MATHÉMATIQUES ET SCIENCES HUMAINES

FRÉDÉRIQUE SEGOND
JEAN-PIERRE CHANOD

Grammaire applicative : traitement informatique de la composante morpho-syntaxique

Mathématiques et sciences humaines, tome 103 (1988), p. 23-43
<http://www.numdam.org/item?id=MSH_1988__103__23_0>

© Centre d'analyse et de mathématiques sociales de l'EHESS, 1988, tous droits réservés.

L'accès aux archives de la revue « Mathématiques et sciences humaines » (http://msh.revues.org/) implique l'accord avec les conditions générales d'utilisation (http://www.numdam.org/conditions). Toute utilisation commerciale ou impression systématique est constitutive d'une infraction pénale. Toute copie ou impression de ce fichier doit contenir la présente mention de copyright.

NUMDAM
Article numérisé dans le cadre du programme
Numérisation de documents anciens mathématiques
http://www.numdam.org/

Math. Inf. et Sci. hum. (26e année, n°103, *1988*, pp. 23-43).

GRAMMAIRE APPLICATIVE : TRAITEMENT INFORMATIQUE DE LA COMPOSANTE MORPHO-SYNTAXIQUE

Frédérique SEGOND [1] Jean-Pierre CHANOD [2]

Cet article présente nos travaux dans le cadre de la grammaire applicative universelle (G.A.U.).
Notre but est d'implanter sur machine une grammaire du français qui utilise le *formalisme des types* propre à cette grammaire.
Dans un premier temps nous situerons rapidement la grammaire applicative pour exposer ensuite les principes et les métarègles que nous proposons d'y ajouter ; enfin nous décrirons un prototype de programme.

1. LA GRAMMAIRE APPLICATIVE : UN SURVOL[3]

La grammaire applicative est une extension de la *grammaire catégorielle*. Celle-ci remonte aux années 1930 avec les travaux du logicien polonais Kazimierz Ajdukiewicz qui met au point en 1935 un système formalisé de vérification syntaxique des expressions. A la base de ce système on trouve les idées du philosophe Edmund Husserl, plus particulièrement celles qui concernent sa théorie sur les catégories de signification.

L'idée d'Ajdukiewicz est, en effet, de considérer les catégories grammaticales comme des fonctions qui s'appliquent à des opérandes pour fournir un résultat.
Il se donne deux types de base : les termes, notés *t*, qui symbolisent les syntagmes nominaux et les propositions notées *s*.

---

[1] E.H.E.S.S. / Centre scientifique I.B.M.-France

[2] Centre scientifique I.B.M.-France

[3] Pour un exposé complet sur la grammaire catégorielle se reporter à J.L. Gardies [19] et pour un exposé plus bref à Lyons [23]

A partir de ces deux types de base (nous pouvons même parler de catégories de base) il va engendrer récursivement une infinité de types dérivés (qu'Ajdukiewicz appelle foncteurs) qui vont représenter les autres catégories syntaxiques du français.
Ainsi un adjectif va être considéré comme une fonction qui agit sur un syntagme nominal (de type *t*) pour donner un syntagme nominal.

Par exemple l'adjectif "bleu" est une fonction qui agit sur le syntagme nominal "ciel" pour donner le syntagme nominal "ciel bleu". De même le verbe intransitif "dort" est une fonction qui agit sur le syntagme nominal "l'enfant" pour former la proposition "l'enfant dort".

La notation adoptée par Ajdukiewicz pour représenter les foncteurs, est celle des fractions formelles avec la simplification des fractions.
Nous donnerons ici un exemple afin que le lecteur puisse se faire une idée du mécanisme de simplification formelle.

Exemple :

Les codages sont les suivants :[4]

bleu: t/t
ciel: t
l': t/t
enfant: t
dort: s/t

Les règles de simplification sont les suivantes :

u * v/u = v
v/u * u = v où u et v sont des types quelconques.

On forme le syntagme nominal (ciel bleu) en simplifiant les fractions selon les règles indiquées précédemment :

ciel bleu → (ciel bleu)
t * t/t = t

Avec les mêmes règles de simplification on forme la proposition (l'enfant dort).

L' enfant dort → (l'enfant) dort→(l'enfant dort)
t/t * t * s/t = t * s/t = s

L'intérêt d'un tel formalisme est de pouvoir, à l'aide d'une seule règle très simple, établir si une proposition est correcte du point de vue de la syntaxe (Ajdukiewicz disait syntaxiquement connectée ou normale).

D'autres auteurs se sont ensuite penchés sur la grammaire catégorielle. Citons les noms de Lambek et de Bar Hillel ; mais si séduisante que soit cette théorie elle doit être enrichie : c'est le propos de la grammaire applicative universelle.

La grammaire applicative apparaît dans les années soixante avec les travaux du linguiste Sébastian Konstantinovitch Shaumyan qui, partant de la distinction entre le niveau des observables et le niveau des construits, va établir ce qu'il appelle un *langage génotype*.

Ce langage génotype est un métalangage de description des opérations constitutives des langues naturelles. La grammaire catégorielle engendre un système génotype d'expressions de différents types. La grammaire applicative universelle est une extension de la grammaire catégorielle obtenue en lui adjoignant des *combinateurs* et des lois spécifiques à la théorie linguistique.
Comme en grammaire catégorielle, on trouve les deux types de base *s* et *t* à partir desquels sont construits les types dérivés appelés, en grammaire applicative, des *opérateurs*.

---

[4] Le numérateur de la fraction symbolise le résultat, le dénominateur l'argument de la fonction.

Remarquons que la notation fractionnaire étant intrinsèquement ambiguë[5], la grammaire applicative adopte les notations suivantes :

t est un type
s est un type
Si *x* est un type et si *y* est un type, alors *Oxy* est un type.

Un opérateur est donc une séquence préfixée par O, suivie du type de son argument (ou opérande), lui-même suivi du type du résultat (résultant).
Ainsi, l'adjectif est codé Ott (au lieu de t/t), le verbe intransitif est codé Ots (au lieu de s/t), le verbe transitif est codé OtOts, etc.

Nous avons les types suivants :

type	catégories syntaxiques	exemples
Ott	adjectif, article	bon, le, la, etc.
OtOts	verbe transitif	déplacer, etc.
Ots	verbe intransitif	bondir, etc.
Ouu où u $\in$ {*Ott, Ots, OtOts*}	adverbe	très, etc.
OtOtt	transposition d'un nom en un adjectif	de (compl. du nom)
Ost	transposition d'une phrase en un terme	que (je dis que)
OOt(OtOts)Ott	pronom relatif objet	que
O(Ots)Ott	pronom relatif sujet	qui
OtO(Ots)Ots	transposition d'un terme en un pronom adverbial	avec

Bien entendu cette liste n'est qu'indicative.

Cette notation a l'avantage de mettre en valeur la notion d'application qui est à la base de la grammaire applicative. Signalons à ce propos que la grammaire applicative emprunte le formalisme de la logique combinatoire.[6] Ainsi tout opérateur sera considéré comme étant une fonction à un argument. Une fonction à n-arguments sera donc envisagée comme une composition de n-fonctions à un seul argument.[7]
En pratique, au lieu de considérer le verbe transitif comme un opérateur qui agit sur deux opérandes (le COD et le sujet), on le considérera comme un opérateur qui agit sur un opérande (le COD) pour former un opérateur qui attend un opérande (le sujet) pour former une proposition ; ceci apparaît clairement dans le codage du verbe transitif : *OtOts.*

D'autre part la grammaire applicative nous permet d'engendrer des arbres applicatifs qui mettent bien en évidence les rapports qu'entretiennent les différents mots de la phrase.

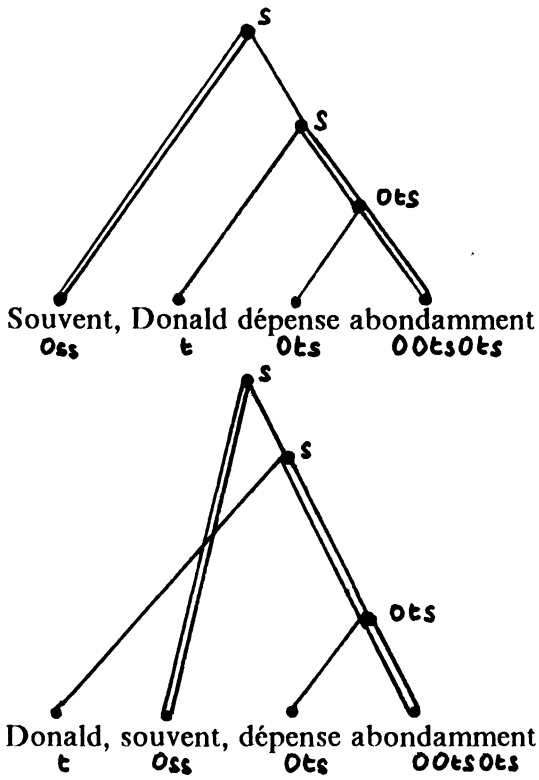
---

[5] Ainsi un adverbe qui modifie un verbe intransitif est noté, de façon ambiguë, s/t/s/t dans le formalisme d'Ajdukiewicz alors qu'il est noté sans ambiguïté OOtsOts dans le formalisme applicatif.

[6] Consulter l'article de Curry [10]

[7] Pour un exposé plus complet cf. J.-P. Desclés [12]

Donnons un exemple :[8]



1.1 La notion de combinateur :

Les combinateurs sont des opérateurs abstraits qui servent dans la composition d'opérateurs plus élémentaires. A chaque combinateur est associé canoniquement un schéma de règle de réduction qui reçoit le nom de *β-réduction* . Ce schéma définit la signification propre du combinateur en tant qu'opérateur.

Exemple :[9]
Le combinateur de composition *B* est défini à l'aide de la règle de β-réduction suivante :

"*B*XYZ→ X(YZ) où X , Y sont des opérateurs quelconques et *Z* est une entité quelconque."[10]

Ainsi, on peut dire que le prédicat "endort" est un prédicat complexe, dérivé du prédicat "dort" à l'aide du prédicat grammatical "cause", au moyen du combinateur de composition *B*. On a donc la relation suivante :
endort ≅ *B* cause dort
où le prédicat "cause" a pour type OsOts et le prédicat "dort" a pour type Ots. En appliquant la β-règle de réduction on obtient le prédicat binaire "endort" de type OtOts.

Considérons la phrase: "Jean endort Marie", et montrons comment le prédicat "endort" est dérivé du prédicat "dort".

1. (endort Marie) Jean
2. endort ≅ *B* cause dort
3. *B* cause dort Marie Jean
4. cause (dort Marie) Jean

---

[8] Les opérateurs sont symbolisés par des lignes doubles, les arguments par des lignes simples

[9] Notons qu'il existe d'autres combinateurs ; cf. Desclés [12]

[10] cf. Desclés[12]

Le type de *B* est calculé en fonction du contexte.

1.2 La notion de métarègle :

Maintenant que nous avons un métalangage pour décrire le langage naturel il va nous falloir établir, sur ce métalangage, un système de calcul qui nous permette de vérifier si une expression est ou n'est pas bien formée.
Pour ce faire, nous avions en grammaire catégorielle la simplification des fractions ; en grammaire applicative, avec les notations adoptées, nous avons la règle de simplification suivante :

y/x * x = y devient : Oxy x → y
x * y/x = y devient : x Oxy → y
où x et y sont des types quelconques

Or cette seule règle de simplification se révèle rapidement insuffisante pour diriger l'analyse. Nous avons donc besoin de trouver d'autres règles qui vont "gouverner" cette règle de simplification, en ce sens qu'elles vont nous permettre de restreindre son emploi, de le préciser. Ces nouvelles règles reçoivent le nom de *métarègles*.

On voit donc que la grammaire applicative est une grammaire catégorielle à laquelle on a ajouté des métarègles et des combinateurs.
On peut dire que le but de la grammaire applicative est de restreindre le système génotype à l'aide de lois (restrictions) qui sont exprimées avec les combinateurs.

Dans la suite de cet article nous proposons un ensemble de métarègles : métarègles syntaxiques dans un premier temps, morphologiques ensuite.
Elles visent d'une part à enrichir les informations données par les types, d'autre part à fournir une interprétation globale des relations impliquées par le langage génotype.
Notons que le souci de réaliser un prototype de grammaire applicative sur machine a guidé notre réflexion.

2. ANALYSE SYNTAXIQUE : METAREGLES :

2.1 Métarègle d'ordre des arguments :

La relation opérateur/opérande, telle que nous l'avons présentée jusqu'à présent, ne prend pas en compte l'ordre des mots.
Or, il convient de pouvoir représenter des informations telles que :

- l'adjectif attend un terme indifféremment à droite ou à gauche,
- l'article attend un terme à droite,
- le verbe intransitif attend un terme à gauche (le sujet),
- le verbe transitif attend un terme à droite (le COD), puis un terme à gauche (le sujet).

On procède de la façon suivante :

1. On code dans le dictionnaire les informations relatives à l'ordre : le type d'un opérateur est suivi, dans le dictionnaire, de l'indication de la position que doit occuper chacun de ses arguments. Pour chacune des catégories syntaxiques que nous venons d'énumérer on trouve :

   chien : t
   gigantesque : Ott, droite ou gauche
   le : Ott, droite
   bondir : Ots, gauche
   parcourir : OtOts, droite, gauche

2. On applique au cours de l'analyse syntaxique la métarègle d'ordre, qui consiste à vérifier dans une relation opérateur/opérande la compatibilité de l'ordre des mots avec les contraintes qui leur sont imposées par le dictionnaire.

   Ainsi sans la métarègle d'ordre on obtient la simplification suivante :

   chien le ----------> * (chien le)
   t Ott t

   Avec la métarègle d'ordre, l'analyseur n'effectuera aucune simplification :

   chien le
   t (Ott,d)

Il est clair que cette seule métarègle ne peut rendre compte de toutes les questions liées à l'ordre des mots, et qu'il faudra, pour traiter de cas tels que l'interrogation, le passif, etc., introduire d'autres métarègles utilisant des combinateurs.[11]

2.2 Métarègle de comptage des arguments :

La métarègle de comptage est une méthode pour éliminer, d'entrée de jeu, des choix de types qui de manière certaine ne conduiront pas à une analyse syntaxique correcte.

Règle de comptage :

Un même mot peut être associé à différents types, selon la catégorie grammaticale qui lui est attribuée. Par exemple le mot "ferme" peut être nom, adjectif, verbe transitif ou intransitif et respectivement associé aux types *t, Ott, OtOts, Ots.* Il en résulte qu'à l'échelle d'une même phrase il existe plusieurs combinaisons possibles de types, dont certaines n'ont, indépendamment des règles de simplification et des métarègles, aucune chance de produire une expression normale de type *s*.

La métarègle de comptage permet de prévoir, sans effectuer de simplification d'opérateur, quelles sont, parmi toutes les combinaisons de types possibles pour une phrase donnée, celles qui sont susceptibles de produire des expressions normales de type *s*. On peut donc rejeter d'emblée certaines combinaisons de types, sans avoir à effectuer la moindre analyse syntaxique. En revanche, si la règle de comptage assure qu'une combinaison choisie pour les types est susceptible de produire une expression normale de type *s*, il reste bien évidemment à poursuivre l'analyse syntaxique pour s'en assurer.

---

[11] Consulter, pour le passif et les réflexifs l'ouvrage de DESCLES-SHAUMYAN-GUENTCHEVA [17]

La métarègle de comptage s'exprime au moyen de deux fonctions de comptage, $compt_t$ et $compt_s$, définies comme suit[12]:

$compt_t(t) = 1$

$compt_t(s) = 0$

$compt_t(Oxy) = compt_t(y) - compt_t(x)$ où x et y sont des types.

$compt_t(A\ B) = compt_t(A) + compt_t(B)$ où A et B sont des séquences de types.

De même :

$compt_s(s) = 1$

$compt_s(t) = 0$

$compt_s(Oxy) = compt_s(y) - compt_s(x)$ où x et y sont des types.

$compt_s(AB) = compt_s(A) + compt_s(B)$ où A et B sont des séquences de types.

Pour une phase donnée, la métarègle de comptage détermine les combinaisons (ou séquences) de types $T_1\ T_2 \ldots T_n$ pour lesquelles :

$compt_t(T_1\ T_2 \ldots T_n) = 0$
et
$compt_s(T_1\ T_2 \ldots T_n) = 1$

Dans le cadre de la métarègle de comptage, les ambiguïtés associées à la notation catégorielle d'Ajdukiewicz n'ont aucun effet, puisqu'on ne considère pour chaque opérateur que son "bilan", en termes d'opérandes et de résultants de types s ou t, sans se soucier des contraintes associées au mécanisme de simplification des opérateurs.
Dans le formalisme d'Ajdukiewicz, la métarègle de comptage revient donc à effectuer des simplifications de fraction entre les t et entre les s, selon les règles usuelles de l'arithmétique, comme dans l'exemple suivant :

le	petit	chien	dort	
t/t	* t/t	* t	* s/t =	t * t * t * s / t * t * t = s

Il peut paraître avantageux, pour la commodité de la lecture, de revenir à cette notation dans la présentation des exemples qui suivent. Considérons la phrase : "la belle ferme le voile" et examinons les entrées du dictionnaire :

la : t/t, s/t / s/tt	Ott, OOtOtsOts
belle : t/t, t	Ott, t
ferme : t/t, t, s/t, s/tt	Ott, t, Ots, OtOts
le : t/t, s/t / s/tt	Ott, OOtOtsOts
voile : t, s/t	t, Ots

L'analyseur doit envisager 64 configurations résultant des différentes combinaisons possibles pour le type de chacun des mots.
La règle de comptage des arguments permet de n'en retenir que 10 susceptibles de produire une expression de type s après simplification.
Ce sont les suivants :

la	belle	ferme	le	voile
OOtOtsOts	t	Ott	Ott	OtOts

la	belle	ferme	le	voile
OOtOtsOts	Ott	t	Ott	OtOts

---

12 On trouve une règle analogue dans l'article de Van Benthem [24]

la	belle	ferme	le	voile
OOtOtsOts	Ott	Ott	OOtOtsOts	OtOts
la	belle	ferme	le	voile
OOtOtsOts	Ott	OtOts	Ott	t
la	belle	ferme	le	voile
Ott	t	t	Ott	OtOts
la	belle	ferme	le	voile
Ott	t	Ott	OOtOtsOts	OtOts
la	belle	ferme	le	voile
Ott	t	OtOts	Ott	t
la	belle	ferme	le	voile
Ott	Ott	t	OOtOtsOts	OtOts
la	belle	ferme	le	voile
Ott	Ott	Ots	Ott	t
la	belle	ferme	le	voile
Ott	Ott	OtOts	OOtOtsOts	t

Parmi ces 10 séquences de types pour lesquelles $compt_s = 1$ et $compt_t = 0$, certaines, bien sûr, ne produiront pas d'expression normale. De fait, seules trois de ces séquences seront réduites à des formes normales par l'analyseur syntaxique (cf. ultérieurement les métarègles de priorité). L'intérêt de la métarègle de comptage est toutefois d'avoir éliminé, à moindre coût, bon nombre de séquences incorrectes, sans même effectuer d'analyse syntaxique. C'est là une parfaite illustration de l'intérêt du langage génotype.

Pour souligner l'intérêt de cette métarègle on peut la comparer avec une approche plus traditionnelle : les ATN.[13] Disposant d'une grammaire qui utiliserait des ATN nous serions obligés d'envisager d'une part toutes les catégories possibles pour chacun des mots et d'autre part pour chacune de ces catégories nous aurions plusieurs chemins à tester. Sur une phrase comportant autant d'ambiguïtés que "la belle ferme le voile" on s'aperçoit rapidement que la métarègle de comptage des arguments représente un gain de temps considérable.

2.3 Métarègles de priorité associées aux opérateurs :

Considérons la proposition suivante :

Le léopard bondit
Ott t Ots

On sait que "le" (Ott) attend un terme à droite, et que "bondit" (Ots) attend un terme à gauche ; "léopard" est donc attendu à droite et à gauche.
Nous avons donc deux analyses possibles :

1er essai :

le léopard bondit ----------> le (léopard bondit)
Ott t Ots Ott s

L'analyseur ne termine pas.

2ème essai :

Le léopard bondit -----> (le léopard) bondit-----> (le léopard bondit)
Ott t Ots . t Ots s

---

[13] cf. Woods [28]

Pour lever un tel type d'ambiguïté[14] nous allons introduire des métarègles qui vont nous indiquer quel est l'opérateur qui agira en priorité. Cette démarche repose sur l'hypothèse que l'information fournie par les types peut déterminer une stratégie d'analyse syntaxique. Pour ce faire nous avons dû opérer une classification des opérateurs.
La terminologie suivante est inspirée de Bar Hillel ; nous l'avons affinée et enrichie.

Définitions : Opérateurs endotypes et exotypes :

1. Un opérateur *endotype unaire relativement à u* a pour type *Ouu* où u est un type quelconque.
   Exemple :

   chien *jaune*
   t *Ott*
   parler *sérieusement*
   Ots *OOtsOts*

2. Un opérateur *endotype composite relativement à u* a pour type *OuOxu* où x est un type quelconque.
   Exemple :

   chapeau *de* paille
   t *OtOtt* t
   le chat *que* voit
   Ott t *OOtOtOtsOtt* OtOts

3. Un opérateur *exotype relativement à u* a pour type *Oux* où x ≠ u et x ≠ Oyu.
   Exemple :

   mange la soupe
   OtOts t
   je *bondis*
   t *Ots*

Ces définitions étant posées, nous allons établir une *hiérarchie* entre les simplifications.
Le principe en est le suivant : *les opérateurs les plus puissants opèrent en priorité*, la puissance des opérateurs étant définie comme suit :
les opérateurs endotypes unaires sont plus puissants que les opérateurs endotypes composites, eux-mêmes plus puissants que les opérateurs exotypes.
Enonçons ce principe sous forme de métarègles.

- Métarègle de priorité 1 :

  Si OP1 est endotype unaire relativement à u, et plus puissant que OP2 alors on applique d'abord la simplification de OP1 sur le terme de type u.
  Autrement dit :
  Si OP1 est de type Ouu et $OP1_u > OP2_u$ alors:
  OP1 u OP2 se simplifie en :(OP1 u) OP2

- Métarègle de priorité 1bis : (cette métarègle est une simple convention)

  Si OP1 et OP2 sont endotypes unaires, alors on applique d'abord la simplification de OP2 sur le terme de type u.
  Autrement dit :
  Si OP1 est de type Ouu et OP1 = OP2 alors :
  OP1 u OP2 se simplifie en : OP1 (u OP2)

---

[14] Cette ambiguïté dans la simplification peut se présenter sous la forme Oxy x Oxz où x, y, z sont des types quelconques.

- Métarègle de priorité 2 :

  Si OP1 est endotype composite relativement à u, et plus puissant que OP2 alors on applique d'abord la simplification de OP1 sur le terme de type u.
  Autrement dit :
  Si OP1 est de type OuOxu et $OP1_u > OP2_u$ alors :
  OP1 u OP2 se simplifie en : (OP1 u) OP2

- Métarègle de priorité 3 :

  Si OP1 est exotype relativement à u, et plus puissant que OP2 alors on applique d'abord la simplification de OP1 sur le terme de type u.
  Autrement dit :
  Si OP1 est de type Oux et $OP1_u > OP2_u$ alors :
  OP1 u OP2 se simplifie en :(OP1 u) OP2

Stratégie :

La hiérarchie entre les simplifications s'établit en appliquant la première métarègle le plus longtemps possible, puis la deuxième, ou si elle même ne s'applique pas, la troisième, et en revenant à la première dès que possible.

Métarègles de priorité : exemples

Le	léopard	bondit
Ott	t	Ots

On repère l'opérateur le plus puissant ; c'est Ott (endotype unaire relativement à t) ; c'est donc lui qui va opérer en premier. Il attend un terme à droite ; il opère donc sur le terme léopard pour donner (le léopard).

(le léopard)	bondit
t	Ots

Il ne nous reste qu'un opérateur exotype Ots ; il opère donc sur t (le léopard) puisqu'il attend un terme à gauche pour former la proposition (le léopard bondit).

Prenons un exemple plus complexe que nous permettent de traiter ces métarègles.

Le	petit	chien	que	poursuit	le	chat	gris	de	la	fille	court	vite
Ott	Ott	t	O(OtOts)OtOtt	OtOts	Ott	t	Ott	OtOtt	Ott	t	Ots	O(Ots)Ots

On repère les opérateurs les plus puissants, c'est-à-dire les opérateurs endotypes unaires (Ouu) et on applique les métarègles de priorité 1 et 1bis ; on utilise également la métarègle sur l'ordre des arguments sans la citer explicitement pour la clarté de la lecture.

le	(petit chien)	que	poursuit	le	(chat gris)	de	(la fille)	(court vite)
Ott	t	O(OtOts)OtOtt	OtOts	Ott	t	OtOtt	t	Ots

(le petit chien)	que	poursuit	(le chat gris)	de	(la fille)	(court vite)
t	O(OtOts)OtOtt	OtOts	t	OtOtt	t	Ots

Il n'y a plus d'opérateurs endotypes unaires. On cherche donc les opérateurs endotypes n-aires (OuOxu) et on applique la métarègle de priorité 2 , toujours en s'aidant de la métarègle sur l'ordre des arguments.

(le petit chien)	que	poursuit	(le chat gris)	(de la fille)	(court vite)
t	O(OtOts)OtOtt	OtOts	t	Ott	Ots

Un opérateur endotype unaire (Ouu) est apparu ; on applique donc à nouveau la métarègle de priorité 1.

(le petit chien) que poursuit (le chat gris de la fille) (court vite)
t O(OtOts)OtOtt OtOts t Ots

On n'a plus ni opérateur endotype unaire, ni opérateur endotype n-aire ; on a des opérateurs exotypes relativement à t et à OtOts ; on applique donc la métarègle de priorité 3.

(le petit chien) (que poursuit) (le chat gris de la fille) (court vite)
t OtOtt t Ots

On a récupéré un opérateur endotype n-aire relativement à t, on applique donc la métarègle de priorité 2, ainsi que la métarègle sur l'ordre des arguments.

(le petit chien) (que poursuit le chat gris de la fille) (court vite)
t Ott Ots

(le petit chien que poursuit le chat gris de la fille) (court vite)
t Ots

Il ne reste plus qu'à appliquer la métarègle de priorité 3 (opérateur exotype) pour récupérer la phrase correcte.

Le petit chien que poursuit le chat gris de la fille court vite.
s

On voit qu'au cours de l'exécution des métarègles, vont se former des groupements qui détermineront la structure de la phrase.

Cette idée d'introduire une hiérarchie entre les types semble également avoir une pertinence linguistique. En effet, elle traduit le fait que la détermination (symbolisée par les opérateurs endotypes) précède la prédication (symbolisée par les opérateurs exotypes) dans la constitution d'une phrase.

3. ANALYSE MORPHOLOGIQUE : PRINCIPES ET METAREGLE

Le type d'analyse dont nous avons parlé jusqu'à présent est purement syntaxique, et par là même nettement insuffisant. L'analyseur que nous avons défini détermine si une phrase est syntaxiquement correcte, c'est-à-dire si, après simplification on obtient une expression de type *s* ou *t*. Toutefois, sans analyse morphologique, on décomposera correctement, au plan de la structure, des phrases telles que :

*la petit chattes mangez le soupe

Une analyse morphologique est donc nécessaire ; elle permettra même, dans certains cas, de lever des ambiguïtés syntaxiques.

Nous avons voulu trouver un formalisme qui, d'une part, permette d'exprimer des généralités telles que : "l'adjectif s'accorde en genre et en nombre avec le mot auquel il se rapporte"... et qui, d'autre part, soit compatible avec le formalisme du langage génotype.

Nous avons donc posé le premier principe suivant :

3.1 Principe de transmission :

Le résultant d'un opérateur hérite de certains des traits morphologiques de l'opérande.

Illustrons ce principe.
Ott hérite son genre et son nombre du terme qui est son opérande :

Exemple :

solide : Ott

amitié : t , fém, sing
(solide amitié) : t, fém, sing

L'opérateur Ott (solide) s'applique à son opérande t (amitié) pour former le terme t (solide amitié) qui a hérité des traits morphologiques de genre et de nombre de l'opérande (amitié).

De même un verbe intransitif de type *Ots* hérite son genre, son nombre et sa personne du terme qui constitue son opérande (le sujet) :

bondis : Ots
je : 1pers, sing
(je bondis) : 1pers, sing

L'opérateur Ots (bondis) agit sur son opérande t (je) pour former la proposition s (je bondis) qui a hérité des traits morphologiques de l'opérande t (je).

On voit donc qu'au fur et à mesure qu'un opérateur opère sur ses opérandes, il forme des expressions progressivement enrichies du point de vue morphologique.

Cependant la première remarque qui s'impose est la suivante : les opérateurs sont eux aussi porteurs de traits morphologiques dans le dictionnaire ; ainsi, les verbes ou les adjectifs vont hériter de certains des traits morphologiques de leurs opérandes alors qu'ils possèdent déjà de tels traits. Par exemple l'adjectif "solide" figure dans le dictionnaire comme masculin/singulier ou comme féminin/singulier indépendamment du nom auquel il se rapporte.

Il convient donc d'assortir le principe de transmission des propriétés morphologiques d'un principe de compatibilité qui s'énonce ainsi :

3.2 Principe de compatibilité :
Un trait morphologique ne peut être hérité par une expression que s'il est compatible avec les traits morphologiques imposés initialement par le dictionnaire à l'opérateur autour duquel s'est formée l'expression.

Autrement dit, les informations morphologiques associées aux opérateurs par le dictionnaire sont considérées comme des contraintes imposées au mécanisme de transmission.

Reprenons les exemples précédents pour illustrer ce principe :

Exemples :

solide : Ott, fém, sing / masc, sing
amitié : t , fém, sing
(solide amitié) : t, fém, sing

On voit que l'expression finale (solide amitié) a pu hériter des traits morphologiques associés à l'opérande (amitié), uniquement parce que ces traits étaient dans le dictionnaire également associés à l'opérateur (solide) : il y a compatibilité.

bondis : Ots, 1pers, sing
bondissions : Ots, 1pers, plur.
je : 1pers, sing
(je bondis) : 1pers, sing
*(je bondissions)

De la même façon l'expression (je bondis) va hériter des traits morphologiques attachés à l'opérande (je) parce qu'ils sont également associés à l'opérateur (bondis) ; alors que l'analyseur va refuser de former l'expression (je bondissions) car les traits morphologiques associés à l'opérande (je) sont

différents de ceux qui sont associés à l'opérateur (bondissions) : il n'y a pas compatibilité.

Ces deux principes nous ont permis d'élaborer le principe suivant donnant naissance à une métarègle morphologique :
Les expressions de même type obéissent aux mêmes règles d'accord morphologique.

Exemples :

1. Les adjectifs et les articles sont codés *Ott* et ces deux catégories ont la particularité de s'accorder en genre et en nombre avec le nom auquel elles se rapportent. On voit donc qu'à des catégories distinctes, articles et adjectifs, mais de même type *Ott*, sont associées les mêmes règles d'accord morphologique.

2. Les verbes intransitifs sont codés *Ots* et ils s'accordent en genre, en nombre et en personne avec leur opérande, le sujet.

3. Les autres catégories semblent a priori ne s'accorder ni en genre, ni en nombre, ni en personne, avec leur opérande. Dans un tel cas l'expression formée conservera les traits morphologiques associés à l'opérateur dans le dictionnaire.

Cette hypothèse est en fait une justification profonde du mécanisme proposé pour la morphologie et une preuve de son adéquation au langage génotype.
Si cette hypothèse n'est pas démentie nous pourrons construire une table indiquant pour chaque type les règles de transmission.

Donnons un exemple synthétisant ce que nous venons d'énoncer :

Exemple :

je mange la soupe

Nous disposons des informations suivantes dans le dictionnaire concernant le type et les traits morphologiques de chacun des mots (on ne garde, pour illustrer notre propos, qu'une ambiguïté parmi celles proposées pour une si petite phrase).

je : t (1pers, sing)
mange : OtOts (1pers, sing)
mange : OtOts (3pers, sing)
la : Ott (fém, sing)
soupe : t (fém, sing)

On peut donc former l'expression "la soupe" qui va hériter des traits de genre et de nombre du terme "soupe", lesquels traits sont compatibles avec ceux de l'opérateur "la" tels que les définit le dictionnaire. On obtient :

je : t (1pers, sing)
mange : OtOts (1pers, sing)
mange : OtOts (3pers, sing)
la soupe : t (fém, sing)

On forme ensuite à partir de l'opérateur "mange" l'expression "mange la soupe", pour laquelle aucune transmission morphologique ne se produit (le COD d'un verbe transitif pouvant être quelconque). Les contraintes attachées à l'opérateur "mange" sont conservées par le résultant "mange la soupe". On obtient :

je : t (1pers, sing)
mange la soupe : Ots (1pers, sing)
mange la soupe : Ots (3pers, sing)

On forme ensuite l'expression "je mange la soupe", pour laquelle genre, nombre et personne sont hérités du sujet "je".

je mange la soupe : s (1pers, sing)

On remarque qu'il y a compatibilité entre les traits hérités du sujet et ceux imposés initialement à l'une des entrées du verbe "mange". On remarque également que l'héritage du genre se ramène à une opération nulle, le genre de "je" étant indéterminé.

Montrons que l'on obtient de la sorte une description très générale de la morphologie bien adaptée aux types et une justification de la distinction opérée entre les notions de transmission des propriétés morphologiques et de compatibilité des traits morphologiques, distinction qui pouvait paraître artificielle a priori.

Considérons les propositions suivantes :

chemise verte
chemise de coton

Nous allons voir que le formalisme des types va faire apparaître le fait que "verte" et "de coton" jouent le même rôle dans la phrase.

Examinons les entrées du dictionnaire :

chemise: t, fém., sing.
verte : Ott, fém., sing.
de : OtOtt, sans genre, sans nombre
coton : t, masc., sing.

"de" a le type OtOtt ; si bien que l'expression "de coton" prend le type Ott (comme "verte") et se trouve, en vertu de la règle énoncée précédemment, hériter du genre et du nombre de son opérande s'il y a compatibilité.

(de coton) : Ott, sans genre, sans nombre
chemise de coton : t, fém., sing.
chemise verte : t, fém. sing.

C'est dire que la notion de transmission morphologique associée au type *Ott* s'applique sans difficulté à "de coton", comme elle s'applique aux adjectifs et aux articles. Si ce n'est que l'expression "de coton" n'ayant de contrainte ni de genre ni de nombre, nul n'est besoin de considérer la règle de compatibilité dans ce dernier cas.

4. PROGRAMME INFORMATIQUE :

Les principes de la grammaire applicative exposés dans cet article ont été implémentés en machine sous forme d'un prototype. L'utilité de ce prototype était de tester, sur un certain nombre d'exemples, la validité des métarègles. Il ne s'agit donc pas d'un système opérationnel à grande échelle. Remarquons toutefois que c'est le souci de produire une réalisation informatique effective de la grammaire applicative qui a guidé notre réflexion et que les métarègles de priorité, si elles semblent avoir une pertinence linguistique particulièrement adaptée aux principes de la grammaire applicative, sont le produit direct de cette volonté d'informatisation.

4.1 Représentation des informations syntaxiques :

La première étape, lors de l'écriture du programme, a été de déterminer une représentation des données qui soit compatible avec les principes de la grammaire applicative et avec la nature des métarègles proposées. Le langage génotype se représente en machine naturellement au moyen de listes canoniquement associées aux différents types d'opérateurs. Si l'on adjoint les informations syntaxiques complémentaires (relatives à l'ordre des opérandes ou à la morphologie par exemple) on comprend la nécessité pour le programme de manipuler des structures de tableaux de tableaux. Comme par ailleurs les métarègles qui sont au coeur du programme (métarègles de priorité) dirigent l'analyse syntaxique de manière procédurale, APL2 nous a paru un langage tout à fait adapté à ce premier prototypage.

4.2 Représentation des types :

Les principes de base du système génotype (*t* est un type ; *s* est un type ; si *x* et *y* sont des types, alors *Oxy* est un type) ont un équivalent naturel, qui produit dans la structure interne du programme des types sous forme de tableaux numériques : (*1* est un type ; *2* est un type ; si *X* et *Y* sont des types, alors *(X Y)* est un type). Nous montrons ci-dessous quelques exemples de la correspondance qui s'établit de la sorte :

```
t      -------->  1
s      -------->  2
Ott    -------->  (1 1)
Ots    -------->  (1 2)
OtOts  -------->  (1 (1 2))
```

4.3 Codage de l'ordre des opérandes :

Nous avons signalé que la métarègle d'ordre exige que des informations relatives à l'ordre des opérandes soient codées dans le dictionnaire pour chaque mot qui est associé à un opérateur du langage génotype. De manière interne, on code l'information de la façon suivante:

- 0 signifie que l'opérande se place indifféremment à droite ou à gauche,
- 1 signifie qu'il se place à gauche,
- 2 signifie qu'il se place à droite.

Pour les entrées du dictionnaire associées à des types de base (essentiellement *t*) cette information n'est pas pertinente, étant donné que de tels mots n'agissent pas comme opérateurs.

4.4 Codage des informations morphologiques :

Les informations morphologiques (genre, nombre et personne dans le cadre de notre prototype) sont codées dans le dictionnaire numériquement (1/2 pour masculin/féminin, par exemple). Le codage est identique, que les informations morphologiques soient associées à des types de base ou à des opérateurs, bien que la métarègle de morphologie traite différemment ces deux types (cf. supra).

4.5 Structure des données (récapitulatif) :

Les entrées du mini-dictionnaire constitué pour le prototype sont structurées sous forme d'un tableau qui comporte :

1. la chaîne de caractères qui représente l'intitulé de l'expression
2. le type de l'expression
3. l'ordre droite/gauche de chacun des opérandes définis par le type
4. les indicateurs morphologiques.

Certains des éléments décrits ci-dessus peuvent être eux-mêmes représentés par des tableaux ; c'est le cas notamment des types.

Les informations relatives à un mot sont donc regroupées dans la structure suivante :

< nom> < type> < ordre> < genre, nombre, personne>

Donnons quelques exemples, pour lesquels les places vides indiquent que l'information requise est non pertinente ou indéterminée :

	< nom>	< type>	< ordre>	< genre, nombre, personne>
chien :	'chien'	(1)	( )	(1,1,3)
le :	'le'	(1 1)	(2)	(1,1,)
petit :	'petit'	(1 1)	(0)	(1,1,)
sévère :	'sévère'	(1 1)	(0)	(,1,)
mange :	'mange'	(1 (1 2))	(2 1)	(,1,3)

L'analyseur syntaxique peut réduire en cours d'exécution de telles structures, et produire des structures résultantes selon le même modèle, conformément aux diverses métarègles qu'il applique :

(le petit chien) : 'le petit chien' (1) ( ) (1,1,3)

Notons qu'il s'agit là de la structure interne au programme, très proche de l'aspect qu'elle prend effectivement en APL2. Rien n'empêche que les données soient codées plus traditionnellement et plus convivialement dans un dictionnaire qui serait mis à jour ou consulté par un utilisateur humain, et qu'un interface traduise de façon transparente ces données dans la structure interne requise par le programme.

4.6 L'analyseur syntaxique :

L'analyseur syntaxique comporte deux éléments essentiels :

1. Une procédure qui détermine la nature du lien opérateur/opérande, c'est-à-dire si l'opérateur est endotype, exotype, unaire, n-aire, etc. Cette procédure peut également opérer la simplification opérateur/opérande, c'est-à-dire créer une expression à partir de deux expressions dont l'une opère sur l'autre, et calculer les attributs de la nouvelle expression.

   Exemple :
   ? le chien
   = = > opérateur à gauche (le), endotype unaire. Morphologie OK
   = = > 'le chien' (1) ( ) (1,1,3)

Cette procédure élémentaire de simplification opérateur/opérande applique *indépendamment* :

- la règle de simplification selon le type
- la métarègle d'ordre
- la métarègle de morphologie

Il est donc possible de produire une analyse syntaxique qui relâche les contraintes d'ordre ou de morphologie, sans pour autant modifier le programme de l'analyseur.

2. Une procédure d'analyse globale. L'interprétation d'une phrase s'effectue selon une procédure qui applique la stratégie associée aux métarègles de priorité :

   - on réduit d'abord tous les couples opérateur/opérande pour lesquels l'opérateur est endotype unaire et tels que l'opérande n'est pas relié à un autre opérateur endotype unaire (métarègle de priorité 1) ;

   - puis lorsque la métarègle 1 ne s'applique plus (ni sa variante 1 bis), on réduit tous les couples opérateur/opérande pour lesquels l'opérateur est endotype n-aire et tels que l'opérande n'est pas relié à un autre opérateur endotype n-aire (métarègle de priorité 2) ;

- puis lorsque ni la métarègle 1 ni la métarègle 2 ne s'appliquent, on réduit tous les couples opérateur/opérande pour lesquels l'opérateur est exotype et tels que l'opérande n'est pas relié à un autre opérateur (métarègle de priorité 3).

Dès que la métarègle 1 s'applique à nouveau, elle est prioritaire sur les deux autres, et de même pour la métarègle 2 par rapport à la métarègle 3.

La nature des opérateurs (endotype, exotype) est déterminée par la procédure décrite précédemment pour chacun des couples d'expressions contiguës dans la phrase. Dès qu'une simplification opérateur/opérande est effectuée, il convient de réévaluer les relations entre les types créés par la simplification.

De même que pour la procédure élémentaire (simplification opérateur/opérande), les métarègles d'ordre et de morphologie sont, au cours de l'analyse globale, exécutées indépendamment des métarègles de priorité. On peut donc, de façon optionnelle, contrôler ou non la compatibilité des traits morphologiques de l'opérateur et de ses opérandes, ou l'ordre de ceux-ci dans la phrase.

Cette procédure produit une expression structurée, qui représente l'analyse la plus fine que puissent produire les métarègles de priorité. Dans les cas les plus favorables on obtient une analyse globale de la phrase. Il se peut toutefois qu'on n'obtienne qu'une simplification partielle de la phrase, c'est-à-dire que des ambiguïtés interdisent la poursuite de l'analyse selon la procédure déterminée par les métarègles de priorité. On peut alors débloquer la situation par des essais-erreurs à partir de l'analyse partielle produite par les métarègles de priorité, lesquelles, en tout état de cause, réduisent la complexité de l'analyse globale.

Le résultat produit par l'analyseur est une structure qui garde la trace des différentes simplifications opérées au cours de l'analyse. On peut choisir de n'afficher, par commodité, qu'une partie des informations recueillies en fin d'analyse, comme dans l'exemple suivant[15] :

? le chien mange l'os de poulet

= = > (le> chien) <(mange> ((l'> os) <(de> poulet)))

Il est à noter que l'introduction de nouvelles catégories grammaticales dans le mini-dictionnaire ne nécessite pas la réécriture de la procédure ou l'introduction de nouvelles règles. En effet, les règles de formation sont implicitement contenues dans le type associé aux mots du dictionnaire. Ainsi la procédure capable d'analyser :

le chat mange la soupe

peut analyser :

je dis que le chat mange la soupe de boeuf froide

grâce à la seule introduction de mots nouveaux dans le dictionnaire, avec leurs types et attributs divers.

Il est clair que ce simple prototype ne permet de prendre en compte qu'un sous-ensemble des phénomènes linguistiques, et même, qu'un sous-ensemble des principes de la grammaire applicative. En particulier, il n'a pas été tenu compte dans ce prototype :

- de la multiplicité et de l'ambiguïté des informations associées à une même entrée lexicale (ainsi le verbe "manger n'a, dans le dictionnaire que nous utilisons dans notre prototype, que l'étiquette de verbe transitif),
- de la notion de type variable qui définit la relation opérateur/opérande non pas entre types, mais entre sous-ensembles de types (nous avons été amenés à envisager la notion

---

[15] les signes > et < indiquent respectivement que l'opérande se trouve à droite ou à gauche

de type variable pour traiter des adverbes, en choisissant de les coder Ouu où u appartient à l'ensemble suivant { verbe transitif, verbe intransitif, adjectif, adverbe, phrase },

- de la notion de combinateur, qui ouvre la voie au traitement de phénomènes linguistiques importants que nous avons ici négligés (le passif, les réfléchis, etc.).

Il nous a paru intéressant d'insister ici sur le caractère procédural qui fait l'originalité de l'analyseur et qui témoigne de la richesse des informations impliquées par le système génotype. Toutefois, les développements futurs, s'ils ne doivent faire renoncer à cet aspect procédural, devront nécessairement faire appel à des aspects "déclaratifs" pour prendre en compte l'incontournable ambiguïté de la langue.

5. CONCLUSION :

Nous ne nous situerons pas ici, comme nous aurons sans doute à le faire dans la suite de nos recherches, par rapport aux grammaires de Gazdar, Montagüe, Bresnam, etc. Situons rapidement la grammaire applicative par rapport aux grammaires de Chomsky et d'Harris.
Shaumyan reproche aux grammaires transformationnelles des années 60[16] de ne pas suivre une démarche mathématique rigoureuse qui serait de concevoir dès le départ un système de calcul avec lequel on opèrerait des transformations élémentaires.[17]
D'autre part, Shaumyan reproche à l'"Harris du début"(1955) de se contenter de donner une liste de toutes les transformations à l'intérieur d'une langue, alors que la grammaire applicative engendre les transformations potentielles par un calcul. Ce calcul établit une relation entre deux ou plusieurs objets construits et mesure ensuite les absences de réalisation dans les langues naturelles.
Cependant nous pouvons dire, qu'avec le temps, Harris et Shaumyan se sont rapprochés. En effet, bien qu'ayant un système différent, Harris aboutit au même type de résultats que Shaumyan. En particulier, il nous semble important qu'ils partent tous deux d'une base catégorielle.
En effet, on trouve durant toute une période de l'histoire de la linguistique de nombreux travaux concernant les catégories grammaticales (en particulier dans la tradition des grammaires universelles) . Il semblerait qu'ensuite les linguistes se soient mis à y accorder moins d'importance. Or la tendance de la grammaire applicative universelle ainsi que celle de Harris est justement de partir de l'étude des catégories grammaticales (catégories syntaxiques et catégories formelles (aspect etc.)) pour écrire une grammaire.
Signalons également que des travaux en cours ont pour but de formaliser Harris en logique combinatoire, ce qui abonde dans notre sens.[18]

Nous voudrions mettre l'accent sur l'intérêt du formalisme de la grammaire applicative.

Précisons tout d'abord qu'il semble que la réalisation de cette maquette soit l'une des premières tentatives d'implantation sur machine du système applicatif. Rappelons , à ce propos, que l'un des intérêts du formalisme applicatif pour l'implantation d'une grammaire sur machine réside dans les faits suivants :

- Les informations syntaxiques et morphologiques sont stockées dans le dictionnaire ; en ce sens, elles sont nettement séparées des règles de simplification et des métarègles ce qui les rend susceptibles d'être modifiées à tout moment sans perturber pour autant le mécanisme de simplification.

- Les règles et les métarègles en très petit nombre, permettent déjà de décrire une partie de la langue qui nécessiterait, avec un autre formalisme, un nombre beaucoup plus élevé de règles imbriquées les unes dans les autres.

---

[16] Cf. Chomsky [8] et [9]

[17] Pour une discussion plus approfondie voir le livre de Z. Guentcheva [21]

[18] Cf. les travaux d'Anne Daladier [11] et de J.-P. Descles [16]

Remarquons que dans le système applicatif, le plus important est de trouver le "bon codage" des mots ; on voit donc qu'il y a une osmose entre ce dictionnaire et le système. Le dictionnaire n'étant plus uniquement un support lexical mais un composant indissociable du mécanisme proposé. Les règles et les métarègles sont, d'une certaine façon moins importantes, vu qu'elles ne servent qu'à diriger et à accélérer l'analyse.

Représenter à ce niveau d'abstraction les relations entre les mots, tout en produisant une analyse fine de la phrase nous permet de pouvoir prétendre à une certaine universalité.
Sur ce sujet nous renvoyons, encore une fois, le lecteur à Shaumyan[19] qui montre que des relations telles que "sujet de", "complément d'objet direct", "complément d'objet indirect", ne sont pas des concepts universels alors que les relations opérateur/opérande étant plus abstraites sont plus universelles.

Essayons de dégager plus précisément l'intérêt des métarègles syntaxiques et morphologiques que nous avons proposées dans cet article.

La maquette que nous avons présentée ici montre bien que le système génotype, au-delà de la simple relation opérateur/opérande entre deux expressions contiguës, fournit d'amples informations sur l'organisation générale de la phrase.
L'analyseur que nous proposons pourrait se résumer ainsi : un automate de simplification et une stratégie faite d'un ensemble de métarègles ordonnées.
L'analyseur détermine en fait une stratégie qui guide l'automate entre les multiples choix et ambiguïtés de la phrase: nous avons montré que, au moins dans une phase préliminaire, il est possible de conduire l'analyse de manière procédurale, en tirant parti du formalisme des types. Ceci n'exclut pas qu'à un stade ultérieur de l'analyse on puisse revenir à une stratégie de choix multiples, en relâchant les contraintes imposées par les métarègles.
Cette notion de métarègle qui nous sert à guider l'analyse, qui nous fournit une stratégie, est possible justement grâce au formalisme des types. La notion de métarègle, telle que nous la présentons n'est pas envisageable dans des systèmes utilisant des ATN.

Une des caractéristiques intéressantes du modèle proposé est que les métarègles soient indépendantes entre elles et indépendantes des informations syntaxiques contenues dans le lexique.
On obtient de la sorte une plus grande maîtrise de l'architecture du programme. Par exemple, il est tout à fait possible de conduire une analyse morphologique indépendante de l'analyse sur les types.
Remarquons cependant qu'il y a d'autres métarègles possibles et que notre travail à venir sera précisément de trouver ces métarègles ainsi qu'un meilleur codage des mots quand cela est possible.

---

[19] Article cité en bibliographie [24]

BIBLIOGRAPHIE :

[1] AJDUKIEWICZ K., *The scientific world-perspective and other essays, 1931-1933"*, Dordrecht, Riedel publishing company, 1978.

[2] ARNAUD A. et LANCELOT C., *Grammaire générale et raisonnée de Port-Royal*, 1660, Réédité par BAILLY, Slatkine reprint, 1968.

[3] BAR HILLEL Y., "A quasi arithmetical notation for syntactic description", *Language* n°29, pp.47-58, 1953, traduit in *Langage* n°9, pp. 9-22, 1968.

[4] BAR HILLEL Y., *Language and information: selected essays on their theory and application*, Reading, Adison Westley, 1964.

[5] BEAUZEE N., *Grammaire générale*, Paris, Barbou, 1767.

[6] BENVENISTE E., *Problèmes de linguistique générale*, Paris, Gallimard, 1966.

[7] CHANOD J.-P. et SEGOND F., "Introduction à la grammaire applicative" *Etude IBM*, Paris, n°F129, 1988.

[8] CHOMSKY N., *Aspects de la théorie syntaxique*, Paris, Seuil, 1965.

[9] CHOMSKY N., *Structures syntaxiques*, Paris, Seuil, 1969.

[10] CURRY H.B., "Some logical aspects of grammatical structure", *Proceedings of symposia in applied mathematics*, volume XII, 1961.

[11] DALADIER A., "Représentations applicatives d'énoncés", *Actes de l'école d'été Linguistique et Informatique*, Toulouse, 1982.

[12] DESCLES J.-P., *Sciences cognitives et grammaire applicative universelle*, Paris, Hermès, 1988, (à paraître).

[13] DESCLES J.-P., "De la notion d'opération à celle d'opérateur ou à la recherche de formalismes intrinsèques" *Mathématiques et sciences humaines* n°76, pp. 5-33, 1981.

[14] DESCLES J.-P., "Opérations constitutives d'énoncé", Linguistique et mathématiques, *Peter Lang*, pp. 123-165, 1982.

[15] DESCLES J.-P., "Systèmes transformationnels de Z. Harris: I. Construction du noyau", *T-a-informations*, n°1, pp. 3-34, 1975.

[16] DESCLES J.-P., "Systèmes transformationnels de Z. Harris: II. Eclatement du noyau", *T-a-informations*, n°2, pp. 2-26, 1975.

[17] DESCLES J.-P., GUENTCHEVA Z., SHAUMYAN S.K., *Theoritical aspects of passivization in the framework of applicative grammar*, Amsterdam, John benjamins publishing company, 1985.

[18] DU MARSAIS C., *Le traité des Tropes* ,1730. Réédition: Le nouveau commerce, (1977).

[19] GARDIES J.-L., *Esquisse d'une grammaire pure*, Paris, Vrin, 1975.

[20] GAZDAR G., "Phrase structure grammar", *The nature of Syntactic Representation*, Reidel publishing company, VOL.15, pp. 131-186, 1982.

[21] GUENTCHEVA Z., "Présentation critique du modèle applicationnel de S.K. Saumjan", *Documents de linguistique quantitative n°30*, Dunod, 1976.

[22] JESPERSEN O., *Philosophie de la grammaire*, Paris, Editions de minuit, 1971.

[23] LYONS J., *Linguistique générale*, Paris, Larousse, 1970.

[24] VAN BENTHEM J., "The Lambeck calculus", *Categorial Grammars and Natural Language Structures*, Reidel publishing company, VOL.32, 35-68, 1988.

[25] SHAUMYAN S.K., "The goals of linguistic theory and applicative grammar", *Mathématiques et sciences humaines* , n°77, pp.7-43, 1982.

[26] SHAUMYAN S.K., *A Semiotic theory of langage*, Bloomington, Indiana University Press, 1987.

[27] TESNIERE L., *Eléments de syntaxe structurale*, Paris, Klincksieck, 1959.

[28] WAGNER R.L. et PINCHON J., *Grammaire du Français*, Paris, Hachette, 1962.

[29] WOODS W.A., Transition Network Grammars for Natural Language Analysis, *Communications of ACM*, Vol. 13, n° 10, October 1970.