ALGORITHMIQUE DE TEXTES

Jacques Chauché

Professeur honoraire à l'université Montpellier 2

TRAITEMENT DE TEXTES

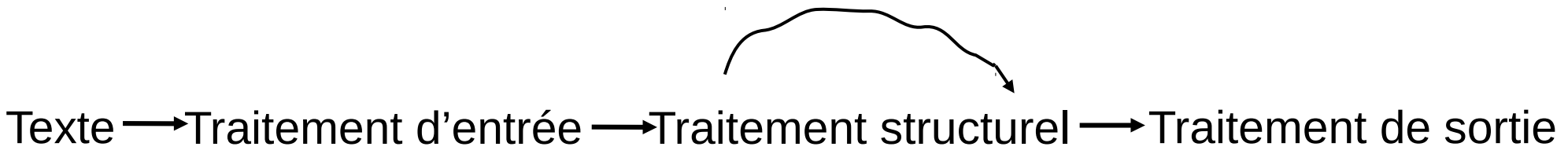
ANALYSE SYNTAXIQUE

TAGGAGES

SÉMANTIQUE

TRADUCTION

Modèle de traitement

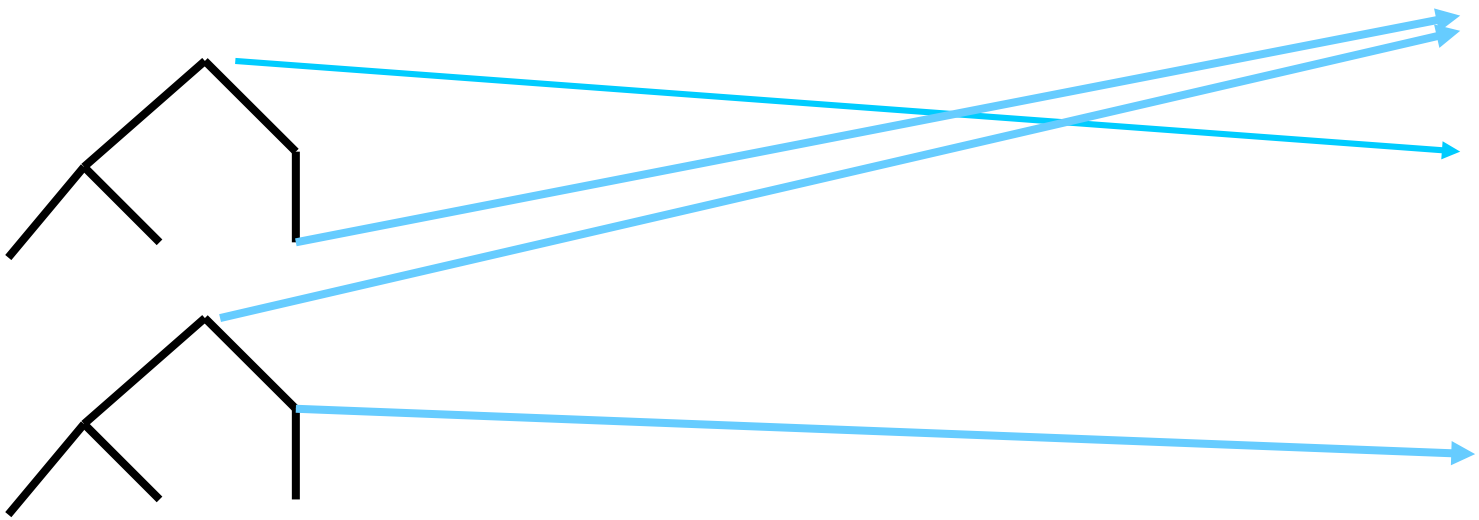


Objets Traités : Éléments structurés

Structures

Fonction d’étiquetage

Ensemble d’étiquettes



Propriété de l’étiquetage :

- Non injectif
- Non surjectif
- Partout défini

Propriété des étiquettes :

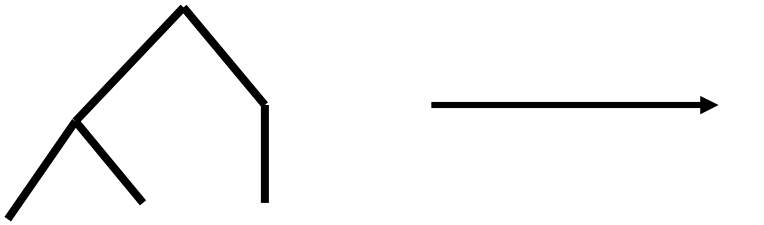
Deux ensembles :

1 Au moins l’image d’un point

2 valeur d’une variable référence issue transitivement d’une étiquette du premier ensemble

Implémentation des éléments structurés

Chaînes de caractères



((( ) ( )) (( )))

Avec étiquetage :

Pt1 Pt2 Pt3) Pt4)) Pt5 Pt6)))

Pti : pointeur vers une étiquette

Conséquence : Tout traitement structurel est réalisé par un traitement de chaîne

Implémentation des éléménts structurés
Structure d’organisation des Variables

Les variables SYGMART ont 9 types possibles :

- Chaîne (qui peut être interprétée comme un tableau à une dimension d’entier ou de flottant simple et double)
- Référence
- Arithmétique
- Arithmétique long
- Flottant
- Double
- Exclusif
- Non exclusif
- Potentielle

L’utilisation d’une variable suppose sa déclaration préalable

Implémentation des éléménts structurés
Structure d'organisation des Variables

Chaque système OPALE, TELESI, ou AGATE utilise un ensemble de variables déclarées dans une « définition »

Chaque définition possède un ensemble de blocs de variables.

Une définition peut contenir un bloc déclaré dans une autre définition

Lors du passage d'un sous système à un autre les blocs communs utilisés par les deux définitions concernées sont conservées.

Un bloc ne peut comprendre qu'une seule variable chaîne

Pour les autres types de variables la place occupée par l'ensemble des variables d'un bloc ne peut dépasser 256 octets

Il ne peut y avoir plus de 65536 blocs (ce qui fait un maximum de 16777216 octets pour l'ensemble des variables …. si la machine sur laquelle est implémanté SYGMART le supporte, chaque point ayant une étiquette qui occupe cette place)

STRUCTURE DU SYSTÈME SYGMART

TROIS COMPOSANTS :

- OPALE : définit un traitement de chaînes. Assure le passage texte → élément structuré
- TELESI : définit un traitement d'éléments structurés
- AGATE : définit un traitement d'une composante d'élément structuré → chaîne

Chacun de ses sous sytèmes utilise un dictionnaire. Obligatoire pour OPALE et AGATE, facultatif pour TELESI

Structures :

- OPALE : automate d 'états finis

- TELESI : Réseau de grammaire avec parcours optionnel de retour arrière.
- Chaque grammaire correspond à une variante d'algorithme de Markov :
  - Les règles sont ordonnées.
  - L'application des règles sont simultanées avec une contrainte de non chevauchement.
  - 3 modes d'applications dont 1 indécidable

- AGATE : automate d'états finis

STRUCTURE DU SYSTÈME SYGMART

Les dictionnaires

Il y a 4 types de dictionnaires :

- Le dictionnaire des formats
- Le dictionnaire des segments
- Le dictionnaire d'étiquettes
- Le dictionnaire dynamique

Le dictionnaire des formats est formé d'étiquettes constantes ou modifiables

Le dictionnaire des segments est spécifique au sous système OPALE

- Il est structuré.
- Son adressage est fonction des variables d'une étiquette

Tous les sous-systèmes ont un dictionnaire d'étiquettes

- Il est structuré
- Sa lecture est univoque : première lecture correcte trouvée

Le dictionnaire dynamique est spécifique au système sous système TELESI

- Il fait partie de l'élément structuré et donc éphémère d'une application à l'autre
- Il permet de définir des valeurs temporaires (comme les noms de variables)

Exemple de traitement de Texte avec SYGFRAN

Les caractéristiques de la grammaire d'analyse du français
SYGFRAN

NOMBRE DE GRAMMAIRES:358
NOMBRE DE REGLES:39397
NOMBRE TOTAL DE POINTS:397877
NOMBRE D'OCTETS CONDITIONS:4733

Soit une moyenne de 10 points par règle.

Exemples d’applications
1 : Analyse syntaxique

chauche@Jacques-PC:/DEMO> ll -h
total 102M
-rwxr-xr-x 1 chauche users 1,8M 24 mai 11:11 applisyg
-rw-r--r-- 1 chauche users 100M 24 mai 12:28 donneesTagPcLinux64

-rw-r--r-- 1 chauche users 86K 24 mai 11:17 Le_petit_prince.txt

-rw-r--r-- 1 chauche users 6,1K 17 mai 08:42 LICENCE.xhtml
chauche@Jacques-PC:/DEMO> time /DEMO/applisyg donneesTagPcLinux64 Le_petit_prince.txt
1
SYSTEME SYGMART,CTR A: SYGTEXT

at: 2 mn 25 s

real 2m24.765s
user 2m24.577s
sys 0m0.192s
chauche@Jacques-PC:/DEMO> ll -h
total 108M
-rwxr-xr-x 1 chauche users 1,8M 24 mai 11:11 applisyg
-rw-r--r-- 1 chauche users 100M 24 mai 12:33 donneesTagPcLinux64
-rw-r--r-- 1 chauche users 4 24 mai 12:35 EXECSYG
-rw-rw-rw- 1 chauche users 5,9M 24 mai 12:35 Le_petit_prince.stx
-rw-r--r-- 1 chauche users 86K 24 mai 11:17 Le_petit_prince.txt
-rw-r--r-- 1 chauche users 6,1K 17 mai 08:42 LICENCE.xhtml
-rw-r--r-- 1 chauche users 0 24 mai 12:33 LOCKSYG

chauche@Jacques-PC:/DEMO> tail Le_petit_prince.stx
LEMME(être),FS(GOV),TYP(ATTRIB,VETAT,AUX),SEMOBJT(TEMPS,LIEU),FLX(être)), **33670** (Balise(été),PLACEMOT(9861),PLACEMOTCHAR(9595)
,POSITION(MOT_TEXTE),FRM(été),GNR(MAS),NUM(SIN),CAT(V),SOUSV(PAPA),POT(AVOIR),AUXCJ(AVOIR),LEMME(être),FS(GOV),TYP(ATTRIB,VETAT,
AUX),SEMOBJT(TEMPS,LIEU),FLX(être)), **33671** (Balise(été),PLACEMOT(17736),PLACEMOTCHAR(17272),POSITION(MOT_TEXTE),FRM(été),
GNR(MAS),NUM(SIN),CAT(V),SOUSV(PAPA),POT(AVOIR),AUXCJ(AVOIR),LEMME(être),FS(GOV),TYP(ATTRIB,VETAT,AUX),SEMOBJT(TEMPS,LIEU),FLX(être)
), **33672** (Balise(été),PLACEMOT(25244),PLACEMOTCHAR(24585),POSITION(MOT_TEXTE),FRM(été),GNR(MAS),NUM(SIN),CAT(V),SOUSV(PAPA)
,POT(AVOIR),AUXCJ(AVOIR),LEMME(être),FS(GOV),TYP(ATTRIB,VETAT,AUX),SEMOBJT(TEMPS,LIEU),FLX(être)), **33673** (Balise(été),PLACEMOT(40162)
,PLACEMOTCHAR(39146),POSITION(MOT_TEXTE),FRM(été),GNR(MAS),NUM(SIN),CAT(V),SOUSV(PAPA),POT(AVOIR),AUXCJ(AVOIR),LEMME(être),FS(
GOV),TYP(ATTRIB,VETAT,AUX),SEMOBJT(TEMPS,LIEU),FLX(être)), **33674** (Balise(trouvé),PLACEMOT(74757),PLACEMOTCHAR(72956),POSITION(
MOT_TEXTE),FRM(trouvé),CATRAC(VB),GNR(MAS),NUM(SIN),CAT(V),SOUSV(PAPA),POT(AVOIR),AUXCJ(ETRE),CASPRNML(NEXCDIR,NEXCFIGE),LEMME(trouver)
FS(GOV),TYP(TRANS,TRANSCONJ,RFLX,TRANSRFL,TRANSRFLCONJ,ATTRIBRFL,CIRCADJ),FLX(trouver))),NOM_ETIQUETTES())
chauche@Jacques-PC:/DEMO>

Exemples d’applications
2 : Taggage

chauche@Jacques-PC:/DEMO> ll -h
total 102M
-rwxr-xr-x 1 chauche users 1,8M 24 mai 11:11 applisyg
-rw-r--r-- 1 chauche users 100M 24 mai 11:20 donneesTagPcLinux64

-rw-r--r-- 1 chauche users 86K 24 mai 11:17 Le_petit_prince.txt

-rw-r--r-- 1 chauche users 6,1K 17 mai 08:42 LICENCE.xhtml

chauche@Jacques-PC:/DEMO> time /DEMO/applisyg donneesTagPcLinux64 Le_petit_prince.txt
SYSTEME SYGMART,CTR A: SYGTEXT

at: 9 mn 11 s

real 9m10.897s
user 9m10.723s
sys 0m0.176s

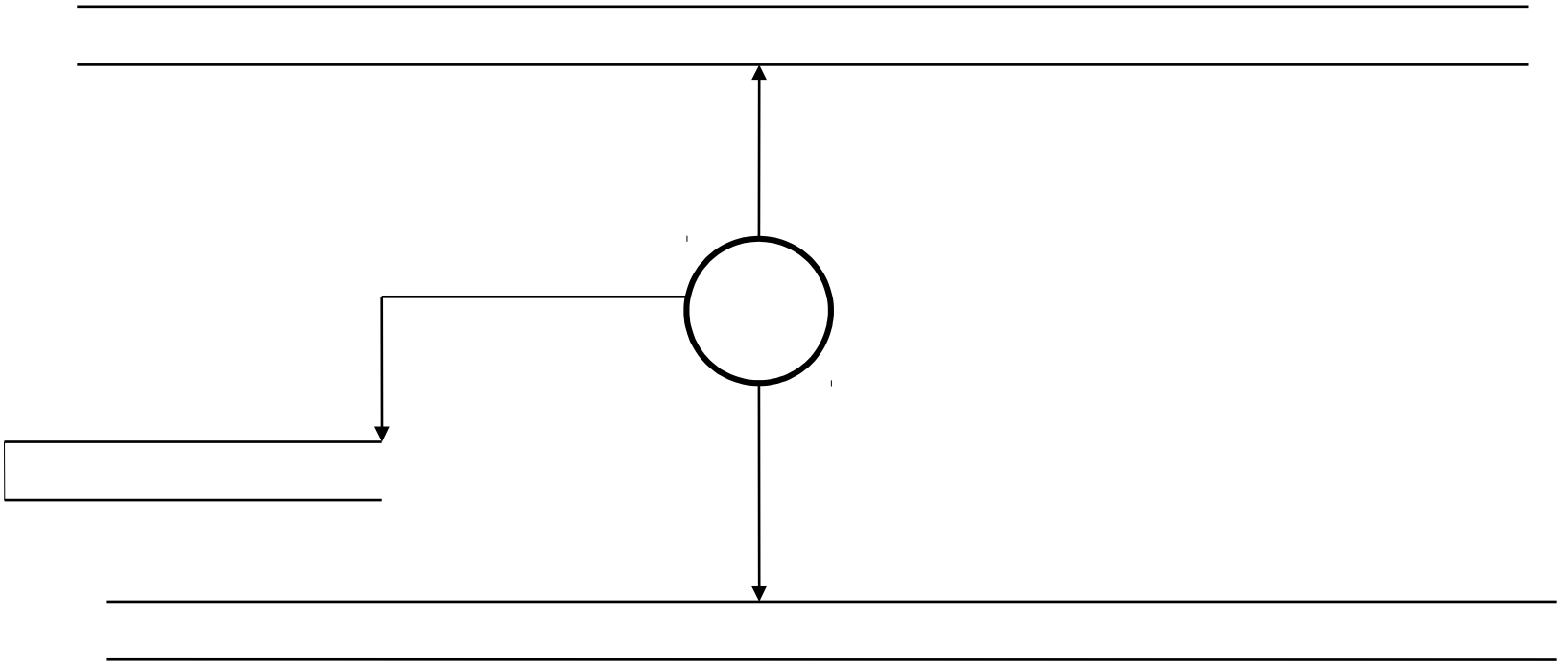
chauche@Jacques-PC:/DEMO> ll -h
total 104M
-rwxr-xr-x 1 chauche users 1,8M 24 mai 11:11 applisyg
-rw-r--r-- 1 chauche users 100M 24 mai 11:31
donneesTagPcLinux64
-rw-r--r-- 1 chauche users 4 24 mai 11:41 EXECSYG
-rw-rw-rw- 1 chauche users 1,7M 24 mai 11:41 Le_petit_prince.tag
-rw-r--r-- 1 chauche users 86K 24 mai 11:17 Le_petit_prince.txt
-rw-r--r-- 1 chauche users 6,1K 17 mai 08:42 LICENCE.xhtml
-rw-r--r-- 1 chauche users 0 24 mai 11:31 LOCKSYG

chauche@Jacques-PC:/DEMO> tail Le_petit_prince.tag

il [Pronom Personnel Masculin Singulier Fonction(Gouverneur) FonctionGroupe(Sujet) ProfondeurSyntaxique(6) 'il']
est [Verbe conjugué VOIX_ACTIVE INDICAT INDICATIF(PRES) Fonction(Gouverneur) ProfondeurSyntaxique(7)
'être']
revenu [Participe Passé Fonction(Gouverneur) ProfondeurSyntaxique(7) 'revenir']
... [Ponctuation ProfondeurSyntaxique(2) '...']

Algorithmique interne du moteur de transformation :

LES TRANSDUCTEURS



Ils peuvent bien sûr se composer

Exemple de composition : Analyse du langage sous contexte $a^n b^n c^n$

Le premier transducteur :

- Reconnaît $a^n b^n c^n$
- Recopie sur sa bande de sortie $b^n c^n$

Le deuxième transducteur :

- Reconnaît $b^n c^n$

Reconnaissance d’un schéma

Automate d’états finis :

Reconnaissance de ((( )) ( ))

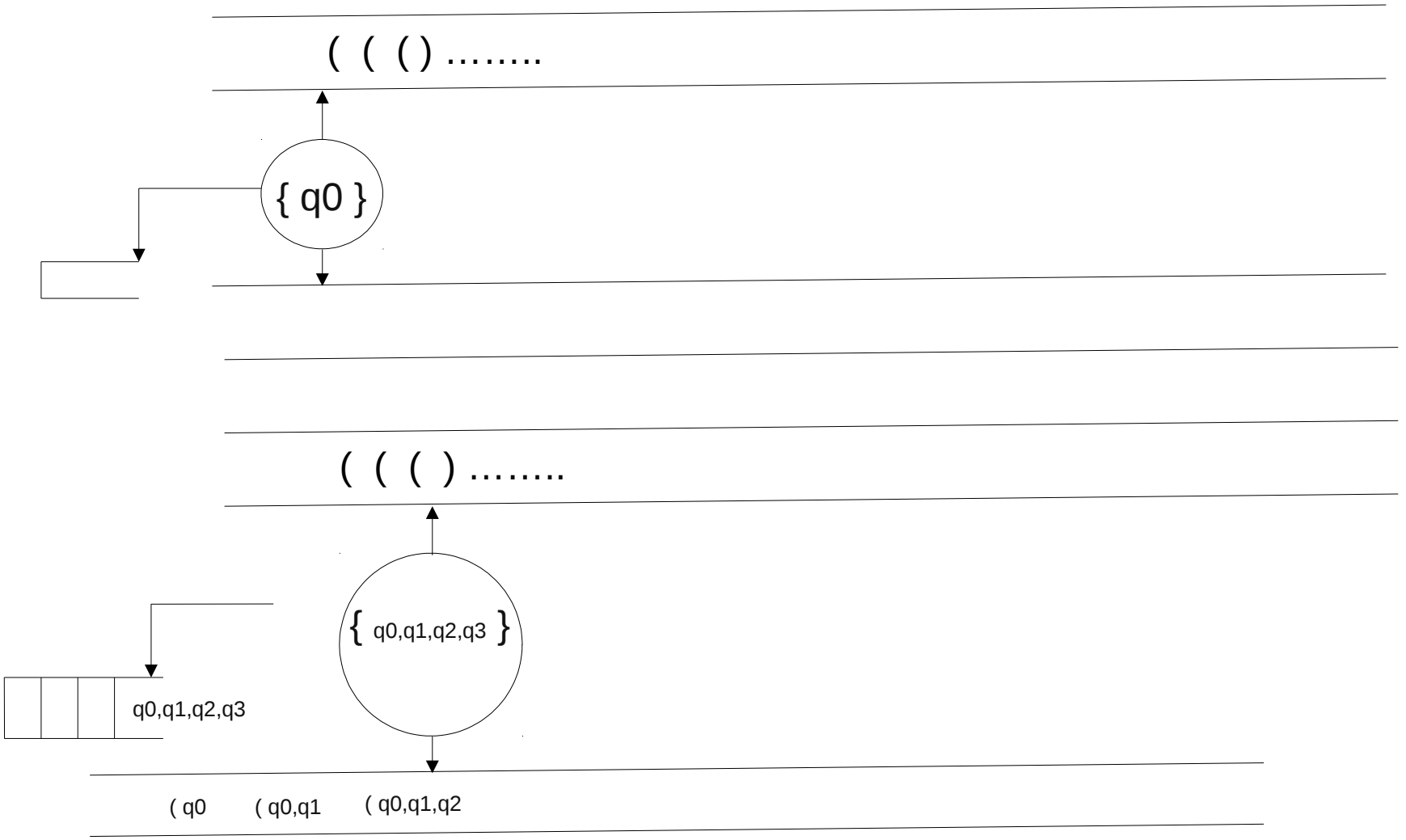
q0 (q1 (q2 (q3) q4) q5 (q6) q7) q8

	(	)
q0	q1	
q1	q2	
q2	q3	
q3		q4
q4		q5
q5	q6	
q6		q7
q7		q8

l’état q8 est un état final il ne possède pas de transition

État du transducteur : Ensemble des parties de { q0,q1,q2,q3,q4,q5,q6,q7,q8 }

Reconnaissance d’un schéma

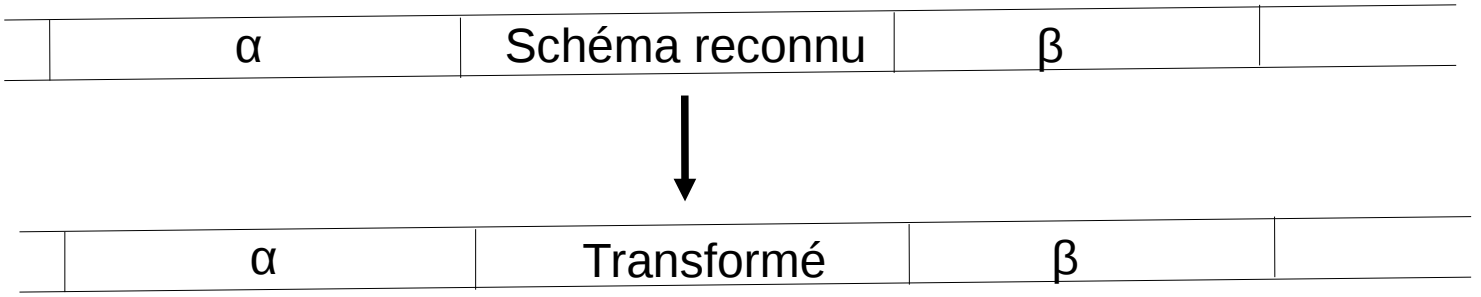


La présence de q8 en fin de lecture signifie que le chemin est présent.

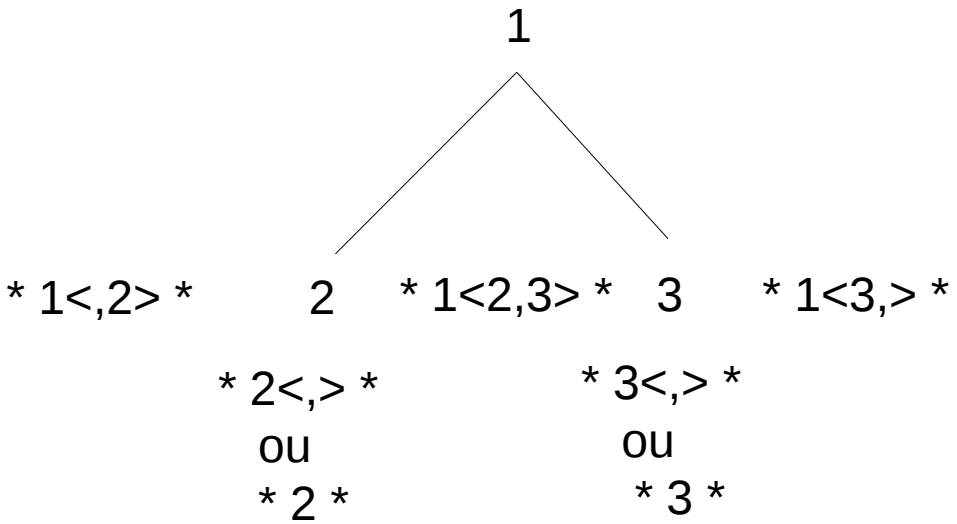
Le second transducteur étiquette les points du schéma à partir de la sortie du premier

Transformation d’un schéma

Sortie d’une recherche :



Pour la définition du transformé :



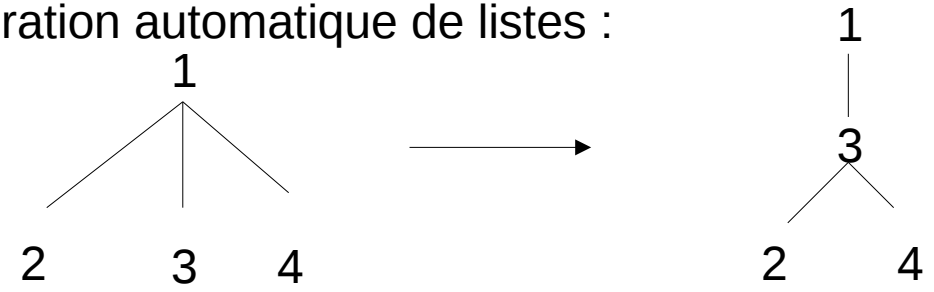
Transformation d'un schéma

Lorsque le schéma est une forêt :

1,2 ⟶ * @<1,2> *

Pas de définition possible pour la structure à gauche de 1 ou à droite de 2

Génération automatique de listes :



Génération :

1 (* 1<,3> * (2 (***** 2 *) , * 3 * , 4 (* 4 *)) , * 1 < 3,> *)

Bien sûr cette génération peut être désactivée

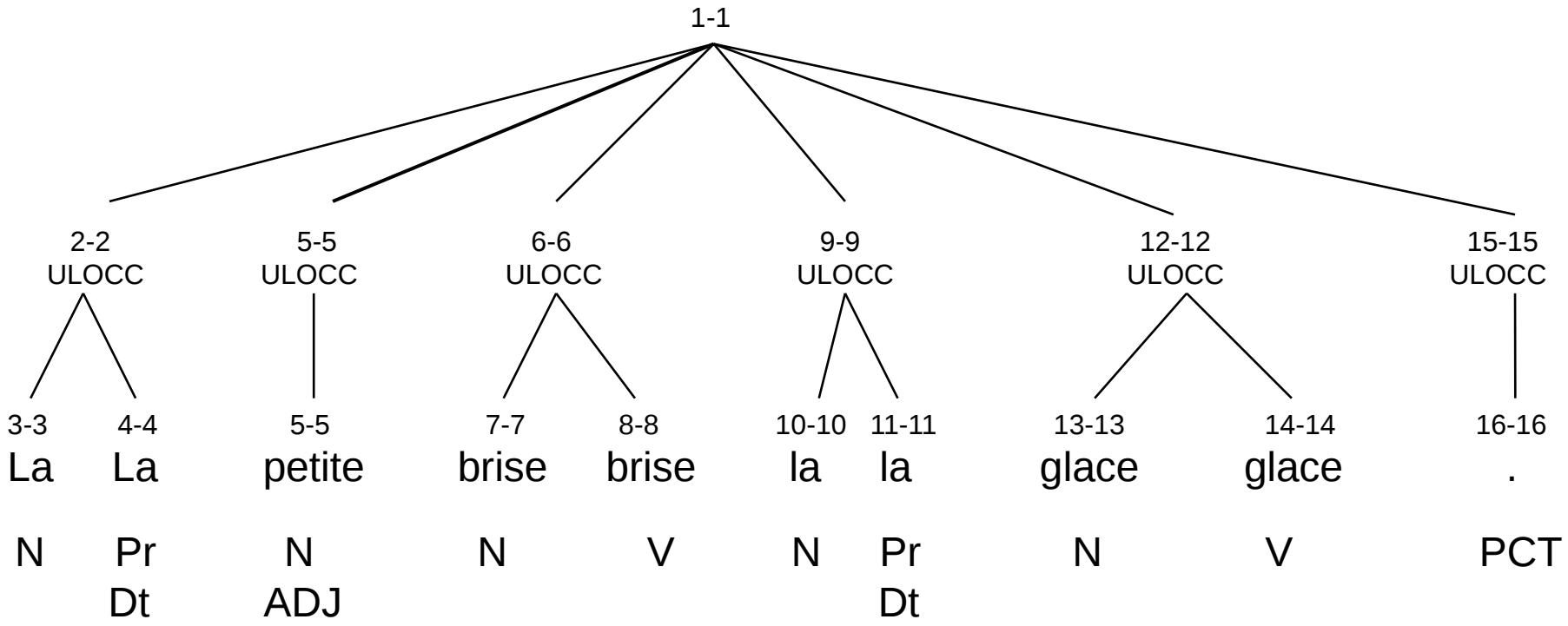
Propriété de la reconnaissance d'un schéma

- La reconnaissance d'un schéma est linéaire : deux lectures de la chaîne d'entrée
  la deuxième lecture à une longueur maximale de (k + 1) x l,
  k étant le cardinal de l'ensemble des états et l la longueur de la chaîne d'entrée
- Il est possible de rechercher plusieurs schémas simultanément
- Pour SYGFRAN la recherche simultanée dépasse le millier de schémas
- Pour l'implémentation du moteur la borne correspond au nombre maximum d'états pouvant être traités (Il est actuellement de 8192)
- Lorsque le système traite des éléments structurés ayant plusieurs dimensions, la recherche s'effectue dans l'ordre sur chaque dimension et pour l'étiquetage des schémas dans des dimensions suivant la première, des contraintes supplémentaires sont ajoutées : des points de ces dimensions suivantes peuvent avoir la contrainte d'avoir la même référence d'étiquette que celle d'un point déjà reconnu
- Les contraintes d'étiquettes sont globalisées : Pour un schéma la contrainte d 'étiquette est associée à la progression d'un état. Si cette contrainte à déjà été évaluée elle n'est bien sûr par réévaluée.
- Un état des conditions est associé à chaque point de la structure (deux bits pour 3 valeurs : non évaluée, vraie, fausse) l'indication :
  NOMBRE D'OCTETS CONDITIONS:4733
  donne la longueur de la chaîne qui sera associée à chaque point.

Taggage : Exemple de traitement sur deux dimensions

Texte : La petite brise la glace.

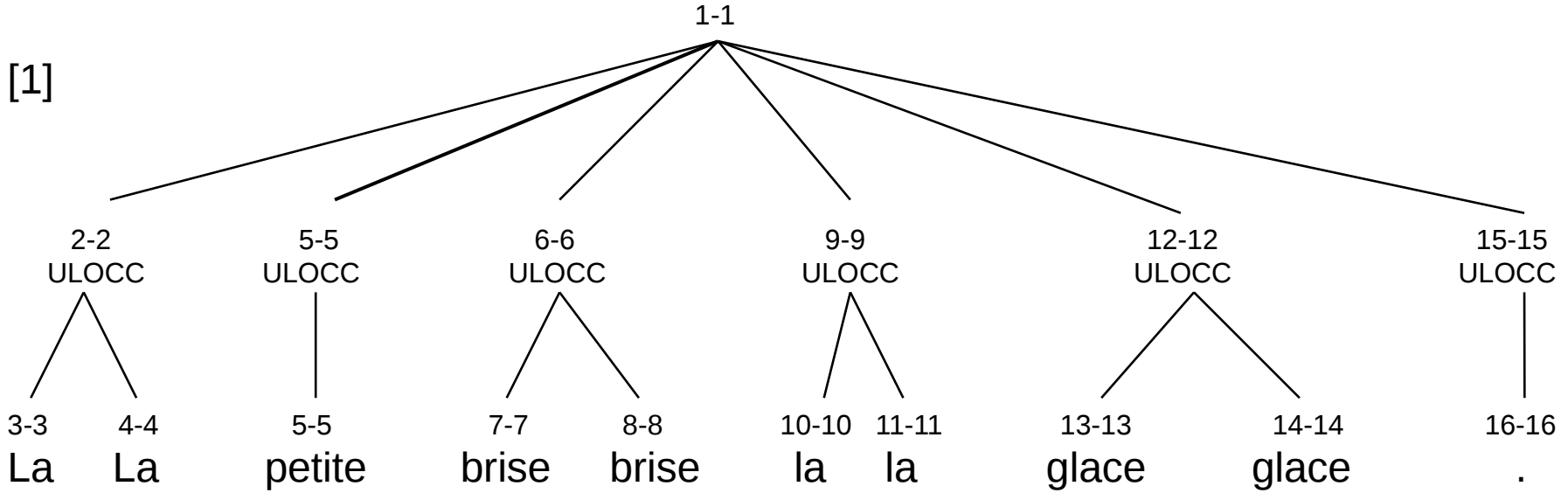
Résultat du traitement morphologique :



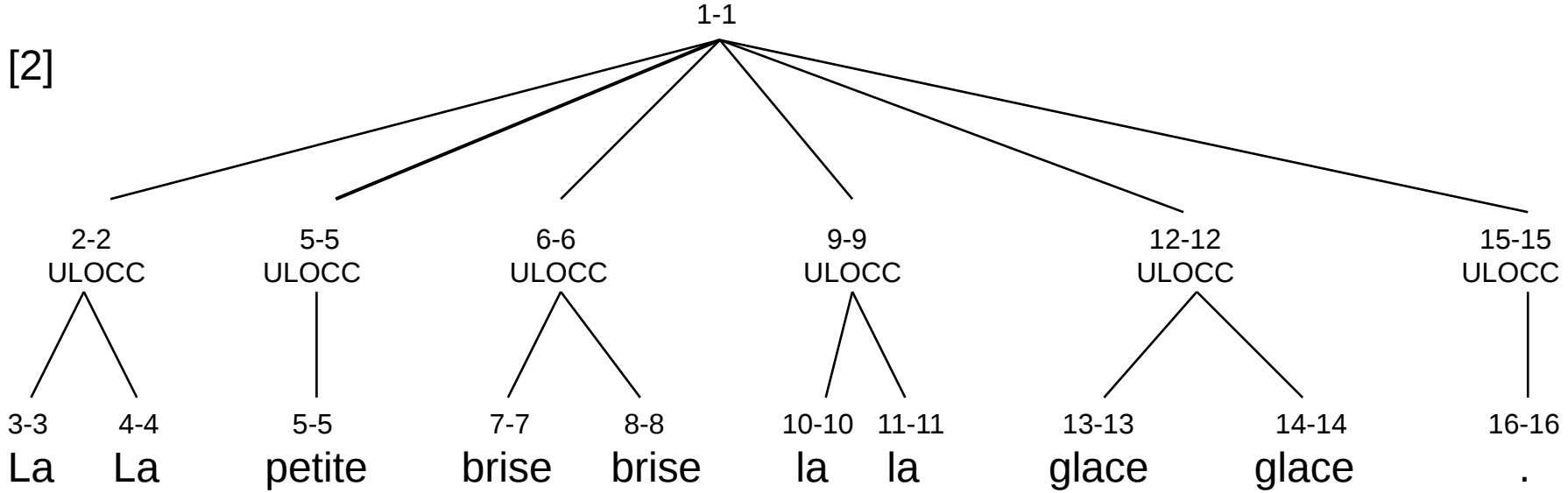
Taggage : Exemple de traitement sur deux dimensions

Entrée traitement Syntaxique :

[1]



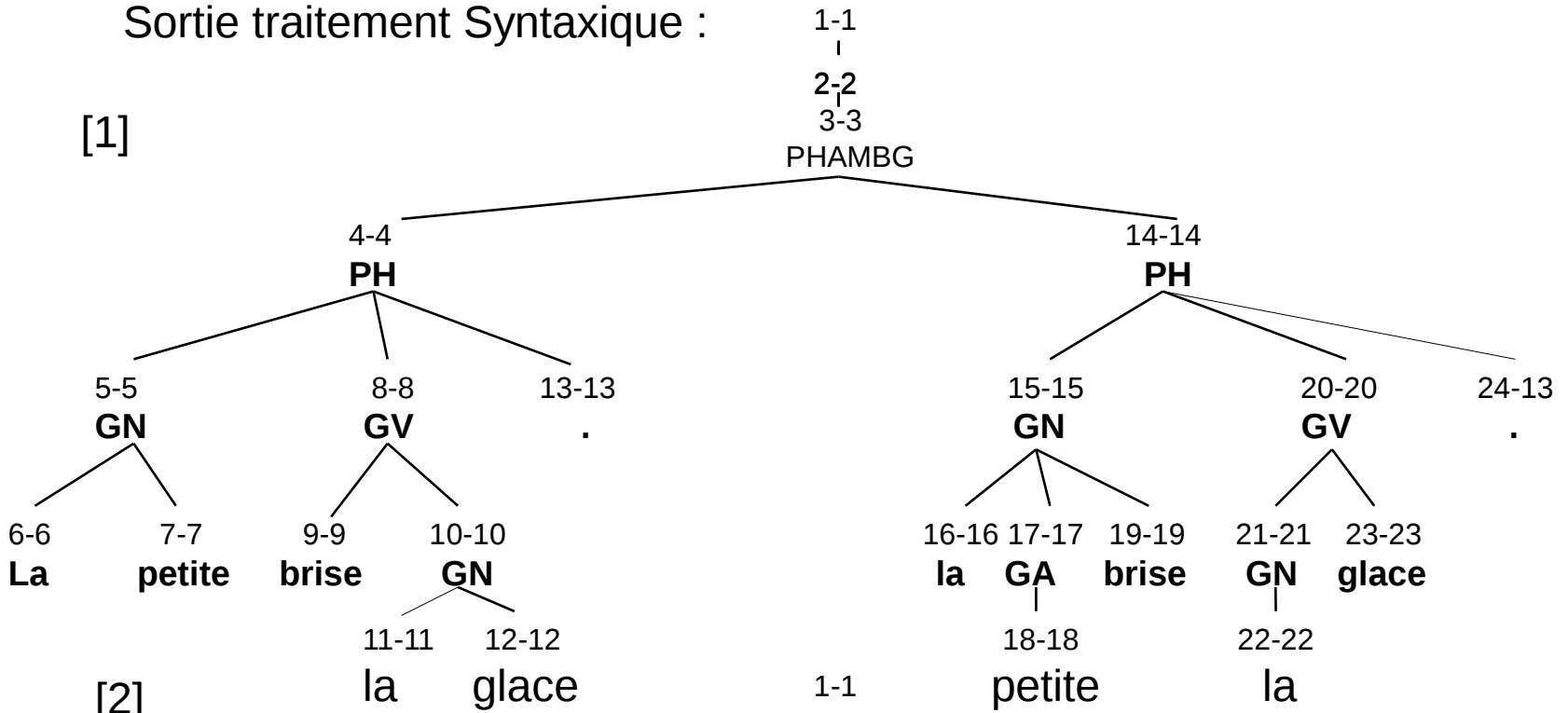
[2]



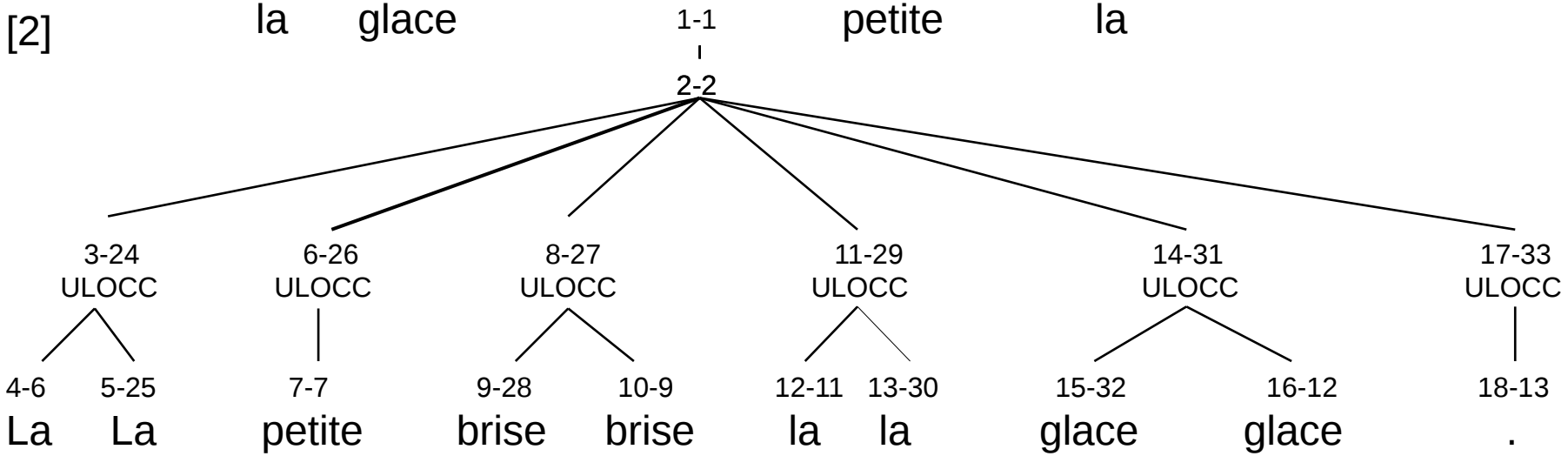
Taggage : Exemple de traitement sur deux dimensions

Sortie traitement Syntaxique :

[1]



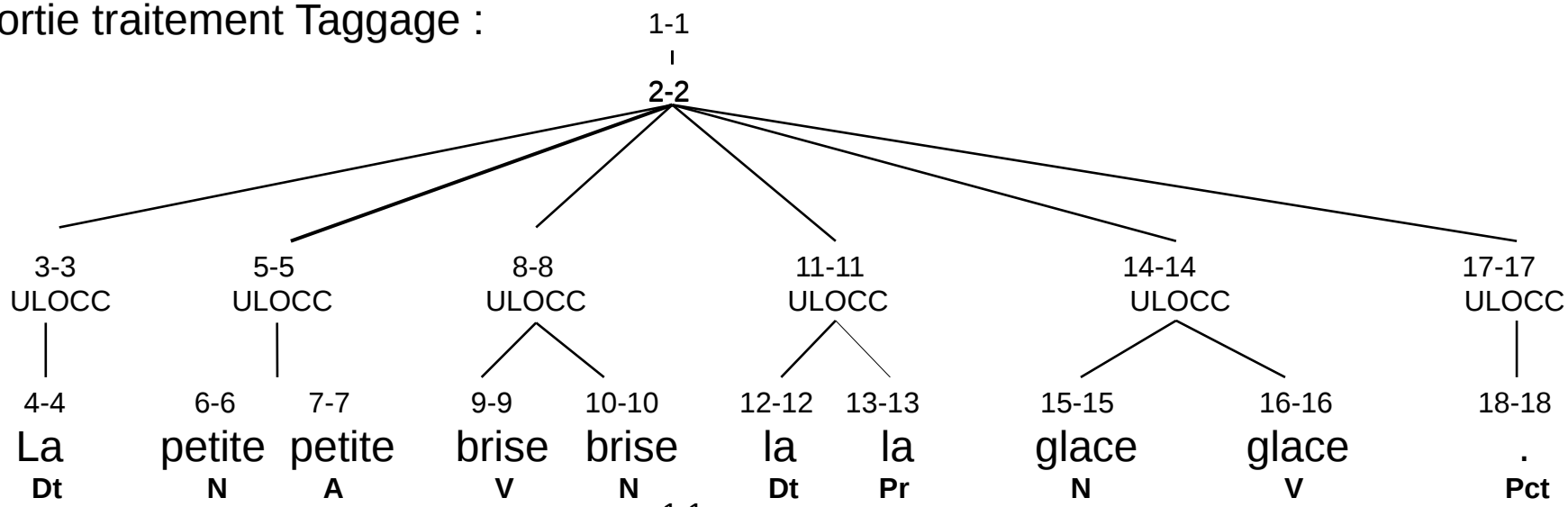
[2]



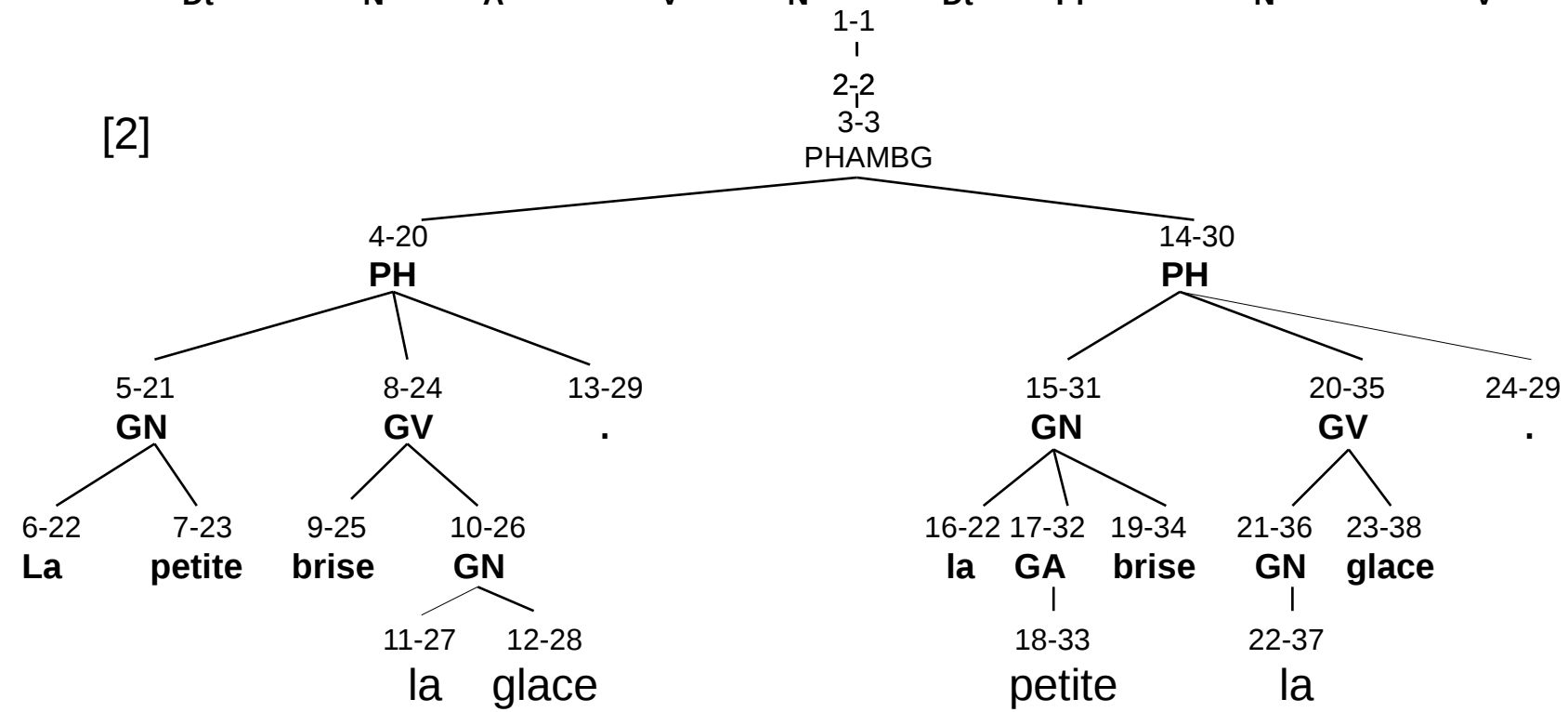
Taggage : Exemple de traitement sur deux dimensions

Sortie traitement Taggage :

[1]



[2]



Taggage : Exemple de traitement sur deux dimensions

RésultatTaggage :

La [Article défini Féminin Singulier ProfondeurSyntaxique(4) 'le']

petite [Nom Commun Féminin Singulier Fonction(Gouverneur) FonctionGroupe(Sujet) ProfondeurSyntaxique(4) 'petit' |

Adjectif Féminin Singulier Fonction(Gouverneur) FonctionGroupe(Attribut)
ProfondeurSyntaxique(5) 'petit']

brise [Nom Commun Féminin Singulier Fonction(Gouverneur) ProfondeurSyntaxique(4) 'brise' |

Verbe conjugué VOIX_ACTIVE INDICAT|SUBJNCT|IMPERAT|INDICATIF(PRES) SUBJONCTIF(PRES) IMPERATIF(PRES)
Fonction(Gouverneur)
ProfondeurSyntaxique(4) 'briser']

la [Article défini Féminin Singulier Fonction(Gouverneur) FonctionGroupe(Objet direct) ProfondeurSyntaxique(5) 'le' | Pronom Personnel Féminin Singulier

Fonction(Gouverneur) FonctionGroupe(Objet direct) ProfondeurSyntaxique(5) 'le']

glace [Verbe conjugué VOIX_ACTIVE INDICAT|SUBJNCT|IMPERAT INDICATIF(PRES) SUBJONCTIF(PRES) IMPERATIF(PRES) Fonction(Gouverneur)

ProfondeurSyntaxique(4) 'glacer' | Nom Commun Féminin Singulier Fonction(Gouverneur) FonctionGroupe(Objet direct)
ProfondeurSyntaxique(5) 'glace']

. [Ponctuation ProfondeurSyntaxique(3) '.']

Reconnaissance des lexies

La reconnaissance des lexies s'appuie sur trois éléments :

1) Un dictionnaire

2) Une lecture de ce dictionnaire par des étiquettes éphémères

3) Une structure support

(1)
```
&VAL(ADVAL,'LEXIEFIGEE3').
 &CLEX(TYPLOC).
  &VAL(TYPLOC,FILTRE).
   &CLEX(VART3[UL1,UL2,UL3]).
     REPER3FILTRE: &VAL.
     'compte' 'tenu' 'de' :TYP=LOC;TYPLOC=FIGEE;TYPCONS=LXGNP3;CAT=PREP;TPREPLOC=DE;
                           SEMA=LEXIECC;  UL='compte tenu de';LOCUTION='compte tenu de';
                           FRM='compte_tenu_de'.
          "                : CAT = N; SOUSN = NCOM;CASLOC=C;TYP=CCNOM.
          "                : CAT = ADJOINT;SOUSA=ADNOM; SOUSV = PAPA;CASLOC=P.


 &VAL(ADVAL,'LXF31').
 &CLEX(TYPLOC).

  &VAL(TYPLOC,FILTRE).
   &CLEX(UL1).
    &VAL:REPER3FILTRE.
```

Reconnaissance des lexies

(2) LCTLXFFILTRE31: TYPLOC=FILTRE;UL1=%(UL1)←UL(#);ADVAL='LXF31'.

LCTLXFFILTRE32: TYPLOC=FILTRE;UL2=%(UL2)←UL(#);ADVAL='LXF32'.

LCTLXFFILTRE33: TYPLOC=FILTRE;UL3=%(UL3)←UL(#);ADVAL='LXF33'.

LCTLXFFILTRE3(x,y,z): TYPLOC=FILTRE;UL1=%(UL1)<-UL(x);UL2=%(UL2)<-UL(y);
UL3=%(UL3)<-UL(z);ADVAL='LEXIEFIGEE3'.

(3) **RLXF3: 0(1),*,2(3),*,4(5) / 0: FLX = 'ULOCC'; 1: (FILTREFFECT = 0)&(TYPLOC(DICT(LCTLXFFILTRE31)) != 0);**
2: FLX = 'ULOCC'; 3: (FILTREFFECT = 0)&(TYPLOC(DICT(LCTLXFFILTRE32)) != 0);
4: FLX = 'ULOCC'; 5: (FILTREFFECT = 0)&(TYPLOC(DICT(LCTLXFFILTRE33)) != 0) /
(TYPCONS(DICT(LCTLXFFILTRE3(1,3,5))) = LX3)|(TYPCONS(DICT(@)) = LX3GNP)&
(CAT(1)&CAT(DICT(@,2)) != 0)&(CAT(3)&CAT(DICT(@,3)) != 0)&(CAT(5)&CAT(DICT(@,4)) != 0)

=> N0(1),N2(3),N4(5) / N0: 0;
1:1(GCATL = GCATL(DICT(LCTLXFFILTRE3(1,3,5),2));
TYP=TYP(DICT(@,2)); <TPREPVB(DICT(@,2)) != 0: TPREPVB = TPREPVB(DICT(@,2))>;
FILTREFFECT = FILTREFFECT(DICT(@,2));
<CAT(*) = V: <SOUSV(DICT(@,2)) != 0: SOUSV = SOUSV(DICT(@,2)) #
SOUSV = SOUSV(1)>;TYP = TYP(DICT(@,2))#TYP = TYP(2)|TYP(DICT(@,2))>;
SEMA = SEMA(DICT(@,2)));
N2:2;
3:3(….. etc...).

TRAITEMENT SÉMANTIQUE

J’ai défini le traitement sémantique à partir de la structure syntaxique dans les années 80.

J’étais alors sous-directeur d’une équipe de NANCY 1 : le CELTA

Le premier essai fut le suivant :

À partir des mots d’un petit texte choisi pour l’expérience :

- Chaque membre de l’équipe (8 participants) :
  - Définissait un ou plusieurs concepts et l’associait à un mot
  - Cette association comprenait 2 pondérations, forte ou faible
- Une fois ce travail effectué les tests sur les calculs vectoriels ont donné de bons résultats.

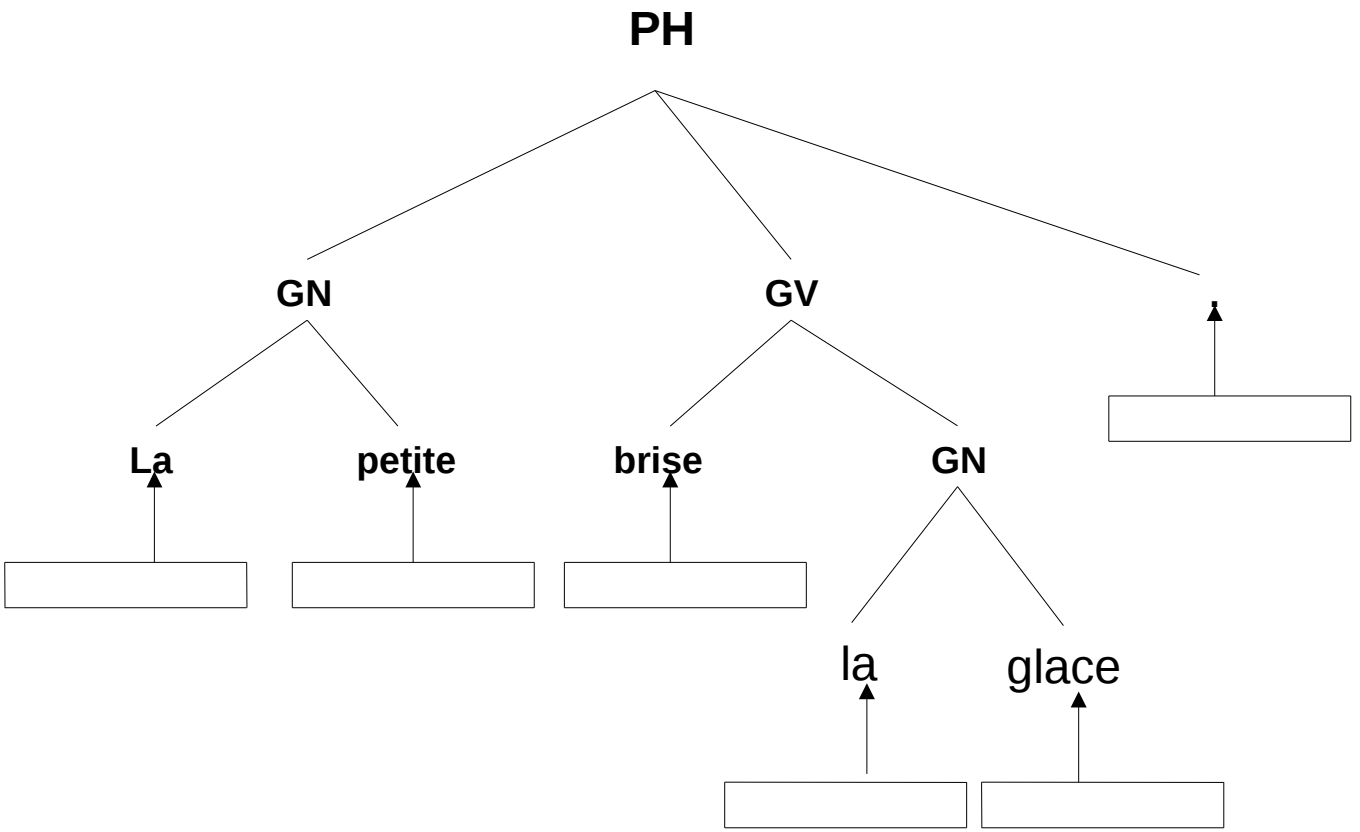
À partir de ces résultats il a fallu rechercher un ensemble prédéfini de concepts.

Le directeur de l’équipe (le professeur Guy Bourquin, angliciste) a proposé de rechercher un équivalent du Roget’s Thesaurus

Le seul faiblement équivalent trouvé fut celui des éditions Larousse

TRAITEMENT SÉMANTIQUE

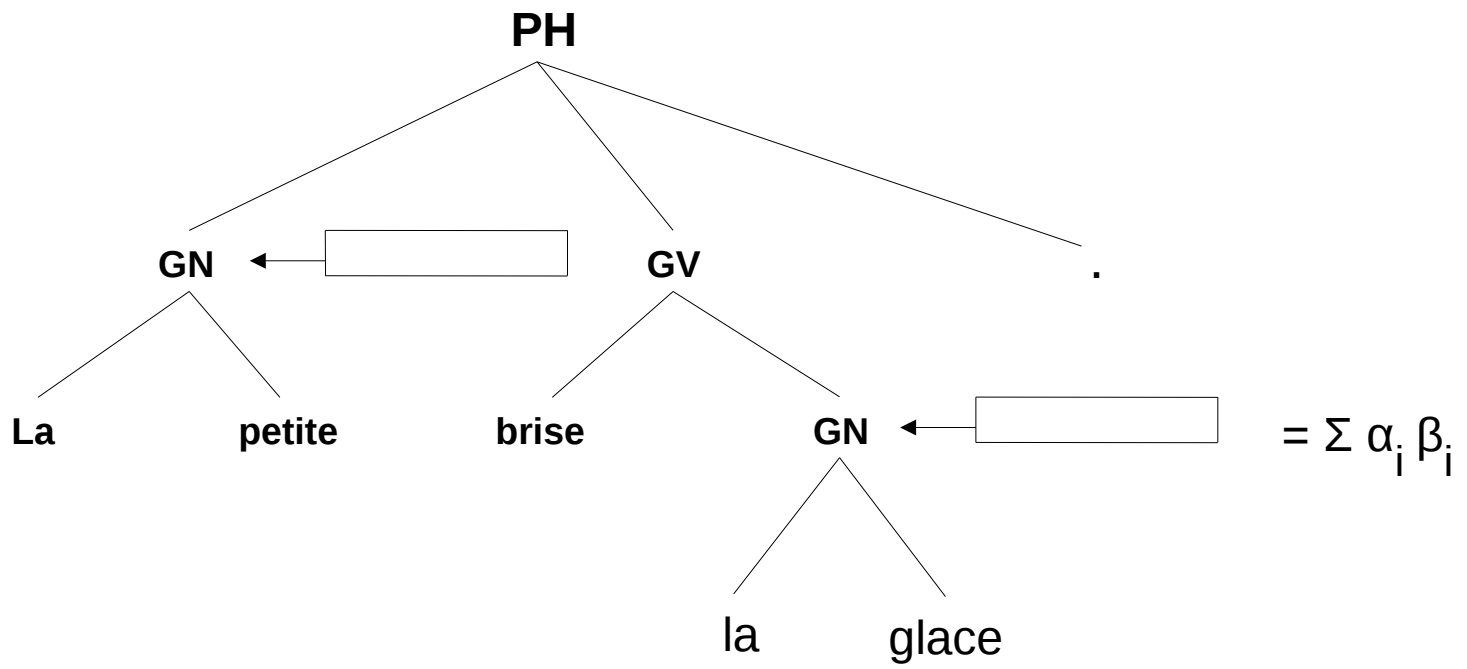
Construction d’un vecteur sémantique à partir de la construction syntaxique :



Première étape : association des vecteurs sémantiques à chaque feuille résultant de la lecture du dictionnaire à partir de la variable lemme

TRAITEMENT SÉMANTIQUE

Construction d'un vecteur sémantique à partir de la construction syntaxique phase 2 :

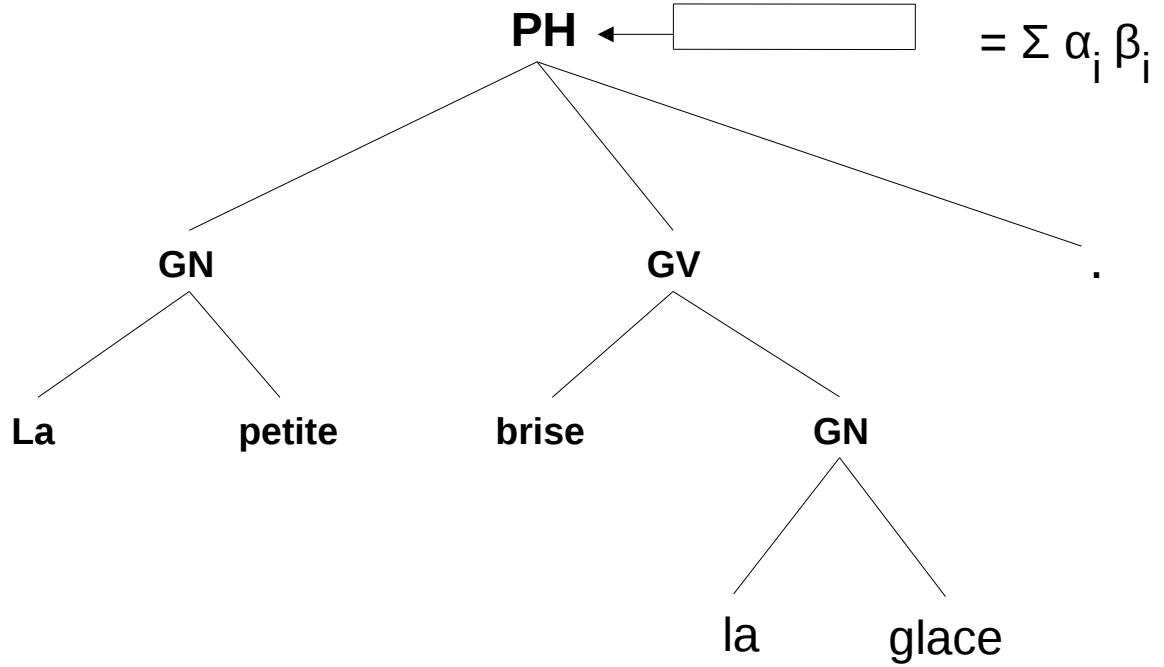


α_i : élément dépendant directement de ce point

β_i : coefficient fonction de la fonction syntaxique de ce point

TRAITEMENT SÉMANTIQUE

Construction d'un vecteur sémantique à partir de la construction syntaxique phase terminale :



Les vecteurs dépendants du thésaurus Larousse ont 874 composantes

Tous les vecteurs sont normés.

Après chaque combinaison linéaire on effectue cette normalisation.

TRAITEMENT SÉMANTIQUE

Comparaison de texte ou de phrase par leur vecteur associé : la fonction de similitude

Raison de la définition de cette fonction :

- Le produit scalaire donne bien la notion de distance
- Le produit scalaire ne tient pas compte de l'importance d'une composante
- Il faut définir une fonction qui dans le cas de vecteurs semblables correspond au produit scalaire
- Dans le cas contraire atténue la valeur du produit en fonction de l'ordre relatif d'une composante
- Cette fonction doit être symétrique

```
for(i = 0; i < n; ++i)
    {
        if (ord1[i] == ord2[i])
            sim += table1[i] * table2[i];
        else {
            k = ord1[i] - ord2[i];
            if (k < 0)
                att = -k;
            else att = k;
            att = 1 + att/100;
            sim += (table1[i] * table2[i]) / att;
        }
    }
```

TRADUCTION AUTOMATIQUE

La première expérience de traduction automatique avec SYGMART
a été réalisée par Christiane Vigroux, linguiste au CELTA de Nancy

Pour montrer la faisabilité de la traduction elle a réalisé une traduction de l'espagnol vers le français d'un article d'un hebdomadaire.
Évidemment cette traduction est basée sur un analyseur syntaxique spécifique au texte traduit.

La traduction français anglais à partir de l'analyseur généraliste SYGFRAN a été réalisé par Violaine Prince au LIRMM.

C'est à partir de cette grammaire de transfert que l'exemple suivant est exposé.

Le texte traité est : Ce marché a des senteurs de Provence.

TRADUCTION AUTOMATIQUE

La suite d’enchainement des systèmes est la suivante :

'OPALE'
'GrammaireAnalyseMorphologique' 'DictionnaireSegmentsFrancais' 'DictionnaireValeurSemantiqueDeriv'
'VariableAnalyseMorphologique'

'TELESI'
'GrammaireAnalyseSyntaxique' 'DictionnaireAnalyseSyntaxique' 'VariableAnalyseSyntaxique'

'TELESI'
'GrammairePresenteResul' '' 'VariableResultatSyntaxique'

'TELESI'
'GrammaireCalculSem' 'DictionnaireVecteurSemantique' 'VariableResultatSyntaxique'

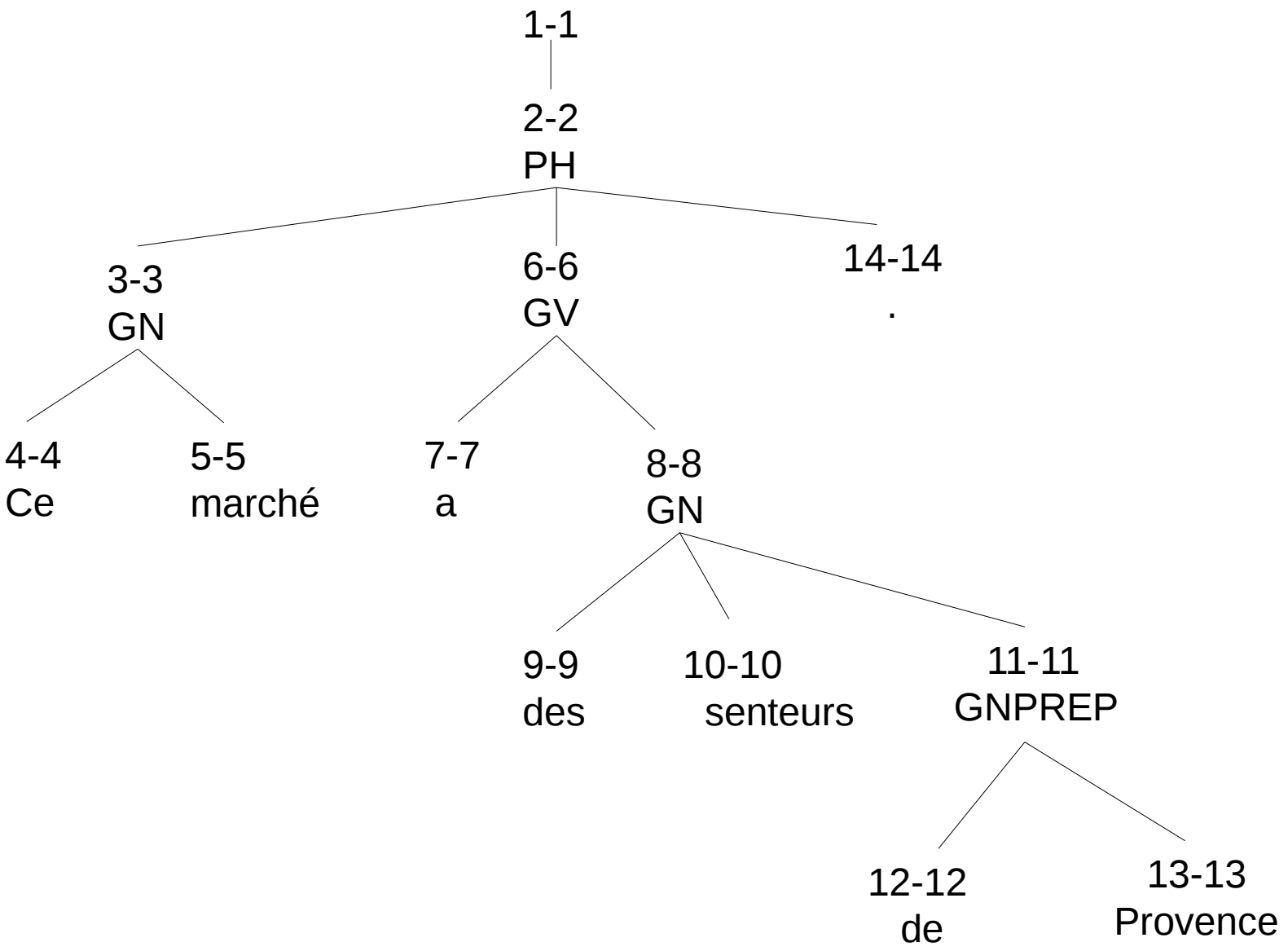
'TELESI'
'GrammaireTranslationAng' 'DictionnaireTranslationAng' 'VariableTranslationAng'

'TELESI'
'GramTrad' 'DictTrad' 'VarExpTrad'

'AGATE'
'GramAGTrad' 'DictAGTrad' 'VarExpTrad'

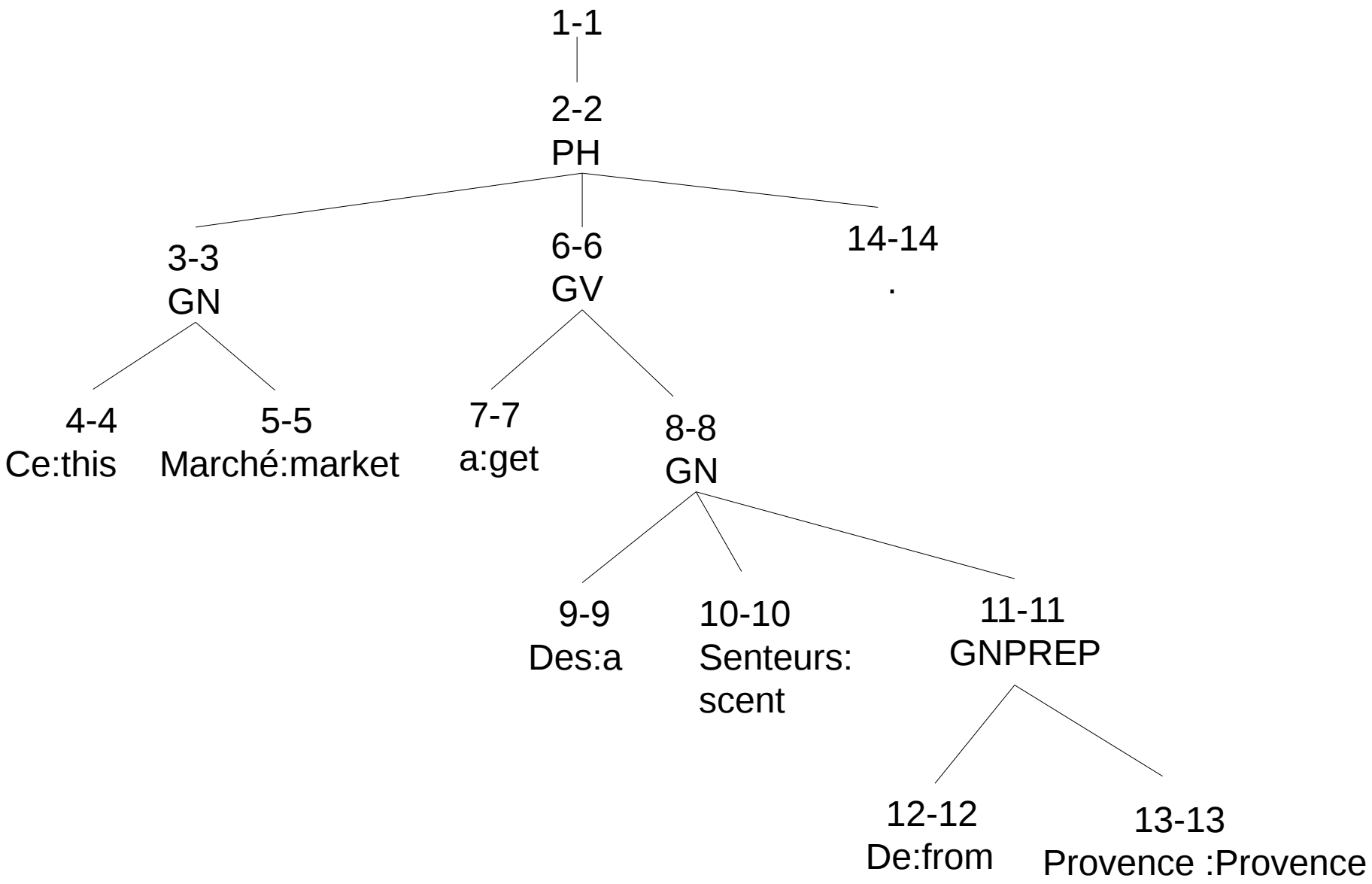
TRADUCTION AUTOMATIQUE

RESULTAT GRAMMAIRE TELESI: GrammairePresenteResul



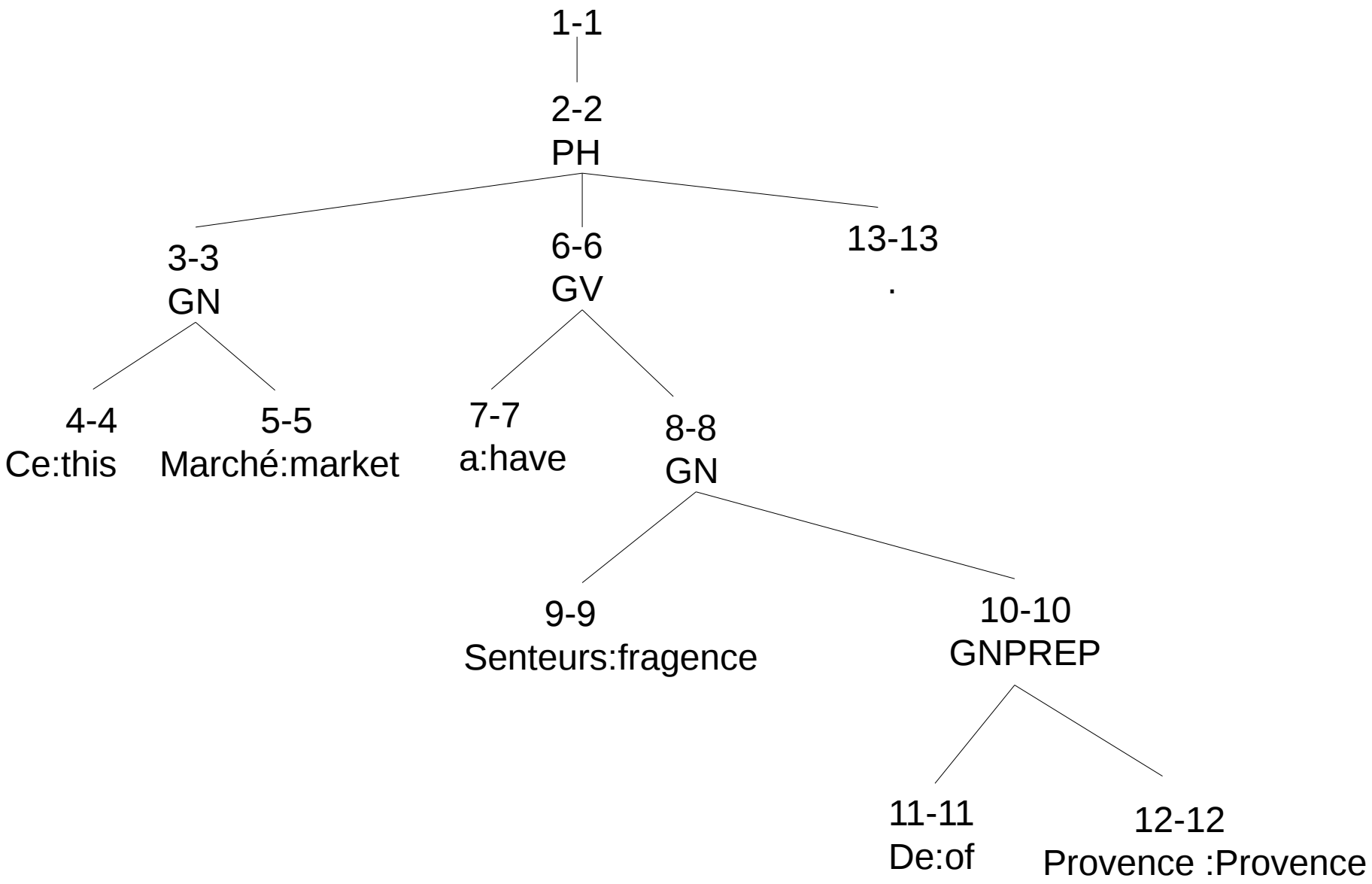
TRADUCTION AUTOMATIQUE

RESULTAT GRAMMAIRE TELESI: GrammaireTranslationAng



TRADUCTION AUTOMATIQUE

RESULTAT GRAMMAIRE TELESI: GramTrad



TRADUCTION AUTOMATIQUE

Texte :

Ce marché a des senteurs de Provence.

Résutat traduction :

this market has fragrences of Provence .

Application applisyg

Cette application regroupe toutes les sorties de **SYGFRAN**

Elle est définie sous LINUX (ou UNIX) ou windows

Elle peut servir de base à la création d'un serveur

La commande générale est :

<chemin absolu>/aplisyg donneesTagPcLinux64 texte [n]

Avec :

N absent => produit un fichier texte.tag

1 => produit un fichier texte.stx structure interne élément structuré

2 => produit un fichier texte.sts structure interne élément structuré
Avec vecteur sémantique

3 => produit un fichier texte.html taggage coloré format html

4 => produit un fichier texte.vs vecteur sémantique du texte 874 flottants (3496 octets)

5 => produit un fichier texte.xhtml dessin svg de la structure syntaxique

Application applisyg

Texte : Un exemple de sortie SYGFRAN.

Sortie html :

<FONT COLOR=RED><B>Un</B></FONT> [Article indÃ©fini Masculin Singulier ProfondeurSyntaxique(3) <FONT COLOR=BLUE><B>'un'</FONT></B>]
<FONT COLOR=RED><B>exemple</B></FONT> [Nom Commun Masculin Singulier Fonction(Gouverneur) ProfondeurSyntaxique(3) <FONT COLOR=BLUE><B>'exemple'</FONT></B>]
<FONT COLOR=RED><B>de</B></FONT> [PrÃ©position ProfondeurSyntaxique(4) <FONT COLOR=BLUE><B>'de'</FONT></B>]
<FONT COLOR=RED><B>sortie</B></FONT> [Nom Commun FÃ©minin Singulier Fonction(Gouverneur) FonctionGroupe(Attribut) ProfondeurSyntaxique(4) <FONT COLOR=BLUE><B>'sortie'</FONT></B>]
<FONT COLOR=RED><B>SYGFRAN</B></FONT> [Nom Propre Fonction(Gouverneur) FonctionGroupe(Attribut) ProfondeurSyntaxique(4) <FONT COLOR=BLUE><B>'SYGFRAN'</FONT></B>]
<FONT COLOR=RED><B>.</B></FONT> [Ponctuation ProfondeurSyntaxique(2) <FONT COLOR=BLUE><B>'.'</FONT></B>]

Certains navigateurs (comme firefox) n’ont pas besoin d’ entête pour le visualiser

Cela permet également d’inclure ce résultat dans une application plus générale.

Application applisyg

Texte : Un exemple de sortie SYGFRAN.

Sortie xhtml :

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN" "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="500" height="595" viewBox="0 0 500 595" xmlns="http://www.w3.org/2000/svg"
   xmlns:xlink="http://www.w3.org/1999/xlink" version="1.2"                    baseProfile="tiny">
<title>Structure SYGMART</title>
<desc>Generated with SYGMART</desc>
<defs>
</defs>
<g fill="none" stroke="black" stroke-width="1" fill-rule="evenodd" stroke-linecap="square" stroke-linejoin="bevel" >
<g fill="none" stroke="#000000" stroke-opacity="1" stroke-width="1" stroke-linecap="square" stroke-linejoin="bevel"
                    font-family="Helvetica" font-size="12" font-weight="400" font-style="normal">
<text fill="#000000" fill-opacity="1" stroke="none" xml:space="preserve" x="227" y="65" font-family="Helvetica" font-size="8" font-weight="100"
 font-style="normal" >1</text>
<polyline fill="none" vector-effect="non-scaling-stroke" points="227,77 227,130 " />
<text fill="#000000" fill-opacity="1" stroke="none" xml:space="preserve" x="225" y="140" font-family="Helvetica" font-size="8" font-weight="100" font-style="normal"
>2</text>
<text fill="#000000" fill-opacity="1" stroke="none" xml:space="preserve" x="216" y="150" font-family="Helvetica" font-size="8" font-weight="100" font-style="normal"
>PHN</text>
<polyline fill="none" vector-effect="non-scaling-stroke" points="227,152 147,205 " />
<text fill="#000000" fill-opacity="1" stroke="none" xml:space="preserve" x="145" y="215" font-family="Helvetica" font-size="8" font-weight="100" font-style="normal"
>3</text>


....etc
```

Même remarque que précédement

SYGTEXT

Les principales informations sur SYGMART et SYGFRAN se trouvent sur le site :

www.sygtext.fr

Avec notamment la possibilité :

- de télécharger des applications de démonstrations de SYGFRAN pour LINUX ou Windows
- De télécharger la structure syntaxique du livre « le petit prince »
- Des résultats de l’analyse des groupes de phrases du corpus de développement

FIN