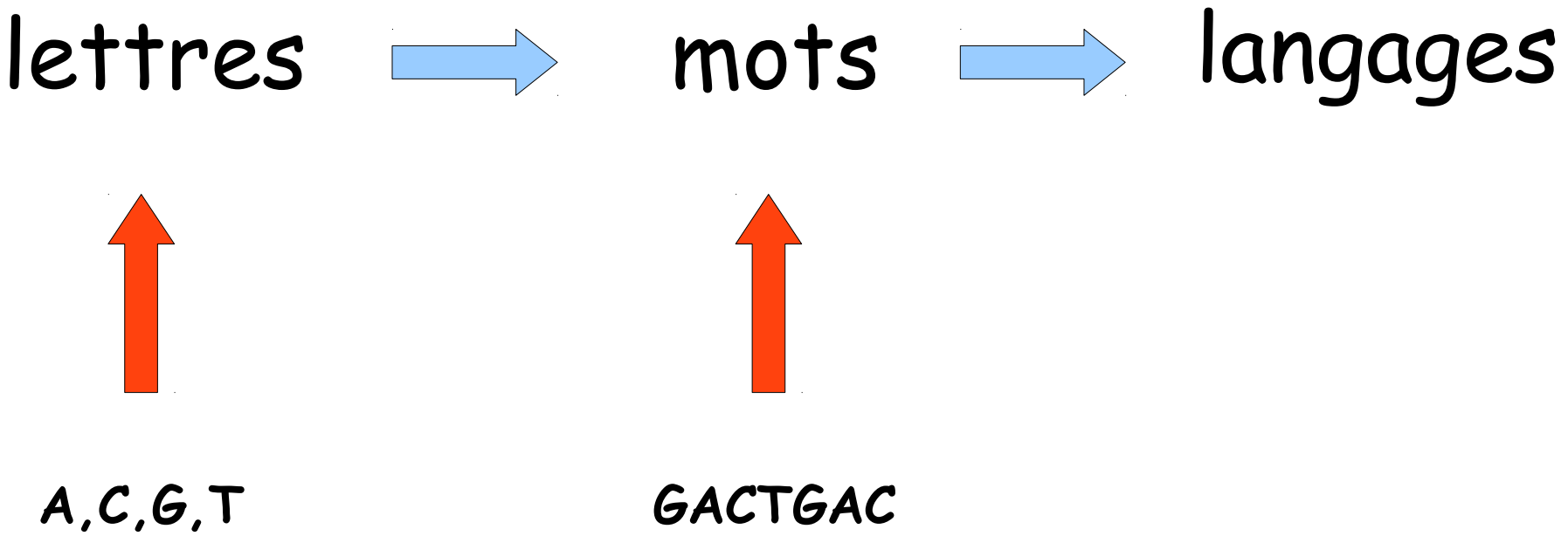
Mots et langages

lettres → mots → langages

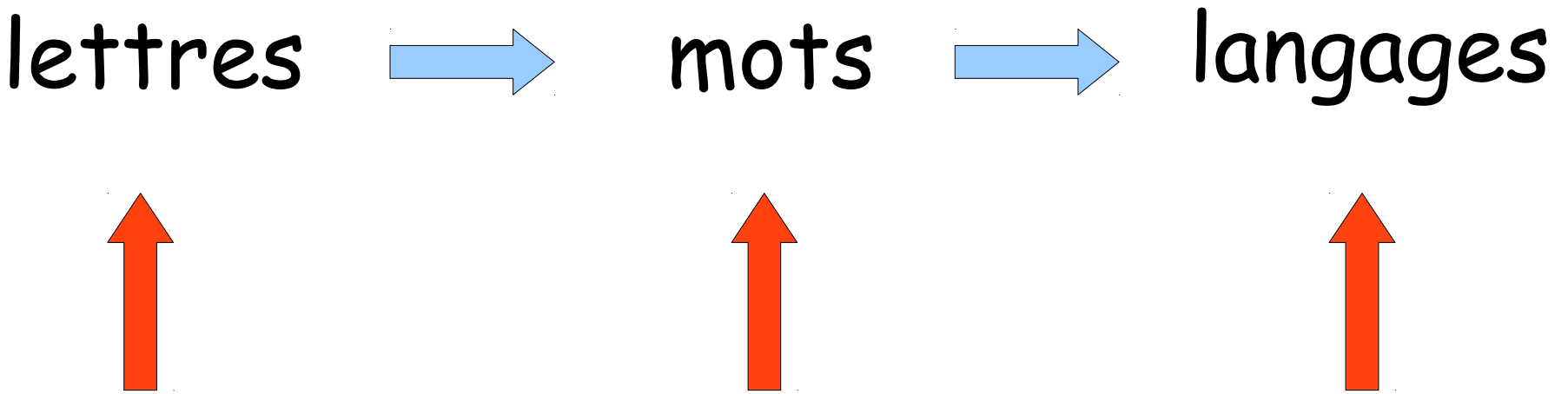


A,C,G,T

Mots et langages



Mots et langages



Mots et langages : alphabet

Un **alphabet** est un ensemble fini de symboles appelés **lettres**.
L'alphabet est souvent noté Σ ou A .

Exemples:

Σ = {a, b, c}
Σ = {0, 1} alphabet binaire
A = {+, -, *, /} alphabet des opérateurs arithmétiques
Σ = {1, b, -}
Σ = {A, C, G, T} alphabet des nucléotides
Σ = {a, r, n, d, c, e, q, g, h, i, l, k, m, f, p o, u, s, t, w, y, v}
alphabet des acides aminés

On travaillera généralement avec les alphabets {a, b, c…} ou {A, C, G, T}.

Mots et langages : mots définitions et notations (1)

Un **mot** sur un alphabet donné est une suite finie de lettres de cet alphabet. On notera ε ou **1** le **mot vide** qui ne contient aucune lettre (suite vide).
Les mots sont souvent notés u, v, w, x, y, z.

Exemples:

u = aba, v = 001, w = *ACAGAG*, x = gadmhhghypgngvtdl

On considère souvent les séquences biologiques comme des mots.

ADN : mots sur Σ = {*A*, *C*, *G*, T}

ARN : mots sur Σ = {*A*, *C*, *G*, U}

séquences protéiques :

mots sur Σ = {a, r, n, d, c, e, q, g, h, i, l, k, m, f, p, s, t, w, y, v} .

Mots et langages : mots définitions et notations (2)

$|u|$ est la **longueur** du mot u. C'est le nombre de lettres de u.
Exemples: $|\varepsilon| = 0$, $|a| = 1$, $|ACAGAG| = 6$

$|u|_a$ est le **nombre d'occurrences** de a dans le mot u.

Exemples: $|\varepsilon|_a = 0$, $|a|_a = 1$, $|ACAGAG|_G = 2$

Ça marche aussi avec les mots : $|ACAGAG|_{GA} = 1$

On note Σ^* **l'ensemble de tous les mots de longueur finie** que l'on peut construire sur l'alphabet Σ (y compris le mot vide).

Mots et langages : mots définitions et notations (3)

. est l'opération de concaténation **des mots :**

ACGAT. GATCA = ACGATGATCA

Si u et v sont 2 mots, u . v est souvent noté **uv** en omettant le . de la concaténation.

Pour tout mot u on a : ε . **u = u** . ε **= u**

Mots et langages : mots définitions et notations (4)

CGATGG

CGA est un **préfixe** ou **facteur gauche** de CGATGG.

C'est un préfixe **propre** parce qu'il est différent du mot.

L'**ensemble des préfixes** de CGATGG est

{ε, C, CG, CGA, CGAT, CGATG, CGATGG}

Mots et langages : mots définitions et notations (5)

CGATGG

GG est un **suffixe** ou **facteur droit** de CGATGG.

C'est un suffixe **propre** parce qu'il est différent du mot.

L'**ensemble des suffixes** de CGATGG est

{ε, G, GG, TGG, ATGG, GATGG, CGATGG}

Mots et langages : mots définitions et notations (6)

CGATGG

ATG est un **facteur** de CGATGG.

C'est un **facteur interne** parce qu'il n'est ni préfixe, ni suffixe de CGATGG.

Les facteurs d'un mot sont les préfixes, les suffixes et les facteurs internes de ce mot.

L'**ensemble des facteurs** de CGATGG est
{ε, C, G, A, T, CG, GA, AT, TG, GG, CGA, GAT, ATG, TGG, CGAT, GATG, ATGG, CGATG, GATGG, CGATGG}

Mots et langages : langages définitions et notations (1)

Langage sur l'alphabet Σ = tout ensemble de mots de Σ^*.

Exemples :

X_1 = {*AACA, CAT, GACTA*} sur Σ = {*A, C, G, T*}

X_2 = {mots de longueur paire}

X_3 = {ε, abaab, aaa, a, b} sur Σ = {a,b}

X_4 = {mots qui contiennent *CC* en facteur} sur Σ = {*A, C, G, T*}

$X_5 = \varnothing$

$X_6 = \Sigma^*$

$X_7 = \{\varepsilon\}$

Mots et langages : langages définitions et notations (2)

Si X et Y sont deux langages sur l'alphabet Σ,

$X \cup Y$ ou $X + Y$ est l'**union** de X et Y.

$X \cap Y$ est l'**intersection** de X et Y.

$\Sigma^* - X$ est le **complémentaire** de X dans Σ^*.

$X . Y$ est le **produit de concaténation** de X par Y.
C'est l'ensemble des mots obtenus en concaténant un mot de X et un mot de Y.
$X . \varnothing = \varnothing . X = \varnothing$ et $X . \{\varepsilon\} = \{\varepsilon\} . X = X$

Mots et langages : langages définitions et notations (3)

On définit la puissance d'un langage X quelconque par :

$X^0 = \{\varepsilon\}$

$X^n = X^{n-1}.\ X$ pour tout $n > 0$

X^n est l'ensemble des mots qui sont la concaténation de n mots de X.

$X^* = X^0 \cup X^1 \cup X^2 \cup \ldots\ldots \cup X^n \cup \ldots\ldots$

X^* est l'ensemble des mots qui sont la «concaténation» d'un nombre quelconque de mots de X.

$\varnothing^* = \{\varepsilon\}$

$X^+ = X^1 \cup X^2 \cup \ldots\ldots \cup X^n \cup \ldots\ldots$

X^+ est l'ensemble des mots qui sont obtenus par « concaténation » d'un nombre quelconque non nul de mots de X.

Mots et langages : langages définitions et notations (4)

$X = \{AT, CGA\}$ et $Y = \{TG, ATG\}$ sur $\Sigma = \{A, C, G, T\}$

$X \cup Y = \{AT, CGA, TG, ATG\}$

$X \cup \Sigma^* = \Sigma^*$, $X \cap Y = \varnothing$, $X \cap \Sigma^* = X$

$X . Y = \{ATTG, ATATG, CGATG, CGAATG\}$

$X^2 = \{ATAT, ATCGA, CGAAT, CGACGA\}$

$\Sigma^2 = \Sigma . \Sigma$ est l'ensemble des mots de longueur 2 sur Σ.

$\Sigma^n = \Sigma . \Sigma . \ldots . \Sigma$ est l'ensemble des mots de longueur n sur Σ.

$\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \ldots \Sigma^n \cup \ldots$ est l'ensemble des mots de longueur quelconque sur Σ.

$\Sigma^+ = \Sigma^1 \cup \ldots \Sigma^n \cup \ldots$ est l'ensemble des mots de longueur > 0.

Mots et langages : langages définitions et notations (5)

Préfixes, suffixes et facteurs d'un langage

Y = {TG, ATG} sur Σ = {A, C, G, T}
Préfixe (Y) = {ε, T, TG, A, AT, ATG}
Suffixe (Y) = {ε, G, TG, ATG}
Facteur (Y) = {ε, T, G, A, TG, AT, ATG}

X = {ATG} . Σ^* sur Σ = {A, C, G, T}
Préfixe (X) = {ε, A, AT} $\cup$ {ATG} . Σ^*
Suffixe (X) = Σ^*
Facteur (X) = Σ^*

Mots et langages : exercices (1)

0- Soit Σ = {a, b}, donnez les 10 premiers mots (par ordre de longueur d'abord puis lexicographique) de Σ^*.

1- Soit X = {*AC*, *CAG*} et Y = {*ACT*, *GGA*}, calculez X . Y et Y . X.

2- Soit X = {*AACA*, *GA*}, calculez préfixe(X), suffixe(X), facteur(X), X^2.

3- Sur Σ = {a, b}, donnez explicitement les langages suivants :

X1 = { w dans Σ^* tq w contient au plus un b et $|w| < 4$ }

X2 = { w dans Σ^* tq w se termine par a et $|w| \leq 3$ }

X3 = { w dans Σ^* tq le nombre de a et de b est pair et $|w| \leq 4$ }

Mots et langages : exercices (2)

4- En utilisant les opérations sur les langages, donnez une expression des langages suivants sur Σ = {A, C, G, T}:

X1 ensemble des mots commençant par AC

X2 ensemble des mots de longueur paire

X3 ensemble des mots de longueur supérieure ou égale à 2

5- Essayer de donner quand c'est possible une formulation plus simple pour les langages suivants :

X1 = {G} . {A}+ . {G} . {A}+ sur Σ = {A, C, G, T}

X2 = {aa, ab, ba, bb}* sur Σ = {a, b}

X3 = ({a}* + {b}*)* sur Σ = {a, b}

X4 = ({a}* . {b}*)* sur Σ = {a, b}

X5 = {A}+ + {A}*. {C} . {A}* sur Σ = {A, C, G, T}

Les expressions rationnelles ou régulières (1)

Quand cela est possible, un langage est donné par l'ensemble de ses mots.

Exemples:

$X_1 = \{AT, GCA\}$

X_2 = {w dans {A, C, G, T}* tel que w commence par A et $|w| \leq 2$}

= {A, AA, AC, AG, AT }

X_6 = {w dans {a,b}* tel que w contient aba en facteur et $|w| \leq 5$}

= {aba, aaba, baba, abaa, abab, aabaa, aabab, babaa, babab}

Les expressions rationnelles ou régulières (2)

Mais quand il y a une infinité de mots ou même un nombre « assez grand » de mots, on ne peut que donner une caractérisation des mots du langage (ensemble de critères qui fait qu'un mot est dans le langage).

Exemples:

X_1 = {w dans {A, C, G, T}* tel que w commence par A et |w| ≤ 2 500}

X_2 = {w dans {A, C, G, T}* tel que w commence par A} = {A} . {A, C, G, T}*

X_3 = {w dans {A, C, G, T}* tel que w contient CC}

= {A, C, G, T}*. {CC} . {A, C, G, T}*

X_4 = {w dans {a, b}* tel que w ne contient que des a}

= {ε, a, aa, aaa, aaaa…} = {a}*

X_5 = {w dans {a, b}* tel que w commence par un a suivi d'un nombre quelconque de b} = {a} . {b}*

Les expressions rationnelles ou régulières (3)

Si un langage peut s'exprimer en utilisant uniquement **les singletons {a}, ∅, {ε}** et les opérations ∪ . * et les ( ), on dit que le langage est **régulier** ou **rationnel.**

On peut alors désigner le langage par une **expression régulière** ou **rationnelle**.

Les expressions rationnelles ou régulières (4)

Question : tout langage peut-il s'écrire à l'aide d'une expression rationnelle ?

Réponse : non, il existe des langages qui ne sont pas réguliers.

Exemple : le langage $\{ a^n b^n, n \geq 0 \}$,
on ne peut pas trouver d'expression du langage n'utilisant que les singletons , $\varnothing$, $\{\varepsilon\}$ et les opérations +, . et *.

Les expressions rationnelles ou régulières (5)

Langage rationnel	expression rationnelle
{a}*	a*
{a} . {b}*	a . (b)*
{A, AA, AC, AG, AT}	A + A.A + A.C + A.G + A. T ou A . (ε + A + C + G + T)
{A} . {A,C,G,T}* = {A} . ({A} ∪ {C} ∪ {G} ∪ {T})*	A . (A + C + G + T)*

Les expressions rationnelles ou régulières exercices

1- Donnez une expression régulière pour désigner les langages suivants sur $\Sigma = \{ A, C, G, T \}$:

X_1 ensemble des mots commençant par *AC*

X_2 ensemble des mots se terminant par *AC*

X_3 ensemble des mots ayant *AC* en facteur

X_4 ensemble des mots de longueur paire

X_5 ensemble des mots contenant un nombre paire de *A*

X_6 ensemble des mots de longueur supérieure ou égale à 2

2- Construisez une expression rationnelle qui désigne l'ensemble des séquences d'ARN qui contiennent un codon start *AUG* et ensuite un codon stop *UAG*, *UGA* ou *UAA*.

Analyseurs d'expressions régulières

Les expressions rationnelles sont utilisées par de nombreux outils et langages informatiques :

Lex, grep, csh et sh, awk, Emacs, Perl, php, python ....

Elles permettent de définir des **motifs** de texte ou **modèles**.

On peut ainsi vérifier si un texte donné suit un modèle particulier mais aussi effectuer des modifications ciblées.

Analyseurs d'expressions régulières
un exemple avec `grep` (1)

`grep` = *Global (search for) Regular Expression and Print*

Problème :

Rechercher à l'aide de la commande `grep` si une séquence d'ARN contient dans le même cadre de lecture un codon start AUG et un codon stop UAG, UGA ou UAA.

Analyseurs d'expressions régulières
un exemple avec grep (2)

Solution :

Décrire à l'aide d'une expression rationnelle les textes commençant par un codon start suivi de codons en nombre quelconque suivis d'un codon stop.

L'expression régulière qui désigne un codon start s'écrit : **AUG**
Celle qui désigne un codon quelconque s'écrit : **[ACGU][ACGU][ACGU]**
Une suite de codons : **\([ACGU][ACGU][ACGU]\)***
La liste des codons stop possibles : **UAG\|UGA\|UAA**

Notre expression régulière pourra donc s'écrire :

AUG\([ACGU][ACGU][ACGU]\)*\(UAG\|UGA\|UAA\)

Analyseurs d'expressions régulières
un exemple avec grep (3)

Commande :

```
grep -n 'AUG\([ACGU][ACGU][ACGU]\)*\(UAG\|UGA\|UAA\)' testGrep.txt
```

Contenu de testGrep.txt :

```
ACAUGUGGCAUCGUUGAGUA
ACAUGUGGCAUCGUGAGUA
ACGUGGCAUCGAUUGAGUA
ACGUGGCAUGGAUUAGUA
AUGUGGCAUCGAUAAUGGUAA
AUGUGGCAUCGAUUGGGUA
```

Résultat :

```
1:ACAUGUGGCAUCGUUGAGUA
4:ACGUGGCAUGGAUUAGUA
5:AUGUGGCAUCGAUAAUGGUAA
```

Analyseurs d'expressions régulières
un exemple avec grep (4)

Commande :

```
grep -n 'AUG\([ACGU][ACGU][ACGU]\)*\(UAG\|UGA\|UAA\)' testGrep.txt
```

Contenu de `testGrep.txt` :

ACAUGUGGCAUCGUUGAGUA	reconnu
ACAUGUGGCAUCGUGAGUA	non reconnu (pb cadre)
ACGUGGCAUCGAUUGAGUA	non reconnu (pas de start)
ACGUGGCAUGGAUUAGUA	reconnu
AUGUGGCAUCGAUAAUGGUAA	reconnu (2 occurrences)
AUGUGGCAUCGAUUGGGUA	non reconnu (pas de stop)

Résultat :

```
1:ACAUGUGGCAUCGUUGAGUA
4:ACGUGGCAUGGAUUAGUA
5:AUGUGGCAUCGAUAAUGGUAA
```

Les expressions rationnelles ou régulières un exemple avec grep (5)

Commande :

```
grep -n 'AUG\([ACGU][ACGU][ACGU]\)*\(UAG\|UGA\|UAA\)' testGrep.txt
```

La commmande ci-dessus permet d'extraire du fichier une information en passant par le **filtre** défini par l'expression régulière.

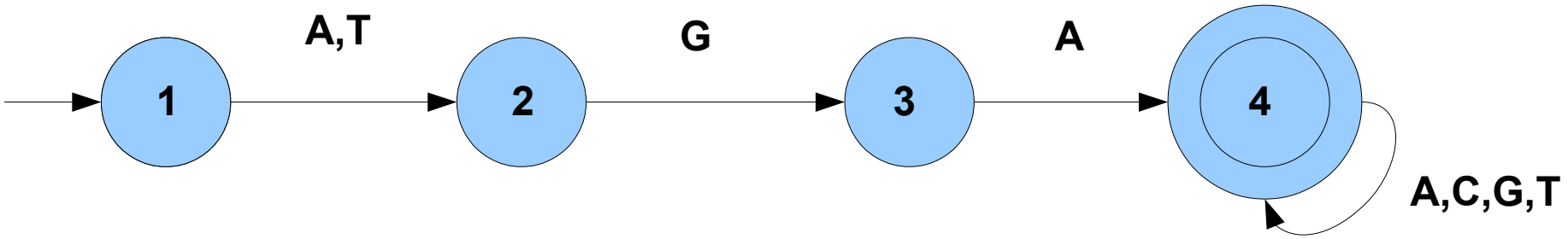
Analyseurs d'expressions régulières un exemple avec php (1)

On peut apporter des modifications ciblées à un texte en utilisant une expression régulière pour filtrer l'information.

Analyseurs d'expressions régulières
un exemple avec php (2)

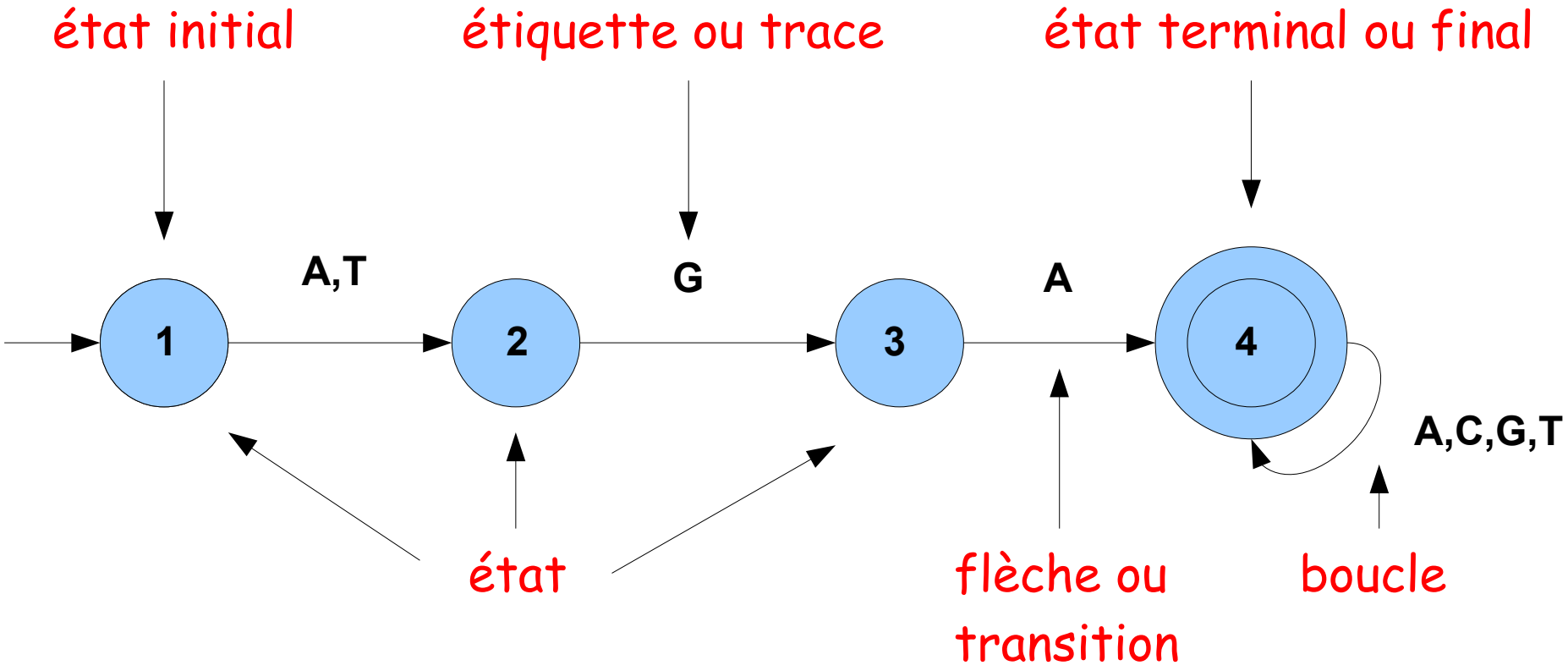
```
<?
Function noaccent($txt) { // Supprime les accents
   // on définit le texte à traiter
   $temp = $txt;
   // puis les caractères recherchés entre [ et ]
   $pattern = "[àâ]";
   // on remplace ces caractères par le caractère a
   $temp = eregi_replace($pattern,"a",$temp);
   // on définit un deuxième filtre
   $pattern = "[éèêë]";
   $temp = eregi_replace($pattern,"e",$temp);
   // et ainsi de suite pour les autres lettres...
   return($temp);
}
?>
```

Automates et langages reconnaissables définitions (1)



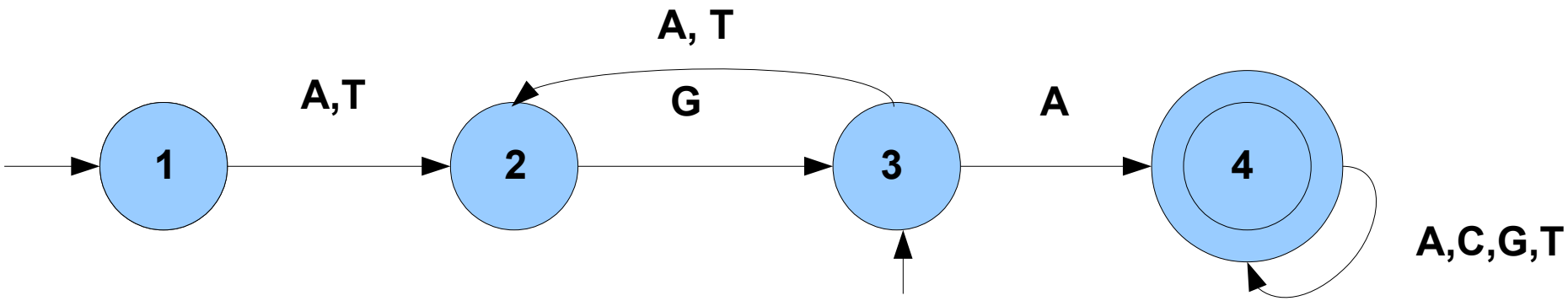
Un **automate** est une machine virtuelle qui **«reconnait»** des mots. L'ensemble des mots reconnus par un automate est le **langage reconnu** par l'automate.

Automates et langages reconnaissables définitions (2)



Un automate est un graphe orienté valué. Il peut avoir plusieurs états initiaux et terminaux. Un état peut être à la fois initial et terminal.

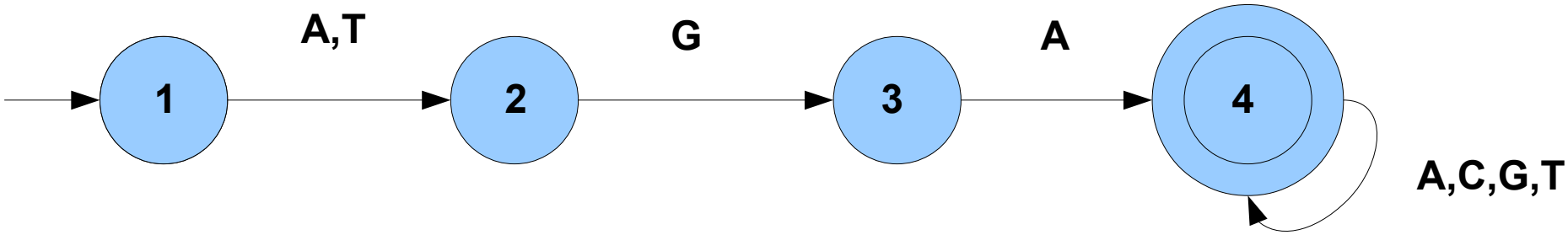
Automates et langages reconnaissables définitions (3)



Un automate peut être représenté à l'aide de sa **table de transitions**:

	A	C	G	T
-> 1	{2}	∅	∅	{2}
2	∅	∅	{3}	∅
-> 3	{2,4}	∅	∅	{2}
<- 4	{4}	{4}	{4}	{4}

Automates et langages reconnaissables définitions (4)



Un **chemin réussi** dans l'automate est un chemin qui part d'un état initial et se termine dans un état terminal.

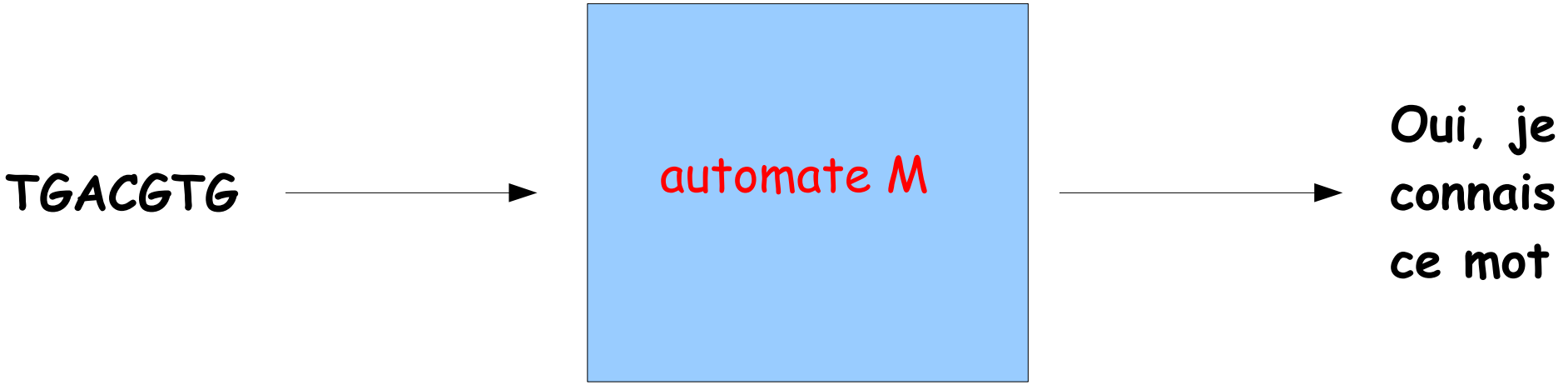
Un mot w est **reconnu** par un automate M si il existe au moins un chemin réussi d'étiquette w dans M.

Le **langage reconnu** par un automate M, noté **L(M)**, est l'ensemble des traces des chemins réussis.

L'automate ci-dessus reconnaît tous les mots qui commencent par un 'A' ou un 'T' suivi de 'GA' puis de n'importe quel mot de {A, C, G, T}* :

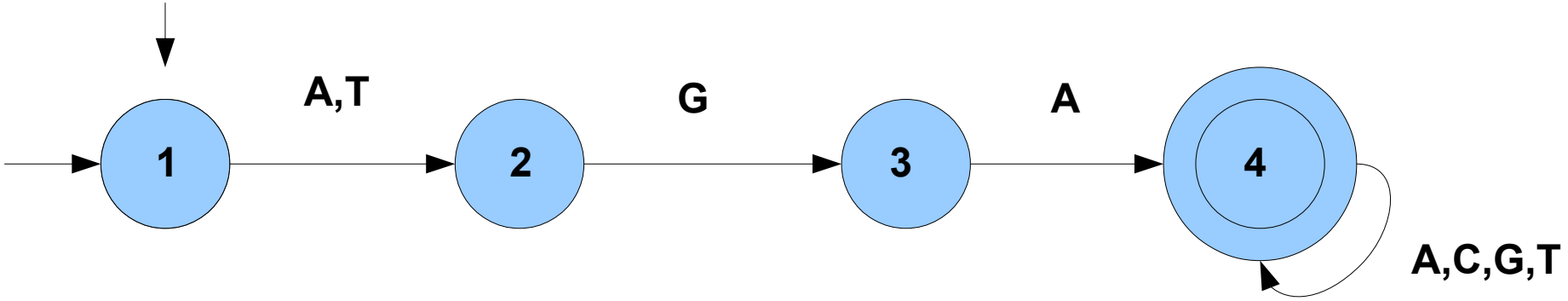
$$L(M) = \{A, T\} . \{GA\} . \{A, C, G, T\}^*$$

Automates et langages reconnaissables fonctionnement (1)



Automates et langages reconnaissables fonctionnement (2)

TGACGTG

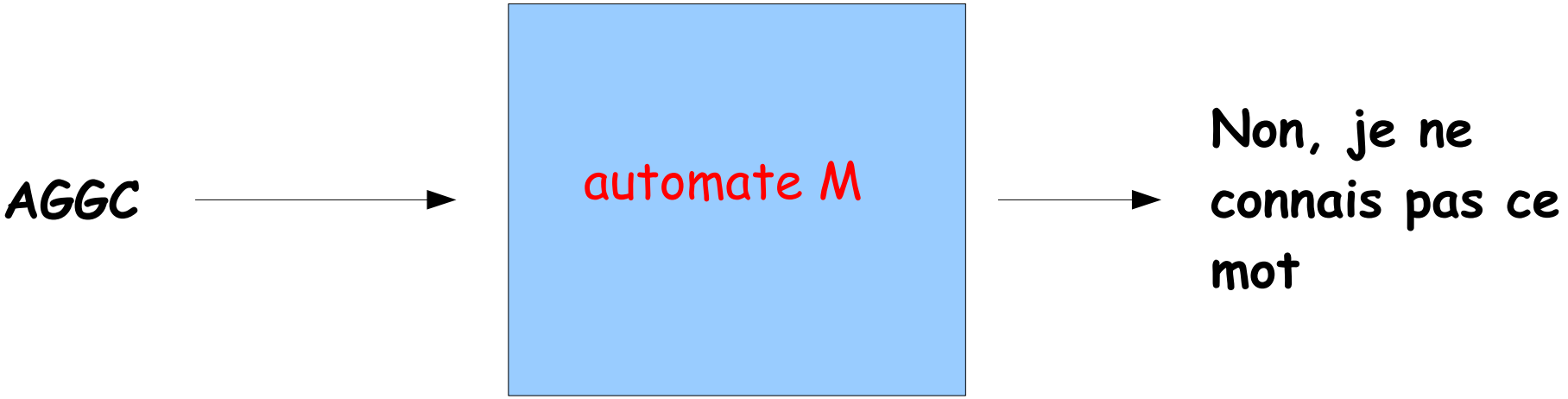


l'automate est dans l'état initial 1
lecture de 'T' => l'automate passe dans l'état 2
lecture de 'G' => l'automate passe dans l'état 3
lecture de 'A' => l'automate passe dans l'état 4
lecture de 'C' => l'automate passe dans l'état 4
lecture de 'G' => l'automate passe dans l'état 4
lecture de 'T' => l'automate passe dans l'état 4
lecture de 'G' => l'automate passe dans l'état 4

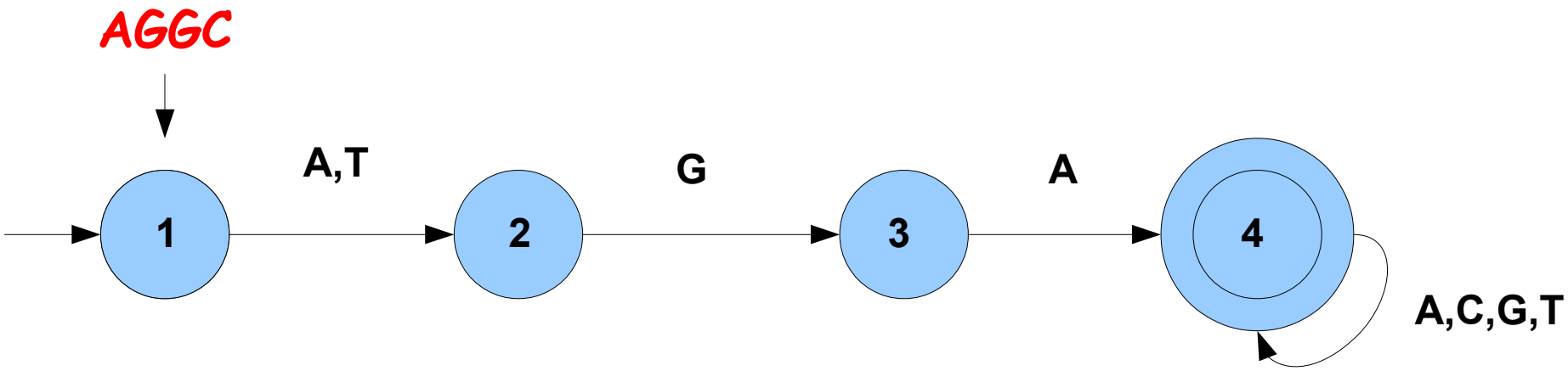
Le chemin se termine dans l'état 4 qui est **terminal**

Oui, je connais ce mot

Automates et langages reconnaissables fonctionnement (3)



Automates et langages reconnaissables fonctionnement (4)

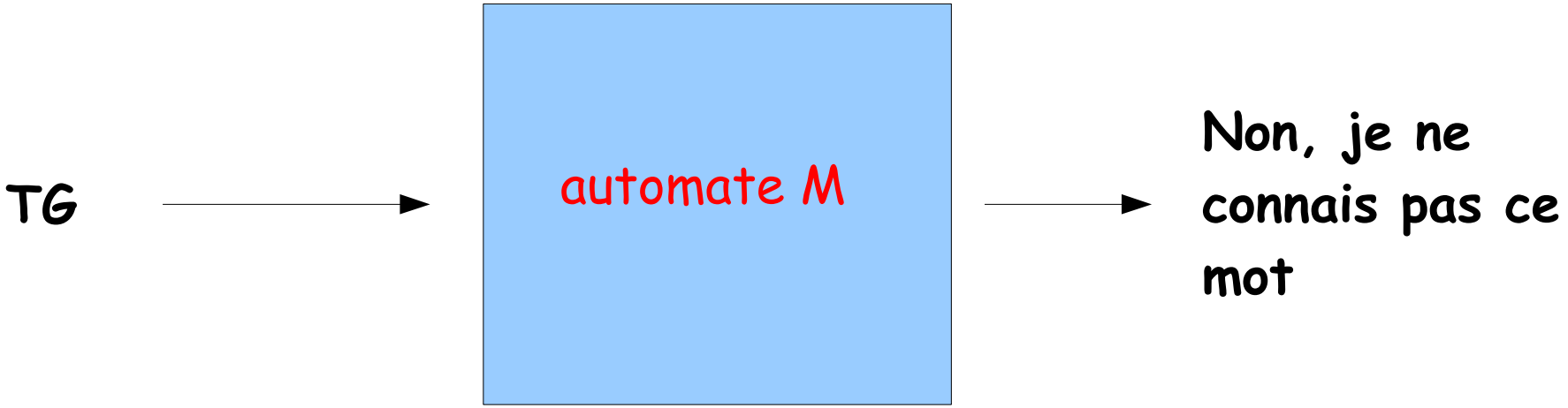


l'automate est dans l'état initial 1
lecture de 'A' => l'automate passe dans l'état 2
lecture de 'G' => l'automate passe dans l'état 3
lecture de 'G' => ??

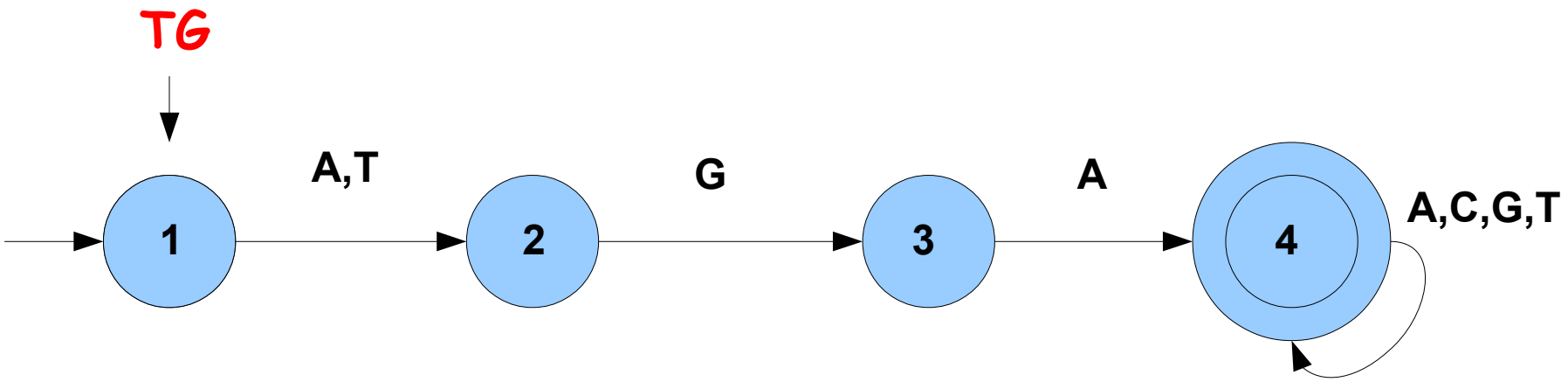
Il n'y a pas de flèche sortant de l'état 3 étiquetée par 'G' => l'automate **se bloque sans avoir lu le mot jusqu'au bout**.

Non, je ne connais pas ce mot

Automates et langages reconnaissables fonctionnement (5)

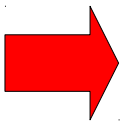


Automates et langages reconnaissables fonctionnement (6)



l'automate est dans l'état initial 1
lecture de 'T' => l'automate passe dans l'état 2
lecture de 'G' => l'automate passe dans l'état 3

L'automate a lu le mot jusqu'au bout mais se trouve dans un état **non terminal**.



Non, je ne connais pas ce mot

Automates et langages reconnaissables déterminisme (1)

Ce n'est pas toujours aussi simple !

L'automate que nous avons étudié a une propriété particulière : il est **déterministe**.

-Il a un unique état initial
-Quels que soient l'état q et la lettre a considérés, il existe au plus une flèche étiquetée par a qui sort de l'état q.

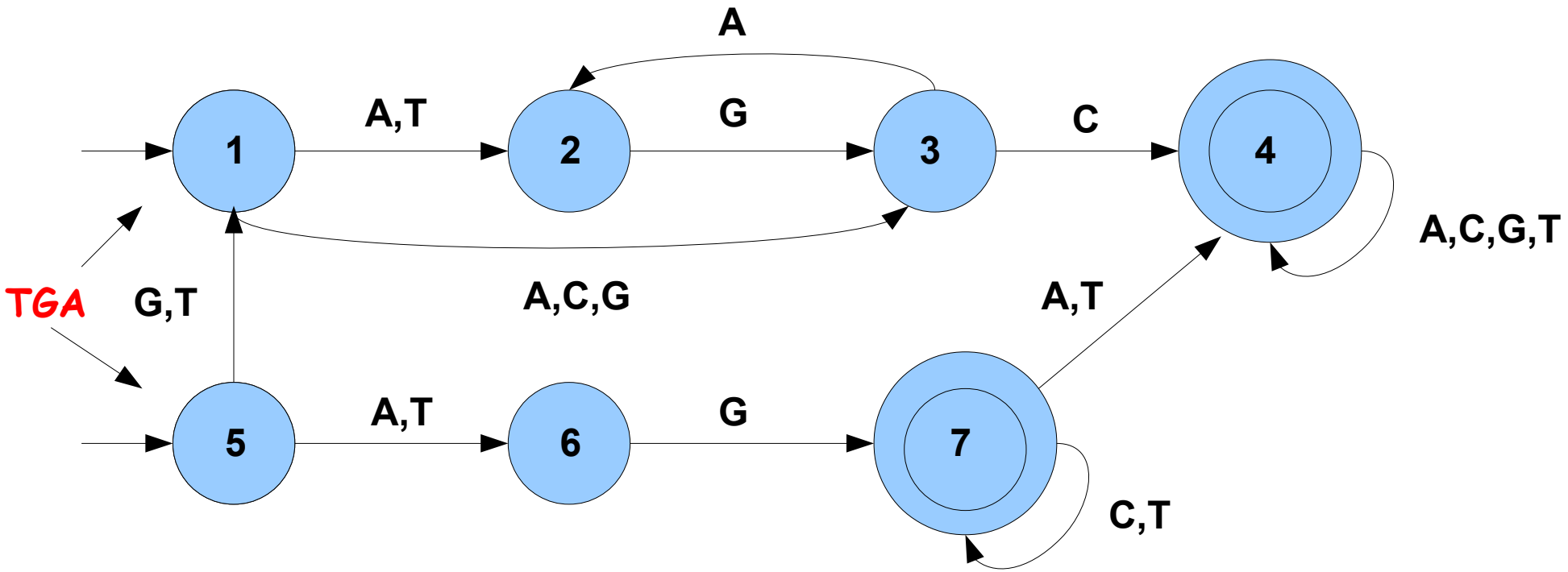
Quand un automate possède cette propriété, quel que soit le mot w qui est lu, si il existe un chemin (réussi ou non) étiqueté par w issu de l'unique état initial alors ce chemin est unique.

Automates et langages reconnaissables déterminisme (2)

Si un automate n'est pas déterministe, il peut exister plusieurs chemins (réussis ou non) étiquetés par w issus d'un état initial.
Pour savoir si un mot w est reconnu, on construit tous les chemins d'étiquette w issus d'un état initial

-soit jusqu'à trouver un chemin réussi et dans ce cas w est reconnu
-soit jusqu'à avoir construit tous les chemins de trace w issus d'un état initial sans en trouver un réussi et dans ce cas, w n'est pas reconnu.

Automates et langages reconnaissables déterminisme (3)

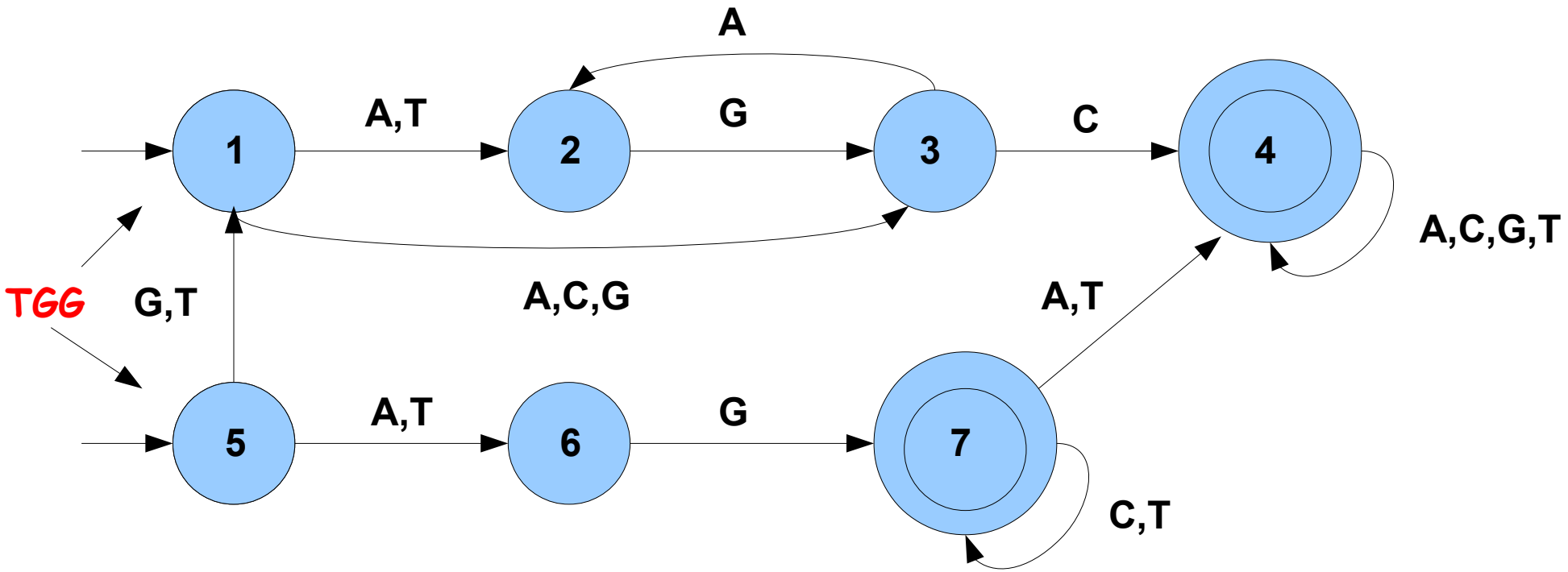


Chemin 1 :
état 1, lettre 'T' => état 2
état 2, lettre 'G' => état 3
état 3, lettre 'A' => état 2
=> chemin non réussi

Chemin 2 :
état 5, lettre 'T' => état 1
état 1, lettre 'G' => état 3
état 3, lettre 'A' => état 2
=> chemin non réussi

Chemin 3 :
état 5, lettre 'T' => état 6
état 6, lettre 'G' => état 7
état 7, lettre 'A' => état 4
=> chemin réussi
=> 'TGA' est reconnu

Automates et langages reconnaissables déterminisme (3)

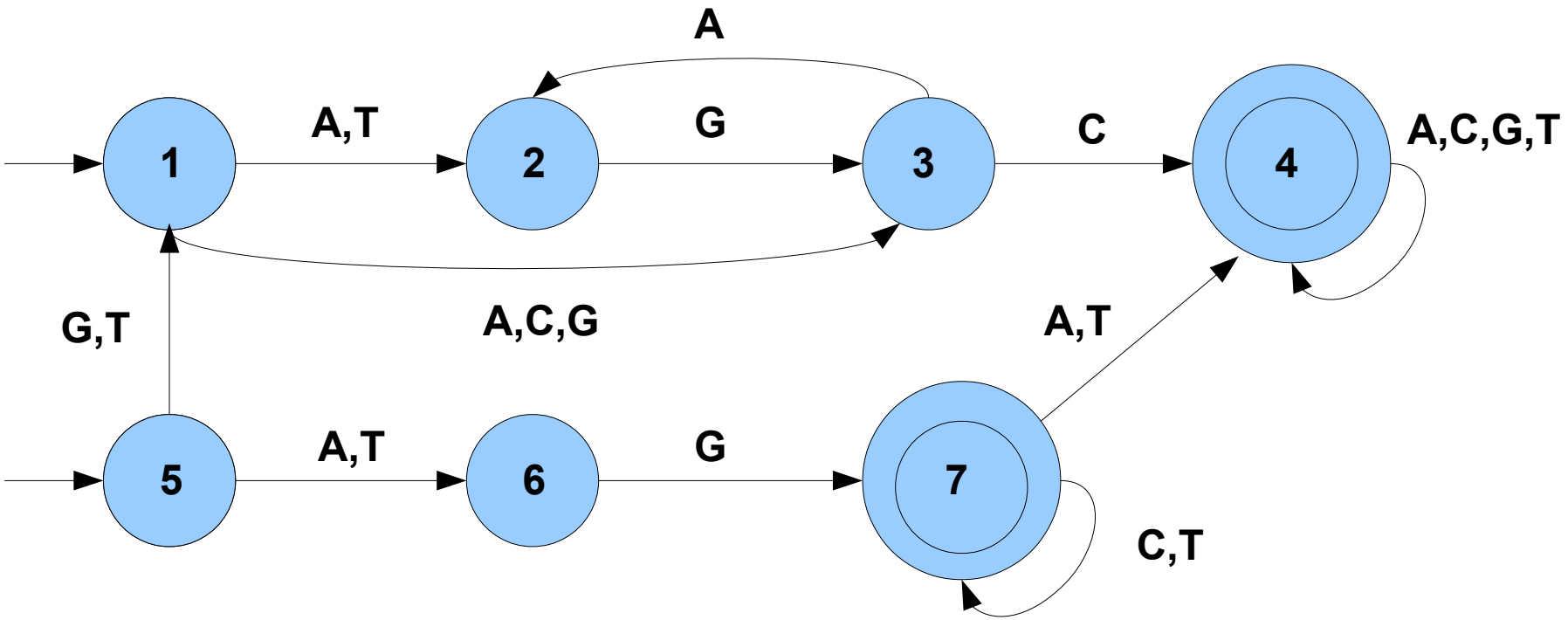


Chemin 1 :
état 1, lettre 'T' => état 2
état 2, lettre 'G' => état 3
état 3, lettre 'G' => ?
=> chemin non réussi

Chemin 2 :
état 5, lettre 'T' => état 1
état 1, lettre 'G' => état 3
état 3, lettre 'G' => ?
=> chemin non réussi

Chemin 3 :
état 5, lettre 'T' => état 6
état 6, lettre 'G' => état 7
état 7, lettre 'G' => ?
=> chemin non réussi
=> 'TGG' n'est pas reconnu

Automates et langages reconnaissables
langage reconnu



$L(M) = \{\varepsilon, G, T\} . (\{A, T\} . \{GA\}^* . \{G\} \cup \{A, C, G\}.\{AG\}^*) . \{C\} . \{A, C, G, T\}^*$
$\cup \{A, T\} . \{G\} . \{C, T\}^* . (\{\varepsilon\} \cup \{A, T\} . \{A, C, G, T\}^*)$

Langages rationnels et langages reconnaissables

Théorème de Kleene :

reconnaissable $\Leftrightarrow$ rationnel

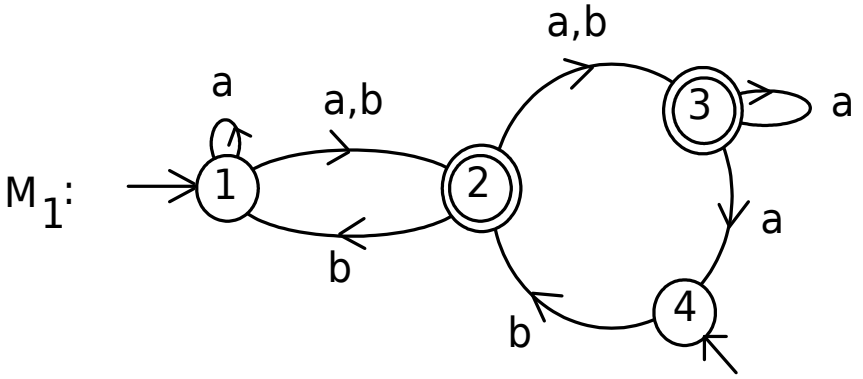
Automates et expressions régulières

Un automate permet de tester si un mot donné est dans le langage reconnu par cet automate. Une expression régulière peut être utilisée de façon analogue. On veut tester si un texte suit les caractéristiques définies par une expression rationnelle. On soumet le texte à un **analyseur d'expressions régulières** (par exemple Lex ou grep), qui utilisera l'expression régulière qui décrit le langage comme une table de transitions. Le texte sera dit reconnu si la lecture de ses caractères successifs permet de parcourir l'expression régulière jusqu'à la fin.

Langages reconnaissables, langages rationnels exercices (1)

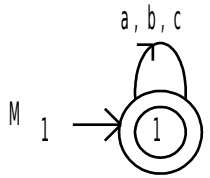
1- Construisez tous les chemins d'étiquette 'baaba' issus d'un état initial dans l'automate M_1 ci-dessous.

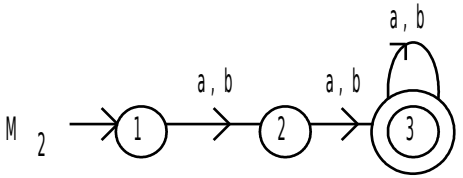
'baaba' est-il reconnu par M_1 ?

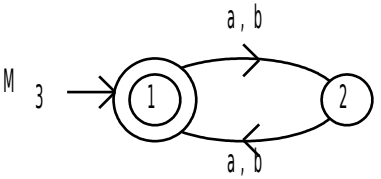


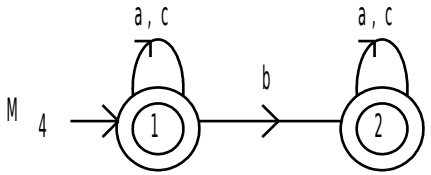
Langages reconnaissables, langages rationnels exercices (2)

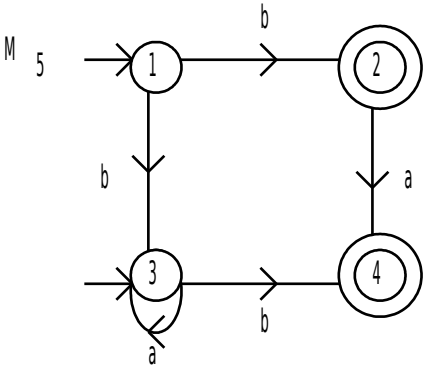
2- Quel est le langage reconnu par chacun de ces automates :

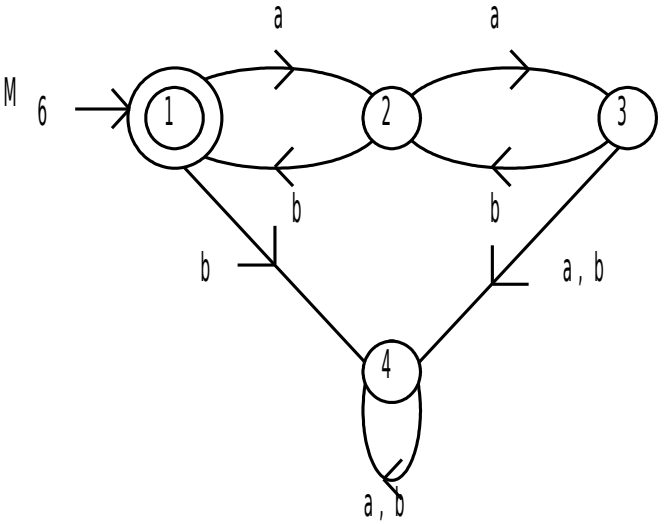












Langages reconnaissables, langages rationnels exercices (3)

3- Construisez des automates reconnaissant les langages suivants sur $\Sigma = \{A,C,G,T\}$. Donnez pour chaque automate un mot reconnu et un mot qui ne l'est pas.

a- { *CAG* }
b- { mots commençant par *CAG* }
c- { mots se terminant par *CAG* }
d- { mots contenant *CAG* }
e- $\{A\}^* . \{C\} \cup \{G\}^* . \{T\}$
f- { mots dont la $3^{ième}$ lettre à partir de la droite est un *A* }
g- $\{AA, ACA, CGA\}^*$
h- { mots contenant un nombre pair de *A* }
i- { mots de longueur impaire qui commencent et se terminent par *A* }

Langages reconnaissables, langages rationnels récréation

5-Un homme *H* voyage avec un loup *L*, une chèvre *C* et une salade *S*. Il arrive au bord d'une rivière que l'on peut traverser en bateau, mais le bateau ne peut prendre que deux passagers à bord. Construisez un automate qui donne toutes les solutions au problème suivant:
Comment faire pour que chacun traverse sans que le loup ne mange la chèvre, ni que la chèvre ne mange la salade ? Vous donnerez d'abord les états initiaux de l'automate, les états terminaux , puis un automate pour résoudre ce problème.

Idée:
On suppose que *H*, *L*, *C* et *S*, sont sur la rive gauche, et qu'ils doivent atteindre la rive droite. Un état du système sera représenté par une chaîne de caractères donnant la liste des protagonistes installés sur la rive gauche de la rivière, suivie d'un tiret, suivi de la liste des protagonistes installés sur la rive droite.
Une transition de l'automate correspond à un changement d'état, soit concrètement à une traversée de la rivière. L'homme effectue chaque traversée puisque ni le loup ni la chèvre ne peuvent ramer. Les flèches seront étiquetées par le symbole, soit du personnage qui accompagne l'homme lors de la traversée considérée, soit *H* lorsque l'homme traverse seul à bord.

Automates et bioinformatique problème (1)

Le code génétique standard est donné par le tableau suivant :

Ala (A) GCU, GCC, GCA, GCG	Lys (K) AAA, AAG
Arg (R) CGU, CGC, CGA, CGG, AGA, AGG	Met (M) AUG
Asn (N) AAU, AAC	Phe (F) UUU, UUC
Asp (D) GAU, GAC	Pro (P) CCU, CCC, CCA, CCG
Cys (C) UGU, UGC	Sec (U) UGA
Gln (Q) CAA, CAG	Ser (S) UCU, UCC, UCA, UCG, AGU, AGC
Glu (E) GAA, GAG	Thr (T) ACU, ACC, ACA, ACG
Gly (G) GGU, GGC, GGA, GGG	Trp (W) UGG
His (H) CAU, CAC	Tyr (Y) UAU, UAC
Ile (I) AUU, AUC, AUA	Val (V) GUU, GUC, GUA, GUG
Leu (L) UUA, UUG, CUU, CUC, CUA, CUG	
START AUG	STOP UAG, UGA, UAA

Automates et bioinformatique problème (2)

But : Construire un traducteur

séquence d'ARN → séquence protéique

0- Construisez un automate M_1 qui reconnaît le langage {A, C, G, U}* . {AUG}.

1- Construisez un automate M_2 qui reconnaît les différents codons qui interviennent dans le code génétique (on se limitera aux codons en bleu).

2- Attachez à chaque état terminal de M_2 l'acide aminé correspondant ou START ou STOP selon les cas.

3- Modifiez M_2 pour qu'il reconnaisse $L(M_2)^*$.

Automates et bioinformatique problème (3)

4- u, v, w sont des mots de {A, C, G, U}*. Donnez une expression rationnelle qui désigne le langage L des mots u suivants :
u est de la forme AUG . v . w où |v| est un multiple de 3 et w est un codon stop.

5- Construisez un automate qui reconnaît L.

6- Modifiez l'automate pour qu'il effectue la traduction
ARN → séquence protéique.