JACQUES COURTIN

UN ANALYSEUR SYNTAXIQUE INTERACTIF POUR LA COMMUNICATION HOMME-MACHINE

Nous envisageons la réalisation d'un système de communication Homme-Machine en langues naturelles, qui pourrait être utilisé par exemple pour l'étude de la langue elle-même ou pour la réalisation d'un système questions-réponses sur un sujet déterminé (consultation d'une banque de données et réponses appropriées).

Avant d'en arriver à la réalisation du modèle sémantique, il faut se définir un modèle d'analyse morphologique et un modèle d'analyse syntaxique. Néanmoins, nous pensons que la sémantique doit pouvoir intervenir le plus rapidement possible afin de lever les ambiguités rencontrées au niveau du modèle précédent.

Dans cet exposé, nous nous limiterons à l'étude des deux premiers modèles.

Dès à présent, nous précisons que nous nous sommes orientés vers des systèmes interactifs afin d'établir un véritable dialogue entre l'utilisateur et la machine.

Pour l'analyse morphologique, nous ne nous sommes pas limités uniquement à la reconnaissance du mot mais également à celle des lo cutions non ambigues afin d'éviter leurs reconstructions au niveau de la syntaxe.

Ce premier modèle constitue à lui seul un véritable système puisque nous avons défini un éditeur morphologique conversationnel dont les grandes lignes seront explicitées en 1.

Quant à l'analyseur syntaxique, nous nous sommes fixés comme objectif d'obtenir directement des structures de dépendances car elles sont facilment interprétables. Nous avons donc abandonné les structures de constituants obtenues à partir des grammaires « *context-free* » pour les raisons suivantes:

— les algorithmes d'analyse avaient l'inconvénient majeur de conduire à une très grande combinatoire. On pouvait certes la limiter en introduisant des « validations-saturations » qui avaient le désavantage de demander au linguiste de définir le cheminement de l'algorithme au niveau de la grammaire. Il en résultait une très grande difficulté de mise au point.

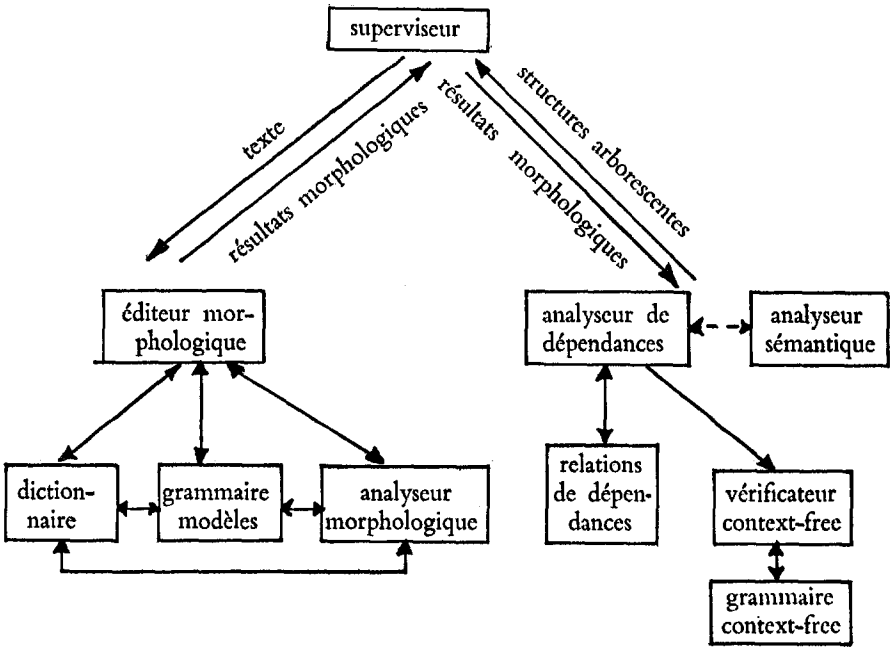
— D'autre part, cette démarche nous semblait peu naturelle et nécessitait ensuite l'écriture d'un modèle supplémentaire permettant d'obtenir des structures plus facilement interprétables que les structures de constituants.

Nous avons également écarté les algorithmes définis à partir d'une grammaire de dépendances du fait même qu'elle nécessite un nombre important de règles car il faut expliciter toutes les configurations possibles de dépendants sous un gouverneur donné.

Le choix que nous avons fait a été d'accepter comme données des structures de dépendances. Cette stratégie, qui sera définie en 2, conduit naturellement à un système d'apprentissage. L'algorithme n'acceptant que des relations entre les catégories, il fera appel à une grammaire *context-free* pour valider une sous-structure terminale en vérifiant les accords entre les variables. Il est à noter que l'écriture de cette grammaire sera extrêmement simplifiée étant donné qu'il suffit de préciser uniquement les conditions de réécriture des variables grammaticales entre les catégories.

Cette démarche nous semble beaucoup plus naturelle. Nous précisons dès maintenant que pour des facilités de réalisation nous ne traiterons que des structures de dépendances projectives.

Nous pouvons résumer le projet par le schéma ci-dessous:



1. EDITEUR MORPHOLOGIQUE CONVERSATIONNEL

Cet éditeur s'adresse à deux catégories d'utilisateurs:

— le spécialiste qui est chargé de la réalisation du système morphologique du point de vue linguistique,

— le non spécialiste qui désire faire de l'analyse morphologique sans être obligé de connaître le fonctionnement du système.

1.1. *Les outils mis à la disposition du spécialiste.*

— L'analyse morphologique est réalisée à l'aide d'une grammaire d'états finis munie de validations et de saturations qui permet non seulement de reconnaître un mot de la langue mais également de réaliser en parallèle une transduction des variables.

— Du point de vue morphologique, chaque mot peut être rangé dans une classe ayant un certain comportement linguistique. Par exemple, en français les verbes du premier groupe: *aimer, chanter, diviser,* etc. ..., admettent la même conjugaison. On a donc choisi arbitrairement un mot de la classe afin de pouvoir, au moment de l'indexage, faire référence à ce mot. C'est ce que nous avons appelé un modèle. Par exemple, si on choisit la base *chant* comme modèle, dans le dictionnaire il suffira de dire que les bases *DIVIS*, *AIM*, etc ... se comportent comme *CHANT* du point de vue de la conjugaison verbale.

— Nous avons donc mis à la disposition du linguiste un petit langage extrêmement simple qui lui permettra d'écrire les règles de grammaire et les modèles. La mise au point se faisant à l'aide d'un terminal par l'intermédiaire du système CP/CMS de l'université de Grenoble, l'utilisateur peut donc à tout moment modifier les règles et les modèles qui sont ensuite compilés en vue du traitement morphologique.

— Quant au dictionnaire, il est important de noter:

a) qu'il est organisé par ordre alphabétique et en fonction de la fréquence des mots.

b) Qu'étant donné que l'on désire obtenir tous les découpages possibles, on a utilisé, pour l'identification, la technique du *longestmatch*; c'est-à-dire que l'on cherche la plus longue superposition possible entre l'occurrence et un élément du dictionnaire; une fois trouvée, on recherche un élément plus court que cette dernière et ainsi de suite. Nous avons donc organisé le dictionnaire de telle sorte qu'ayant trouvé

la plus longue superposition possible, on obtienne « presque gratuitement » les éléments qui sont plus courts.

c) qu'étant donné que l'on désire analyser des formes complètes telles que: *à gauche de, à droite de,* etc. ..., le blanc ne sera plus considéré comme un séparateur.

1.2. *Les outils mis à la disposition du non spécialiste et du spécialiste.*

Nous avons voulu donner à l'utilisateur, en cours d'exécution d'analyse morphologique, la possibilité:

— de corriger un mot mal orthographié.

— d'interroger le dictionnaire;

— d'indexer automatiquement, sans être obligé de connaître les modèles, grâce à un système d'équivalence;

— de donner une équivalence morphologique sans pour cela être obligé d'introduire le mot dans le dictionnaire (cette option est particulièrement intéressante pour les noms propres, noms de villes, les mots rares, etc ...);

— de faire du pas à pas et ainsi pouvoir vérifier les résultats après chaque mot.

Nous avons énuméré ici les fonctions principales de cet éditeur, étant bien entendu que toutes ces possibilités sont offertes à l'utilisateur en cours d'exécution et qu'elles ne remettent pas en cause la partie du texte déjà analysée.

2. ANALYSE SYNTAXIQUE

Pour réaliser cette analyse, nous nous sommes servis de la notion de poids qui avait été introduite dans le système de traduction automatique du C.E.T.A. pour effectuer la génération du français.

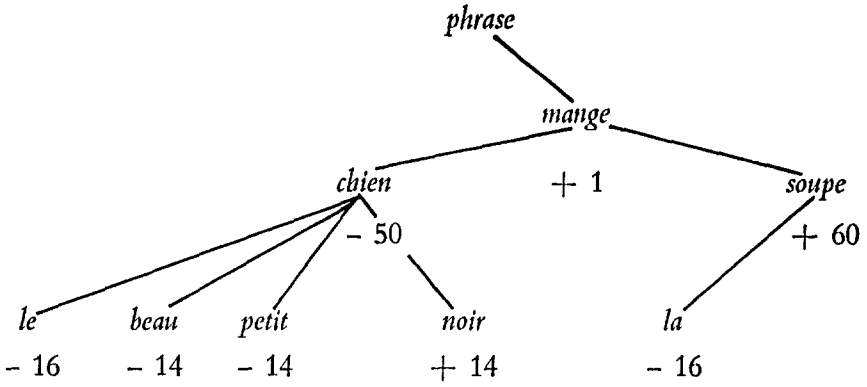
Nous rappelons ici la définition du poids.

Poids. A chaque sommet de la structure de dépendances, on va affecter un poids. Les poids négatifs correspondront aux mots placés à gauche du gouverneur, et les poids positifs à ceux placés à droite, le gouverneur ayant un poids zéro par rapport à ses dépendants.

D'autre part, il est également important de noter que le poids est en même temps caractéristique d'une fonction syntaxique.

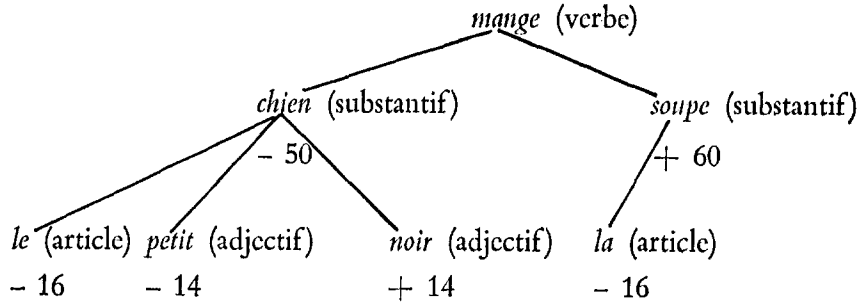
Nous avons également ajouté une catégorie syntaxique appelée phrase qui est telle que tous les gouverneurs d'une phrase auront un poids + 1 par rapport à cette catégorie.

Exemple:



Nous allons donc demander à l'utilisateur de décrire la langue en fournissant au système des relations de dépendances munies de poids. Nous nous sommes donc définis un tableau à trois dimensions A (n,n,p) où n est le nombre de catégories, et p le nombre maximum de relations différentes entre deux catégories.

Exemple: si pour la phrase *le petit chien noir mange la soupe*, le linguiste nous a fourni la structure suivante:



le tableau aura la forme suivante:

gouverneur / dépendant	phrase	verbe	substantif	article	adjectif
phrase					
verbe	1				
substantif		— 50 + 60			
article			— 16		
adjectif			— 14 + 14		

Nous donnons maintenant les points principaux de l'analyseur en terminant par un exemple qui permettra de déterminer la philosophie de la méthode.

Soit une phrase $X_1\ X_2$ ..... X_n où X_i est une catégorie syntaxique.

On extrait du tableau précédent A les informations propres à cette phrase.

	phrase	X_1	X_2	----------	X_n
phrase	0				
X_1	1	0	P_{12}		P_{1n}
X_2		P_{21}	0		P_{2n}
, , , , , , , , ,					
X_n	1	P_{n1}	P_{n2}		0

Pij: ensemble des poids entre le dépendant X_i et le gouverneur X_j.

Propriétés de ce tableau:

a) $P_{ii} = O$

b) Étant donné que nous ne traitons que des structures de dépendances projectives, nous avons les propriétés suivantes:

*b*1 $\forall_i$, $\forall_j$, avec $i > j$, si l'ensemble P_{ij} n'est pas vide, il ne peut être constitué que de poids strictement positifs.

*b*2 $\forall_i$, $\forall_j$, avec $i < j$, si l'ensemble P_{ij} n'est pas vide, il ne peut être constitué que de poids strictement négatifs.

Le premier pas de l'algorithme consiste donc en la suppression des poids positifs ou des poids négatifs des ensembles P_{ij} suivant que $i < j$ ou $i > j$.

Etude de plusieurs dépendants à droite d'un gouverneur.

Etant donné qu'il n'y a aucune relation entre les dépendants à droite et les dépendants à gauche, nous nous posons le problème suivant: soit X_j un gouverneur, et X_{i1} X_{i2} ..... X_{in} n dépendants possibles de X_j avec

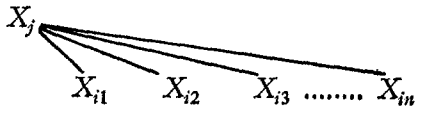
$$X_j < X_{i1} < X_{i2} \text{ ..... } < X_{in}$$

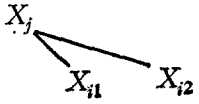
Nous avons d'abord la solution évidente d'un seul dépendant. On les prendra donc successivement les uns après les autres pour savoir si l'on peut construire une telle structure.

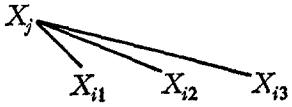
Ensuite, il faudra examiner la possibilité de deux dépendants, puis de trois, etc. ... jusqu'à n dépendants.

Nous allons donner maintenant l'algorithme qui permet de déterminer si n dépendants sont possibles.

Soient P_{i1j}, P_{i2j} ..... P_{inj} l'ensemble des poids correspondant à



pour obtenir  il faut trouver un poids de l'ensemble

P_{i2j} qui soit supérieur ou égal à un poids de P_{i1j}: soit P_{i2kj} ce poids. De même, pour que  existe, il faut trouver

un poids de P_{i3j} qui soit supérieur ou égal à P_{i2kj}, etc. ... Nous avons donc défini l'algorithme suivant:

soit M_{i1j} l'élément minimum de P_{i1j}

$$M_{i1j} = MIN\ (P_{i11j},\ P_{i12j} \ldots\ldots\ P_{i1ni1j})\ P_{i1kj} \in P_{i1j}$$

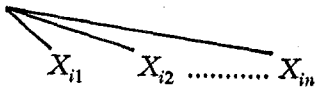
Appelons M_{i2j} l'élément de P_{i2j} qui est tel que

a) $M_{i2j} \geqslant M_{i1j}$

b) Il n'existe pas d'éléments $P_{i2kj} \in P_{i2j}$ tels que: $M_{i2j} > P_{i2kj} \geqslant M_{i1j}$

De même, on définit les éléments M_{i3j}, M_{i4j} ..... M_{inj}:

$$M_{i1j} = MIN\ (P_{i11j},\ P_{i12j} \ldots\ldots\ P_{i1ni1j})\ P_{i1kj} \in P_{i1j}\ M_{ikj} \geqslant M_{ik\text{-}1j}$$

et il n'existe pas d'éléments $P_{ilkj} \in P_{ilj}$ tels que: $M_{ikj} > P_{ilkj} \geqslant M_{ik\text{-}1j}$ avec $2 \leqslant k \leqslant n$. On peut donc construire 

si on a pu trouver les éléments M_{i1j}, M_{i2j} ..... M_{inj}.

Pour les dépendants à gauche, il est facile de voir qu'on est amené à appliquer le même algorithme.

Le principe de l'algorithme est donc le suivant: prendre un gouverneur, étudier ses dépendants à gauche, construire toutes les structures possibles en allant toujours prendre des dépendants à gauche et ceci récursivement, ensuite remonter en prenant des dépendants à droite jusqu'à obtenir toutes les solutions possibles.

Exemple:

soit la phrase: *le petit chien noir mange la soupe chaude.*

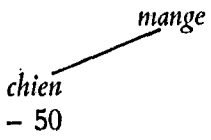
En fonction des relations stockées en mémoire, on extrait le tableau associé à cette phrase:

	phrase	*le*	*petit*	*chien*	*noir*	*mange*	*la*	*soupe*	*chaude*
phrase	0								
le		0		— 16				— 16	
petit			0	— 14 + 14				— 14 + 14	
chien				0		— 50 + 60			
noir				— 14 + 14	0			— 14 + 14	
mange	1					0			
la				— 16			0	— 16	
soupe						— 50 + 60		0	
chaude				— 14 + 14				— 14 + 14	0

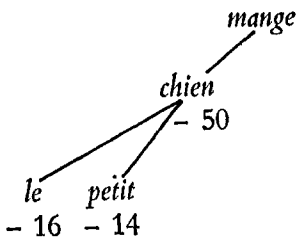
1) Enlever les poids négatifs en dessous de la diagonale puis les poids positifs au-dessus. On obtient:

	phrase	*le*	*petit*	*chien*	*noir*	*mange*	*la*	*soupe*	*chaude*
phrase	0								
le		0		— 16				— 16	
petit			0	— 14				— 14	
chien				0		— 50			
noir				+ 14	0			— 14	
mange	1					0			
la							0	— 16	
soupe						+ 60		0	
chaude				+ 14				+ 14	0

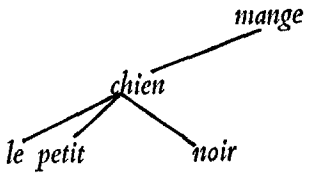
2) prendre le gouverneur *mange* et étudier ses dépendants à gauche. On trouve *chien*.



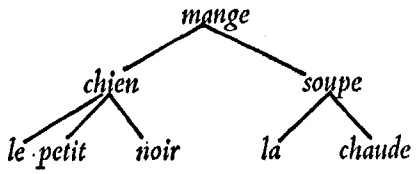
— Étudier les dépendants à gauche de *chien*.
Nous avons trois possibilités: *le*, *petit*, ou *le petit*.
Il n'y a que le couple *le petit* qui convienne



— Étude des dépendants à droite de *chien*: on trouve *noir*.



— A gauche de mange, c'est terminé: donc étude des dépendants à droite.
— Il est facile de voir que l'on obtient:



Comme nous l'avions annoncé, nous précisons que l'analyse syntaxique a été scindé en deux niveaux:

1) étude des relations entre les catégories
2) étude des variables morphologiques.

L'algorithme de dépendances fait appel à une grammaire *context-free* pour vérifier les variables chaque fois que l'on a déterminé une sous-structure terminale à gauche ou à droite. Par exemple, dès que l'on a trouvé 

on fait appel à la *context-free* qui a pour but de vérifier la concordance des variables.

Nous venons de définir les grandes lignes de l'analyseur de dépendances. Ce programme est général et indépendant des applications. Afin d'améliorer l'efficacité de l'algorithme, on peut d'ores et déjà se rendre compte qu'il sera intéressant de:

— déterminer les catégories qui ne sont jamais gouverneurs: article, adjectif, etc... afin d'éviter une recherche inutile de dépendants.

— Déterminer une priorité pour les gouverneurs principaux de la phrase. Par exemple, quand il y a présence d'un verbe, exclure les substantifs (qui sont gouverneurs dans le cas d'un titre qui n'a pas de verbe).

— Etc. ...

D'autre part, dans des cas très particuliers tels que la conjonction, il nous semble nécessaire de déterminer un algorithme spécial qui serait déclenché automatiquement par le programme principal.

L'expérience en ce domaine sera notre meilleur guide.

ANNEXE 1

```
edits
EXECUTION BEGINS ...
EDITS
$morphologie texte stop nbavard
<les poules du couvent couvent.
LES                    1 0 CL3   ARD   FIM  PLU MAS FEM
LES                    2 0 CL4   POP   FIM  PLU ACC MAS FEM
"go

13
```

```
POULE                   0 0 RIB   SBC   FEM
      S                 1 0 E8    SBC   FIM  PLU  FEM
"go
DU                      1 0 RIB   APE   FIM  SIN  MAS
DU                      2 0 CL0   COT   FIM  SIN  MAS
"go
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
COUV                    0 0 RIB   VRB
      ENT               2 0 PR4   VRB   FIM  TRE PLU PRE IND
      ENT               3 0 SU1   VRB   FIM  TRE PLU PRE SUB
"go
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
COUV                    0 0 RIB   VRB
      ENT               2 0 PR4   VRB   FIM  TRE PLU PRE IND
      ENT               3 0 SU1   VRB   FIM  TRE PLU PRE SUB
"go
.                       1 0 RIB   POC   FIM  FIP
"go
<
$dict
EXTENSION DES CLASSES? (O/N)
- oui
> ?/couv/
/COUVENT/HOMME/        /CRAI/PEI/
/COUV/AIM/
/COU/COU
>¬/couvent/
> ?/couv/
/COUV/AIM/             /CRAI/PEI/
/COU/COU/
>¬/couv/
> ?/couv/
/COURS/NEZ/            /CRAI/PEI/
/COUR/COUR/
/COU/COU/
>

$morphologie texte nstop nbavard
<les poules du couvent couvent.
DECOUPAGE PARTIEL
"texte
LES POULES DU COUVENT COUVENT.
                  |
"/couvent/homme/
```

```
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
"indexer
LA OU LES BASES ONT ETE INDEXEES
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
"go
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
"/couvent/aiment/
COUV                    0 0 RIB   VRB
      ENT               1 0 PR4   VRB   FIM  TRE PLU PRE IND
      ENT               2 0 SU1   VRB   FIM  TRE PLU PRE SUB
"indexer
LA OU LES BASES ONT ETE INDEXEES
COUVENT                 0 0 RIB   SBC   MAS
                        1 0 E1    SBC   FIM  SIN  MAS
COUV                    0 0 RIB   VRB
      ENT               2 0 PR4   VRB   FIM  TRE PLU PRE IND
      ENT               3 0 SU1   VRB   FIM  TRE PLU PRE SUB
"go nstop?
<les poulent mangent.
DECOUPAGE PARTIEL
"bavard
POULE                   0 0 RIB   SBC   FEM
P                       0 0 RIB   VRB   MAS
"ortho
CMG poules noires
CMG texte
LES  POULES NOIRES MANGENT.
     |
CMG go
POULE                   0 0 RIB   SBC   FEM
      S                 1 0 E8    SBC   FIM  PLU FEM
P                       0 0 RIB   VRB   MAS
"go nbavard
NOIR                    0 0 RIB   ADJ   MAS
     E                  0 0 E7    ADJ   FEM
      S                 1 0 E8    ADJ   FIM  PLU FEM
"go
MANG                    0 0 RIB   VRB
      ENT               1 0 PR4   VRB   FIM  TRE PLU PRE IND
      ENT               2 0 SU1   VRB   FIM  TRE PLU PRE SUB
"go
.                       1 0 RIB   POC   FIM  FIP
"go
<
$fin
```

ANNEXE 2

print entree syntaxe

LE(PRAR) PETIT(ADJQ) CHIEN(SUBC) NOIR(ADJQ) MANGE(VERB)
LA(PRAR) SOUPE(SUBC) CHAUDE(ADJQ).

LE(PRAR) PETIT(ADJQ) CHIEN(SUBC) JAUNE(ADJQ) ET(COCO) NOIR(ADJQ)
BOIT(VERB) DE L'(DELA) EAU(SUBC) ET(COCO) MANGE(VERB) UN(ARTI)
OS(SUBC) DE(DE) POULET(SUBC).

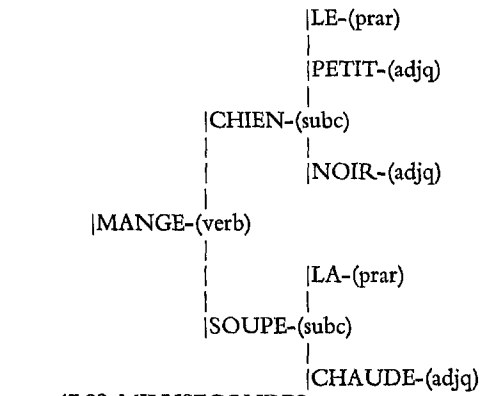
LE(PRAR) CHIEN(SUBC) QUI(PRL) EST(ETRE) JAUNE(ADJQ) ET(COCO)
NOIR(ADJQ) VEUT(VERB) ATTRAPER(INFI) L'(PRAR) OISEAU(SUBC)
QUI(PRL) SE(PPLC) TROUVE(VERB) SUR(SUR) LA(PRAR) TRES(ADV)
HAUTE(ADJQ) BRANCHE(SUBC) DU(DELA) CHENE(SUBC).

LA(PRAR) MAISON(SUBC) SUR(SUR) LE(PRAR) TOIT(SUBC) DE LAQUELLE(GPRL)
SE(PPLC) TROUVE(VERB) UN(ARTI) NID(SUBC) D'(DE) HIRONDELLES(SUBC)
N'(NE) A(AVOI) PAS(PAS) DE(DE) CHEMINEE(SUBC).

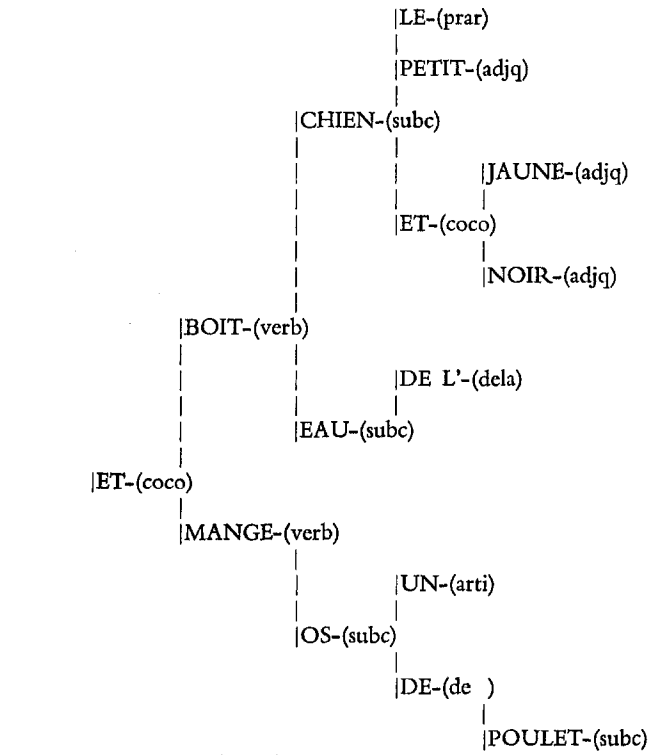
IL(PPLS) DIT(VERB) :(2.) PETIT(ADJQ) CHIEN(SUBC) ,(VIRG)
VA(VERB) DANS(DANS) TA(ADJP) NICHE(SUBC).

LE(PRAR) PILOTE(SUBC, VERB) FERME(SVA, VERB) LA(PRAR) PORTE(SUBC, VERB).

POUR(POUR) LES(PRAR) GRANDES(ADJQ) VACANCES(SUBC) ,(VIRG) PAPA(SUBC)
A(AVOI) PROMIS(PPAS) UN(ARTI) PETIT(ADJQ) VELO(SUBC) A¬(A¬) JEAN(SUBC).

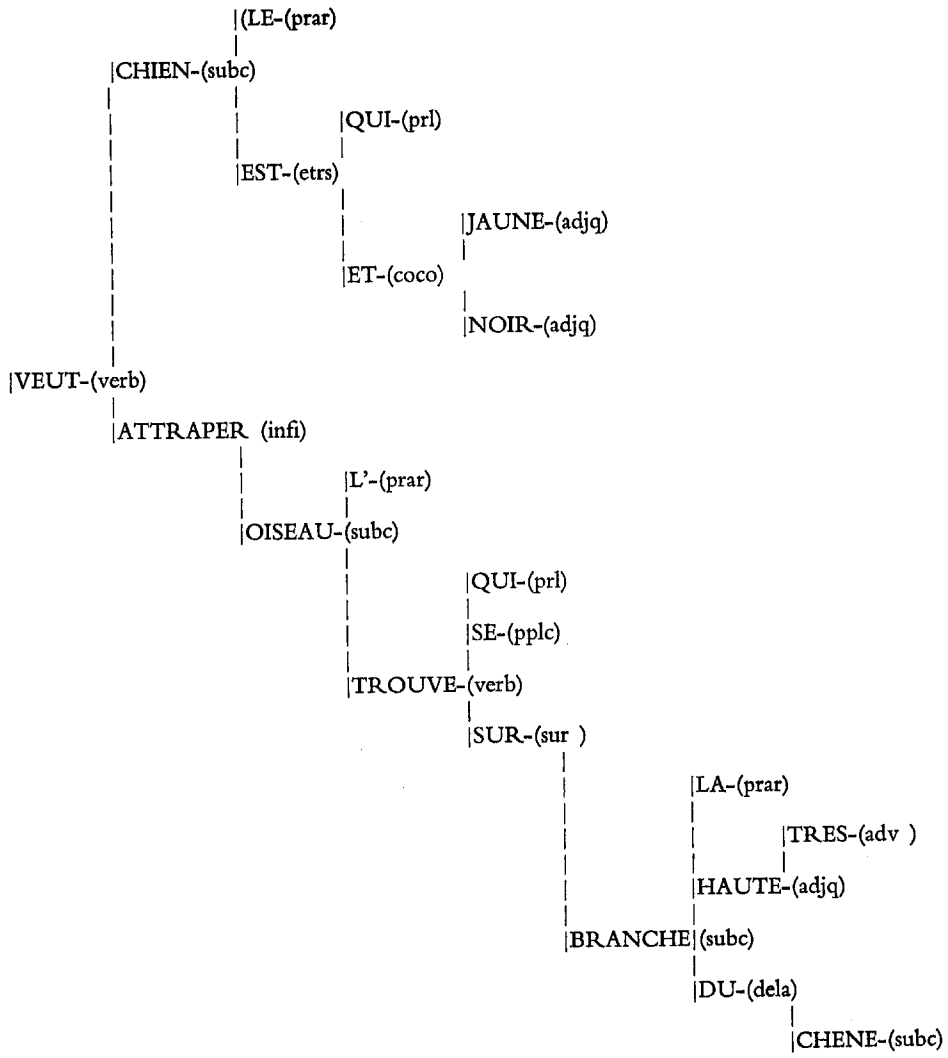


47,99 MILLISECONDES
>



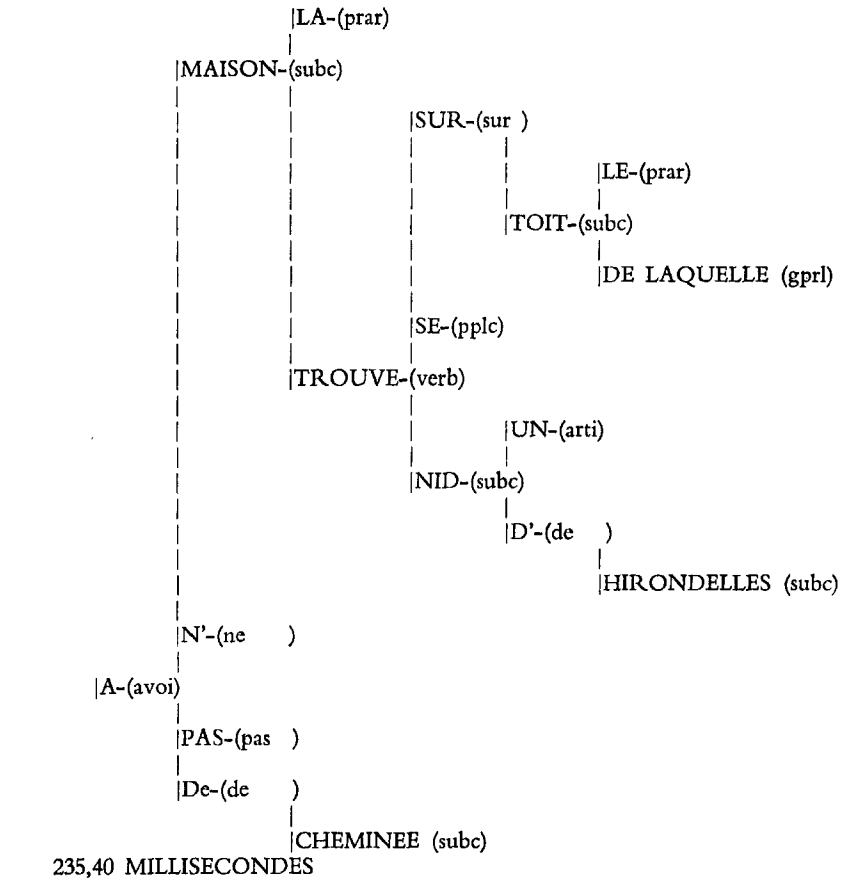
353,65 MILLISECONDES

>



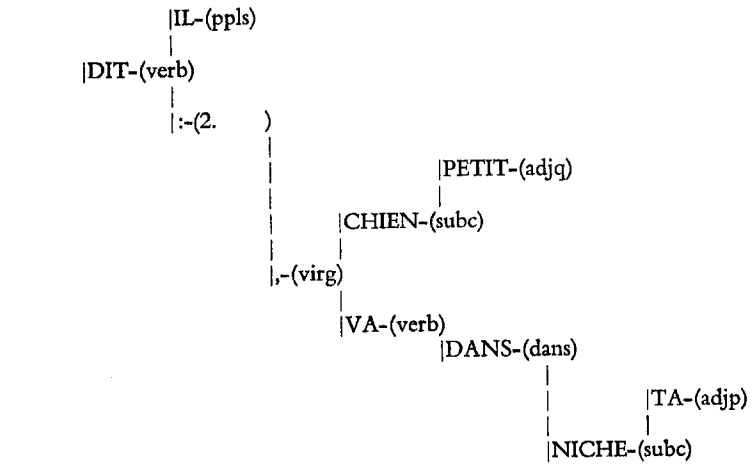
336,73 MILLISECONDES

>



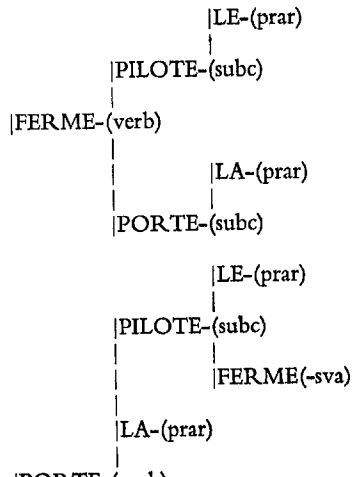
235,40 MILLISECONDES

>



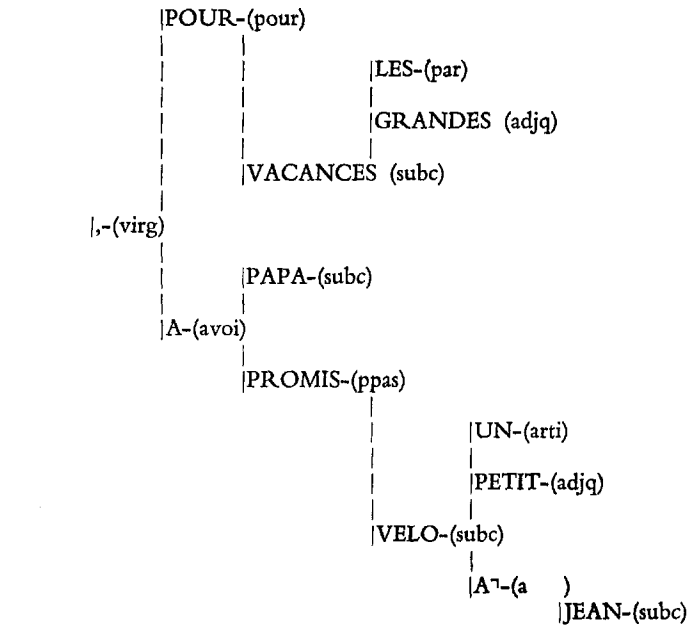
79,04 MILLISECONDES

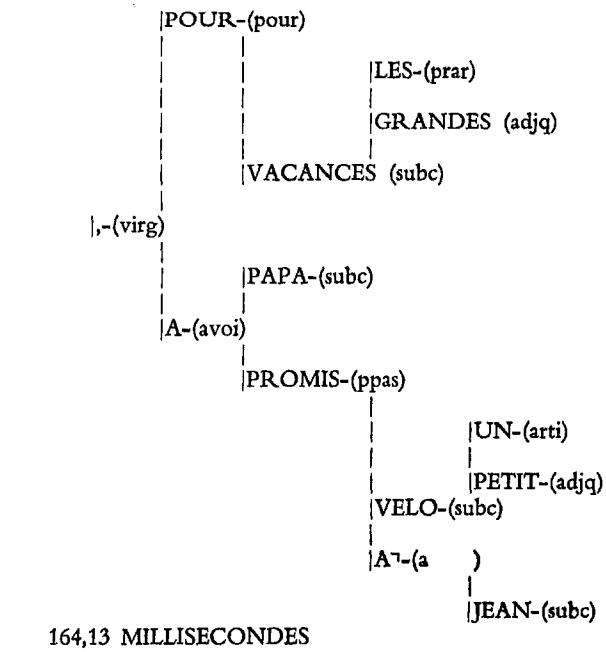
>



67,76 MILLISECONDES

>





164,13 MILLISECONDES

>

FILE: REGLES DEP P1

A¬*INFI:=16;
A¬*PPLC:=32;
A¬*SUBC:=32;
ADJQ*ADV:=-32;
ADV*ADV:=-32;
AVEC*COCO:=32;
AVEC*PPCS:=32;
AVEC*SUBC:=32;
AVOI*A¬:=48;
AVOI*ADJQ:=32;
AVOI*COCO:=-32, 32;
AVOI*DE:=32;
AVOI*NE:=-24;
AVOI*PAS:=16;
AVOI*PPAS:=96;
AVOI*PPLS:=-32;
AVOI*SUBC:=-32, 32;
AVOS*A¬.:=48;
AVOS*ADJQ:=32;
AVOS*COCO:=-32,32;
AVOS*DE:=32;
AVOS*NE:=-24;
AVOS*PAS:=16;
AVOS*PPAS:=96;
AVOS*PPLS:=-32;
AVOS*PRL:=-48;
AVOS*SUBC:=-32,32;
COCO*ADJQ:=-32,32;
COCO*ADV:=-32,32;
COCO*AVOI:=-32,32;
COCO*AVOS:=-32,32;
COCO*DE:=-32,32;
COCO*ETRE:=-32,32;
COCO*ETRS:=-32,32;
COCO*INFI:=-32,32;
COCO*PAR:=-32,32;
COCO*SUBC:=-32,32;
COCO*SVA:=-32,32;
COCO*VERB:=-32,32;
COCO*VERS:=-32,32;
DANS*SUBC:=32;
DE*INFI:=16;
DE*SUBC:=32;
DE*SVA:=32;
DELA*SUBC:=32;
DVAN*SUBC:=32;
EN*SUBC:=32;
ETRE*ADJQ:=32;
ETRE*ADV:=16;
ETRE*AVEC:=32,48;
ETRE*COCO:=-32,32;
ETRE*DANS:=32;
ETRE*DE:=32;
ETRE*DVAN:=32;
ETRE*NE:=-24;

FILE: REGLES DEP P1

ETRE*PAS:=16;
ETRE*PPAS:=96;
ETRE*PPLC:=-8;
ETRE*PPLS:=-32;
ETRE*PRDE:=32;
ETRE*SUBC:=-32,32;
ETRE*SUR:=32,48;
ETRE*SVA:=-32;
ETRS*ADJQ:=32;
ETRS*ADV:=16;
ETRS*AVEC:=32,48;
ETRS*COCO:=-32,32;
ETRS*DANS:=32;
ETRS*DE:=32;
ETRS*DVAN:=32
ETRS*NE:=-24;
ETRS*PAS:=16;
ETRS*PPAS:=96;
ETRS*PPLC:=-8;
ETRS*PPLS:=-32;
ETRS*PRDE:=32;
ETRS*PRL:=-48;
ETRS*SUBC:=-32,32;
ETRS*SUR:=-47,32,48;
ETRS*SVA:=-32;
ILYA*DAN:=-32,48;
ILYA*PRDE:=-32;
ILYA*SUBC:=32;
INFI*A¬:=32;
INFI*ADV:=16;
INFI*AVEC:=32,48;
INFI*DANS:=32,48;
INFI*DE:=48;
INFI*DVAN:=32;
INFI*INFI:=16;
INFI*PRDE:=32;
INFI*SUBC:=32;
IYAS*DANS:=32,48;
IYAS*PRDE:=-32;
IYAS*PRL:=-48;
IYAS*SUBC:=32;
IYAS*SUR:=-48;
PAR*SUBC:=32;
PHR*AVOI:=32;
PHR*COCO:=32;
PHR*ETRE:=32;
PHR*ILYA:=32;
PHR*SUBC:=32;
PHR*VERB:=32;
PHR*VIRG:=32;
POUR*INFI:=16;
POUR*SUBC:=32;
PPAS*ADJQ:=16;
PPAS*A¬:=32,48;
PPAS*ADV:=-16,16;

VERB*2.:=32;
VERS*A¬:=32;
VERS*ADV:=16;
VERS*AVEC:=32,48;
VERS*COCO:=-32,32;
VERS*DANS:=32,48;
VERS*DE:=32;
VERS*DVAN:=32;
VERS*INFI:=96
VERS*NE:=-24;
VERS*PAS:=16;
VERS*POUR:=32;
VERS*PPCS:=-32;
VERS*PPLC:=-8;
VERS*PPLS:=-32;
VERS*PRAER:=-16;
VERS*PRDE:=32;
VERS*PRL:=-48;
VERS*SUBC:=-32,32;
VERS*SUR:=-48,32,48;
VERS*2.:=48;
VIRG*AVEC:=-32
VIRG*AVOI:=-32,32;
VIRG*DANS:=-32;
VIRG*DVAN:=-32;
VIRG*ETRE:=-32,32;
VIRG*ILYA:=32;
VIRG*POUR:=-32;
VIRG*PRDE:=-32;
VIRG*SUBC:=-32;
VIRG*VERB:=-32,32;
2.*AVOI:=32;
2.*ETRE:=32;
2.*VERB:=32;
2.*VIRG:=32;
2.*VERS:=32;

PPAS*AVEC:=32,48;
PPAS*COCO:=32;
PPAS*DANS:=48;
PPAS*DE:=48;
PPAS*DELA:=47;
PPAS*INFI:=96;
PPAS*PAR:=32;
PPAS*PRDE:=32,47;
PPAS*SUBC:=32;
PPAS*SUR:=32,48;
PPAS*SVA:=42;
PRDE*PPCS:=32;
PRDE*PPLC:=32;
PRDE*SUBC:=32;
SUBC*A¬:=32;
SUBC*ADJP:=-32;
SUBC*ADJQ:=-16,16;
SUBC*ARTD:=-32;
SUBC*ARTI:=-32;
SUBC*AVOS:=32;
SUBC*COCO:=32;
SUBC*DE:=32;
SUBC*DELA:=-32,32;
SUBC*EN:=32;
SUBC*ETRS:=32;
SUBC*GPRL:=96;
SUBC*IYAS:=32;
SUBC*PAR:=32,48;
SUBC*POUR:=32;
SUBC*PRAR:=-32;
SUBC*SVA:=-16,16;
SUBC*VERS:=32;
SUR*SUBC:=32;
SVA*ADJP:=-16;
SVA*ARTD:=-32;
SVA*ARTI:=-32;
SVA*PRAR:=-32;
VERB*A¬:=32;
VERB*ADV:=16;
VERB*AVEC:=32,48;
VERB*COCO:=-32,32;
VERB*DANS:=32,48;
VERB*DE:=32
VERB*DVAN:=32;
VERB*INFI:=96;
VERB*NE:=-24;
VERB*PAS:=16;
VERB*POUR:=32;
VERB*PPCS:=-32;
VERB*PPLC:=-8;
VERB*PPLS:=-32;
VERB*PRAR:=-16;
VERB*PRDE:=32;
VERB*SUBC:=32,32;
VERB*SUR:=32,48;

BIBLIOGRAPHIE

J. COURTIN, *Organisation d'un dictionnaire pour l'analyse morphologique*, I.R.M.A., Grenoble, 1973.

J. COURTIN, J. L. RIEU, P. SGALL, *Un métalangage pour l'analyse morphologique*, Document C.E.T.A., Grenoble, 1969.

E. GRANDJEAN, *Compilateur syntaxique; Programme d'analyse syntaxique* », Documents C.E.T.A., Grenoble.

D. G. HAYS, *Dependency theory: A formalism and some observations*, Memorandum RM-4087-PR, The Rand Corporation, 1964.

G. VEILLON, *Consultation du dictionnaire et analyse morphologique en traduction automatique* (thèse), Grenoble, 1962.

G. VEILLON, *Modèles et algorithmes pour la traduction automatique* (thèse), Grenoble, 1970.

T. WINOGRAD, *Procedures as a representation for data in a computer program for understanding natural language* (1971).