Aix*Marseille université

BULLETIN D'INFORMATIQUE APPROFONDIE ET APPLICATIONS

COMPUTATION - INFORMATION

Volume 91 - Mars 2012

FONDATEUR *Edmond Bianco*

Publication trimestrielle, gratuite, de l'Université d'Aix - Marseille

http ://www.univ-provence.fr/biaa

Impression : septembre 2013

ISSN 0291 - 5413

Le bulletin d'informatique approfondie et applications est une revue pluridisciplinaire destinée à éclairer les connaissances fondamentales informatiques.
Les fondements sont un domaine vaste allant de la structure intérieure de l'ordinateur, où se matérialise la machine universelle, à l'algorithme qui devient programme, pour aboutir à la notion de système.

Nous contribuons ainsi à ce que les autres disciplines plus anciennes (sciences humaines et de la société, sciences de la matière et de l'énergie, sciences mathématiques, sciences de la nature, sciences de la terre, sciences de l'univers, sciences de la vie, etc.) n'aient pas tendance à considérer l'informatique comme un simple outil définitivement figé.

Il importe de continuer à maîtriser les développements fondamentaux de l'informatique pour que nos disciplines puissent en tirer un meilleur parti.
Notre publication est ouverte à l'ensemble de la communauté scientifique. Le périodique est diffusé vers les bibliothèques universitaires de France et vers quelques bibliothèques des cinq continents.

Dépôt légal : décembre 2011

ISSN 0291 - 5413

Couverture : graffiti gare Saint Charles à Marseille. Graffeur inconnu.

EDITORIAL

Le jeune Bianco aux prises avec *un* texte, acte 1

André Lentin

Résumé. – La journée du 30 mars 2012 est consacrée à un Edmond Bianco, oeuvrant, tout au long de sa pleine maturité et jusqu'au seuil de la vieillesse, « au bénéfice » de la nouvelle science que l'on nomme l'informatique. Fondements logico-épistémologiques, mais aussi avancement des techniques de pointe concernant la pratique, tout l'aura intéressé. Que l'on veuille bien permettre à un très vieil homme d'évoquer « en contrepoint » un jeune Edmond Bianco, analyste et programmeur, à l'aube de sa carrière.

J'ai fait la connaissance de Bianco à l'époque où le Laboratoire de Calcul Numérique du CNRS, ayant pour directeur le professeur René de Possel, et pour sous-directeur un certain André Lentin, venait enfin de quitter les caves de l'Institut Henri Poincaré et de s'installer dans de vastes locaux au nord de Paris, rue du Maroc, où il était appelé à s'épanouir en Institut Blaise Pascal. De nombreux postes ont été créés pour lui, qu'il s'agit donc maintenant de pourvoir. Et c'est alors que nous arrive de Grenoble, bardé de diplômes, le jeune Edmond Bianco.

(Il semble ne pas s'être harmonieusement entendu avec le professeur Kuntzmann qui dirigeait le Centre d'enseignement de la nouvelle discipline : leurs caractères ne s'accordaient guère). Je l'ai accueilli. Le courant passe et s'éveille le désir de collaborer. Assez rapidement s'est présentée une occasion extrêmement intéressante dont je vais longuement parler : ce sera le noyau central de ma contribution.

Il s'agit d'une demande que nous présente un jeune latiniste, ancien élève de l'Ecole Normale Supérieure, du nom de Charpin. Une demandè qui nécessite de longues explications. Elle concerne un livre écrit par un grammairien latin de l'antiquité tardive nommé Nonius Marcellus.

Charpin est d'ailleurs arrivé avec un gros livre relié à l'ancienne : c'est la présentation et la publication en latin de l'œuvre en question qu'en a faite en 1872 notre grand et vénérable Ludovic Quicherat. L'éditeur – pardon, : le bibliopola - en était Hachette.

Nonius Marcellus ? De nos jours, bien des latinistes ne connaissent (au mieux !) que son nom. Il n'en a pas toujours été ainsi. Au cours de ce que l'on appelle « La Renaissance Carolingienne », c'était le « manuel de base » pour enseigner la langue des auteurs classiques.

A suivre dans le numéro 92 du bulletin.

Hommage à Edmond Bianco : De la Machine Universelle au Traitement Automatique du Langage Naturel

Nadia Mesli

nadia.mesli@univ-amu.fr

1. PREAMBULE

Le *Bulletin d'Informatique Approfondie & Applications* (BIAA) fête en 2012 ses 30 ans d'existence, ce qui me ramène inévitablement 26 ans en arrière, c'est-à-dire à l'année 1986, date de ma rencontre avec le fondateur de cette revue scientifique, Edmond Bianco. J'étais alors étudiante en thèse de doctorat nouveau régime à la Faculté de Lettres d'Aix-en-Provence – Département d'Etudes Germaniques. Ma thèse était dirigée par Daniel Bresson, professeur de linguistique allemande, et avait pour domaine d'étude les locutions verbales dans les écrits de Martin Luther (Mesli 1989). Par « locution verbale », il faut entendre une construction verbo-nominale à statut de verbe composé telle que *faire du sport*, *donner un conseil* ou *prendre une décision*, où le verbe (appelé *verbe support*) a perdu une partie plus ou moins grande de son sens lexical de verbe plein et où le nom (appelé *nom prédicatif*), souvent dérivé d'un verbe, est porteur du sens lexical de l'expression. A ce stade de mon récit, on pourrait se demander quel rapport il y aurait entre la linguistique allemande que j'étudiais et l'informatique fondamentale enseignée par Edmond Bianco à cette époque !

Encore aucun lien direct, mais un lien indirect si l'on considère le double corpus dont je disposais dans le cadre de mes recherches et qui était prêt à être exploité sous forme informatique. Il s'agissait en effet de fiches papier recensant tous les exemples trouvés pour les locutions verbales que j'étudiais et qui étaient classées par ordre alphabétique des noms prédicatifs ainsi que des verbes supports qu'elles contenaient. Le traitement automatique de mes données devenait nécessaire et par chance, au même moment, la Faculté de Lettres commençait à doter tous ses bureaux d'équipements informatiques. De nombreux stages d'initiation à la micro-informatique y étaient délivrés pour le personnel de même que des cours d'informatique plus avancés pour les étudiants. Poussée par mon directeur de recherche, j'ai donc participé à tous les stages et cours d'informatique susceptibles de m'aider dans cette nouvelle tâche, ce qui m'a amenée à créer ma base de données centrale sous le logiciel DBASE III+ comprenant 6201 fiches (ou enregistrements) de 30 rubriques (ou champs) chacune. Mon aventure informatique démarrait alors et ma rencontre avec Edmond Bianco devenait imminente.

En effet, tous ces cours d'informatique appliquée avaient ouvert mon horizon et l'idée de combiner mes compétences linguistiques à mes nouvelles aptitudes informatiques commençait à germer dans mon esprit. J'envisageais alors de suivre un enseignement plus poussé en informatique théorique à la Faculté des Sciences de Luminy, ce qui n'était pas une chose aisée en tant qu'étudiante en Lettres, même en étant titulaire d'un baccalauréat série C (Mathématiques & Sciences Physiques). Comment intégrer une Faculté de Sciences tout en étant une étudiante en Lettres et quelle formation suivre lorsque l'on sait qu'il n'existait pas vraiment de passerelles entre ces deux disciplines ? Ma rencontre avec Edmond Bianco fut décisive. Son ouverture d'esprit ainsi que sa foi en l'interdisciplinarité lui ont permis de me faire confiance et d'accepter mon inscription au Certificat d'Etudes Supérieures d'Université (C.E.S.U.) en Informatique Fondamentale, formation délivrée à l'époque par le Département

de Mathématiques de Luminy (Université d'Aix-Marseille II). Il s'agissait d'un sous-ensemble de modules d'enseignement du deuxième cycle de Mathématiques aboutissant à la Maîtrise mention « Ingénierie Mathématique ». Edmond Bianco assurait dans le programme de l'année universitaire 1986-1987 le module intitulé « Algorithmique – Structures d'ordinateurs » alors que Mmes Preller et Donnadieu assuraient le module de « Logique et Calculabilité » et Mr Lehmann celui des « Structures câblées – Structures logicielles » :

M2 : Algorithmique – Structures d'ordinateurs (Mr Bianco)

Algorithmique

Modèles mathématiques de l'ordinateur
- Machine de Turing
- Machine de Nolin
- Thèse de Church, hypothèse fondamentale de la théorie des algorithmes

Algorithme d'une machine universelle
- La notion de compilateur
- Démonstrations algorithmiques d'équivalences des machines

L'algorithmique et des implications sur la structure des langages de programmation

Algorithmique des processus simultanés
- Réseaux de Pétri
- Grafcet

Structures d'ordinateurs

Unité centrale
- Processeur, machine universelle
- Mémoire, configuration

Périphériques
- Calculateurs d'échanges
- Structures de dispositifs d'échanges
  - Disques, bandes, clavier, écran, imprimante
- Structure de bus
- Notion de système

M3 : Logique et Calculabilité (Mme Preller, Mme Donnadieu)

Déduction automatique
- Formes de Skolem
- Formes prénexes
- Clause de Horn
- Principe de la programmation logique
- Satisfaisabilité au sens de Gentzen
- Satisfaisabilité au sens de Herbrand

Théorie générale des algorithmes
- Récursivité (au sens arithmétique)

λ – calcul
Codage des programmes
Programme universel

Décidabilité
Semi-décidabilité – Indécidabilité
Insolvabilité des problèmes
Contrôle des procédures dans un algorithme

M9 : Structures câblées – Structures logicielles (Mr Lehmann)

Automates combinatoires
Automates d'états finis
Mémoires
L'ordinateur et le processeur comme assemblage d'automates
Séquence microprogrammation
Isomorphismes entre structures câblées et logicielles

Je dois dire que tous ces cours étaient d'un abord très difficile pour un public non scientifique et mes connaissances acquises jusque là en informatique appliquée ne semblaient guère m'aider dans la compréhension de ces enseignements fondamentaux dont je ne percevais pas encore tout l'intérêt. J'avais le sentiment que mon profil de littéraire / linguiste avait éveillé dès le départ la curiosité d'Edmond Bianco qui semblait être le seul à voir le lien entre nos deux disciplines respectives. Et lien il y avait effectivement, sa grande intelligence ainsi que la clarté de son raisonnement m'ont montré progressivement tout l'intérêt d'un enseignement fondamental dans le cadre de l'informatique et au-delà. Qui allait croire que, quelques années plus tard, cet enseignement des modèles mathématiques de l'ordinateur tels que la machine de Turing, la machine de Nolin, la notion de compilateur ou la machine universelle allaient me conduire vers le traitement automatique du langage naturel et me montrer le véritable lien entre la linguistique et l'informatique, cette nouvelle discipline que je ne connaissais pas encore et que l'on nommait la *linguistique informatique* (*computational linguistics*) !

2. INFORMATIQUE FONDAMENTALE ET LINGUISTIQUE ALLEMANDE

En hommage à Edmond Bianco, je vais présenter ici un extrait de mon travail effectué dans le cadre du C.E.S.U. (Mesli 1987) qui illustre une application de l'informatique fondamentale au traitement automatique du langage naturel. En tant qu'étudiante spécialisée en linguistique allemande, j'avais choisi l'allemand comme langue d'application ainsi qu'un domaine grammatical ciblé : la notion de rection verbale. Je cite ici Bresson (2001 : 111) qui explique cette notion à travers l'exemple du verbe allemand **schenken** (*faire cadeau, offrir*) :

> « On appelle **rection des verbes et des adjectifs** la sélection des marques grammaticales (cas, préposition) qui caractérisent les diverses fonctions syntaxiques. Dans le cas du verbe **schenken**, le donneur est réalisé par une expression de type nominal (**Peter, mein Freund, er, wer auch immer**) au nominatif. Le bénéficiaire est représenté par une expression nominale de même type au datif (**Peter, meinem Freund, ihm / ihr, wem auch immer**). L'objet est représenté par une expression nominale à l'accusatif ou une subordonnée (**ein Buch, es, was du willst**). C'est ainsi que la rection du verbe **schenken** peut être décrite au moyen de l'expression suivante :
>
> jemand schenkt jemandem etwas *quelqu'un fait cadeau de quelque chose à quelqu'un* »

Il s'agissait alors de l'élaboration d'un langage de recherche et de calcul appliqué à un échantillon de verbes allemands pouvant se construire avec un objet au datif (verbes régissant le datif), de sa compilation en Procédure Formelle (langage machine) ainsi que de la réalisation de cette Machine Universelle en langage PASCAL. La structure et la syntaxe de ce langage y étaient tout d'abord présentées, puis la sémantique de ses instructions détaillée avant compilation. Je ne présenterai ici que la structure et la syntaxe de ce langage symbolique pour illustrer le lien entre Machine Universelle (MU) et Traitement Automatique du Langage Naturel (TALN). Les travaux de Bianco (1979, 1997, 1998) ou Bisière / Iwanesko / Knippel / Massat (1989) peuvent être cités en référence pour situer le cadre théorique de cette étude.

3. PRESENTATION DES PROCEDURES

Le langage de recherche et de calcul élaboré dans ce cadre est constitué de quatre procédures. Par « procédure », nous entendons un texte qui débute par un *en-tête* décrivant le nom de ce texte et qui s'achève sur le mot « fin ». Ces procédures seront explicitées dans les paragraphes qui suivent :

```
Système EXPL ;          déclarations
index Q ;
file F ;

début                   instructions

Boucle Syst ;
lire F ;
lire Q ;
Aig F = 'VERBDAT1' : e1, 'VERBDAT2' : e2, 'VERBDAT3' : e3 ;
e1 :    insérer VERBDAT1 (Q) ;
        print Q ;
        sortie F ;
e2 :    insérer VERBDAT2 (Q) ;
        print Q ;
        sortie F ;
e3 :    insérer VERBDAT3 (Q) ;
        print Q ;
        sortie F ;
        nœud F ;
rep Syst ;

fin
```

```
Programme VERBDAT1 ; déclarations
par Q ;
index J, K, N, R, S, T ;
booléen Z, Z1 ;
file    mot (1000), liste (4000), tabint (5000), L (200) ;
filext  table (200000) ;
expr
```

```
E1 = # $0$ ($1$) * jmd, $2$ * jmdm, $3$ # $4$,
E2 = & $5$ & $6$ ;

début                          instructions

table := vide ;                initialisation des files / de la filext
liste := vide ;
mot := vide ;

Boucle F (K := Q) ;            introduction et écriture de l'information
tabint := vide ;
        Boucle I (J) ;
        lire mot [.] ;
        tabint := tabint U mot ;
        tabint := tabint U '#' ;
        rep I (J = 5 ou mot = vide) ;
table (L(K)) := tabint ;
rep F (mot = vide) ;
        Q := K ;

Boucle R3 (N) ;                recherche de l'information
tabint := vide ;
        Boucle R1 (R) ;
        T := N - 1 ;
        T := T * 5 ;
        T := T + R ;
        input tabint, table (L(T)) ;
        rep R1 (R = 5) ;

        Boucle R2 (S) ;
                  m
        tabint, Z ← E1 ;
        Aig Z : suite ;
        suite : Boucle C
                liste := liste U '&' ;
                liste := liste U $0$ ;
                         m
                $4$, Z1 ← E1 ;
                Aig Z1 : suit1 ;
        suit1 : rep C ($4$ = vide) ;
        nœud Z1 ;
        nœud Z ;
        rep R2 (S = 5 ou tabint = vide) ;

rep R3 (N = K) ;

         m
liste, Z ← E2 ;                impression de l'information
        Boucle P ;
        print $5$ ;
```

```
                m
        $6$, Z1 ← E2 ;
        rep P ($6$ = vide) ;

fin
```

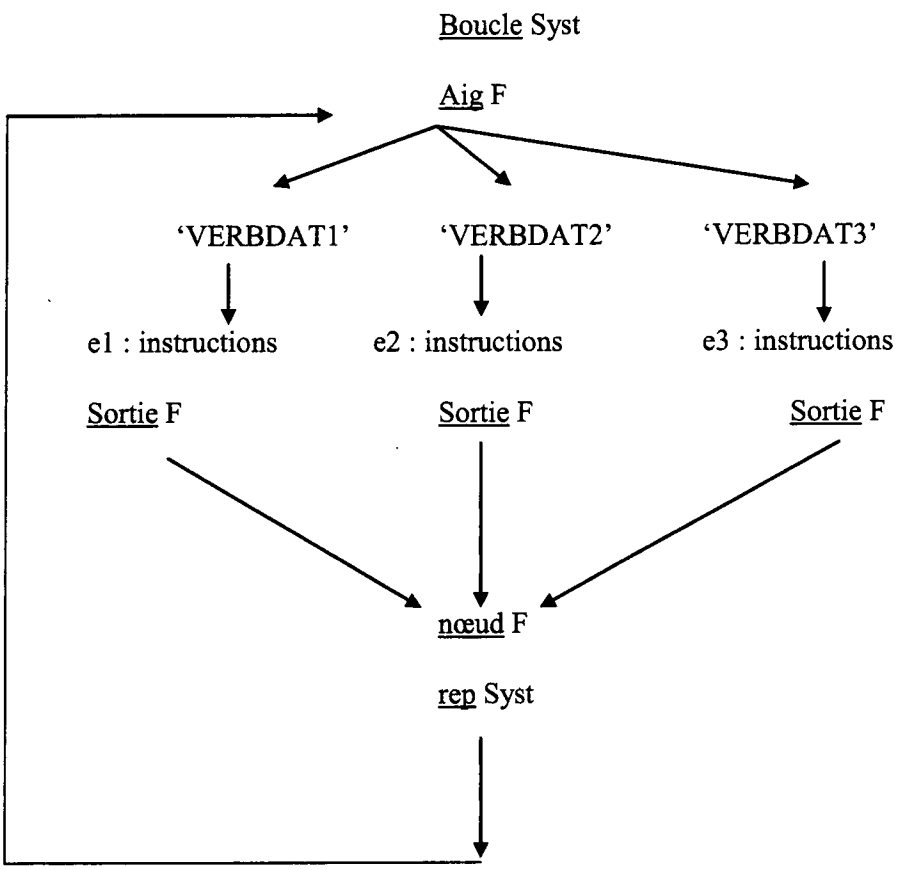
```
Programme VERBDAT2 ; déclarations
par Q ;
index J, K, N, R, S, T ;
booléen Z, Z1 ;
file    mot (1000), liste (4000), tabint (5000), L (200) ;
filext  table (200000) ;
expr
E1 = # $0$ ($1$) * $2$ GVdass $3$ * $4$ # $5$,
E2 = & $6$ & $7$ ;

début                          instructions

table := vide ;                initialisation des files / de la filext
liste := vide ;
mot := vide ;

Boucle F (K := Q) ;            introduction et écriture de l'information
tabint := vide ;
        Boucle I (J) ;
        lire mot [.] ;
        tabint := tabint U mot ;
        tabint := tabint U '#' ;
        rep I (J = 5 ou mot = vide) ;
table (L(K)) := tabint ;
rep F (mot = vide) ;
        Q := K ;

Boucle R3 (N) ;                recherche de l'information
tabint := vide ;
        Boucle R1 (R) ;
        T := N – 1 ;
        T := T * 5 ;
        T := T + R ;
        input tabint, table (L(T)) ;
        rep R1 (R = 5) ;

        Boucle R2 (S) ;
                m
        tabint, Z ← E1 ;
        Aig Z : suite ;
        suite : Boucle C
                liste := liste U '&' ;
                liste := liste U $0$ ;
```

```
                    m
            $5$, Z1 ← E1 ;
            Aig Z1 : suit1 ;
      suit1 : rep C ($5$ = vide) ;
      nœud Z1 ;
      nœud Z ;
      rep R2 (S = 5 ou tabint = vide) ;

rep R3 (N = K) ;

      m
liste, Z ← E2 ;              impression de l'information
      Boucle P ;
      print $6$ ;
            m
      $7$, Z1 ← E2 ;
      rep P ($7$ = vide) ;

fin
```

```
Programme VERBDAT3 ; déclarations
par Q ;
index J, K, N, R, S, T ;
booléen Z, Z1 ;
file    mot (1000), liste (4000), tabint (5000), L (200) ;
filext  table (200000) ;
expr
E1 = # $0$ ($1$) * $2$ + humain ± individu $3$ * $4$ + humain – individu + groupe $5$ #
$6$,
E2 = & $7$ & $8$ ;

début                        instructions

table := vide ;              initialisation des files / de la filext
liste := vide ;
mot := vide ;

Boucle F (K := Q) ;          introduction et écriture de l'information
tabint := vide ;
      Boucle I (J) ;
      lire mot [.] ;
      tabint := tabint U mot ;
      tabint := tabint U '#' ;
      rep I (J = 5 ou mot = vide) ;
table (L(K)) := tabint ;
rep F (mot = vide) ;
      Q := K ;

Boucle R3 (N) ;              recherche de l'information
tabint := vide ;
```

```
        Boucle R1 (R) ;
        T := N – 1 ;
        T := T * 5 ;
        T := T + R ;
        input tabint, table (L(T)) ;
        rep R1 (R = 5) ;

        Boucle R2 (S) ;
                m
        tabint, Z ← E1 ;
        Aig Z : suite ;
        suite :  Boucle C
                liste := liste U '&' ;
                liste := liste U $0$ ;
                        m
                $6$, Z1 ← E1 ;
                Aig Z1 : suit1 ;
        suit1 :  rep C ($6$ = vide) ;
        nœud Z1 ;
        nœud Z ;
        rep R2 (S = 5 ou tabint = vide) ;

rep R3 (N = K) ;

        m
liste, Z ← E2 ;                    impression de l'information
        Boucle P ;
        print $7$ ;
                m
        $8$, Z1 ← E2 ;
        rep P ($8$ = vide) ;

fin
```

La première procédure (Système EXPL) se distingue des autres procédures de par le rôle qu'elle joue : elle simule un *système*. Ce système se matérialise ici par une boucle (Boucle Syst) qui tourne sans fin sur elle-même et un aiguillage (Aig F). Il déroulera au choix le programme voulu :



C'est ici qu'intervient la notion d'*insertion de procédure* (insérer VERBDAT1 (Q) / VERBDAT2 (Q) / VERBDAT3 (Q)) et d'*initialisation de paramètre* (lire Q). Les trois autres procédures (Programme VERBDAT1 / VERBDAT2 / VERBDAT3) décrivent un calcul sur des mots – des chaînes de caractères dont la structure est particulière.

La notion de *mot* utilisée dans ce cadre est différente de la notion communément admise. Il faut entendre par « mot » un *enregistrement*, assimilable à un tableau d'un certain volume (longueur de l'enregistrement). Ces mots contiennent un certain nombre d'informations susceptibles d'être exploitées. Dans notre cas, ces mots contiennent des informations concernant l'emploi d'un échantillon de verbes allemands pouvant se construire avec un objet au datif. Ces mots sont constitués alors de trois parties : la première contient le verbe, suivi entre parenthèses de traductions possibles en français ; la seconde contient des informations sur la forme syntaxique et les caractéristiques sémantiques du sujet, et la troisième des informations sur la forme syntaxique et les caractéristiques sémantiques de l'objet au datif.

Le début et la fin du mot sont marqués par le signe « # » : le premier « # » fait partie du mot, le second est mis par le programme. La limite entre les différentes parties du mot se fait à l'aide du signe « * », ainsi que le montre l'exemple suivant que l'on pourra retrouver dans l'échantillon du fichier joint au paragraphe 3 :

ähneln (ressembler à) * jmd / etwas, GNn, non restreint * jmdm / einer Sache, GNd, non restreint

Les deux parties réservées au sujet et à l'objet donnent tout d'abord des formes syntaxiques possibles :

- Sujet : GNn, GNn / GVdass (groupe nominal ou pronominal au nominatif, même groupe au nominatif ou groupe verbal dépendant, introduit par la conjonction de subordination *dass*) ;

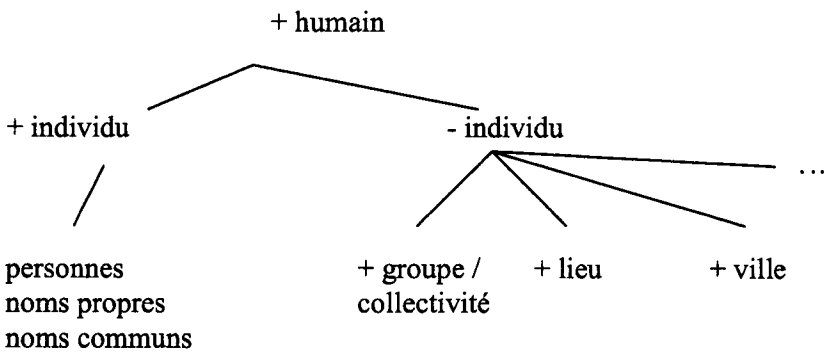
- Objet : GNd (groupe nominal ou pronominal au datif).

Les caractéristiques sémantiques générales du sujet et de l'objet peuvent être les suivantes (il s'agit d'abréviations de la terminologie allemande, *jmd* (forme du nominatif) pour *jemand* et *jmdm* (forme du datif) pour *jemandem*) :

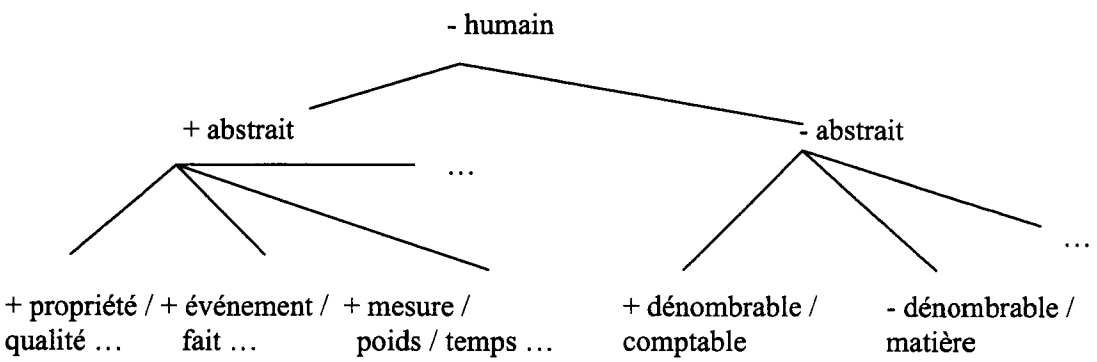
- Sujet : *jmd, jmd / etwas* (quelqu'un, quelqu'un ou quelque chose), sujet humain, sujet humain ou non humain) ;

- Objet : *jmdm, jmdm / einer Sache* (quelqu'un, quelqu'un ou quelque chose), objet humain, objet humain ou non humain).

En ce qui concerne les différents traits sémantiques utilisés, différentes combinaisons peuvent apparaître de façon linéaire à partir des schémas initiaux suivants (+ *humain,* - *humain*), le trait pouvant être *non restreint* :



Exemple : + humain ± individu
+ humain – individu + groupe



Exemple : - humain + abstrait + événement
+ humain ± abstrait

Chacun de ces traits peut être suivi par un exemple prototypique entre parenthèses. Ainsi, à chaque trait sémantique d'un sujet donné correspondra un trait sémantique de l'objet. Nous prendrons à titre d'exemple le verbe *absagen* qui peut se construire avec un objet au datif (*se décommander auprès de / renoncer à ...*). La présentation sous forme de tableau du mot mettra en évidence la symétrie entre les deux parties du sujet et de l'objet :

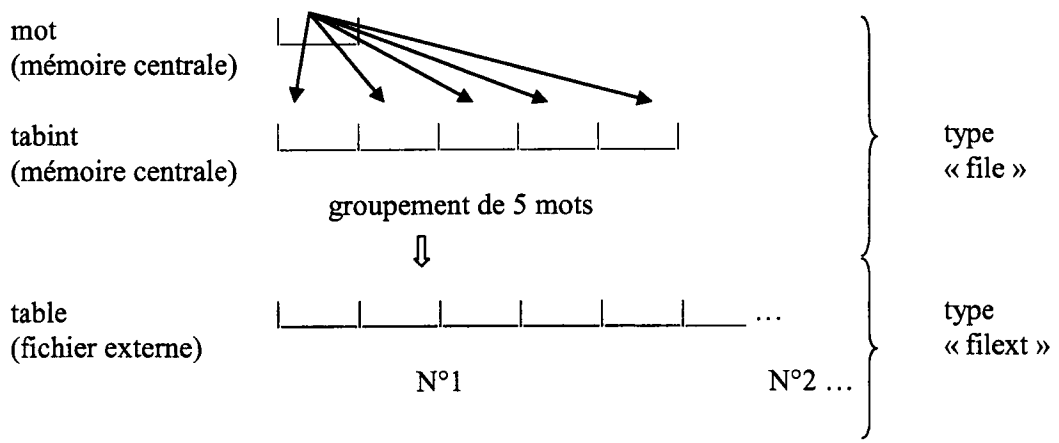
SUJET	**OBJET**
jmd	jmdm / einer Sache
GNn	GNd
+ humain ± individu	+ humain ± individu
+ humain ± individu	- humain - abstrait (*dem Alkohol / einem Laster absagen*)
+ humain ± individu	- humain + abstrait + pensée / parole / croyance (*seinem Glauben / einer Lehre absagen*)

Ces trois procédures ne se distinguent que par la nature de l'information recherchée, explicitée dans les règles **E1, E2** apparaissant dans la partie déclaration de chacune de ces procédures (nous verrons plus loin la sémantique exacte de ces règles) :

- La première (VERBDAT1) devra sortir la liste des verbes qui n'ont pour sujet et objet que *jmd* (sujet humain) et *jmdm* (objet humain) ;
- La seconde (VERBDAT2) devra sortir la liste des verbes dont la forme syntaxique du sujet peut être un groupe verbal dépendant, introduit par *dass* (GVdass) ;
- La troisième (VERBDAT3) devra sortir la liste des verbes dont le sujet a pour trait sémantique + *humain* ± *individu* et l'objet + *humain* - *individu* + *groupe*.

Toutes les trois ont pour tâche :

- d'introduire l'information en mémoire centrale :
  - lecture au clavier du mot et écriture de celui-ci dans un emplacement mémoire réservé (mot) ;
  - transfert de ce mot dans un tableau en mémoire (tabint) de volume suffisamment grand pour pouvoir contenir cinq mots.
- de transférer le contenu de ce tableau (tabint) dans un fichier externe : table et ce jusqu'à ce que le mot soit vide. Ce fichier contiendra alors l'ensemble des mots lus au clavier :



On attribuera à chaque bloc de 5 mots un numéro d'enregistrement que l'on saura retrouver à l'aide d'une file spéciale L (sorte de *directory*) lors de la manipulation inverse qui consistera à lire l'information depuis l'extérieur.

- de rechercher l'information voulue et de la stocker dans une file prévue à cet effet (liste)
- et d'imprimer l'information souhaitée.

4. STRUCTURE ET SYNTAXE DU LANGAGE SYMBOLIQUE

Chaque programme se divise en deux parties : la partie *déclarations* où l'on définit la configuration des données et la partie *instructions* où l'on décrit l'algorithme. Il débute par le mot « Programme » et s'achève sur le mot « fin ». Le mot « début » marque la fin de la partie *déclarations* et le début de la partie *instructions*.

4.1. LES DECLARATIONS

On déclare dans l'ordre six sortes de types :

- **les paramètres « par » :**

Les identificateurs qui représentent les paramètres peuvent être en nombre quelconque et sont précédés du mot « par », ainsi que le montre l'exemple : par Q.

Nous entendons par « identificateur » un terme qui se compose de lettres et de chiffres, mais qui doit obligatoirement commencer par une lettre. Cette définition peut être formulée de la façon suivante dans la notation de Backus (« ::= » signifiant « le terme précédant ce symbole est défini par ce qui suit ») :

<identificateur> ::= <lettre>
<lettre> ::= A
<lettre> ::= B
...
<lettre> ::= Z

<identificateur> ::= <identificateur > <lettre>
<identificateur> ::= <identificateur > <chiffre>
<chiffre> ::= 0
<chiffre> ::= 1
...
<chiffre> ::= 9

Exemple : A est un identificateur
A9 est un identificateur
A9B est un identificateur...

Les listes d'identificateurs se composent de suites d'identificateurs séparés par des virgules et ponctuées par un point-virgule. Les identificateurs doivent être tous différents les uns des autres.

- **les index « index » :**

Les index sont des variables simples. Elles sont précédées par le mot « index », ainsi que le montre l'exemple : index J.

- **les booléens « booléen » :**

Les booléens sont aussi des variables simples, mais ce sont des variables auxquelles on attribue une autre signification. Une variable de type « booléen » est une variable logique indiquant une alternative : « vrai » ou « faux ». Par convention, on assignera à l'entier arithmétique **1** la valeur « vrai » et à l'entier arithmétique **0** la valeur « faux ». Les variables de ce type sont précédées du mot « booléen », ainsi que le montre l'exemple : booléen Z.

- **les files « file » :**

Les files peuvent être assimilées à des variables indicées, des sortes de tableaux d'un certain volume. Chaque variable de type « file » est donc suivie de l'indication, entre parenthèses, de son volume. La liste des files est précédée par le mot « file », par exemple : file mot (1000).

- **les files externes « filext » :**

Les files externes sont des files auxquelles on attribue une autre signification : elles simulent en mémoire des supports externes, d'où la dénomination « filext ». Les remarques faites à propos des files sont valables ici aussi. La liste des files externes est précédée du mot « filext », par exemple : filext table (200000).

- **les expressions « expr » :**

Le type « expression » est tout à fait différent des types que nous venons de voir. En effet, une expression n'est pas considérée dans sa totalité comme une variable de type « expr ». Une expression, dans le sens du Système de Post, est une règle qui contient des caractères imposés et des variables appelées « Symboles coïncidants » (Sc) susceptibles de recevoir une certaine information par application de cette règle sur un mot quelconque (application par « matching »).

Exemple : Soit le mot M = aaa**bbcabc**ccaa**b** et la règle E = x1**bbc**x2**bc**x3**ab**. Par application de la règle E sur M par *matching*, noté ($\leftarrow^{m}$), nous obtenons :

$$M \overset{m}{\leftarrow} E : \quad x1 = aaa \quad x2 = a \quad x3 = cca$$

Comme dans le Système de Post, nos expressions sont constituées de caractères ou groupes de caractères imposés et de symboles coïncidants, ainsi que le montre la règle E1 du programme VERBDAT1 :

E1 = # 0 (1) * **jmd,** 2 * **jmdm,** 3 # 4

Caractères imposés : 6 (5 + le vide)

#	(	) * jmd,	* jmdm,	#	
1	2	3	4	5	6

Symboles coïncidants : 5

0 1 2 3 4

Le signe « $ » est un méta-symbole : il n'appartient pas à l'alphabet du mot et ne peut donc pas être confondu avec celui-ci. Ce sont les constituants de ces règles (les caractères imposés et les symboles coïncidants) qui prennent le statut de variable de valeur connue (pour les caractères imposés) ou inconnue au stade de la déclaration (pour les symboles coïncidants). La liste des expressions est précédée du mot « expr », par exemple : expr E1.

4.2. LES INSTRUCTIONS

La partie *instructions* s'ouvre sur le mot « début » et s'achève sur le mot « fin » qui ponctue également la procédure. Il existe neuf types d'instructions :

- **les affectations simples ou arithmétiques :**

  - **affectation simple :**

    Exemple : Q := K

    Elle comprend une variable (paramètre, index, booléen) ponctuée par « := » et suivie d'une expression. Cette expression peut être une variable (paramètre / index), un entier (plus petit que la plus grande valeur entière représentable sur l'ordinateur utilisé – noté K) si le premier opérande est un paramètre ou un index, un entier (0 ou 1) si le premier opérande est un booléen.

  - **affectation arithmétique :**

    Exemple : T := N – 1
    T := T * 5
    T := T + R

    Pour ce type d'affectation, l'expression est un couple dont chacun des éléments peut être pris dans l'ensemble des variables déclarées (paramètres / index)* et des entiers, et relié par un des quatre opérateurs arithmétiques bornés par K (addition, soustraction, multiplication, division) : « $+^K$, $-^K$, $*^K$, $\div^K$ ». Ce qui signifie que ces opérations n'ont un sens qu'à la condition que la représentation du résultat n'exige pas un entier plus grand que K, valeur fixée à l'avance définitivement. Les différentes combinaisons de cette expression peuvent être :

variable	variable
variable	entier
entier	variable
entier	entier

* Remarque : Nous excluons ici les booléens car le calcul les concernant est de toute autre nature. Il est cependant tout à fait licite d'envisager de simuler un calcul logique à l'aide de l'un des quatre opérateurs arithmétiques « +, -, *, ÷ », l'opérateur « * » simulant le ET logique.

Exemple : Soit les deux booléens Z et Z1

Z	Z1	ET logique	Valeur	*	valeur
0	0	0	faux	0	faux
0	1	0	faux	0	faux
1	0	0	faux	0	faux
1	1	1	vrai	1	vrai

Les autres opérateurs sont à exclure. Le seul opérateur arithmétique qui aurait pu éventuellement servir à simuler le OU logique est l'opérateur « + » :

Z	Z1	OU logique	Valeur	+	valeur
0	0	0	faux	0	faux
0	1	1	vrai	1	vrai
1	0	1	vrai	1	vrai
1	1	1	vrai	2	?

L'entier 2 n'étant pas autorisé, la valeur reste indéterminée.

- **la mise à vide (initialisation) d'une file ou d'une file externe :**

Exemple : tabint := vide
table := vide

Cette instruction comprend une variable de type « file » ou « filext » ponctuée par « := » et suivie du mot clé « vide ».

- **les affectations de files simples ou de type « concaténation » :**

  - **affectation simple :**

Elle se compose d'une variable de type « file » ponctuée par « := » et suivie d'une expression qui peut être soit une variable de type « file » ou éventuellement un symbole coïncidant, soit une constante. Nous appelons *constante* un caractère ou une chaîne de caractères mis entre guillemets droits simples / apostrophes ('...') :

Exemple : tabint := '#'

  - **affectation de type « concaténation » :**

Pour ce type d'affectation de file, l'expression est un couple dont chacun des éléments peut être pris dans l'ensemble des files déclarées et des constantes, et relié par

l'opérateur de concaténation « union » noté « U ». Les différentes combinaisons de cette expression peuvent être :

file	file
file	constante
constante	file
constante	constante

file	Sc	éventuellement
Sc	file	
Sc	Sc	
Sc	constante	
constante	Sc	

Exemple : tabint := tabint U mot
tabint := tabint U '#'

- **les instructions d'échange :**

  - **output :**

Exemple : table (L(K)) := tabint

Cette instruction se compose d'une variable de type « filext » suivie d'une expression entre parenthèses ponctuée par « := », le tout suivi d'une variable de type « file ». Cette expression est elle-même constituée d'une variable de type « file » suivie entre parenthèses d'un index.

  - **input :**

Exemple : input tabint, table (L(T))

Cette instruction débute par le mot clé « input » suivi d'une variable de type « file » ponctuée par une virgule, le tout suivi d'une expression. Cette expression se compose d'une variable de type « filext » suivie entre parenthèses d'une variable de type « file », elle-même suivie entre parenthèses d'un index.

  - **lire :**

Exemple : lire Q
lire mot [.]

Dans ce cas, le mot clé « lire » est suivi d'une expression qui peut avoir les deux formes suivantes : une variable (paramètre / index / booléen) ; une file, suivie entre crochets d'un marqueur de fin de mot n'appartenant pas au mot lui-même, le point « . ».

  - **print :**

Exemple : print Q
print 5

Le mot clé « print » est suivi d'une expression qui peut être une variable (paramètre / index / booléen), une file ou un symbole coïncidant.

- **les aiguillages simples ou booléens :**

  - **aiguillages simples :**

Exemple :
```
Aig F = 'VERBDAT1' : e1, 'VERBDAT2' : e2, 'VERBDAT3' : e3 ;
...
sortie F
noeud F
```

Cette instruction débute par le mot clé « Aig » suivi d'une variable, d'un opérateur relationnel et d'une expression assez complexe qui varie selon le type de la variable choisie :

a. Si la variable est une file : l'opérateur relationnel de cet aiguillage ne pourra être que « = » ou « ≠ » et l'expression sera une constante, une file (ou un Sc) ou le mot clé « vide » suivis d'une étiquette (voir le paragraphe consacré à l'étiquetage) et éventuellement de plusieurs autres expressions. Les expressions sont séparées entre elles par des virgules.

b. Si la variable est un paramètre ou un index* : tous les opérateurs relationnels seront admis (=, ≠, >, ≥, <, ≤) et l'expression pourra être un paramètre / un index ou un entier suivis également de plusieurs autres expressions. Les expressions sont aussi séparées entre elles par des virgules.

* Remarque : Du fait de la définition du type « booléen », la relation d'ordre sur les valeurs « faux » et « vrai » est telle que « faux » < « vrai ». Il est donc possible d'exprimer les opérations logiques en utilisant les six opérateurs de relation. Les remarques précédentes sont valables ici aussi, à la condition d'avoir pour expression un booléen, ou l'entier 0 ou 1.

L'aiguillage simple est lié aux deux instructions suivantes :

a. l'instruction « sortie » qui débute par ce mot suivi du nom de la variable ;
b. l'instruction « nœud » qui débute également par ce mot suivi du nom de la variable.

Pour cet aiguillage simple, les différentes combinaisons de l'expression peuvent être :

file	file		
file	constante		
file	Sc	→	éventuellement
file	vide		
variable	variable		
variable	entier		
Sc	Sc	}	éventuellement
Sc	vide	}	
Sc	constante	}	

  - **aiguillages booléens :**

Exemple : Aig Z : suite ;
...
nœud Z

Cette instruction débute par le mot clé « Aig » suivi d'un booléen et d'une étiquette. Ici, une seule étiquette est autorisée. Cet aiguillage est lié à l'instruction « nœud » définie plus haut (dans ce cas, l'instruction « nœud » comporte le nom du booléen).

- **les boucles simples ou conditionnelles :**

  - **boucles simples :**

Exemple : Boucle Syst
...
rep Syst

Cette instruction est composée d'une instruction ou d'un groupe d'instructions délimité par les mots clés « Boucle » et « rep » suivis du nom de la boucle. Dans ce cas, le groupe d'instructions est exécuté indéfiniment puisque l'on n'impose aucune condition d'arrêt de boucle.

  - **boucles conditionnelles :**

Exemple : Boucle I (J)
...
rep I (J = 5 ou mot = vide)

Boucle F (K := Q)
...
rep F (mot = vide)

Les boucles conditionnelles sont composées d'une instruction ou d'un groupe d'instructions délimité par les deux mots clés suivants :

1. « Boucle » suivi du nom de la boucle et éventuellement une expression entre parenthèses pouvant contenir un index ou une instruction d'affectation de type simple ayant pour opérandes deux variables ;
2. « rep » suivi du nom de la boucle et d'une expression entre parenthèses indiquant la ou les conditions d'arrêt. Cette condition se compose d'un couple d'opérandes dont le premier détermine la nature de l'opérateur relationnel ainsi que le type du deuxième opérande :

   a. Si le premier opérande est une file : seuls les opérateurs relationnels « = » et « ≠ » seront autorisés. Le deuxième opérande pourra être une file ou un Sc ; une constante ou le mot clé vide.
   b. Si le premier opérande est une variable (paramètre ou index)* : tous les opérateurs relationnels seront autorisés. Le deuxième opérande pourra être une variable (paramètre / index) ou un entier.

* N.B. : Les remarques faites sur les booléens dans le paragraphe consacré aux aiguillages simples sont valables ici aussi.

L'expression qui suit le mot clé « rep » pourra être simple, c'est-à-dire ne comporter qu'une condition, ou complexe. Elle comportera dans ce dernier cas une série de conditions séparées par des ET ou des OU logiques, ou à la fois des ET et des OU logiques (notés et, ou). Les différentes combinaisons de chacune des conditions peuvent donc être :

file	file
file	constante
file	Sc
file	vide
variable	variable
variable	entier
Sc	Sc
Sc	vide
Sc	constante

Le groupe d'instructions se trouvant à l'intérieur de ce type de boucle sera exécuté jusqu'à ce que l'expression booléenne (résultat du calcul logique sur la ou les conditions d'arrêt) devienne vraie.

- **l'insertion :**

Exemple : insérer VERBDAT1 (Q)

L'instruction d'insertion commence par le mot « insérer » suivi du nom de la procédure à insérer, le tout suivi entre parenthèses par la liste des paramètres effectifs. Le nom de la procédure est l'identificateur tel qu'il a été déclaré dans le texte de la procédure que l'on veut insérer (c'est l'identificateur qui suit le mot « Programme »). La liste des paramètres effectifs représente, quant à elle, la liste des variables (index) déclarées dans la procédure qui contient l'instruction d'insertion, séparées par des virgules. Un paramètre effectif peut également être un entier.

La liste des paramètres effectifs doit correspondre à la liste des paramètres de la procédure insérée. A chaque paramètre doit correspondre un et un seul paramètre effectif, et la correspondance s'effectue dans l'ordre de présentation de chacune des deux listes.

- **le matching :**

Exemple :
$$\text{tabint, } Z \overset{m}{\leftarrow} E1$$

$$\$4\$, Z1 \overset{m}{\leftarrow} E1$$

Cette instruction se compose du signe « $\overset{m}{\leftarrow}$ » (signifiant « application par *matching* ») précédé d'une expression à deux composants et suivi du nom de la règle qui intervient dans le matching. Les deux composants de cette expression sont une variable de type file ou un symbole coïncidant et un booléen, tous deux séparés par une virgule.

- **l'étiquetage :**

Exemple : Aig Z : suite ;
suite : Boucle C
...
rep C (4 = vide)

L'étiquette est un identificateur séparé de l'instruction qu'il précède par le signe « : ». Un identificateur d'étiquette doit être unique dans un même corps de procédure. A toute instruction est susceptible d'être collée une étiquette. En revanche, plusieurs instructions (ici conditionnelles) peuvent être attachées à une même étiquette.

L'étiquette désigne l'instruction qui doit être déroulée juste après l'instruction en train de se dérouler et qui se réfère à un identificateur d'étiquette identique (à la condition, bien sûr, que la condition soit vraie).

5. ECHANTILLON DU FICHIER

sich anschließen (se joindre / se rallier à, se ranger à) * jmd, GNn, + humain ± individu, + humain ± individu * jmdm / einer Sache, GNd, + humain – individu + groupe (sich einer Partei / den Demonstranten anschließen) + activité / action (sich einem Streit / einer Besichtigung anschließen), - humain + abstrait + idée / pensée / avis (sich einer Ansicht / einem Vorschlag anschließen)

absagen (se décommander auprès de, renoncer à / abjurer / renier) * jmd, GNn, + humain ± individu, + humain ± individu, + humain ± individu * jmdm / einer Sache, GNd, + humain ± individu, - humain – abstrait (dem Alkohol / einem Laster absagen), - humain + abstrait + pensée / parole / croyance (seinem Glauben / einer Lehre absagen)

ähneln (ressembler à) * jmd / etwas, GNn, non restreint * jmdm / einer Sache, GNd, non restreint

angehören (appartenir à / être membre de, être très attaché à, faire partie de) * jmd / etwas, GNn, + humain ± individu, + humain ± individu, - humain + abstrait + fait / événement * jmdm / einer Sache, GNd, + humain – individu + groupe (einer Partei / einem Verein angehören), + humain ± individu (einem Mann angehören), - humain + abstrait + époque (das gehört dem Mittelalter an)

ausweichen (faire place à / éviter, esquiver) * jmd, GNn, + humain ± individu, + humain ± individu * jmdm / einer Sache, GNd, + humain ± individu, - humain ± abstrait (einem Motorrad / einem Schlag / einer Frage / einer Schwierigkeit ausweichen)

begegnen (rencontrer, empêcher / prévenir, arriver (emploi impersonnel)) * jmd / etwas, GNn, + humain ± individu, + humain ± individu, - humain + abstrait + événement + mal (etwas Schlimmes / ein Unheil) * jmdm / einer Sache, GNd, + humain ± individu, - humain + abstrait + attitude / pensée (kühler Zurückhaltung / einer Meinung begegen) + événement + mal (einer Gefahr / einem Unheil / einem Fehler begegnen), + humain ± individu

beipflichten (approuver, adhérer à, se ranger à l'avis de) * jmd, GNn, + humain ± individu, + humain ± individu * jmdm / einer Sache, GNd, + humain ± individu, - humain + abstrait + pensée / opinion, avis (einem Vorschlag, einer Ansicht beipflichten)

beispringen (aider / venir à la rescousse de) * jmd, GNn, + humain ± individu * jmdm, GNd, + humain ± individu

beistehen (secourir, aider, assister) * jmd, GNn, + humain ± individu * jmdm, GNd, + humain ± individu

entgehen (échapper à, manquer) * jmd / etwas, GNn / GVdass, + humain ± individu (der Flüchtling / der Dieb), + humain ± individu, - humain + abstrait + fait (mir ist nicht entgangen, dass...) + avantage (ein Vorteil / ein Gewinn / eine gute Gelegenheit) + énoncé / signe (ein Wort / ein Fehler / eine Bemerkung) * jmdm / einer Sache, GNd, + humain ± individu (seinem Verfolger entgehen) – humain + abstrait + mal + oppression physique / morale (einer Gefahr / einer Strafe / dem Tode / einem Tadel / einem Schicksal entgehen), + humain ± individu

6. CONCLUSION

Je dirais en conclusion que ce travail qui illustre une application de l'informatique fondamentale au traitement automatique du langage naturel a marqué une étape dans l'évolution de ma formation en linguistique allemande et informatique. Les travaux que j'ai menés quelques années plus tard en tant que chercheur en linguistique informatique à l'IAI de Sarrebruck (*(Institut der Gesellschaft zur Förderung der Angewandten Informationsforschung e.V. an der Universität des Saarlandes*) dans le cadre de la traduction automatique reflètent de façon plus poussée l'interaction entre linguistique et informatique.

Je présentais dans mon préambule le sujet de ma thèse qui portait sur les locutions verbales dans les écrits de Martin Luther lors de ma rencontre avec le fondateur du *Bulletin d'Informatique Approfondie & Applications*. Je terminerai mon hommage à Edmond Bianco avec ma première contribution au BIAA en citant mes travaux sur l'analyse et l'implémentation des constructions à verbe support dans le formalisme CAT2 (Mesli 1991) que j'ai eu l'honneur de présenter à la Première Conférence Internationale de la Computation des Systèmes Informatiques du LITAM à Marseille (Mesli 1992).

7. REPERES BIBLIOGRAPHIQUES

BIANCO Edmond, 1979 : *Informatique Fondamentale : De la Machine de Turing aux ordinateurs modernes*. Birkhäuser Verlag, Basel, Boston, Stuttgart.

BIANCO Edmond, 1997 : « Chapitre 2 : Une Machine Elémentaire, Chapitre 3 : Compléments à la Machine de Nolin, Chapitre 4 : La Machine Formelle ». *Bulletin d'Informatique Approfondie & Applications* N° 47, 3-26, Marseille.

BIANCO Edmond, 1998 : « Chapitre 5 : La Machine Universelle, Chapitre 6 : Notion de Système ». *Bulletin d'Informatique Approfondie & Applications* N° 51, 3-34, Marseille.

BISIERE Christophe, IWANESKO Denis, KNIPPEL Jean-Michel et MASSAT Jean-Luc, 1989 : « Compilateur de machine universelle : MU ». *Bulletin d'Informatique Approfondie & Applications* N° 23, 26-45, Marseille.

BRESSON Daniel, 2001 : *Grammaire d'usage de l'allemand contemporain*, Hachette, Paris.

MESLI Nadia, 1987 : *Programmation en Procédure Formelle d'un compilateur pour un langage expérimental de recherche et de calcul.* Projet de recherche du Certificat d'Etudes Supérieures d'Université (C.E.S.U.) en Informatique Fondamentale, Département de Mathématiques de Luminy, Université d'Aix-Marseille II, Marseille.

MESLI Nadia, 1989 : *Les locutions verbales dans les écrits de M. Luther.* Thèse de Doctorat, 2454 p. (531 p. et 4 annexes), Université d'Aix-Marseille I, Aix-en-Provence. Microfiches Lille-Thèses, ISSN : 0294-1767, N° d'identification : 0342.08880/90.

MESLI Nadia, 1991 : *Analyse et traduction automatique de constructions à verbe support dans le formalisme CAT2.* EUROTRA-D Working Papers 19b, IAI, Saarbrücken.

MESLI Nadia, 1992 : « Le formalisme CAT2. Un exemple d'implémentation : les constructions à verbe support ». *Bulletin d'Informatique Approfondie & Applications* N° spécial, *Première Conférence Internationale de la Computation des Systèmes Informatiques*, LITAM, Marseille.

VOUZZAVEDIBISAR

Affiche :
Les 30 ans du B.I.A.A. Hommage à Edmond Bianco

Eric Olivier