2 - Traitement des mots à l'aide de méthodes d'états finis

1. Expressions régulières
2. Automates d'états finis
3. Automates d'états finis et expressions régulières
4. Transducteurs d'états finis

2.1 - Expressions régulières : motivation

- Dans le traitement de textes écrits, il est souvent nécessaire de rechercher dans ces textes des segments ayant une caractéristique particulière : les phrases, les mots, les entités nommées, un mot particulier, les formes fléchies d'un lemme particulier …

- Un moyen d'effectuer une telle tâche est d'utiliser des **expressions régulières**. Une expression régulière est une formule permettant de caractériser un ensemble de chaînes de caractères.

2.1 - Expressions régulières : définition

- Formellement, une expression régulière (ou rationnelle) sur un alphabet de symboles Σ est une expression construite à partir de symboles de Σ et du mot vide ε, à l'aide de 3 opérations : la **concaténation** (les expressions concaténées sont simplement juxtaposées), la **disjonction** « | » et la **clôture de Kleene** « * ».

- Une expression régulière définit un ensemble de chaînes de symboles de Σ qui forment un **langage régulier**.

- A partir des trois opérations primitives, on définit habituellement 2 opérations complexes qu'on exprime à l'aide des opérateurs ? et + :

Si E est une expression régulière, $E? = E \mid \varepsilon$ *et* $E^+ = E\,E^*$

2.1 - Expressions régulières : définition

- Dans certains langages formels comme Python, on utilise d'autres opérations complexes :

.	Remplace n'importe quel symbole (sauf éventuellement le caractère spécial \n de passage à la ligne)
[ ]	Remplace l'un quelconque des symboles placés entre les crochets
[^]	Remplace l'un quelconque des symboles qui ne sont pas entre les crochets
{m,n}	m et n sont des entiers qui indiquent respectivement le nombre minimum et maximum de fois que la sous-expression située juste avant est répétée
\w	Remplace n'importe quel caractère alphanumérique (lettre ou chiffre) plus le caractère _
\W	Remplace n'importe quel symbole qui n'est ni un caractère alphanumérique, ni le caractère _
\d	Remplace n'importe quel chiffre
\D	Remplace n'importe quel symbole qui n'est pas un chiffre
\s	Remplace l'un quelconque des caractères d'espacement ' ', \t, \n, \f, \r \v
\S	Remplace n'importe quel symbole qui n'est pas un caractère d'espacement

2.1 - Expressions régulières : définition

- Dans des expressions régulières complexes, les opérations unaires qui sont toutes postfixées, sont effectuées en premier. Ensuite, ce sont les concaténations et enfin les disjonctions. Les parenthèses permettent de forcer les priorités.

2.1 - Expressions régulières : utilisation

- L'application la plus répandue des expressions régulières en TAL consiste à rechercher dans un texte des chaînes de caractères ayant une caractéristique donnée.

- La caractéristique donnée est exprimée par une expression régulière.

- La plupart des algorithmes utilisés parcourt le texte du début à la fin en essayant de trouver la plus longue chaîne de caractères correspondant à l'expression régulière. Dès qu'une chaîne est trouvée, le processus est réitéré à partir du caractère suivant la fin de la chaîne trouvée.

2.1 – Expressions régulières : utilisation

Algorithme de recherche d'expressions régulières dans une chaîne

Fonction RECHERCHE_EXPR (*expr, chaine*)
index ← 0
resultat ← [ ]
tantque *index* < longueur*(chaine)* **faire**
  si *chaine[index]* est le début d'un mot du langage L(*expr*) représenté par *expr* **alors**
    fin_pref = *index*
    fin_mot = -1
    tantque *chaine[index : fin_pref* + 1*]* est le début d'un mot du langage L(*expr*)
      et *fin_pref* < longueur*(chaine)* **faire**
        si *chaine[index : fin_pref* + 1*]* ∈ L(*expr*) **alors**
          fin_mot = *fin_pref*
        fin_pref = *fin_pref* + 1
    si *fin_mot* > -1 **alors**
      resultat ← *resultat* + [(*index, fin_mot*)]
      index ← *fin_mot* + 1
    sinon
      index ← *index* + 1
  sinon
    index ← *index* + 1
retourner *resultat*

2.1 - Expressions régulières : utilisation

- Pour exprimer certaines contraintes sur la position des chaînes de caractères recherchées, la syntaxe des expressions régulières est étendue par des constantes de contrôle ou **ancres**. Les ancres ne représentant aucune chaîne de caractères (le mot vide) mais elles indiquent des positions particulières dans le texte.

^	Début d'une ligne
$	Fin d'une ligne
\b	Début ou fin d'un mot (un mot est une suite continue de caractères alphanumériques ou de caractères _)
\B	Position qui n'est ni un début, ni une fin de mot
\A	Début du texte
\Z	Fin du texte

2.1 - Expressions régulières : applications au TAL

- Les expressions régulières jouent un rôle essentiel pour segmenter un texte en phrases et en mots. Elles permettent de détecter grossièrement les séparateurs de phrases et de mots.
- Dans les éditeurs de textes, elles servent à rechercher des mots.
- En recherche d'informations, elles permettent de retrouver des documents et des informations à l'intérieur de ces documents.
- Elles sont aussi utilisées pour indexer des documents.
- Elles peuvent être aussi employées dans l'analyse morphologique et l'étiquetage morpho-syntaxique.

2.1 - Expressions régulières : exercices

1. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Pour chacune des expressions régulières ci-dessous, décrivez le langage qu'elle représente, c'est-à-dire l'ensemble des chaînes de caractères qui correspondent à l'expression.

   a) ab^{+}| b?a

   b) a\d*\s^{+}

   c) \s |[ea] \s

   d) [0-7]$^{+}$(,[0-7]*[1-7])?

2. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Trouver une expression régulière caractérisant chacun des langages suivants :

   a) Les sommes d'argent en euros avec au maximum deux chiffres après la virgule,

   b) Les dates écrites dans un format illustré par l'exemple 04/05/2004 (on suppose que le mois de février peut avoir 29 jours sans vérification des années bissextiles)

   c) Les chaînes de caractères comportant au moins deux fois le mot « cher » (mais ni en début, ni en fin de chaîne).

2.1 - Expressions régulières : exercices

3. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Trouver une expression régulière permettant de rechercher dans un texte en français les chaînes de caractères ayant chacune des caractéristiques suivantes :

a) Le mot « cheval » ou « chevaux »,

b) Les lignes se terminant par un point d'interrogation (le point d'interrogation est représenté par « \? » dans une expression régulière),

c) Les phrases interrogatives (on suppose qu'une phrase est caractérisée comme se terminant par un point, un point-virgule, un point d'exclamation ou un point d'interrogation)

4. On applique l'algorithme de recherche des chaînes de caractères dans un texte correspondant à une expression régulière donnée. Dans chacun des cas, donnez le résultat de la recherche

	Expression régulière	texte
a)	'a\w*a'	'acddcceeacvvv bbaa'
b)	'[^a-z]*ba'	'Aaa Abbca bac'
c)	'^ .\b'	'Je viens.\n Et toi?'
d)	'\b[A-Z].*'	'Tu connais Paul et Marie ?'
e)	'[\s .] [A-Z]'	'Tu connais l'O.N.U. C'est difficile.'

2.2 - Automates d'états finis : motivation

- Etant donné une expression régulière, le problème de détecter dans un texte les chaînes de caractères qui appartiennent au langage défini par l'expression régulière nécessite, pour être résolu automatiquement, un mécanisme de reconnaissance de ce langage.

- Un **automate d'états finis** est une machine abstraite destinée à reconnaître un **langage régulier**, c'est-à-dire un langage défini par une **expression régulière**.

- Il peut être normalisé pour avoir une grande efficacité en temps et en espace de calcul.

- Les automates peuvent être combinés de manières très variées pour construire des automates plus complexes.

2.2 - Automates d'états finis : définition

Un **automate d'états finis** est composé de deux parties :

- Un **ruban** infini avec des positions numérotées 0, 1, … pouvant contenir chacune un symbole d'un alphabet donné. Le ruban est muni d'un pointeur sur la position courante qui peut être seulement lue.

- Une **unité de contrôle** qui pilote l'avancée pas à pas du pointeur sur le ruban en fonction de son état et du symbole lu sur le ruban.

2.2 - Automates d'états finis : définition

Définition : un automate d'états finis sur un alphabet Σ, est un 5-uplet $(Q, \Sigma, q_0, F, \tau)$ tel que :

- Q est un ensemble fini d'états.
- Σ est un alphabet fini de symboles. En lui ajoutant le mot vide ε, on obtient l'ensemble des étiquettes d'entrée de l'automate.
- q_0 est un élément particulier de Q, l'état initial de l'automate.
- F est une partie de Q rassemblant les états acceptants de l'automate.
- τ est une relation de transition qui associe un état de départ q_1 et une étiquette d'entrée a avec un état d'arrivée q_2. Ceci est noté : $\tau(q_1, a, q_2)$.

2.2 - Automates d'états finis : définition

- **Définition :** un **calcul** sur un automate est une suite (éventuellement vide) de transitions de la forme : $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} q_n$ où q_0 est l'état initial.
  On note alors : $q_0 \xrightarrow{a_1 a_2 \cdots a_n} * q_n$

- Le mot $a_1 a_2 \ldots a_n$ est **reconnu** par l'automate si q_n est un état acceptant.

- Le langage reconnu par l'automate est l'ensemble de tous les mots reconnus par l'automate.

2.2 - Automates d'états finis : exercices

1. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Dans chacun des cas suivants, un automate est donné par sa table de transition. Les symboles de transition sont indiqués sur la première ligne et les états de départ des transitions dans la première colonne. L'état initial est toujours l'état 0 et les états acceptants sont marqués par un astérisque.
Décrire le langage reconnu par chaque automate.

a)

	a	b	ε
0	1	2	
1	1	3	
2	2, 3		
3*			0

b)

	0	1	,
0	1	2	
1			3
2	2	2	3
3*	3	3	

c)

	a	[^a]	[^ab]
0*	1	0	
1*	1		0

2.2 - Automates d'états finis : exercices

2. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Dans chacun des cas suivants, définir un automate qui reconnaisse le langage présenté

a) Les formes fléchies du verbe marcher à l'indicatif présent,

b) Les nombres décimaux en virgule flottante, c'est-à-dire dans le format : *signe mantisse E signe exposant*
La mantisse est soit zéro, soit un nombre décimal avec un chiffre non nul avant la virgule et l'exposant est un nombre entier.

c) Les chaînes de caractères ne comportant pas la sous-chaîne « du » (commencer par définir un automate qui reconnaisse les chaînes contenant au moins une occurrence de « du »).

2.2 - Automates d'états finis : opérations

Union - Soit A_1 et A_2 deux automates d'états finis. Soit l'automate $A_1 \cup A_2$ obtenu de la façon suivante :

- on considère comme états les états de A_1 et de A_2 plus un nouvel état q_0;
- on considère comme transitions les transitions de A_1 et de A_2 plus deux transitions silencieuses de q_0 aux états initiaux A_1 et de A_2
- l'état initial est q_0;
- Les états acceptants sont les états acceptants de A_1 et de A_2 .

Le langage reconnu par $A_1 \cup A_2$ est l'union des langages reconnus par A_1 et A_2 .

2.2 - Automates d'états finis : opérations

Concaténation - Soit A_1 et A_2 deux automates d'états finis. Soit l'automate $A_1 A_2$ obtenu de la façon suivante :

- on considère comme états les états de A_1 et de A_2 ;
- on considère comme transitions les transitions de A_1 et de A_2 plus une transition silencieuse de chaque état acceptant de A_1 jusqu'à l'état initial de A_2 ;
- l'état initial est l'état initial de A_1;
- les états acceptants sont les états acceptants de A_2 .

Le langage reconnu par $A_1 A_2$ est le langage reconnu par A_1 concaténé avec le langage reconnu par A_2 .

2.2 - Automates d'états finis : opérations

Clôture de Kleene - Soit A un automate d'états finis. Soit l'automate A^* obtenu de la façon suivante :

- on considère comme états les états de A plus un nouvel état q_0;
- on considère comme transitions les transitions de A plus une transition silencieuse de chaque état acceptant de A jusqu'à l'état q_0 et de q_0 jusqu'à l'état initial de A;
- l'état initial est l'état q_0;
- l'état q_0 est aussi le seul état acceptant de A^*.

Le langage reconnu par A^* est le langage reconnu par la clôture de Kleene du langage reconnu par A .

2.2 - Automates d'états finis : applications au TAL

- Etant une forme d'implémentation des expressions régulières, les automates d'états finis peuvent se retrouver dans toutes les applications qui font intervenir des expressions régulières.

- Les automates d'états finis permettent de représenter les lexiques et dictionnaires électroniques de manière compacte avec un accès efficace.

- Ils ont une manière simple et efficace d'implémenter différentes formes de traitement des langues (plutôt dans les bas niveaux) : reconnaissance et synthèse de la parole, segmentation d'un texte, analyse morphologique, analyse syntaxique …

2.3 - Automates d'états finis et expressions régulières : équivalence

- A partir de l'union, de la concaténation et de la clôture de Kleene d'automates, on peut construire un automate d'états finis pour n'importe quelle expression régulière, qui reconnaisse le langage associé.

- Réciproquement, pour n'importe quel automate d'états finis, on peut trouver une expression régulière dont le langage associé est celui reconnu par l'automate.

- **Théorème** : un langage est régulier si et seulement si il est reconnu par un automate d'états finis

2.3 - Automates d'états finis et expressions régulières : exercices

1. On considère l'alphabet usuel des caractères utilisés pour écrire un texte en français. Dans chacun des cas suivants, construire un automate qui reconnaisse le langage défini par l'expression régulière

   a) a b* | a* b

   b) [a b]?([^a] | b)*

   c) [a b c]{1,3} e+

2. On considère l'expression régulière ([^a]* (a [^b])?)*.

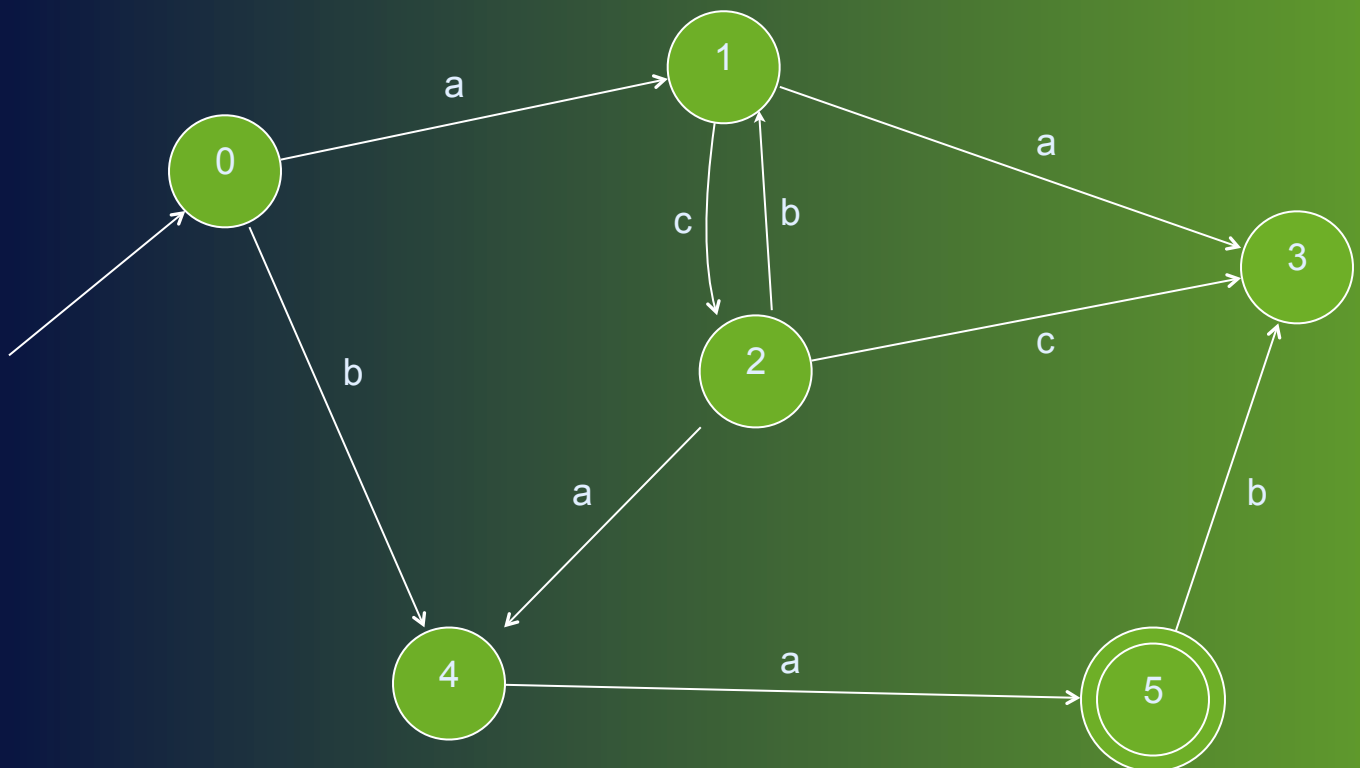
   a) Construire un automate d'états finis qui reconnaît le langage défini par cette expression régulière.

   b) Montrez que ce langage n'est pas celui des chaînes ne contenant pas « ab ».

   c) Modifier l'automate et l'expression régulière pour que le langage le devienne.

2.3 - Automates d'états finis et expressions régulières : exercices

3. Déterminer une expression régulière représentant le langage reconnu par l'automate ci-dessous.



2.4 - Transducteurs d'états finis : motivation

- Les automates d'états finis sont des machines destinées à **reconnaître** certains langages.
- Parfois, il est nécessaire de traduire un langage dans un autre. C'est le rôle des **transducteurs d'états finis**.
- Un transducteur d'états finis définit une **relation** entre un langage d'entrée et un langage de sortie.

2.4 - Transducteurs d'états finis : définition

Définition : un transducteur d'états finis sur un alphabet d'entrée Σ_i et un alphabet de sortie Σ_o est un 6-uplet $(Q, \Sigma_i, \Sigma_o, q_0, F, \tau)$ tel que:

- Q est un ensemble d'états finis.
- Σ_i est un alphabet fini de symboles d'entrée.
- Σ_o est un alphabet fini de symboles de sortie.
- q_0 est un élément particulier de Q représentant l'état initial du transducteur.
- F est une partie de Q représentant les états acceptant du transducteur.
- τ est une relation de transition qui associe un état de départ q_1 de Q et un mot d'entrée w_i de Σ_i^* avec un état d'arrivée q_2 de Q et un mot de sortie w_o *de* Σ_o^* . On écrit alors : $\tau(q_1, w_i, q_2, w_o)$.

2.4 - Transducteurs d'états finis : définition

- **Définition :** un **calcul** sur un transducteur est une suite éventuellement vide de transitions de la forme : $q_0 \xrightarrow{a_1 : b_1} q_1 \xrightarrow{a_2 : b_2} \dots \xrightarrow{a_n : b_n} q_n$ où q_0 est l'état initial. On écrit alors : $q_0 \xrightarrow{a_1 a_2 \cdots a_n : b_1 b_2 \cdots b_n} * q_n$

- Le mot $a_1 a_2 \dots a_n$ est dit **traduit** par le transducteur dans le mot $b_1 b_2 \dots b_n$ si q_n est un état acceptant.

2.4 - Transducteurs d'états finis : définition

- La relation réalisée par le transducteur T est l'ensemble des couples (w_1, w_2) tels que w_1 est transformé en w_2 par le transducteur. Cette relation est appelée une **relation régulière (rationnelle)**.

- On associe à tout transducteur T la fonction $|T|$ qui associe chaque mot d'entrée w_i à l'ensemble $|T|(w_i)$ de tous les mots résultant de la transduction de w_i par T.

- Si tout mot d'entrée est transformé en un mot de sortie au plus, le transducteur réalise une **fonction rationnelle**.

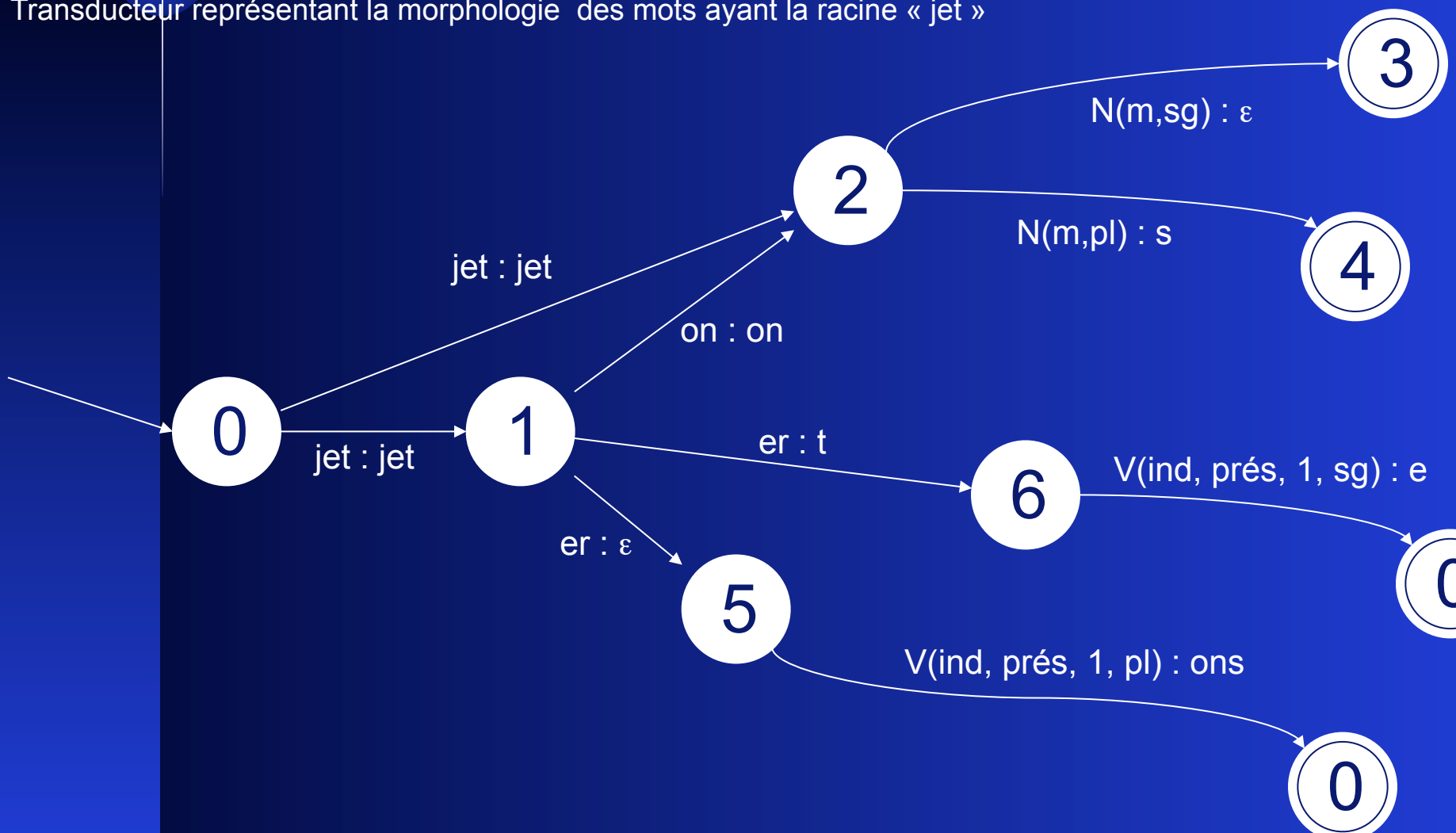
2.4 - Transducteurs d'états finis : application au TAL

- Pour les langues qui n'ont pas une morphologie trop riche (anglais, français…), on peut stocker tous les mots fléchis de la langue dans un lexique morphologique, sous forme d'un transducteur.

- Pour les langues riches morphologiquement (turc, arabe …), ce sont les règles morphologiques qui sont représentées par des transducteurs.

2.4 - Transducteurs d'états finis : application au TAL

Transducteur représentant la morphologie des mots ayant la racine « jet »



2.4 - Transducteurs d'états finis : application au TAL

- Les transducteurs peuvent être utilisés pour la segmentation de textes (complétée éventuellement par des règles phonologiques).

- Ils sont particulièrement efficaces dans l'étiquetage morpho-syntaxique fondé sur des règles de transformation apprises d'un corpus (étiquetage à la Brill).

2.4 - Transducteurs d'états finis : exercices

1. L'algorithme du Soundex est utilisé dans les bibliothèques pour représenter les noms de personnes. Il a l'avantage d'être peu sensible aux variations d'écriture des noms; par exemple, Jurafsky, Jarofsky, Jarovsky et Jarovski seront tous représentés par J612.

   En utilisant des transducteurs, réaliser les différentes tâches suivantes qui composent l'algorithme dans l'ordre où elles sont décrites (on suppose que le texte d'entrée a été découpée en mots séparés par le caractère d'espacement):

a) Conserver la première lettre du nom et supprimer toutes les occurrences non initiales des lettres a, e, h, i, o, u, w, y.

b) Remplacer les lettres autres que la première par des entiers selon les règles :

   b, f,p,v → 1 c,g,j,k,q,s,x,z → 2 d,t → 3 l → 4 m,n → 5 r → 6

c) Remplacer les suites de chiffres identiques par un chiffre unique (666 → 6)

d) Enlever les chiffres au-delà des 3 premiers ou, s'il en manque pour aller jusqu'à 3, compléter par des zéros.

2.4 - Transducteurs d'états finis : exercices

2. Concevoir un transducteur qui segmente un texte en phrases en ajoutant le caractère # pour séparer deux phrases. Le point virgule, le point d'exclamation et le point d'interrogation sont considérés comme des séparateurs absolus tandis que le point peut avoir aussi d'autres fonctions. Pour simplifier, on considèrera comme autres fonctions possibles :

a) exprimer un sigle (O.N.U.),

b) exprimer un numéro de téléphone (03.83.54.77.32),

c) Exprimer une suspension au sein d'une phrase (Ne seriez-vous pas … son ami ?).

On supposera qu'après un point marquant la fin du phrase, il y a un espace suivi d'une majuscule au début d'une nouvelle phrase.

2.4 - Transducteurs d'états finis : exercices

3. Construire un transducteur pour analyser la morphologie des verbes du français du premier groupe (verbe se terminant par « er »).
   En fonction du nombre de racines et des variations orthographiques ou phonologiques, on distingue plusieurs sous-groupes :

a) Les verbes sans exception (*chanter*),

b) Les verbes dont la racine se termine par « c » ou « g » (*placer, manger*),

c) Les verbes ayant une « e » atone sur la dernière syllabe de leur racine (*semer, jeter, appeler*),

d) Les verbes ayant un « é » sur la dernière syllabe de leur racine (*espérer*)

e) Les verbes se terminant par « ayer », « oyer » et « uyer ».

f) Le verbe « *aller* ».