MATHÉMATIQUES ET SCIENCES HUMAINES

JEAN-PIERRE DESCLES

Théorème de Church-Rosser et structuration des langues naturelles

Mathématiques et sciences humaines, tome 103 (1988), p. 67-92

<http://www.numdam.org/item?id=MSH_1988__103__67_0>

Math. Inf. et Sci. hum. (26e année, n°103, *1988*, pp.67-92).

THEOREME DE CHURCH-ROSSER ET STRUCTURATION DES LANGUES NATURELLES

Jean-Pierre DESCLES[1]

1. NOTION DE FONCTION "PENSEE EN SOI".

1.1. Dans la théorie des ensembles axiomatisée dans le cadre Z-F (Zermelo-Fraenkel), la notion de fonction est dérivée de celle plus primitive d'ensemble. Une fonction f de E dans F, où E et F sont des ensembles, *est un ensemble* tel que :

(i) $f \subset E \times F$;
(ii) $\forall x : [x \in E] \Rightarrow \exists y : [y \in F] \wedge (x,y) \in f]$;
(iii) $\forall x \; \forall y \; \forall y' : [(x,y) \in f \wedge (x,y') \in f] \Rightarrow [y=y']$.

Une fonction est donc un ensemble statique de couples liés par une relation fonctionnelle ((ii) et (iii)).

Dans le cadre des langages applicatifs, la notion de fonction est première. Elle est pensée de façon dynamique sous forme d'une procédure qui associe une sortie déterminée à une entité donnée en entrée.

Nous appellerons désormais *opérateur* l'expression d'une fonction et *opérande* l'expression qui désigne un argument d'une fonction. "Fonction" sera employée pour désigner plus particulièrement l'interprétation sémantique d'un opérateur dans le cadre de Z-F.

L'*application* est une opération binaire que l'on désigne ici par ':'. Cette opération est primitive (tout comme la relation d'appartenance '$\in$' est primitive dans Z-F). L'application de l'opérateur f à l'opérande a donne pour résultat b. Désignons par '$\geq$' la relation entre l'opération d'application *à effectuer* et le résultat de l'opération *effectuée*. Nous avons :

$$f : a \geq b.$$

La lambda-notation de A. Church permet de représenter une "fonction pensée en soi" sous forme d'un opérateur. Etant donnée une expression $A(\xi)$, où ξ présente certaines occurrences, une *opération d'abstraction* permet de construire la lambda-expression '$\lambda\xi \; .A(\xi)$' qui exprime l'opérateur qui a été abstrait de l'expression $A(\xi)$. Cet opérateur représente la notion de "fonction

[1] Université de Paris-Sorbonne et Centre d'Analyse et de Mathématique Sociales, Unité Mixte CNRS-EHESS.

pensée en soi" dont le mécanisme opératoire de la procédure associée est décrit par cette expression : $A(\xi)$ décrit l'agencement de tout ce qui peut être *substitué* aux occurrences de la variable ξ.

Par exemple, la fonction "carré" est représentée par la lambda-expression suivante : '$\lambda\xi \,.\, \xi^2$' qui décrit le mécanisme opératoire de la procédure "élévation au carré".

Le nom de la variable ξ, liée par l'opération d'abstraction $\lambda\xi$, n'intervient pas dans l'expression linguistique de l'opérateur.

Lorsque nous appliquons l'opérateur '$\lambda\xi \,.\, A(\xi)$' à un opérande a, nous construisons une nouvelle expression obtenue en substituant a à chaque occurrence de ξ dans $A(\xi)$. Nous écrivons :

$$(\lambda\xi \,.\, A(\xi)) : a \geq [a/\xi]\, A(\xi)$$

où '$[a/\xi]\, A(\xi)$' désigne le résultat de la substitution de a aux occurrences de ξ dans $A(\xi)$.

Par exemple, en appliquant la fonction "carré" à l'opérande 5, nous obtenons :

$$(\lambda\xi \,.\, \xi^2) : 5 \geq [5/\xi]\, \xi^2 = 5^2 = 25.$$

Le lambda-calcul de Church précise la notion de substitution d'un opérande à une variable liée par l'opération d'abstraction.

1.2. Bien que très analogues, lambda-calcul et logique combinatoire diffèrent cependant sur certains points. En particulier, le lambda-calcul donne des règles explicites pour gérer et automatiser la substitution d'expressions à des occurrences de variables liées. En revanche, la *logique combinatoire* vise à éliminer complètement la notion de variable liée en faisant appel à des opérateurs abstraits, appelés *combinateurs*, qui offrent la possibilité de construire des opérateurs complexes à partir d'opérateurs plus élémentaires. La logique combinatoire possède de ce fait une puissance abstractive plus forte que le lambda-calcul car elle permet de mieux cerner la notion de "fonction en soi" et de définir un *calcul intrinsèque* sur des fonctions, indépendamment de tout domaine d'interprétation de ces fonctions.

1.2.1. Le linguiste S.K. Shaumyan a pensé à utiliser, dès 1965, les combinateurs de la logique combinatoire, pour une analyse des transformations interphrases dans les langues naturelles. Il a proposé un modèle, appelé *Grammaire Applicative Universelle*, qui représente les phrases d'une langue sous forme d'expressions applicatives, c'est-à-dire sous forme d'agencements combinatoires d'opérateurs et d'opérandes de différents types.

1.2.2. On sait que la sémantique des langages de programmation fait de plus en plus appel aux formalismes applicatifs (lambda-calcul ou logique combinatoire). Un langage de programmation comme LISP est une variété de langage applicatif qui emprunte certains des mécanismes du lambda-calcul de Church. Un nouveau style de programmation - la programmation fonctionnelle - émerge depuis les propositions de J. Backus conduisant à une "programmation sans variables" fondée sur les formalismes applicatifs de la logique combinatoire.

1.2.3. Dans le domaine des représentations des connaissances, nous avons proposé de recourir à des formalismes applicatifs pour représenter les significations de prédicats linguistiques au moyen de combinateurs qui agencent des primitives sémantiques entre elles.

1.2.4. Reprenant l'analyse de la négation propositionnelle effectuée par J. Piaget, L. Frey en a donné une "version combinatoire" qui éclaire très certainement la nature cognitive et opératoire de la négation.

1.3. Tous ces domaines soulignent l'importance d'un calcul intrinsèque sur des fonctions. Le pouvoir d'expressivité conceptuelle des formalismes applicatifs vient de l'abstraction qui est attachée à la structure applicative mettant en jeu des opérateurs appliqués à des opérandes. L'effectivité conduit à des utilisations fécondes dans des domaines aussi variés que les langages de programmation, les langues naturelles, les opérations cognitives ; elle prend appui sur la propriété de Church-Rosser que possèdent la logique combinatoire et le lambda-calcul.

Nous allons dégager quelques conséquences du théorème de Church-Rosser pour la linguistique, en particulier pour l'étude des catégories grammaticales, des opérations paraphrastiques et les structurations qu'elles entraînent.

2. EXPRESSIONS APPLICATIVES.

On se donne un ensemble d'*atomes* comprenant des atomes constants et éventuellement des variables.

La classe des expressions applicatives (e.a.) est définie récursivement par :
(1) *les atomes sont des e.a. ;*
(2) *si* X *et* Y *sont des e.a., alors* (XY) *est une e.a.*

En représentant l'opération d'application par la juxtaposition, l'expression '(XY)' représente l'application de X à Y, où X joue le rôle d'un opérateur et Y le rôle d'un opérande.

Nous adoptons quelques simplifications d'écriture :
(i) *Si* (XY) = (UV) *alors* $X \equiv U, Y \equiv V$;
(ii) $(XYZ) \equiv ((XY)Z)$ (simplification à gauche) ;
(iii) $XY \equiv (XY)$

Nous en déduisons que :
$XY_1 \ldots Y_n \equiv (\ldots ((XY_1)Y_2) \ldots Y_n)$
$X(Y_1Y_2)Z \equiv ((X(Y_1Y_2))Z) \neq (((XY_1)Y_2)Z) \equiv XY_1Y_2Z$

Remarque : Nous distinguons l'identité '$\equiv$' entre e.a. et l'égalité '=' qui est une congruence entre e.a. équivalentes.

3. EXPRESSIONS APPLICATIVES TYPEES.

Aussi bien en logique, qu'en linguistique et en informatique, on a besoin de manipuler différents types d'expressions. En linguistique, on distingue les termes nominaux, les prédicats, les déterminants, ... ; en logique, on manipule des termes, des prédicats et des propositions ; en informatique, on a des chaînes de caractères, des nombres entiers, des nombres réels, des valeurs booléennes...

Les types que nous allons considérer, seront des *types fonctionnels* engendrés à partir d'un ensemble S de sortes. Ils sont engendrés par les règles suivantes :
(3) *les sortes de* S *sont des types (dits élémentaires) ;*
(4) *si* x *et* y *sont des types, alors* 0xy *est un type, dit fonctionnel.*

Le symbole "0" est un opérateur binaire qui sert à former des types fonctionnels à partir de types déjà définis.

Une classe d'e.a. ayant des types assignés, la classe des *expressions applicatives typées* (e.a.t.) est récursivement engendrée par les règles suivantes :

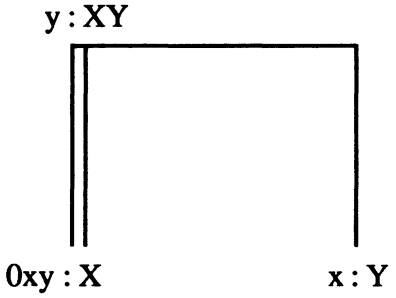
(5) *les e.a. ayant des types assignés sont des e.a.t. ;*

(6) *si* X *est une e.a.t. de type* 0xy *et* Y *une e.a.t. de type* x, *alors* (XY) *est une e.a.t. de type* y.

Nous désignons par 'x : X' une e.a.t. "X de type x". Le type s'interprète comme un opérateur qui en s'appliquant à une e.a. X, fait de X une e.a.t.. Nous représentons l'application de l'e.a.t. X, de type 0xy, à l'e.a.t. Y, de type x, par la liste suivante :

$$((0xy : X)\ (x : Y))$$

ou encore par l'arbre applicatif suivant :



La règle (6) est représentée par la règle constructive suivante :

$$(6^1)\quad \frac{0xy : X \qquad x : Y}{y : XY}$$

On a d'autres règles analogues à (6^1) :

$$(6^2)\quad \frac{0xy : X \qquad y : XY}{x : Y}$$

$$(6^3)\quad \frac{x : Y \qquad y : XY}{0xy : X}$$

Le schéma de règle (6^1) a un fonctionnement comparable à la règle de Modus Ponens relative à l'élimination de l'implication dans la "déduction naturelle" de G. Gentzen.

Remarque : Dans le cadre ensembliste de Z-F, on peut interpréter 'x : X' par l'appartenance de l'e.a. X à une classe x : 'X $\in$ x'. Dans une interprétation moins ensembliste, on peut interpréter 'x : X' par "x s'applique à X" ou encore par "la proposition x X est vraie".

Soit X une e.a.t. de type $0x_1\ 0x_2 \ldots 0x_n\ y$. On dit alors que X est un *opérateur n-aire*. Nous définissons :

$$0^n\ x_1\ x_2 \ldots x_n\ y \equiv 0x_1\ 0x_2 \ldots 0x_n\ y\ .$$

Soit une séquence d'éléments $<a_1, \ldots, a_n>$ de types respectifs $x_1, \ldots, x_n$. L'application de l'opérateur X de type $0^n\ x_1 \ldots x_n\ y$ à la séquence $<a_1, \ldots, a_n>$ conduit à n opérations successives d'application :

$$\frac{0^n\ x_1\ x_2 \ldots x_n\ y\ :\ X \qquad <x_1 : a_1, x_2 : a_2, \ldots, x_n : a_n>}{y : X\ a_1\ a_2 \ldots a_n}$$

4. PRINCIPE APPLICATIF (de Schönfinkel).

Nous n'allons pas considérer le type "produit cartésien fini" qui permettrait de construire des uples au moyen de la règle suivante :

(7) *si* $x_1, \ldots, x_n$ *sont des types, alors* $<x_1, \ldots, x_n>$ *est un type.*

En effet, en nous fondant sur la bijection suivante dans les ensembles ($E_1, \ldots, E_n$, F sont des ensembles) : $\mathrm{Hom}\ (E_1 \times \ldots \times E_n, F) \simeq \mathrm{Hom}\ (E_1, \mathrm{Hom}\ (E_2, \ldots, \mathrm{Hom}\ (E_n, F) \ldots))$, nous pouvons poser la bijection suivante entre les types :

$$0\ <x_1, \ldots, x_n>\ y \ \simeq\ 0^n\ x_1 \ldots x_n\ y \ \equiv\ 0x_1 \ldots 0x_n\ y\ .$$

Le principe applicatif (ou principe de Schönfinkel) fait correspondre à chaque opérateur n-aire X de type $0^n x_1 \ldots x_n\ y$ un opérateur unaire *Cur* (X) de type $0x_1\ z$, avec $z \equiv 0^{n-1} x_2 \ldots x_n\ y$ (pour $n > 1$). L'opérateur unaire *Cur* (X) est obtenu par la correspondance, dite de curryfication, à partir de l'opérateur n-aire X.

Remarque : Lorsqu'on "oublie" les types, on peut considérer que tous les opérateurs sont unaires puisque tout opérateur n-aire se voit associer un opérateur unaire canonique obtenu par curryfication.

5. TYPES MORPHO-SYNTAXIQUES.

Les types syntaxiques des langues naturelles sont récursivement engendrés à partir de deux types élémentaires de base : 'termes nominaux' et 'phrases'. Désignons par t et p ces types élémentaires ou sortes de base. Les autres types syntaxiques sont engendrés par la règle (6). Nous avons, par exemple, les types syntaxiques suivants pour le français :

0tp : type des *verbes intransitifs* et des *prédicats* (linguistiques) *unaires* ;
0^2ttp : type des *verbes transitifs* et des *prédicats* (linguistiques) *binaires* ;
0tt : type des *déterminants nominaux* ;
0pp : type des *opérateurs unaires de phrases* (négation, modalités, ...) ;
0^2ppp : type des *connecteurs binaires de phrases* ;
0Ott0tt : type des *adverbes* portant sur des déterminants nominaux.

Un *prédicat* (linguistique) n-aire est un opérateur de type $0^n t_1 \ldots t_n\ p$ ($n \geq 1$). Une proposition peut être considérée comme un prédicat zéro-aire. Un prédicat n-aire sera désigné

par P_n. Un terme de type t sera désigné par T. Une *opération élémentaire de prédication* consiste à appliquer un prédicat à un terme nominal T pour construire soit un prédicat, soit une proposition. Le dernier terme qui entre dans une prédication s'appelle terme primaire (assumant en français la fonction syntaxique de *sujet*) ; l'avant-dernier terme qui entre dans la prédication s'appelle *terme secondaire* (*objet direct* en français) (voir l'article de S.K. Shaumyan dans *Mathématique et Sciences humaines*, n°77, 1982, pp.7-42).

Une relation prédicative intransitive (exemple : *Socrate court*) est construite par une seule opération de prédication :

$$\frac{0tp : P_1 \qquad t : T^1}{p : P_1\, T^1}$$

Une relation prédicative transitive (exemple : *Noémon admire Marie*) est construite par deux opérations successives de prédication :

$$\frac{\dfrac{0t0tp : P_2 \qquad t : T^2}{0tp : P_2\, T^2} \qquad t : T^1}{p : P_2\, T^2\, T^1}$$

Remarque : Pour une discussion sur les choix linguistiques, le lecteur se reportera à la bibliographie générale, notamment aux ouvrages de S.K. Shaumyan et de J.P. Desclés.

5.1. Une phrase étant considérée comme un agencement d'opérateurs et d'opérandes de différents types, elle est analysée comme une e.a.t. ayant le type p de phrase. Ses constituants sont analysés comme des e.a.t.. Ainsi, l'analyse applicative d'une phrase comme :

Noémon a épousé une très jeune fille

conduit à une représentation applicative dès que l'on a assigné des types syntaxiques élémentaires aux unités de la phrase. Supposons que l'on ait, par exemple, le système d'assignations suivantes :

t : *Noémon* ; 0^2ttp : *a-épousé* ; 0tt : *une* ; 00tt0tt : *très* ; 0tt : *jeune* ; t : *fille.*

Nous avons alors l'analyse applicative qui est représentée par la liste suivante (exprimable dans un langage de programmation de haut niveau comme LISP) :

(p : (0tp : (0^2ttp : *a-épousé*)
(t : (0tt : *une*)
(t : (0tt : (00tt0tt : *très*)
(0tt : *jeune*)
(t : *fille*))))
(t : *Noémon*))

qui correspond canoniquement à l'arbre applicatif de la figure 1 :

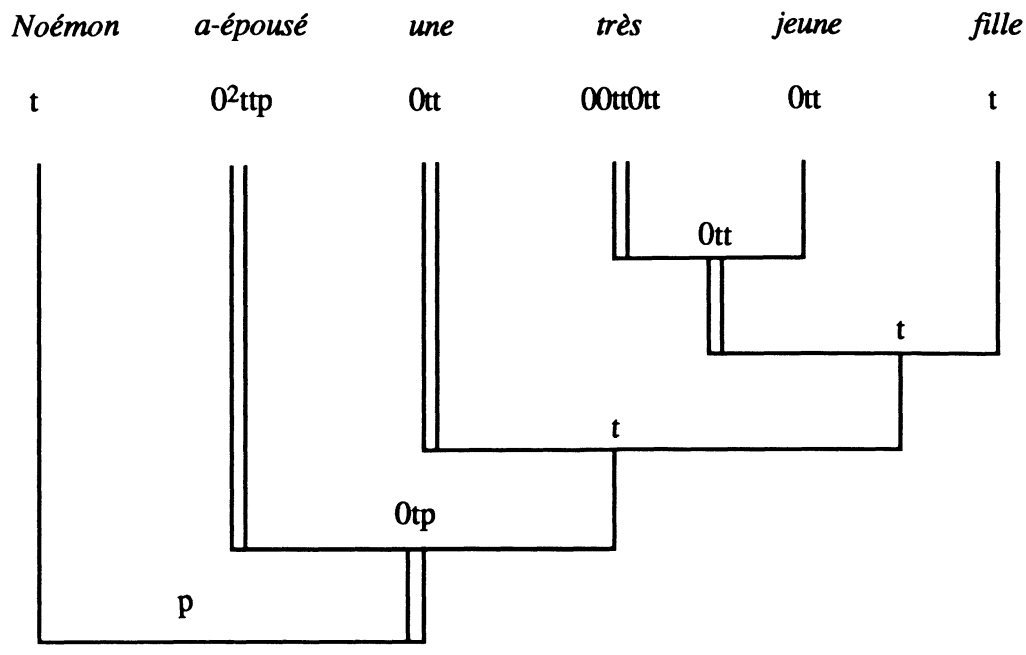


Figure 1.

En adoptant une *présentation préfixée* (selon les positions : opérateur - opérande) la phrase reçoit une décomposition applicative exprimée sous forme d'une autre liste (sans indication, ici, des types) :

((*a-épousé* (*une* ((*très jeune*) *fille*))) *Noémon*).

5.2. Nous pouvons pousser l'analyse jusqu'à une décomposition applicative morphologique en morphèmes. Pour cela, nous introduisons des nouvelles sortes :
- des *sortes morphologiques* d'un ensemble Sμ de sortes morphologiques ;
- des *sortes positionnelles* d'un ensemble Sπ de sortes de position sur l'axe syntagmatique.

Pour Sμ, nous avons, entre autres, les sortes suivantes :
'préfixe nominal' ; 'suffixe nominal' ; 'suffixe adjectival' ; 'base verbale' ; 'base nominale' ; 'genre' ; 'nombre' ; ...

Les éléments de Sπ sont au nombre de quatre :
'position gauche (g)' ; 'position droite (d)' ; 'position neutre (n)' ; 'position indéfinie (?)'.

Une unité de *position neutre* a un fonctionnement d'opérande morphologique ; une unité de *positon gauche* (ou *droite*) a un fonctionnement d'opérateur morphologique qui vient se positionner dans la séquence linguistique à gauche (respectivement à droite) de l'opérande morphologique auquel il est appliqué. Une unité de *position indéfinie* est un constituant linguistique qui peut ultérieurement s'intégrer dans une autre construction.

Les *types morpho-syntaxiques* sont alors engendrés au moyen d'un opérateur M et de la règle suivante :
(8) *si* μ est une sorte morphologique de Sμ ,
π est une sorte positionnelle de Sπ,
σ est un type syntaxique,
alors Mπ : μ : σ est un type morpho-syntaxique assignable à une unité linguistique.

Soit X une e.a. à laquelle on assigne le type morpho-syntaxique Mπ : μ : σ. Nous écrivons :

(Mπ : μ : σ : X).

Dans ce cas, l'e.a. X a le type syntaxique σ, le type morphologique μ et fonctionne comme : soit un opérateur morphologique qui vient se positionner à gauche (si π = g) ou à droite (si π = d) de son opérande morphologique, soit un opérande morphologique (si π = n).

Par exemple, le syntagme nominal *une loi anticonstitutionnelle* se voit analyser sous forme d'un agencement d'opérateurs et d'opérandes morpho-syntaxiques. Nous avons la liste (avec types) suivante :

(M ? : Syntagme nominal fem. : t :
((Mg : article indéf. fem. : 0tt :
((Md : (genre : fem.) : 00tt0tt : - *e*)
(Mn : article : 0tt : *un*)))

(Mn : nom fem : t :
((Md : adj. fem. : 0tt :
((Mg : préf. adject. : 00tt0tt : *anti-*)
((Md : (genre : fem.) : 00tt0tt : *-le*)
((Md : suf. adject. : 0t0tt : *-nel*)
((Md : suf. nomin. : 00t0tpt : *-tion*)
(Mn : base verbale : 0t0tp : *constitu*))))))

(Mn : nom fem. : t :
((Md : (genre : fem.) : 0tt : -)
(Mn : base nominale : t : *loi*))))))

Cette liste correspond à l'arbre applicatif de la figure 2 :

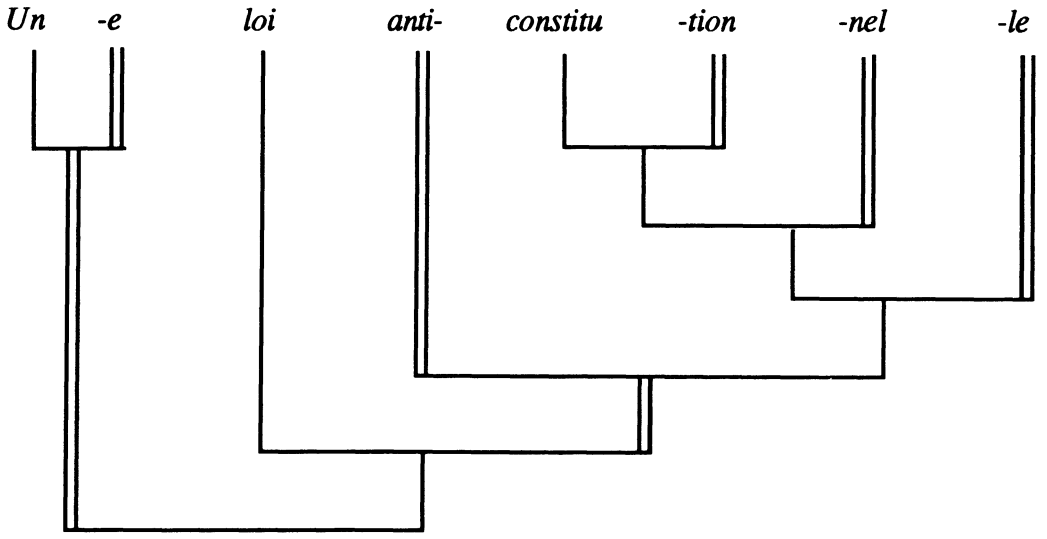


Figure 2.

En oubliant les types et en prenant un ordre préfixé (opérateur-opérande), nous obtenons la liste suivante :

((*-e* (*un*)) ((*anti-* (*-le* (*-nel* (*-tion* (*constitu*))))) (*loi*))).

5.3. Revenons à l'exemple précédent : *Noémon a épousé Marie*. Considérons le verbe conjugué '*a-épousé*' comme étant décomposable en un morphème discontinu *a-é* de temps (tense) avec pour valeur le "passé composé". En tenant compte de la décomposition morpho-syntaxique, nous avons la liste suivante :

(M ? : phrase : p :
((Md : prédicat : passé composé : 0tp :
((Mg : verbe passé composé : 0^2ttp :
((Mg-d : (tense : passé composé) : 00^2ttp0^2ttp : *a -é*)
(Mn : base verbale : 0^2ttp : *épous*)))
(Mn : nom : t : *Marie*)))
(Mn : nom : t : *Noémon*)))

Dans cette analyse, l'opérateur *a -é*, qui se réalise par un morphème discontinu (d'où le type de position : Mg - d), s'applique à l'opérande *épous*, prédicat binaire de type 0^2ttp, pour constituer le nouveau prédicat binaire *a-épousé* de même type. Ce dernier est appliqué à l'opérande nominal *Marie*. Le résultat est un nouvel opérateur unaire (un prédicat de type 0tp) qui est appliqué à *Noémon* pour constituer finalement la phrase.

6. COMBINATEURS.

Les grammaires catégorielles (Bar-Hillel, 1953) entrent dans le cadre des systèmes applicatifs avec types : les e.a.t. représentent des agencements d'opérateurs et d'opérandes linguistiques de différents types syntaxiques, agencements que l'on représente par des arbres applicatifs.

La logique combinatoire est un langage applicatif qui a un plus grand pouvoir expressif que celui des langages engendrés ou reconnus par les grammaires catégorielles. Ce pouvoir lui vient de l'adjonction d'opérateurs particuliers que l'on appelle des *combinateurs*.

6.1. Intuitivement, un combinateur est un opérateur abstrait dont la sémantique d'action est déterminée intrinsèquement à l'intérieur du langage. Explicitons cette idée. Le formalisme de la logique combinatoire est, dans son essence, indépendant des interprétations dénotatives. En général, un opérateur élémentaire f qui s'applique à un opérande a construit une nouvelle entité appelée b . L'action sémantique de f appliquée à a est alors complètement déterminée lorsqu'on sait construire explicitement le résultat. Cette action sémantique peut être définie soit à l'intérieur du langage, soit en faisant appel à un domaine interprétatif externe. Dans le second cas, un opérateur f doit être interprété par une fonction f° définie sur un domaine D : f étant appliqué à l'opérande a , l'entité dénotative [[b]] de D, exprimée dans le langage par b , est explicitement déterminée par l'image de la fonction f° appliquée à la valeur dénotative [[a]] de a. Nous avons : $[[b]] = f^\circ([[a]])$.

Les combinateurs sont des opérateurs dont les actions sémantiques sont toujours définies à l'intérieur du formalisme, de façon intrinsèque, sans faire référence à quelque domaine interprétatif externe. La sémantique des combinateurs est donc indépendante des interprétations dénotatives. Il s'ensuit qu'une entité qui est construite à l'aide des combinateurs restera indépendante des interprétations dénotatives. Les combinateurs définissent ce que nous proposons d'appeler une *sémantique intrinsèque*. Nous verrons, à l'aide de quelques exemples, l'usage que l'on peut en faire pour une analyse des significations intrinsèques dans les langues naturelles, notamment pour les significations grammaticales.

6.2. Donnons une définition plus formelle d'un combinateur. Considérons des e.a. (ou des e.a.t.) X_1, ..., X_p et Y_1, ..., Y_q. Un combinateur $\mathfrak{X}$ appliqué à ces e.a. (ou e.a.t.) est destiné à les combiner entre elles pour construire une nouvelle e.a. (ou e.a.t.) 'Z_1 ... Z_m'. Plus précisément, à chaque combinateur $\mathfrak{X}$ est associée une règle, dite *règle de réduction* ou encore β-réduction, de la forme suivante :

$$[\mathfrak{X}] \quad \mathfrak{X}X_1 \ldots X_p\, Y_1 \ldots Y_p \geq Z_1 \ldots Z_m$$

L'e.a. (ou e.a.t.) 'Z_1 ... Z_m' est le résultat de l'action du combinateur $\mathfrak{X}$: chaque Z_i (i=1,...,m) est un agencement combinatoire des X_1, ..., X_p, Y_1, ..., Y_q (pas nécessairement tous !), c'est-à-dire que Z_i est une e.a. (ou e.a.t.) formée à l'aide d'occurrences des X_1, ..., X_p, Y_1, ..., Y_q. La partie gauche de [$\mathfrak{X}$] est le *programme d'opération applicative* consistant à appliquer le combinateur $\mathfrak{X}$ à ses opérandes successives de façon à former une nouvelle entité du langage ; l'e.a. (ou e.a.t.) décrite dans l'expression de droite exprime le résultat du programme d'action de $\mathfrak{X}$.

Supposons que X_1, ..., X_p soient des opérateurs (unaires) et Y_1, ..., Y_q des e.a. (ou e.a.t.) quelconques. Un combinateur donné $\mathfrak{X}$ est en général caractérisé par deux paramètres : p et q (et une portée p+q). Le combinateur $\mathfrak{X}_{pq}$ combine entre eux les opérateurs X_1, ..., X_p de façon à former un nouvel opérateur complexe '$\mathfrak{X}_{pq}\ X_1 \ldots X_p$' dont l'action sur les opérandes successifs Y_1, ..., Y_q est décrite explicitement par la β-réduction associée à $\mathfrak{X}_{pq}$; dans ce cas, Y_1, ..., Y_q jouent le rôle d'argument de l'opérateur complexe $\mathfrak{X}_{pq}\ X_1 \ldots X_p$.

D'une façon générale, à un combinateur est associé un schéma de règle formulé avec des variables (nécessairement libres). Par exemple, le combinateur **B** combine deux opérateurs (unaires) f_1 et g_1 de façon à former un opérateur complexe **B** $f_1 g_1$ tel que l'action de '**B** $f_1 g_1$' sur l'argument a, donnera pour résultat l'e.a. $f_1(g_1\ a)$: '**B** $f_1 g_1$' a $\geq f_1(g_1\ a)$. D'une façon plus générale, le schéma de règle canoniquement associé au combinateur **B** est :

$$\mathbf{B}\ X\ Y\ Z \geq X(Y\ Z)\,.$$

6.3. Les combinateurs seront tous engendrés à partir de *combinateurs élémentaires* par l'opération d'application.

DEFINITION : Un **combinateur** *est une e.a. ne comportant que des occurrences de combinateurs élémentaires.*

Donnons quelques exemples de *combinateurs élémentaires* avec leurs règles de réduction :

BXYZ	$\geq$	X(YZ)
CXYZ	$\geq$	XZY
WXY	$\geq$	XYY
KXY	$\geq$	X
IX	$\geq$	X
SXYZ	$\geq$	XZ(YZ)
ΦXYZU	$\geq$	X(YU)(ZU)
ΨXYZU	$\geq$	X(YZ)(YU)

. Le combinateur **B** compose directement X et Y entre eux. En interprétant X et Y comme des fonctions [[X]] et [[Y]], le combinateur **B** représente la composition ordinaire de fonctions :

$$[[\mathbf{BXYZ}]] = [[\mathbf{BXY}]]\ ([[Z]]) = [[X]]\ ([[Y]]\ ([[Z]]))$$

. Le combinateur **C** exprime la permutation des arguments d'un opérateur X binaire. Il sera utilisé pour exprimer intrinsèquement *l'idée de "conversion abstraite"*.

. Le combinateur **W** sert à dupliquer un opérande. Il sera utilisé dans les *procédés de diagonalisation.*

. Le combinateur **K** sert à contruire des opérateurs qui n'agissent plus sur certains opérandes.

. Le combinateur **I** sert à exprimer *l'idée d'"identité"*.

. Les combinateurs **S**, $\mathbf{\Phi}$, $\mathbf{\Psi}$, sont des opérateurs de composition qui généralisent le combinateur **B**.

6.4. Les combinateurs élémentaires sont des opérateurs dont les actions sont purement formelles et internes au formalisme. Ils possèdent plusieurs propriétés intéressantes.

(**P1**) Ils ne sont pas indépendants les uns des autres. On peut en effet définir des combinateurs élémentaires à partir d'autres *combinateurs élémentaires de base*. Si l'on prend comme combinateurs de base les combinateurs **S** et **K**, on démontre alors que : $\mathbf{I} \equiv \mathbf{S\,K\,K}$; $\mathbf{B} \equiv \mathbf{S(KS)\,K}$; $\mathbf{W} \equiv \mathbf{SS(K\,I)}$; $\mathbf{C} \equiv \mathbf{S(BBS)(KK)}$ etc.

(**P2**) Certains combinateurs élémentaires représentent des transformations. Les combinateurs *propres* aboutissent à des réductions où les membres de droite de la β-réduction sont formées avec les *seules* occurrences (pas nécessairement toutes !) des éléments $X_1, ..., X_p, Y_1, ..., Y_q$. Les combinateurs *réguliers* sont des combinateurs propres qui laissent invariant l'opérande qui est en première position, c'est-à-dire que le schéma d'un combinateur régulier est donné par :

$$[X_r] \quad \mathfrak{X}_r\, X_0\, X_1 \ldots X_p Y_1 \ldots Y_q \geq X_0\, Z_1 \ldots Z_m .$$

Le combinateur régulier $\mathfrak{X}_r$ représente alors des transformations sur les seuls opérandes de l'opérateur X_0. Cette transformation est représentable par une transformation d'arbres.

Les combinateurs élémentaires précédents sont tous *réguliers*. En revanche, le combinateur **W W W** n'est pas propre car :

$$\mathbf{W\,W\,W} \geq \mathbf{W\,W\,W} \geq \mathbf{W\,W\,W} \ldots$$

Le combinateur $\mathfrak{Y} \equiv \mathbf{W\,S(B\,W\,B)}$ n'est pas propre non plus car on vérifie :

$$\mathfrak{Y}X \geq X(\mathfrak{Y}X)$$

($\mathfrak{Y}$ est un combinateur constructeur de points fixes).

(**P3**) Les combinateurs (en particulier les combinateurs réguliers) sont composables entre eux dans une structure algébrique de monoïde. Ainsi, pour effectuer une transformaiton sur une séquence d'éléments, on peut écrire un programme de transformations, représentable pour un combinateur, en composant des transformations élémentaires représentées par des combinateurs élémentaires réguliers effectuant des permutations, des duplications, des constructions de sous-expressions, des effacements ...

Remarque : Le lecteur se reportera à l'article de J.P. Ginisti dans ce numéro.

(**P4**) A chaque combinateur est attribué un *schéma de type*. Le type d'un combinateur est calculé en fonction du contexte, c'est-à-dire du type de ses opérandes.

Désignons par α, β, γ des variables de types. Les schémas de type associés aux combinateurs **B, C, W, K, I, S** sont respectivement :

$0^2(0\beta\gamma)\ (0\alpha\beta)\ (0\alpha\gamma)$: **B**
$0(0^2\alpha\beta\gamma)\ (0^2\beta\alpha\gamma)$: **C**
$0(0^2\alpha\alpha\beta)\ (0\alpha\beta)$: **W**
$0^2\alpha\beta\alpha$: **K**
$0^2(0^2\alpha\beta\gamma)\ (0\alpha\beta)\ 0\alpha\gamma$: **S**
$0\alpha\alpha$: **I**

Le type d'un combinateur **B** dans l'e.a.t. **B**fga sera calculé à partir de son schéma et des types assignés aux opérandes f,g et a, par la construction suivante :

$$\dfrac{\dfrac{\dfrac{0^2(0\beta\gamma)\ (0\alpha\beta)\ (0\alpha\gamma) : \mathbf{B} \qquad 0yz : f}{0(0\alpha\beta)\ (0\alpha\gamma) : \mathbf{B}f} \qquad 0xy : g}{0\alpha\gamma : \mathbf{B}fg} \qquad x : a}{\gamma : \mathbf{B}fga}$$

d'où l'on tire successivement :

$0\beta\gamma = 0yz$ d'où $\beta = y$ et $\gamma = z$
$0\alpha\beta = 0xy$ d'où $\alpha = x$ et $\beta = y$
$\alpha = x$

c'est-à-dire : $\alpha = x, \beta = y$ et $\gamma = z$.

Remarque 1 : Certaines constructions aboutissent à des expressions où il est impossible d'assigner, de façon cohérente, des types aux constituants de ces expressions. On dit, dans ce cas, que l'expression est *non stratifiable*. Par exemple, dans un système typé, l'expression XX est non stratifiable. En effet, d'après (6^1), (6^2), (6^3), on doit avoir, si X est un opérateur de type $0\alpha\beta$:

$$\dfrac{0\alpha\beta : X \qquad \alpha : X}{\beta : X}$$

Il est alors nécessaire de poser $0\alpha\beta = \alpha$. Dans une interprétation ensembliste, il faudrait trouver un ensemble E tel que l'on ait la bijection : Hom (E,E) $\simeq$ E. Signalons toutefois qu'il existe des modèles ensemblistes interprétant le système de la logique combinatoire (modèle de D. Scott) car, dans un système sans type, l'expression auto-applicative XX a parfaitement un sens.

Remarque 2 : Pour les expressions applicatives typées représentatives d'agencements linguistiques, les expressions seront toutes des expressions stratifiées.

Les schémas de type expriment le rôle des transformations assumées par les combinateurs.

. Le combinateur **B** compose entre eux des opérateurs de types respectifs $0\beta\gamma$ et $0\alpha\beta$ pour construire un nouvel opérateur complexe de type $0\alpha\gamma$.

. Le combinateur **C** construit un nouvel opérateur binaire de type $0^2\beta\alpha\gamma$, associé canoniquement à un opérateur binaire de type $0^2\alpha\beta\gamma$.

. Le combinateur **W** construit un nouvel opérateur unaire de type $0\alpha\beta$ canoniquement associé à un opérateur binaire de type $0^2\alpha\alpha\beta$.

. Le combinateur **K** construit un opérateur de type $0\beta\alpha$ à partir d'une entité de type α.

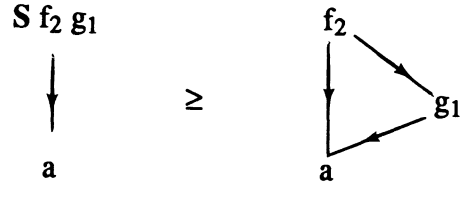
. Le combinateur **S** compose entre eux un opérateur binaire de type $0^2\alpha\beta\gamma$, un opérateur unaire de type $0\alpha\beta$ et une entité de type α de façon à ce que le résultat obtenu, en appliquant le nouvel opérateur ainsi composé à une opérande approprié, soit de type γ.

. Le combinateur **I** identifie une entité de type α avec elle-même.

(**P5**) Les combinateurs (tout particulièrement les combinateurs réguliers) sont des opérateurs abstraits qui construisent, à l'intérieur du formalisme, des opérateurs complexes à partir d'opérateurs plus élémentaires.

Reprenons le schéma général de réduction ($[\mathfrak{A}_r]$) où nous supposons que $\mathfrak{A}_r$ est un combinateur régulier et que $X_0, X_1, ..., X_p$ sont des opérateurs. Le schéma exprime que le combinateur $\mathfrak{A}_r$ compose entre eux les opérateur $X_0, X_1, ..., X_p$ de façon à construire un opérateur complexe $\mathfrak{A}_r X_0 X_1 ... X_p$. En appliquant cet opérateur complexe à la séquence de ses arguments $Y_1,..., Y_q$ on obtient le résultat sous forme de l'e.a. (ou e.a.t.) $Z_1 ... Z_m$ où chaque $Z_i (i = 1, ..., m)$ est une e.a. (ou e.a.t.) formée des *seules* occurrences des opérateurs $X_1,..., X_p$ et des e.a. (ou e.a.t.) $Y_1, ..., Y_q$.

Par exemple, le combinateur **S** compose entre eux les opérateurs f_2 et g_1 de types assignés respectifs 0^2xyz, $0xy$; il construit ainsi l'opérateur complexe Sf_2g_1 de type $0xz$. L'opérateur Sf_2g_1 étant appliqué à une entité a, de type x, donne pour résultat une entité, de type z, définie explicitement sous forme d'une e.a.t. où seules interviennent les occurrences de f_2, g_1 et a ; l'e.a.t. est, par définition de la réduction associée à S, l'expression : $f_2a(g_1a)$. Dans ce cas, on a $X_0 \equiv f_2$, $X_1 \equiv g$, $Y_1 = a$ et $Z_1 \equiv a$, $Z_2 \equiv g_1a$. En utilisant des représentations sous forme de graphes, nous pouvons représenter la β-réduction par une *réécriture de graphes* (en fait des treilles au sens de Desclés, 1981) :



7. PREDICATS COMPLEXES ET LOIS LINGUISTIQUES.

Les combinateurs donnent la possibilité de construire des prédicats complexes (et plus généralement, nous l'avons déjà dit, des opérateurs complexes) à partir de prédicats plus élémentaires, ce qui est impossible dans les *langages du premier ordre*. Donnons quelques exemples de prédicats grammaticaux complexes que nous avons utilisés dans plusieurs

publications (voir les références bibliographiques).
7.1. Nous introduisons l'*opérateur grammatical* © de *contrôle intentionnel*, lié à l'analyse de la "transitivité" qui implique l'intentionalité d'un agent. Cet opérateur, de type 0^2ptp (où p et t sont les types élémentaires des propositions ou des phrases d'une part, et des termes nominaux, d'autre part), construit une relation qui tient entre une proposition statique et un terme nominal assumant le rôle d'un agent. L'expression © $(P\ T^2)T^1$ signifie que T^1 dénote un agent qui "exerce un contrôle intentionnel sur un processus affectant un patient représenté par T^2, le processus étant orienté vers la situation stative représentée par la proposition '$P\ T^2$' où P est un prédicat unaire". Par contre, l'opérateur grammatical 'CAUSE' a pour type 0^2ptp ; il exprime la relation qui tient entre un causateur T^1 (pas nécessairement agent) et une situation représentée par la relation prédicative : CAUSE $(P_1T^2)\ T^1$ exprime que T^1 est la cause de la situation représentée par l'expression P_1T^2.

Introduisons aussi les termes constants $\varnothing_0$ et $\varnothing_\Delta$ qui dénotent respectivement "agent non spécifié" et "place syntaxique d'un terme sans dénotation".

Nous avons alors par exemple les *prédicats complexes* suivants construits à l'aide des combinateurs **B**, **C**, **W** et **K** :

B © P_1 ; **C** $P_{trans.}$ $\varnothing_0$; **W**(**B** © P_1) ; **K** P_0 ; **W** $P_{trans.}$; **B** CAUSE P_1.

7.1.1. Ainsi, si P_1 désigne le prédicat descriptif 'est-ouvert', le prédicat transitif 'ouvre' est construit à partir de P_1 et du prédicat grammatical de contrôle ©, à l'aide du combinateur **B** : ['*ouvre*' ≡ **B** © '*est-ouvert*'].

La relation prédicative sous-jacente à la phrase transitive active '*Jean ouvre la porte*' est obtenue en appliquant le prédicat '*ouvre*' aux deux opérandes successifs *la-porte* et *Jean* : (*ouvre la-porte*) *Jean* . A partir de ce prédicat transitif '*ouvre*', nous en déduisons le prédicat passif '*a-été-ouverte*' par dérivation : ['*a-été-ouverte* ≡ **C** '*ouvre*' $\varnothing_0$]. La phrase passive est alors construite en appliquant ce prédicat complexe à un terme nominal comme '*la-porte*', d'où : *a-été-ouverte la-porte.*

Remarque : Pour l'analyse du passif, se reporter à la bibliographie citée.

Prenons maintenant pour prédicat P_1, le prédicat descriptif '*est-assis*'. Le prédicat moyen '*s'asseoit*' est dérivé de '*est-assis*' au moyen d'une composition à l'aide de **B** et **W** par : ['*s'asseoit*' ≡ **W**(**B** © '*est-assis*')].

7.1.2. Si P_0 désigne un prédicat zéro-aire (en fait une proposition) comme '*il-y-a-pluie*', alors le prédicat impersonnel unaire '*pleut*' est dérivé à l'aide du combinateur **K** : ['*pleut*' ≡ **K** '*il-y-a-pluie*'].

7.1.3. Soit un prédicat transitif comme '*lave*'. Le prédicat réflexif '*se-lave*' est dérivé de '*lave*' par ['*se-lave*' ≡ **W** '*lave*'].

7.1.4. Considérons le prédicat unaire *meurt*. Le combinateur **B** compose le prédicat '*meurt*' avec le prédicat grammatical CAUSE d'où le prédicat causatif complexe '*fait-mourir*' défini comme : **B** CAUSE '*meurt*'.

7.1.5. L'opérateur de contrôle intentionnel © joue un rôle important dans l'organisation des voix (voir 9.2.) et dans l'analyse de l'agentivité. Son recours n'est pas *ad hoc* mais justifié par

de nombreuses observations et analyses. La notion de contrôle est l'un des invariants grammaticaux les plus importants nécessaire à l'organistion des relations casuelles abstraites. Citons, par exemple, B. Comrie (*Languages Universals and Linguistic Typology*, 1981, pp. 55 et 53) :

"On the parameter of control, it might seem that there is no distinction between experience and patient, since in general one does have control over one's own sensory experiences : one can choose whether or not to look at something (...). The most important point that we want to make concerning the relations among agent, force, instrument, and patient is that this is not so much a set of discrete relation is but rather a continuum, the labels representing different points along thes continuum. The continuum as a whole can be regarded as a continuum of control, and we shall use this term rather the set of discrete labels (...)".

On peut aussi citer C. Hagège (*La structure des langues*, Presses Universitaires de France, 1982, p. 50) :

"Cette opposition entre des *degrés de contrôle* tient une grande place dans les langues [...]. Selon les langues s'opposent : (i) *les degrés de volonté de l'agent* ; par ex., correspondant aux paires fr. *entendre/écouter* ou *recevoir/prendre* [...] ; (ii) *les degrés d'affectation du patient* [...] ; (iii) *les degrés d'achèvement d'un procès* [...] ; (iv) *les degrés de définitudes du patient* [...]".

Nous pourrions citer encore d'autres linguistes comme P. Hopper, T. Givón, C. Tchekhoff qui attachent une grande importance à la notion de contrôle pour l'analyse de la transitivité sémantique.

7.2. Une loi élémentaire est une relation (le plus souvent une identification) entre un prédicat P' complexe (plus généralement un opérateur complexe) et une e.a. (ou une e.a.t.) E : [P' $\mathfrak{R}$ E].

Si $\mathfrak{R}$ est une identification (désignée '≡') alors P' pourra être remplacée par l'expression E dans toutes les occurrences de P'. Ainsi, dans une définition d'un prédicat complexe, P' représente le *definiendum* et E le *definiens* , d'où la loi linguistique : [P' ≡ E].

Si $\mathfrak{R}$ est un préordre, alors, dans une sous-expression, on pourra remplacer P' par E au cours d'une réduction. Dans certains cas, la relation $\mathfrak{R}$ est sensible au contexte ; dans ce cas, le remplacement, dans une réduction donnée, pourra se faire lorsque les conditions contextuelles seront vérifiées.

Il est clair que les lois élémentaires sont liées à un domaine d'utilisation du formalisme de la logique combinatoire ; elles ne font pas partie des lois logiques élémentaires de la logique combinatoire (comme dans la définition d'un combinateur en terme des combinateurs de base ainsi : [**I** ≡ **S K K**]. Il y a donc des lois élémentaires spécifiques à l'arithmétique, à la logique inférentielle, à la programmation fonctionnelle...

En linguistique, les lois élémentaires expriment des *relations intrinsèques* entre des prédicats et entre des opérateurs. Ces lois sont formulées à la suite d'une *démarche abductive* propre aux règles argumentatives de la linguistique.

Les lois linguistiques sont intrinsèques ; elles traduisent des relations de significations notamment des *relations grammaticales indépendantes* des domaines externes (univers perçus et organisés par la perception) que les langues naturelles, en tant que systèmes symboliques, représentent.

Les prédicats complexes précédents donnent naissances aux formulations des lois élémentaires suivantes :

$[P_{trans.} \equiv \mathbf{B} \copyright P_1]$
$[P_{passif} \equiv \mathbf{C}\, P_{trans.}\, \varnothing_0]$
$[P_{moyen} \equiv \mathbf{W}(\mathbf{B} \copyright P_1)]$
$[P_{impers.} \equiv \mathbf{K}\, P_0]$
$[P_{refl.} \equiv \mathbf{W}\, P_{trans.}]$
$[P_{caus.} \equiv \mathbf{B}\ \text{CAUSE}\ P_1]$

7.3. Les lois élémentaires doivent respecter la condition suivante d'isotypicalité :
Dans la loi élémentaire $[P' \equiv E]$, P' *et* E *sont isotypes.*

Nous allons donner deux exemples de relations. La première respecte cette condition ; la seconde ne la respecte pas.

Exemple 1 : Soit P_0 une proposition de type p. Formons l'e.a.t. $\mathbf{K}\, P_0$ que nous appliquons à un terme T. Calculons le type de **K** à partir de son schéma (voir en 6.4 $\mathbf{P_n}$) :

$$\frac{0\alpha 0\beta\alpha : \mathbf{K} \qquad p : P_0}{0\beta\alpha : \mathbf{K}\, P_0} \qquad t : T$$

$$\alpha : \mathbf{K}\, P_0 T$$

Nous en déduisons que : $\alpha = p$, $\beta = t$. La relation $[P'_1 \equiv \mathbf{K} P_0]$ vérifie la condition d'isotypicalité : P'_1 est un prédicat unaire de type 0tp construit à partir d'une proposition P_0.

C'est ainsi que nous analysons le prédicat '*pleut*' dans la construction impersonnelle. Le terme fictif $\varnothing_\Delta$ introduit par le combinateur **K** est le marqueur d'une place syntaxique non dénotative, d'où la représentation applicative de *il pleut* par l'e.a.t. : $\mathbf{K}\, P_0\, \varnothing_\Delta$, où P_0 est la proposition "*il-y-a-pluie*" et où $\varnothing_\Delta$ représente *il*. Nous avons en effet : $\mathbf{K}\, P_0\ \varnothing_\Delta \geq P_0$.

Exemple 2 : Sur le même modèle, on pourrait essayer d'établir une relation entre l'infinitif *partir* (T_0), considéré comme un terme nominal, et le prédicat *part* (P'_1), par la relation de réduction *part* $\geq$ **K** *partir* $\varnothing_0$, où $\varnothing_0$ est un terme qui désigne un "agent non spécifié". Calculons le type que l'on devrait assigner à **K** :

$$\frac{0\alpha 0\beta\alpha : \mathbf{K} \qquad t : T_0}{0\beta\alpha : \mathbf{K}\, T_0} \qquad t : \varnothing_0$$

$$y : \mathbf{K}\, T_0\, \varnothing_0$$

Nous voudrions considérer $\mathbf{K}\,T_0$ comme un prédicat unaire P'_1 de type 0tp qui s'applique à un terme nominal $\varnothing_0$, d'où la relation $[P'_1 \equiv \mathbf{K}\,T_0]$. Le type γ doit donc être p. Nous en déduisons que nécessairement, $\alpha = t$. Alors, d'une part, en considérant le résultat, on a : $\alpha = \gamma = p$; d'autre part, par construction, on a : $\beta = t$. La relation $[P'_1 \equiv \mathbf{K}\,T_0]$ où P'_1 serait un prédicat unaire ne respecte pas la condition d'isotypicalité, cette relation ne peut donc pas être une loi élémentaire puisque l'on aboutit simultanément à : $\alpha = t$ et à : $\alpha = p$.

8. RELATION DE REDUCTION ENTRE EXPRESSIONS APPLICATIVES.

DEFINITION : *La relation de réduction '→'entre e.a.t. ("... se réduit à ..." ou "... est obtenu à partir de ...") est engendrée par la relation de réduction immédiate '≥' associée aux combinateurs ; elle est définie comme suit :*

Schéma d'axiomes :

[**I**]	: **I**X	≥ X	*pour tout X*
[**K**]	: **K**XY	≥ X	*pour tout X, Y*
[**S**]	: **S**XYZ	≥ XZ(YZ)	*pour tout X, Y, Z*
[ρ]	: X	≥ X	*pour tout X.*

Schémas de règles de réduction :

[μ] : [X → X'] ⇒ [ZX → ZX']

[ν] : [X → X'] ⇒ [XZ → X'Z]

[τ] : [X → Y] ∧ [Y → Z] ⇒ [X → Z]

PROPOSITION : *'X → Y' si et seulement si Y est obtenu à partir de X par une suite finie (éventuellement de longueur nulle) de réductions immédiates.*

Nous utilisons la présentation des réductions comme dans la "déduction naturelle" de Gentzen.

Exemple : Nous avons la loi (de la logique combinatoire) qui exprime le combinateur **B** en terme de **S** et de **K** : [**B** ≡ **S**(**KS**)**K**]. Prouvons que : **BXYZ** → X(YZ).

1	**BXYZ**	hyp.
2	[**B** ≡ **S**(**KS**)**K**]	loi logique
3	**S**(**KS**)**K** XYZ	1, 2, rempl.
4	**KSX**(**KX**)YZ	3,[**S**]
5	**S**(**KX**)YZ	4,[**K**]
6	**KXY**(YZ)	5,[**S**]
7	X(YZ)	6,[**K**]

DEFINITION : *La relation d'égalité ' = ' est engendrée à partir de la relation de réduction '→' comme étant sa fermeture symétrique.*

Autrement dit, on a [X = Y] lorsqu'on déduit cette assertion des schémas [**I**], [**K**], [**S**] et [ρ] et des schémas de règles [μ], [ν] et [τ] en remplaçant '→' par '=' et en ajoutant le schéma de règle [σ] : [X = Y] ⇒ [Y = X].

L'égalité '=' est alors la relation d'équivalence, monotone à gauche et à droite, qui est engendrée par la réduction '→'.

DEFINITION : *Une e.a. (ou une e.a.t.) X est dans une* forme normale *lorsqu'il est impossible de procéder par l'un des schémas [I], [K], [S], à une réduction de X ou de toute e.a. (ou e.a.t.) qui serait une sous-expression de X.*

On dit que X qui est dans une forme normale, est une expression irréductible.

DEFINITION : *Soit X une e.a. (ou e.a.t.). Si X se réduit à Z et si Z est dans une forme normale, alors Z est appelée* "forme normale de" *X.*

Exemple : X(YZ) est une forme normale de **B**XYZ.

Toute e.a. n'a pas nécessairement une forme normale. Par exemple, **W W W** est irréductible !

8.1. Nous allons donner des exemples de réduction engendrés par les lois linguistiques que nous avons considérées précédemment (voir 7.2.).

Nous avons, par exemple, les réductions suivantes (les phrases sont représentées sous forme d'expressions applicatives préfixées) :

transitivité : *lit un-livre Jean* → © *(est-lu un-livre) Jean*

passif : *a-été-lu le-livre* → *a-lu le-livre on*

moyen : *se-asseoit Jean* → © *(est -assis Jean) Jean*

impersonnel : *pleut il* → *il-y-a-pluie*

réflexivité : *se-lave Jean* → *lave Jean Jean*

causatif : *fait-mourir le-daim le-chasseur* → CAUSE *(est-mort le-daim) le-chasseur*

Nous avons les théorèmes suivants, démontrés avec l'aide des lois linguistiques élémentaires introduites en 7.2..

THEOREME 1 : $P_{trans.}\ T^2T^1 \rightarrow \copyright\ (P_1T^2)T^1$

1	$P_{trans.}\ T^2T^1$	hyp.
2	$[P_{trans.} \equiv \mathbf{B} \copyright P_1]$	loi linguistique
3	$\mathbf{B} \copyright P_1\ T^2\ T^1$	1, 2, rempl.
4	$\copyright\ (P_1T^2)T^1$	3, [**B**]

THEOREME 2 : $P_{passif}\ T \rightarrow P_{trans.}\ T\ \varnothing_0$

1	$P_{passif}\ T$	hyp.
2	$[P_{passif} \equiv \mathbf{C}\ P_{trans}\ \varnothing_0]$	loi linguistique
3	$\mathbf{C}\ P_{trans.}\ \varnothing_0\ T$	1, 2, rempl.
4	$\mathbf{C}\ P_{trans.}\ \varnothing_0\ T$	3, [**C**]

THEOREME 3 : $P_{moyen}\ T \rightarrow \copyright\ (P_1T)T$

1	$P_{moyen}\ T$	hyp.
2	$[P_{moyen} \equiv \mathbf{W}(\mathbf{B} \copyright P_1)]$	loi linguistique
3	$\mathbf{W}(\mathbf{B} \copyright P_1)\ T$	1, 2, rempl.
4	$\mathbf{B} \copyright P_1\ T\ T$	3, [**W**]
5	$\mathbf{C}(P_1T)T$	4, [**B**]

THEOREME 4 : $P_{impers.}\varnothing_\Delta \rightarrow P_0$

1	$P_{impers.}\ \varnothing_\Delta$	hyp.
2	$[P_{impers.} \equiv \mathbf{K}\ P_0]$	loi linguistique
3	$\mathbf{K}\ P_0\ \varnothing_\Delta$	1, 2, rempl.
4	P_0	3, [**K**]

THEOREME 5 : $P_{refl.}\ T \rightarrow P_{trans.}\ T\ T$

1	$P_{refl.}\ T$	hyp.
2	$[P_{refl.} \equiv \mathbf{W}\ P_{trans.}]$	loi linguistique
3	$\mathbf{W}\ P_{trans.}\ T$	1, 2, rempl.
4	$P_{trans.}\ T\ T$	3, [**W**]

THEOREME 6 : $P_{caus.}\ T^2T^1 \rightarrow \text{CAUSE}\ (P_1T^2)T^1$

1	$P_{caus.}\ T^2T^1$	hyp.
2	$[P_{caus.} \equiv \mathbf{B}\ \text{CAUSE}\ P_1]$	loi linguistique
3	$\mathbf{B}\ \text{CAUSE}\ P_1T^2T^1$	1, 2, rempl.
4	$\text{CAUSE}\ (P_1T^2)\ T^1$	3, [**B**]

8.2. Revenons maintenant à la phrase *Noémon a épousé Marie* que nous avons déjà analysée d'un point de vue strictement morpho-syntaxique. Nous allons maintenant considérer que le morphème discontinu *a-é* est un opérateur complexe obtenu en combinant, au moyen du combinateur **B**, les deux opérateurs *a* (de "avoir") et *é* (trace morphologique du "participe passé"), selon la loi linguistique suivante :

$[a\text{-}é \equiv \mathbf{B}\ a\ é]$

Nous avons (en notation préfixée) la réduction suivante :

(a-é) (épous) Marie Noémon → *a (é(épous)) Marie Noémon.*

En effet, nous avons :

1	*(a-é) (épous) Marie Noémon*	hyp.
2	$[a\text{-}é \equiv \mathbf{B}\ a\ é]$	loi linguistique
3	**B** *a é (épous) Marie Noémon*	1, 2, rempl.
4	*a(é(épous)) Marie Noémon*	3, [**B**]

Il est clair que le verbe "*avoir*" fonctionne au pas 4 comme un auxiliaire. Désignons par a_{aux} cette occurrence de "*avoir*" à valeur d'auxiliaire. Nous allons maintenant relier le a_{aux} au relateur de possession, désigné par a_{poss}, au moyen de la loi linguistique suivante :

$[a_{aux} \equiv \mathbf{B}\ a_{poss}]$

Nous pouvons alors continuer la réduction précédente :

5	a_{aux}*(é(épous)) Marie Noémon*	hyp.
6	$[a_{aux} \equiv \mathbf{B}\ a_{poss}]$	loi linguistique
7	**B** a_{poss} *(é(épous)) Marie Noémon*	5,6, rempl.
8	a_{poss} *(é(épous) Marie) Noémon*	7, [**B**]

Nous avons ainsi démontré, à l'aide des deux lois précédentes, les relations de réduction :

(a-é) (épous) Marie Noémon

→ a_{aux} *(é(épous)) Marie Noémon*

→ a_{poss} *(é(épous) Marie) Noémon*

8.2.1. Les types syntaxiques des opérateurs *a-é*, *é*, a_{aux} et a_{poss} sont respectivement :

00^2ttp0^2ttp	: *a-é*
$00^2ttp0tt$	: *é*
$00tt0^2ttp$	: a_{aux}
0t0tp	: a_{poss}

En effet, le type de l'opérateur *a-é* est celui d'un déterminant de prédicat ; ce morphème de temps ("tense") associe à un prédicat non temporalisé un prédicat temporalisé, d'où son type : 00^2ttp0^2ttp. L'opérateur *é* est un morphème de "participe passé" qui transforme un prédicat en un déterminant nominal (ou adjectif). L'opérateur a_{aux} transforme l'opérande adjectival *épous-é* en un prédicat binaire *a-épous-é*, d'où son type : $00tt0^2ttp$. Par contre, l'opérateur a_{poss} transforme le terme nominal (*Marie épousée*) en un prédicat unaire, d'où son type : 0t0tp.

La loi $[a_{aux} \equiv \mathbf{B}\ a_{poss}]$ qui relie a_{aux} à a_{poss} fait apparaître le changement de fonction syntaxique du marqueur *a*. En tant qu'auxiliaire, *avoir* transforme un participe, ayant une fonction syntaxique d'adjectif, en un prédicat unaire. En tant que relateur de possession *avoir* transforme un terme nominal (dans l'exemple : *Marie épousée*) en un prédicat unaire (dans l'exemple : *a* (*Marie épousée*)). Dans les deux cas cependant *avoir* assume la fonction syntaxique de construction d'un prédicat. Quant à l'opérateur *a-é*, il assume la fonction d'un affixe temporel qui détermine le prédicat sans changer sa fonction syntaxique. La loi linguistique précédente nous indique explicitement que le rôle d'auxiliaire est dérivé de celui du relateur de

possession, mais les fonctions syntaxiques ne sont pas conservées, c'est ce qu'indiquent les deux types différents attribués à a_{aux} et a_{poss}.

8.2.2. La cohérence des types assignés aux opérateurs *a-é*, *é*, a_{aux} et a_{poss} est bien assurée par les deux lois précédentes. En effet, rappelons que le schéma de type assigné au combinateur **B** est :

$00\beta\gamma 00\alpha\beta 0\alpha\gamma$: **B**

où α, β, γ sont des variables de type.

Pour la loi [$a_{aux} \equiv$ **B** a_{poss}], il suffit de poser : α = t ; β = t ; γ = 0tp, pour vérifier la condition d'isotypicalité :

$$\frac{00t0tp00tt0t0tp : \mathbf{B} \qquad 0t0tp : a_{poss}}{00tt0t0tp : a_{aux} \qquad 0tt : \textit{épousé}}$$

$$0t0tp : a_{aux}(\textit{épousé})$$

Pour la loi [*a-é* ≡ **B** a_{aux}*é*], il suffit de poser : α = 0t0tp ; β = 0tt ; γ = 0t0tp, pour vérifier l'isotypicalité.

8.2.3. En généralisant ce qui précède, nous démontrons immédiatement les réductions suivantes :

$$(a\text{-}é)\ P_2\ T^2T^1 \rightarrow a_{aux}\ (é\ P_2)\ T^2T^1 \rightarrow a_{poss}(é\ P_2T^2)T^1$$

où : P_2 est un prédicat transitif de type 0t0tp ;
T^1 et T^2 sont deux termes nominaux de type t ;
é un opérateur transformant un prédicat en un déterminant nominal ("participe passé") ;
a_{aux} et a_{poss} sont deux opérateurs reliés par la loi :

[$a_{aux} \equiv$ **B** a_{poss}] ;

a-é est un opérateur complexe défini à partir de a_{aux} par la loi :

[*a-é* ≡ **B** a_{aux} *é*].

Les réductions précédentes trouvent un appui dans l'évolution diachronique des langues. Citons, par exemple :

"Un exemple frappant de déplacement du contenu sémantique dû à l'extension progressive d'une forme grammaticale nous est offert, en anglais, par le *parfait* (*perfect*). En anglais moderne deux phrases comme "*he has written the book*" (il a écrit le livre en question) et "*he has the book written*" (il fait écrire le livre) sont commodément distincte l'une de l'autre par le simple effet d'un ordre rigide des mots. Elles sont la résultante d'une différenciation qui remonte à une époque où l'ordre des mots restait libre. A cette époque, *le verbe "have" n'était pas un*

verbe auxiliaire, mais un verbe plein, avec le sens de posséder, avoir à soi [c'est nous qui soulignons] (...). Le déplacement du sens entre "to have the book written" et "to have written the book" est compréhensible. La première construction met l'accent sur l'état qui résulte d'une action antérieure (exprimée par le participe passé) ; la seconde met l'accent sur l'action antérieure incluse dans le résultat de l'action (...) dans la première construction *have* joue le rôle de verbe *auxiliaire*, dans la seconde il garde la valeur du verbe *plein*. (...)". Jerzy Kurylowicz : "L'évolution des catégories grammaticales"; *Problèmes du langage*, coll. Diogène, Gallimard, 1966, pp. 57-58.

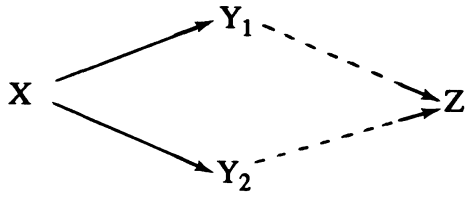
Nous pourrions citer d'autres linguistes, comme E. Benveniste ou R. Jakobson, dont les analyses du parfait seraient formalisées adéquatement par le changement de statut de *avoir* de "verbe plein" en "auxiliaire".

9. THEOREME DE CHURCH-ROSSER ET PARAPHRASES GRAMMATICALES.

La propriété de Church-Rosser (ou propriété du carreau) est possédée par une relation transitive '$\rightarrow$' lorsque nous avons :

[C.R.] *Si* $X \rightarrow Y_1$ *et* $X \rightarrow Y_2$ *alors* $\exists$ Z tel que $Y_1 \rightarrow Z$, $Y_2 \rightarrow Z$

ce que nous résumons par le diagramme :



Il existe des relations qui n'ont pas la propriété [C.R.].

THEOREME (de Church-Rosser) : *La relation de réduction* '$\rightarrow$'*de la logique combinatoire possède la propriété de Church-Rosser*.

La preuve est complexe. Le lecteur peut se reporter à celle de P. Martin-Löf et de W.W. Tait (voir Hindley, Seldin, cité dans la bibliographie générale).

L'importance du théorème repose sur ses corollaires, dont les démonstrations sont immédiates.

COROLLAIRE 1 : *Une e.a. a au plus une seule forme normale.*

COROLLAIRE 2 : *Si* X = Y, *alors il existe un* Z *tel que* $X \rightarrow Z$ *et* $Y \rightarrow Z$.

COROLLAIRE 3 : *Si* X = Y *et* Y *est dans une forme normale, alors* $X \rightarrow Y$.

COROLLAIRE 4 : *Si* X = Y *alors : soit* X *et* Y *n'ont pas de forme normale, soit* X *et* Y *ont la même forme normale.*

COROLLAIRE 5 : *Si* X *et* Y *sont deux e.a. qui sont dans une forme normale, alors :*
$[X \not\equiv Y] \Rightarrow [X \neq Y]$.

9.1. Le théorème de Church-Rosser a une interprétation structurante pour les langues naturelles.

Les formes normales (désormais f.n.) auxquelles se réduisent des e.a. (ou e.a.t.) sont les représentants canoniques d'une famille d'e.a. (ou e.a.t.) équivalentes :

X *et* Y *sont deux e.a. (ou e.a.t.) qui sont équivalentes lorsque* X *et* Y *se réduisent à une même f.n.* Z.

La f.n. Z commune à X et à Y et à toutes les e.a. (ou e.a.t.) de la famille représente l'invariant irréductible de la famille. La relation de réduction modélise ainsi un type très important de relations paraphrastiques, tout particulièrement les relations paraphrastiques fondées sur les relations grammaticales.

L'assertion 'X → Y' signifie que l'e.a. X est réductible à Y ou encore que Y *est une réduction paraphrastique de* X. Lorsque X et Y se réduisent à la même f.n. Z, X et Y admettent pour paraphrase Z et, par symétrisation, X et Y sont paraphrastiquement équivalentes entre elles.

L'explicitation d'une expression X est donnée par une paraphrase censée être plus claire. Par exemple, pour expliquer la construction d'un prédicat transitif $P_{trans.}$ on a recours à l'explicitation à l'aide de la notion de "contrôle intentionnel" : $P_{trans.}\ T^2T^1 \rightarrow \copyright\ (P_1T^2)\ T^1$; c'est-à-dire : l'agent T^1 a le contrôle intentionnel du processus de transformation qui affecte le patient T^2 et qui est orientée vers la situation exprimée par P_1T^2.

9.2. Pour expliquer la différence entre la *voix moyenne*, la *voix transitive active* et la *voix passive*, nous avons eu recours aux trois réductions suivantes :

(1) $P_{trans.}\ T^2T^1 \rightarrow \copyright\ (P_1T^2)\ T^1$
(2) $P_{moyen}\ T \rightarrow \copyright\ (P_1T)\ T$
(3) $P_{pass.}\ T \rightarrow P_{trans.}\ T\ \varnothing_0 \rightarrow \copyright\ (P_1T)\ \varnothing_0$

Introduisons maintenant la *voix médio-passive*, (exemple : *les fleurs se fanent trop rapidement*), on a la réduction suivante :

(4) $P_{médio\text{-}pass.}\ T \rightarrow \copyright\ (P_1T)\ \varnothing_\Delta$

en analysant le prédicat complexe $P_{médio\text{-}pass.}$ par :

$[P_{médio\text{-}pass.} \equiv \mathbf{C}(\mathbf{B} \copyright P_1)\ \varnothing_\Delta]$

En effet, on a le théorème suivant : $P_{médio\text{-}pass.}\ T \rightarrow \copyright\ (P_1T)\ \varnothing_\Delta$.

1	$P_{médio\text{-}pass.}\ T$	hyp.
2	$[P_{médio\text{-}pass.} \equiv \mathbf{C}\ (\mathbf{B} \copyright P_1\ \varnothing_\Delta]$	loi linguistique
3	$\mathbf{C}(\mathbf{B} \copyright P_1)\ \varnothing_\Delta\ T$	1, 2, rempl.
4	$\mathbf{B} \copyright P_1\ T\ \varnothing_\Delta$	3, [**C**]
5	$\copyright\ (P_1\ T)\ \varnothing_\Delta$	4, [**B**]

Les quatre constructions (voix transitive active, moyenne, passive, médio-passive) sont ramenées à quatre f.n. différentes qui *expriment les significations intrinsèques de chaque voix.*

. Pour la voix transitive active, l'agent T^1 a le contrôle intentionnel sur le processus de transformation affectant le patient T^2.

. Pour la voix moyenne, le terme nominal T assume un double rôle : il contrôle le processus, il est affecté par ce processus.

. Pour la voix passive (courte), on se ramène à un schéma transitif actif, où l'agent n'est plus spécifié ($\varnothing_0$) ; c'est pourquoi, bien qu'affirmant la présence d'un agent, la voix passive lui fait jouer un rôle "effacé" et, à la suite d'une conversion, "l'objet" de l'actif assume la fonction syntaxique de sujet du prédicat intransitif P_{passif}.

. Pour la voix médio-passive, il n'y a aucun agent (même non spécifié) qui contrôle le processus de transformation, ce dernier affecte le sujet syntaxique de la construction. Le terme $\varnothing_\Delta$ est uniquement un terme qui occupe une position syntaxique sans dénotation, comme dans les constructions impersonnelles.

Ces f.n. représentent les significations intrinsèques des voix, en explicitant, à l'aide d'écritures symboliques, les rôles qui sont assumés par les actants. Ces f.n. explicitent aussi les relations qui existent entre ces quatre types de voix, notamment en mettant en évidence le caractère moyen et intermédiaire de la "voix moyenne" et de la "voix médio-passive" (intermédiaire entre le passif et le moyen).

(1) Dans l'actif, les rôles d'agent et de patient sont clairement distingués :*Jean écrit une lettre* ; *Paul lit un livre* ; *Pierre monte les valises.*

(2) Dans le moyen, les rôles d'agent et de patient sont assumés par le même actant, mais il y a toujours contrôle par un agent explicite, comme dans l'actif : *Jean s'asseoit* ; *Jean se soigne.*

(3) Dans le passif, le rôle de l'agent est toujours présent mais on tend à l'effacer et à lui faire jouer un rôle secondaire : *le daim a été tué.*

(4) Dans le médio-passif, le rôle de l'agent n'est plus assumé par un actant (même non spécifié) ; le mécanisme de contrôle n'est plus assumé par un agent intentionnel : *les feuilles se fanent ; la boue se sèche facilement (au soleil) ; le cancer se soigne bien.*

Dans certaines langues (dont le français), des marqueurs morphologiques identiques codent des fonctionnements différents (voix différentes) mais cependant assez proches : le moyen, le médio-passif sont souvent codés à l'aide du même marqueur *se* : *Jean se fâche* (moyen) ; *la machine se détériore trop vite* (médio-passif). Le passif lui-même a deux encodages, l'un avec "*être* + le participe passé" (*Les livres ont-été-rangés* (*par quelqu'un*)), l'autre avec le marqueur *se* (*ces fruits se mangent facilement* → *on mange facilement ces fruits*) ; le premier encodage du passif marque une opération aspectuelle d'état, le second une valeur aspectuelle de processus.

Remarque : Pour la discussion linguistique de ce que nous avançons ici, le lecteur se reportera à nos publications, en particulier à celles qui ont été écrites en collaboration avec S. Shaumyan et Z. Guentchéva en 1985 et 1986.

9.3. L'ambiguïté est inséparable de la paraphrase. Dire qu'une expression X est ambiguë, c'est montrer qu'elle est paraphrastiquement équivalente à au moins deux autres expressions X_1 et X_2 qui se réduiront à deux f.n. distinctes Z_1 et Z_2 : l'expression X entre alors dans deux familles paraphrastiques. Par exemple, affirmer que la phrase *la peur du gendarme effraie Paul* est ambiguë, c'est montrer qu'elle admet deux réductions, l'une à : (a) *le gendarme a peur et cela effraie Jean* ; l'autre à : (b) *Paul est effrayé parce qu'il a peur du gendarme.* De même, la

phrase ambiguë *Paul fait servir le café à Pierre* a deux réductions : (a) l'une active : *Paul fait que Pierre serve le café* ; (b) l'autre passive : *Paul fait qu'on serve le café à Pierre.* Les deux réductions conduisent à deux représentations différentes :

(a') CAUSE (QU (*sert café Pierre*)) *Paul*
(b') CAUSE (QU (C *sert* $\varnothing_0$ *à* (*Pierre*) *café*) *Paul* .

9.4. Etant donnée une langue naturelle L_N que l'on peut considérer comme un ensemble de phrases empiriques (au moins en première approximation). Chaque phrase est représentée par e.a.t. X d'un *langage applicatif* L_1 engendré à partir des opérateurs linguistiques de différents types, des combinateurs et des prédicats complexes. A chaque phrase ambiguë X, on associe autant d'occurrences de X qu'il y a d'ambiguïtés. Ces occurrences X_1, X_2, ..., X_n appartiennent à un langage applicatif L_D. Dans ce langage L_D, on procède aux réductions, en faisant appel aux lois linguistiques. En particulier, une expression X de L_1, dupliquée dans L_D en X_1, ..., X_n, ne fera pas appel aux mêmes mécanismes de réduction et donc aux mêmes lois linguistiques. Chaque réduction de X_i conduit à une f.n. Y_i. Par le théorème de Church-Rosser, on sait que si la f.n. Y_i existe, alors elle est unique. Par conséquent, si X_i et X_j se réduisent à une même f.n. Y_{ij}, les deux expressions de L_D appartiennent à la même famille paraphrastique et il n'y a pas lieu de distinguer X_i de X_j.

L'ensemble des f.n. constitue un sous langage L_0 du langage L_D. Les f.n. sont les représentants des familles paraphrastiques. Chaque f.n. est l'invariant prédicatif commun à la famille des paraphrases dans L_D. Les processus de réductions déterminent une surjection de L_D sur L_0 qui apparaît comme l'ensemble quotient de L_D par la relation d'équivalence "X = Y *si et seulement si* X *et* Y *appartiennent à la même famille paraphrastique de f.n.* Z". Dans le cas où X = Y, X et Y se réduisent donc à la même f.n. Z de L_0.

Le langage L_D est donc *stratifié* puisqu'il contient un sous langage L_0 ; il est aussi structuré en classes d'équivalences Z_1, ..., Z_n, ... associées aux f.n. de L_0. En revenant à la langue L_N, on voit qu'elle est aussi stratifiée puisqu'elle contient une sous langue L_{N0} qui est représentée par le langage L_0.

Cette analyse est une des conséquences du théorème de Church-Rosser. En effet, si l'unicité de la réduction n'était pas garantie, alors aucune stratification ne serait possible et aucune structuration en classes d'équivalence de paraphrases ne serait alors plus envisageable.

9.5. Les stratifications et structurations précédentes appellent trois remarques.

(1) Il ne s'agit nullement d'une stratification comparable à celle de la Grammaire Générative de N. Chomsky. Le langage L_0 ne doit pas être assimilé aux structures profondes et L_D aux structures de surface. Les relations entre structures profondes et superficielles sont d'une toute autre nature.

(2) Le théorème de Church-Rosser et le formalisme de la logique combinatoire transcendent des "écoles" linguistiques particulières. Une stratification analogue se retrouve en effet à la fois dans le modèle d'analyse de Z.S. Harris (1968, 1982) et dans le modèle de la Grammaire Applicative Universelle de S. K. Shaumyan (1977, 1987). Dans ce dernier modèle, le langage génotype est un métalangage de description qui est comparable au langage L_1. Le génotype comprend un sous langage (qui ne doit pas être confondu avec la structure profonde de la Grammaire Générative).

Sur la parenté des modèles de Harris et de Shaumyan, le lecteur se reportera à notre article de 1975.

(3) On peut continuer la décomposition des éléments du sous-langage L_0, notamment en analysant les prédicats lexicaux à l'aide de primitives sémantiques combinées entre elles par les combinateurs de la logique combinatoire. Le lecteur se reportera à nos publications parues sur ce sujet depuis 1986 (voir la bibliographie générale).

10. CONCLUSION

Le théorème de Church-Rosser inséré dans le cadre de la logique combinatoire nous paraît être un exemple de théorème mathématique profond qui a une interprétation non triviale dans un domaine des sciences humaines, au même titre que le célèbre théorème de von Neumann dans la théorie des jeux, ou le théorème de Chomsky-Schützenberger dans l'étude de la complexité syntaxique des langues naturelles.

REFERENCES BIBLIOGRAPHIQUES

Voir la bibliographie commentée dans ce même numéro.