# LA DECOMPOSITION AUTOMATIQUE DES COMPOSES NEERLANDAIS (*)

Maurice VAN OVERBEKE, Chargé de cours associé, U.C.L.

*Introduction.*

A une époque où un nombre toujours croissant de linguistes semblent gagnés à l'idée que l'ordinateur peut rendre des services appréciables en tant que classificateur infatigable ou comme calculateur expéditif (entre autres pour la confection d'index, de concordances, de vocabulaires inverses, de listes de fréquences etc.), il peut être utile de réfléchir à la question de savoir si ce recours généralisé aux facultés les plus banales de la machine ne risque pas d'atrophier celle-ci, et par là-même le linguiste, dans leurs capacités les plus importantes.

Dans un petit pays comme le nôtre, qui grâce à une situation économique favorable, peut se permettre l'emploi presque illimité de l'équipement électronique le plus à jour, alors que rien dans sa tradition ne le préparait à un développement parallèle de la *soft-ware*, ce malaise n'est somme toute qu'une conséquence logique. Encore faudrait-il qu'il revête pour nous l'importance d'un signal d'alarme, auquel il convient de réagir sans tarder par des moyens appropriés. Si notre diagnostic du malaise est exact, le remède devra être tel que nos efforts aillent principalement dans le sens d'une *programmation linguistique* plus approfondie. Entendons par là : toute description explicite d'un nombre fini d'instructions (algorithmes), permettant à la machine d'analyser ou de synthétiser une partie d'une grammaire naturelle. Cela revient à dire que les linguistes ont intérêt à s'engager toujours davantage sur la voie de ce qu'il est convenu d'appeler désormais : la simulation automatique. Certes, voilà un mot voué à toutes les vicissitudes de la terminologie à la mode mais qui, lorsqu'il est employé pour désigner la fonction d'un modèle physique tel que l'ordinateur, constitue la preuve la plus adéquate de l'hypothèse scientifique. Rien de plus convaincant pour un biologiste à la recherche de la structure de l'A.D.N. (1) que la simulation de cette structure en un modèle physique : la célèbre double hélice. Produire la réplique exacte d'un phénomène à base de composantes minima hypothétiques et suivant des règles de reconstruction explicites, voilà une forme nouvelle de *connaissance*, à laquelle nous obligera toujours davantage l'emploi des machines mises à notre disposition.

On sait que, dans le domaine de la linguistique, le principe de la simulation est à la base-même de toute la théorie générativiste et transformationaliste. Pourtant, lorsqu'on constate aujourd'hui les flagrants désaccords à l'intérieur de ce courant rénovateur (désaccords que, de manière très cartésienne les pionniers pensaient avoir tués dans l'oeuf en vertu du principe-même de la simulation explicite) on est en droit de se demander si là encore, l'arbitraire ne commence pas à s'instaurer parce que, à quelques exceptions près (2), la référence à un modèle physique, à une machine qu'on ne peut soupçonner d'aucune



Extrait de la Revue (R.E.L.O.)
VII, 1 à 4, 1971. C.I.P.L. - Université de Liège - Tous droits réservés.

complicité, d'aucune bienveillance envers l'hypothèse émise, fait défaut. On a dit et répété que la description grammaticale ne devait supposer aucune connaissance préalable de la part du lecteur (3). Or, comme l'ont bien vu A. Kraak et W. Klooster, cette condition n'est réellement remplie que dans le cas de l'ordinateur qui assimile dans l'indifférence le programme qu'on lui propose et qui n'en tirera rien de plus que ce que le linguiste a bien voulu y mettre (4). L'important est donc pour nous de mettre a profit la présence de ce juge impartial à la logique imperturbable, chaque fois que nous tentons de simuler un processus linguistique pour mieux le connaître, et surtout, pour en acquérir une connaissance non-contradictoire. C'est dans ce contexte que nous nous sommes proposés d'élaborer un programme de simulation automatique portant sur les composés néerlandais, programme qui, dans l'éventail proposé par W. Martin (5), peut se classer sous la dénomination 3.1 : *automation lexicale.*

1. *Utilité du programme.*

Les linguistes qui ont essayé d'établir des listes de fréquences pour des fins didactiques ou pour la solution de problèmes de lexicométrie, n'ont pu rester indifférents au fait que dans un grand nombre de cas et de langues, ces listes étaient truffées de vocables composés dont les composantes apparaissaient sous une forme lemmatisée à un autre endroit de la liste. Ainsi, pour ne donner qu'un exemple français, une liste pouvait facilement contenir d'une part les vocables comme *casser* et *croûte*, et d'autre part le vocable composé *casse-croûte*, sans qu'il soit fait état de leurs rapports visiblement privilégiés. Cela pouvait donner lieu à deux conséquences fâcheuses. D'un point de vue didactique d'abord, on pouvait se demander dans quelle mesure il ne fallait pas favoriser la composition lexicale chaque fois qu'elle pouvait éviter l'encombrement de la mémoire verbale par un vocable nouveau. "Si l'on possède le mot *garde-fou*, il n'est pas nécessaire de connaître le mot *parapet*. Pour celui qui connaît *garde* et *fou*, l'apprentissage de *garde-fou* est peu coûteux" (6). Ensuite, d'un point de vue lexicométrique, il était utile de savoir dans quelle mesure les différentes approximations statistiques n'étaient pas d'emblée faussées par la définition opérationnelle du mot comme : "toute suite de graphèmes entre deux blancs". C'est ainsi qu'en cherchant l'approximation la plus adéquate de l'accroissement du vocabulaire d'un texte, on pouvait se demander si on n'était pas dupe du fait qu'un nombre important de composés tels que *pomme de terre* avaient été enregistrés mécaniquement comme trois vocables différents, alors qu'une expression comme *c'est-à-dire,* grâce à ses traits d'union, prenait la valeur d'un seul nouveau vocable. En outre, le problème des limites du vocabulaire dans les différentes langues n'avait-il pas été mal posé dès lors que dans une langue (comme le français) les néologismes sont le plus souvent formés à partir de racines non-motivées empruntées aux langues classiques, alors que dans telle autre langue (comme le néerlandais ou l'allemand) les mots les plus recherchés, et donc les moins fréquents, sont en grande partie des composés (7) ? Finalement, comment comparer automatiquement deux vocabulaires (par exemple français et néerlandais), eussent-ils le même nombre de vocables, ou comment tirer des résultats valables d'une comparaison automatique d'un texte néerlandais et de sa traduction française (8), quand on sait que, selon la remarque pertinente de A. Martinet : "Un lexique s'étend soit en tirant de nouveaux termes de son propre fonds, soit en cherchant

ailleurs et en intégrant ceux dont il a besoin. Les langues diffèrent beaucoup en ces matières : les unes se créent facilement des ressources par la composition ou la dérivation; d'autres ont plutôt recours à l'emprunt/.../ Le français est très réceptif aux éléments empruntés aux langues classiques/.../ Cette réceptivité particulière, accompagnée d'une tendance croissante à se passer des ressources de la composition et de la dérivation, a abouti à donner au lexique français un caractère fort peu motivé : il est impossible de reconnaître *aveugle* dans *cécité* et difficile de retrouver *oeil* dans *oculaire*" (9). On sait que les représentants anglo-saxons de la statistique linguistique ne sont guère préoccupés par cette difficulté (10). Cela se comprend quand on pense aux latitudes que prévoit l'orthographe anglaise. Lorsqu'en 1935, G.E. Vander Beke publia son vocabulaire de base du français, son ouvrage avait pour titre : French Word Book. Ce titre comporte trois vocables nettement séparés par des blancs, de sorte que pour un lecteur non averti, il n'est pas aisé de savoir si le mot *French* détermine *Word* ou *Book* ou encore l'amalgame de ces deux mots. Pour le néerlandais comme pour l'allemand, les règles d'orthographe sont plus strictes. On devine donc l'intérêt de pouvoir décomposer automatiquement les unités lexicales se présentant sous une forme unique, alors qu'elles contiennent des composantes pouvant mener par ailleurs une existence lexicale autonome.

Le programme présenté ici pourra être utilisé à des fins diverses. Il permettra d'abord d'exécuter des calculs statistiques automatiques susceptibles de nous éclairer sur les questions suivantes :

a) Quel est le rapport *types/tokens* à l'intérieur-même du vocabulaire composé (d'un texte mais également d'un dictionnaire ou d'une liste de fréquence) ? C'est ainsi que dans la série de composés bicéphales *WERKMAN, HANDWERK, VLIEGTUIG, HANDBOEK* et *BOEKWERK,* nous pourrons dénombrer 5 X 2 soit 10 tokens, contre 6 types puisque la composante *WERK* est répétée 3 fois, *HAND* et *BOEK* 2 fois etc.

b) Quel est le rapport numérique entre les fréquences des monèmes dans leur contexte composé et leur fréquence en tant que monème isolé. Ainsi, existe-t-il une corrélation significative entre le vocable HAND pris isolément et le même monème incorporé dans un composé (par ex. HANDWERK) ?

c) Y a-t-il une corrélation négative significative entre la longueur des monèmes et leur répétition dans les composés; en d'autres mots : les monemes courts sont-ils mieux représentés dans les composés et y ont-ils un degré de répétition supérieur ?

d) Y a-t-il proportion inverse entre le nombre d'emprunts et le nombre de composés dans un texte ? S'il est vrai que le choix d'emprunts représente une économie syntaxique (estimée en longueur de mots) et le choix d'un composé une économie lexicale (on évite l'adoption d'un nouveau terme), cette double économie se présente-t-elle dans la production linguistique comme une alternative ?

En second lieu, notre programme vise à dégager des renseignements concernant la charpente syntaxique du composé. Si ce dernier peut être considéré comme un syntagme ordinaire (11), dont les différents

membres entretiennent des rapports de déterminants à déterminés, il s'agit de savoir tout d'abord quel membre est déterminé par quel autre. Cela ne pose aucun problème pour les composés bicéphales. A quelques rares exceptions près, on peut dire qu'en néerlandais ils contiennent un noyau à place fixe, précédé d'un élément déterminatif. Mais dès que le nombre des composantes dépasse ce niveau minimal, l'automation devient plus complexe. Ainsi pour les composés tricéphales, il y aura lieu d'opter pour l'une des deux solutions suivantes : (A(BC)) ou ((AB)C). Or, dans beaucoup de cas aucun indice formel ne fournit suffisamment de renseignements pour qu'on soit en mesure de formuler l'algorithme du choix. A cet égard, comparons les composés ZAKWOORDENBOEK (dictionnaire de poche) et SPREEKWOORDENBOEK (dictionnaire de proverbes).

C'est pourquoi un troisième volet devra nécessairement venir compléter ce programme. Pour être en mesure d'énoncer explicitement les rapports déterminatifs entre les membres des composés par un *bracketing* automatique, il faudra spécifier exactement de quelle nature sont les relations sémantiques et syntaxiques entre les composantes. Aussi faudra-t-il prévoir dans la programmation un dispositif permettant de faire la part des relations possibles et de celles qu'il convient d'écarter, compte tenu de la subcatégorisation des "entrées lexicales" en présence. Pour ce faire, nous nous trouverons dans l'obligation de préciser (de façon provisoire et fort incomplète sans doute) les marqueurs sémantiques, les marqueurs syntaxiques et les restrictions sélectives de chaque composante virtuelle. On comprend aisément que cette troisième partie du programme, qui s'annonce à n'en pas douter comme la plus ardue, devrait nous permettre à la fois de solutionner les problèmes du *bracketing* et de procéder à la simulation automatique d'un nombre limité de composés néerlandais.

## 2 *Le programme : "AUTOMATIC COMPOUND WORD SPLIT".*

Le programme *AUTOMATIC COMPOUND WORD SPLIT* comporte deux parties dont la distinction est plus théorique que pratique : 1. la séparation et 2. la lemmatisation. Ces deux parties forment une unité opérationnelle pour la bonne raison que bien souvent la séparation automatique, qui présuppose la reconnaissance des limites des vocables, passe par l'application des règles de lemmatisation (règles de délétion et de réécriture des phonèmes de liaison propres au néerlandais). Il va de soi que pour la reconnaissance de composantes dans les mots composés, la machine devra disposer de tous les points de comparaisons nécessaires à l'identification automatique. On peut, pour y arriver, procéder de deux façons. La première consiste à enregistrer dans la mémoire (disk pack) un nombre limité de lexemes simples (non composés) que la machine devra retrouver dans une seconde liste d'entrées lexicales ou figurent les composés. La seconde consiste à n'introduire qu'une seule liste comprenant indifféremment des lexèmes simples et composés en orientant le programme dans le sens de l'auto-instruction. Pour y parvenir, il faudra d'abord exécuter le programme de la classification alphabétique inverse. Il va sans dire que cette classification, comme d'ailleurs toute classification alphabétique normale, comprend déjà la classification secondaire par longueur croissante. C'est ainsi que le programme en question appliqué aux lemmes (de pure inven-

tion) : BA, DCBA, DCA, DA, CDA, CA nous donnera automatiquement l'ordonnance suivante :

BA
DCBA
CA
DCA
DA
CDA

L'on constate que là où les lemmes plus longs précèdent des lemmes plus courts, les raisons prioritaires de l'ordre alphabétique ont joué. Ainsi la séquence DCBA, tout en étant plus longue que CA, précède pourtant celle-ci parce que la logique du programme veut qu'un lemme se terminant par BA soit cité avant celui qui se termine par CA, quelle que soit par ailleurs leur longueur respective. Ce fait apparemment anodin a deux consequences importantes pour ce que nous avons appelé : l'auto-instruction de la machine.

a) Pour des raisons d'économie d'abord. En effet, les instructions permettent à la machine de lire les lemmes dans l'ordre de ce classement préalable. Chaque fois qu'elle rencontre un lemme inconnu (par ex. : le premier BA), elle a pour mission de l'enregistrer dans sa mémoire. A chaque nouvelle lecture la question est posée de savoir si des parties du nouveau lemme (composé) ne sont pas identiques à un des lemmes déjà connus, et cela en lisant de droite à gauche. Ainsi dans DCBA, le lemme BA est automatiquement retrouvé et séparé du reste, ce qui donne à la sortie : DC/BA. Si le lemme DCBA devait être lu avant BA, les deux lemmes ne pourraient être mis en regard que par lecture récurrente, ce qui signifierait une perte de temps.

b) Pour le bracketing automatique des lemmes comportant plus de deux composantes. Dans beaucoup de cas (mais pas dans tous) on peut répondre à la question, où passe la frontière entre le noyau et le déterminant, en se référant à l'existence de tel composé secondaire et à la non-existence de tel autre. Ainsi, un lecteur n'ayant aucune connaissance du néerlandais (comme c'est le cas de l'ordinateur) pourrait trouver les rapports syntaxiques du composé ACHTERKLEINKIND, c'est-à-dire (A(BC)) en constatant que le composé secondaire KLEINKIND figure dans la liste, alors que celui de ACHTERKLEIN ne s'y trouve guère. Or, la classification préalable dont nous venons de parler, permet précisément à la machine de lire le lemme KIND avant celui de KLEINKIND et de lire KLEINKIND avant ACHTERKLEINKIND. Grâce à ce classement, la machine sera en mesure de représenter la charpente syntaxique de VOLBLOEDPAARD et de DWERGNIJLPAARD par des schémas différents. Ne connaissant pas BLOEDPAARD dans le premier cas, elle va vérifier si VOLBLOED existe et optera pour la solution ((AB)C). Dans l'autre cas, elle vient d'enregistrer NIJLPAARD et optera provisoirement pour la solution : (A(BC)) (12).

Les deux procédures ont leur pour et leur contre. Pour la première, il est évident qu'il s'agit d'une programmation nettement "ad hoc". C'est ainsi que pour la séparation de HANDSTAND, tout dépendra de

la question, si vous avez adopté dans le lexique des lexèmes simples TAND et/ou STAND. Si vous avez donné TAND, la machine peut se contenter de la ressemblance la plus courte et considérer le *s* comme un graphème de liaison, ce qui aboutirait à une séparation fausse. Par contre, la procédure permet au linguiste de concentrer toute son attention autour des problèmes de lemmatisation, quitte à mettre provisoirement entre parenthèses les questions de séparation. C'est la solution que nous avons adoptée ici.

3. *La lemmatisation.*

La lemmatisation comporte des règles de délétion de graphèmes et des règles de réécriture. La délétion porte sur l'ensemble des morphèmes flexionnels entre les composantes. La réécriture permettra de reconstituer la composante conformément à l'aspect qu'il présente en tant que lexème isolé. Ainsi, dans le composé SLAVENHANDEL, le morphème flexionnel EN sera supprimé et, moyennant des instructions appropriées mentionnées plus loin, la séquence AV sera réécrite comme AAF. C'est seulement ensuite que la composante SLAAF sera comparée au lexique de la mémoire.

Une autre question qu'il nous faut aborder ici, est celle du sens ou de la direction de la lecture. En effet pour être en mesure d'identifier une séquence quelconque du mot composé, la machine doit avoir une raison suffisante de s'arrêter avant la fin du mot. Or, il y a parmi les composantes des lemmes d'entrée (tout au moins en néerlandais) une séquence à laquelle la lemmatisation ne s'applique plus, c'est la dernière, celle qui se trouve à l'extrême droite du composé. On part donc du principe que le mot composé, tel qu'il figure dans le dictionnaire, ne se termine jamais par un morphème flexionnel ou par une désinence. Dès lors, il est tout à fait indiqué de commencer l'analyse par une lecture de droite à gauche, de s'arrêter dès qu'une identite entre la séquence lue et un des termes du lexique est établie, et de considérer ce qui reste du composé comme une structure complexe contenant éventuellement des graphèmes de liaison. Or, dans une première version du programme, nous avions prévu la lecture de gauche à droite de deux listes de mots "spéculaires" ou inversés. Ainsi la séquence LEDNAHNEVALS (mot spéculaire de SLAVEN—HANDEL) était comparée à LEDNAH (mot spéculaire de HANDEL, adopté par le lexique) et ceci par une lecture de gauche à droite. Une fois l'identité établie, la composante détectée (LEDNAH) était reconvertie pour retrouver son aspect normal (HANDEL). Dans l'état présent du programme, cette inversion est abandonnée pour des raisons d'économie. La machine lit d'abord la dernière lettre du composé et écarte dans le lexique de la mémoire tous les termes qui ne se terminent pas par cette lettre; après quoi, elle lit l'avant-derniere et rejette parmi les termes encore retenus, ceux dont l'avant-derniere ne coincide pas avec celle-ci, et ainsi de suite jusqu'à ce que l'identité entre un terme entier du lexique et la composante terminale du composé soit parfaite. On voit donc l'intérêt qu'il peut y avoir à exécuter d'abord le programme de l'alphabétisation inverse.

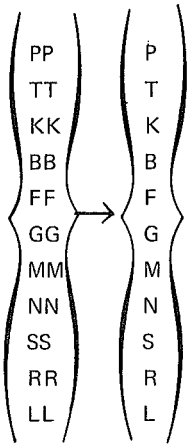
La véritable lemmatisation concerne donc tout ce qui précède la dernière composante du mot composé. Le programme doit prévoir un vecteur exhaustif des graphèmes de liaison en néerlandais. Les voici, avec

pour chaque cas représentatif un exemple :

| | | |
|---|---|---|
| /E/ | comme dans | KRANT/E/MAN |
| /EN/ | comme dans | BOER/EN/PSALM |
| /IEN/ | comme dans | KOE/IEN/HANDEL |
| /ER/ | comme dans | KIND/ER/SPEL |
| /S/ | comme dans | RAAD/S/LID |
| /–/ | comme dans | NA/–/AVOND |

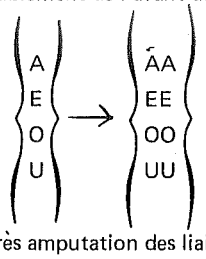
Il est bien évident que ce dernier trait, tout en étant un graphème de liaison, ne peut être considéré comme un morphème flexionnel, ni comme un phoneme de liaison. Mais étant donné les règles de l'orthographe néerlandaise, il était commode de l'integrer dans la série des graphemes à expulser. Munie de ces renseignements, la machine peut maintenant reprendre la lecture de gauche a droite et essayer d'identifier ce qui reste du lemme après l'identification de la dernière composante. Ce reste, qui peut contenir *n* composantes, est appelé LEFTWORD dans le programme. Jusqu'ici le programme doit permettre de résoudre tous les cas bicephales des types mentionnés.
Viennent ensuite les règles de réécriture. Envisageons-les cas par cas et dans l'ordre de leur application.

1. Si, à la fin d'un LEFTWORD déjà amputé de son morphème flexionnel et non encore résolu par les instructions précédentes, on trouve une consonne redoublée, expulser le dernier symbole.



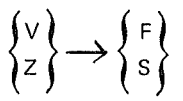

LAPP(...) ⟶ LAP
RATT(...) ⟶ RAT
TAKK(...) ⟶ TAK
WEBB(...) ⟶ WEB
STRAFF(...) ⟶ STRAF
EGG(...) ⟶ EG
STEMM(...) ⟶ STEM
VINN(...) ⟶ VIN
VISS(...) ⟶ VIS
STERR(...) ⟶ STER
VELL(...) ⟶ VEL

2. Redoublement de l'avant-dernière voyelle :



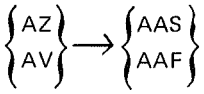

SOLDAT(...) ⟶ SOLDAAT
SUDET(...) ⟶ SUDEET
OR(...) ⟶ OOR
UR(...) ⟶ UUR

3. Si, après amputation des liaisons flexionnelles, le mot se termine par V ou Z :




BOEV(...) ⟶ BOEF
GLAZ(...) ⟶ GLAS

4. Application conjuguée des règles 3 et 2 :




HAZ(...) ⟶ HAAS
GRAV(...) ⟶ GRAAF

5. Application inverse de la règle 3 :

| | | | |
|---|---|---|---|
| F | V | SCHRIJF | SCHRIJV(...) |
| S | Z | LEES | LEZ(...) |

La cinquième règle est prévue pour le cas d'une composante sans morphème flexionnel se terminant par F ou S et dont le lexème figure dans le lexique sous la forme d'un infinitif. Ainsi, dans SCHRIJF/KUNST, la composante SCHRIJF ne figurant pas comme telle dans les dictionnaires, doit être lemmatisée d'abord avec la règle de réécriture 5, puis avec l'addition de la désinence /EN/.

L'application de l'ensemble de ces instructions de lemmatisation permet de résoudre (à quelques exceptions près) tous les composés bicéphales du type suivant : 1) HAND/WERK, 2) BOER(EN)KOST, 3) RAT(TE)VEL, 4) SOLDAAT(EN)PAK, 5) BOEF(EN)EED, 6) HAAS(E)PAD, 7) SCHRIJVEN/KUNST. L'ambiguïté de certaines instructions commence à poindre dès qu'on veut s'attaquer aux composés tricéphales. Souvent, il n'y a pas suffisamment de renseignements permettant à la machine de choisir entre l'interprétation de certains graphèmes du LEFTWORD comme *morphèmes flexionnels* ou comme *partie d'une des composantes*. Pour illustrer ces difficultés, analysons systématiquement quelques cas représentatifs restés non résolus dans la première version du programme.



Extrait de la Revue (R.E.L.O.)
VII, 1 à 4, 1971. C.I.P.L. - Université de Liège - Tous droits réservés.

a) PIJLSCHEEFKELK fut décomposé comme PIJL .S / CHEEF / KELK avec la remarque : "NO LEFTWORD HAS BEEN FOUND IN THE LEXIS", ce qui, selon les conventions du programme, signifie qu'une ou plusieurs composantes du LEFTWORD n'ont pas été identifiées. Que s'est-il passé ? La machine s'est arrêtée après la lecture de PIJL, qui fut identifié. Suivant l'ordre des instructions, elle a d'abord posé la question : y a-t-il, dans la suite, des graphèmes de liaison ? La réponse fut positive puisque SCHEEF commence par S, qui figure dans le vecteur des morphèmes flexionnels. Aussitôt après, elle a appliqué à la composante restante (CHEEF), non identifiée, la règle de réécriture 5 et a obtenu de cette manière CHEVEN. Cette forme lemmatisée ne coïncidant pas avec un lemme du lexique, elle l'a rejetée. Ce traitement a duré 1,340 seconde, ce qui est relativement long.

b) KERKERAADSLID fut décomposé comme KERK .ER / AAD .S / LID. L'origine de la faute est la même; cette fois, c'est ER qui est considéré comme morphème flexionnel. L'erreur peut facilement être évitée en postposant les règles de lemmatisation et en n'y recourant qu'au cas où les composantes ne sont pas autrement identifiables. La même erreur revient dans la décomposition de SCHOORSTEENGELD (SCHOOR .S/TEEN/GELD) et ZWALUWSTAARTVERBAND (ZWALUW .S/TAART/VERBAND) mais, cette fois, sans la remarque précitée parce que les mots TEEN et TAART se trouvent bel et bien dans le lexique.

c) WETSTAAL fut décomposé comme WET .S / TAAL, qui est une des décompositions possibles. Toutefois, dans ce cas, l'ambiguïté fait échec à toute tentative de solution parce que nous avons affaire à un composé dont la signification est double selon qu'on considère le /S/ comme un morphème flexionnel ou comme la première lettre de la composante STAAL. Dans l'adaptation ultérieure du programme, nous avons exigé les deux solutions.

d) VOLKORENBROOD fut décomposé comme VOLK / OOR / BROOD. Il s'agit ici d'une erreur tout à fait différente de celles traitées précédemment. En effet, par un effet de coïncidence, la machine a réussi à résoudre le problème (fût-ce de façon erronée) sans poser les questions de lemmatisation après l'identification de la première composante (fausse, elle aussi). Elle ne s'est pas contentée de l'identification de VOL (lemme figurant pourtant dans le lexique) parce qu'elle a obéi à une instruction donnant priorité à la ressemblance la plus longue. Cette instruction est évidemment à double tranchant : elle corrige de façon satisfaisante un grand nombre d'erreurs (par ex. dans HANDSTAND ou STAND est préféré à TAND) mais en entraîne aussi d'autres (par ex. KASTRAND ou STRAND est préféré à RAND). Dans ce dernier cas, le programme prévoit un doloop permettant de reprendre les instructions précédentes jusqu'à ce qu'une solution satisfaisante soit trouvée pour l'ensemble du composé en question. Ainsi, après l'amputation de STRAND, la machine ne trouve pas KA dans le lexique, recommence sa lecture de droite à gauche et propose RAND. Par le fait-même le LEFTWORD (KAST) est facilement identifié.



Extrait de la Revue (R.E.L.O.)
VII, 1 à 4, 1971. C.I.P.L. - Université de Liège - Tous droits réservés.

4. *Le bracketing automatique.*

Nous avons déjà attiré l'attention sur le fait que les problèmes posés par le bracketing automatique ne trouvent pas de solution convenable sans référence aux rapports syntaxiques et/ou sémantiques entre les composantes. Avant d'en arriver à cette conclusion, nous avons essayé d'élucider de façon empirique le problème de la distribution des morphèmes flexionnels pour savoir si celle-ci ne pouvait pas nous fournir les indications formelles du bracketing. D'une analyse portant sur quelque 500 composés tricéphales, il apparut :

1° : que, de façon générale, les graphèmes de liaison indiquent la séparation principale ou primaire, là où il n'y a qu'une seule flexion.

2° : que dans les composés où il n'y a aucune flexion ou plus d'une seule, la séparation primaire concerne la deuxième et la troisième composante.

3° : que lorsqu'il y a deux flexions de type différent dans un seul composé, c'est /S/ qui a une certaine priorité (statistique) par rapport aux autres flexions : /EN/ etc.

Les exemples suivants illustrent les constatations empiriques qu'on vient d'énumérer :

| | | | |
|---|---|---|---|
| ZOMERNACHTSDROOM | : | A/B//C | (une seule flexion : voir 1°) |
| DORPSSCHOOLMEESTER | : | A//B/C | (une seule flexion : voir 1°) |
| LEKENRECHTSPRAAK | : | A//B/C | (une seule flexion : voir 1°) |
| AALBESSESTRUIK | : | A/B//C | (une seule flexion : voir 1°) |
| KNOOPSGATESCHAAR | : | A/B//C | (deux flexions /S/ et /EN/ : concorde avec 2° mais s'oppose à 3°) |
| VOLLEMAANSGEZICHT | : | A/B//C | (deux flexions /E/ et /S/ : concorde avec 2° et 3°) |
| STAATSMANSGENIE | : | A/B//C | (deux flexions /S/ et /S/ : voir 2°) |
| KRANTEMANNENSYNDIKAAT | : | A/B//C | (deux flexions /E/ et /EN/ : voir 2°) |
| KOUDWATERBAD | : | A/B//C | (aucune flexion : voir 1°) |

Ces données empiriques peuvent avoir une signification statistique mais elles sont bien loin de suffire à l'instruction de la machine, non seulement parce qu'il y a un certain nombre de contre-exemples (pensons par ex. à GODSDIENSTOEFENING où le morphème flexionnel indique la séparation secondaire) mais également parce que ces indications formelles, même si elles ne souffraient aucune exception, ne permettraient pas pour autant de simuler ou de générer automatiquement des composés néerlandais grammaticaux. Un autre inconvénient, c'est que ces règles ne s'appliquent plus aux composés comprenant plus de trois composantes (du type STEENBOKSKEERKRING et STIKSTOFWATERSTOFZUUR).

Lorsqu'on consulte la littérature spécialisée pour trouver des analyses linguistiques susceptibles d'inspirer la programmation des rapports syntaxiques et/ou sémantiques des composés, on ne voit guère, pour ce



Extrait de la Revue (R.E.L.O.)
VII, 1 à 4, 1971. C.I.P.L. - Université de Liège - Tous droits réservés.

qui concerne le néerlandais, que les traités déjà relativement anciens de A. Reichling et W. De Groot (13). Chez le premier, c'est la théorie de l'enchaînement qui prime, enchaînement qu'on est enclin d'appeler "ensembliste" puisqu'il envisage les composantes des mots comme des ensembles partageant des éléments avec une ou plusieurs autres composantes. Ainsi, A peut partager un élément avec B, qui partage à son tour un élément avec C, sans que pour autant A et C aient un élément commun. A cette théorie, De Groot préfère celle du noyau central auquel les différentes variantes se rattachent. Il prévoit trois sortes de composés : a) les *coordonnés* du type TUINMAN-CHAUFFEUR, b) les composés construits sur un rapport de déterminant à déterminé (DAGCHAUFFEUR) et c) les composés dont une des composantes n'a pas de sens prise isolément (CRANBERRY). Plus importante pour notre étude est son affirmation selon laquelle la signification du composé n'est pas déductible de l'addition des sens des composantes : même si on connaît la signification des mots ROOD et BORST, rien ne permet d'en conclure que le ROODBORSTJE est un oiseau. Cette remarque, qui s'oppose à une théorie énoncée par H. Schultink (14), nous paraît importante dans la mesure où, sans même envisager la possibilité d'une subcatégorisation sémantique, elle exclut en principe toute simulation automatique de composés métaphoriques au sens large (entendons : de tout composé dont le signifié n'est pas la somme des signifiés des composantes). Cependant, pour intéressantes que puissent être ces remarques, et encore quelques autres (15), elles ne suffisent pas à l'élaboration d'un modèle théorique ou automatique du composé néerlandais, que nous poursuivons.

## 5. *Profondeur et surface.*

La nécessité de se référer à autre chose que ce qui apparaît à la surface du mot composé, a trouvé pour la première fois une expression claire et cohérente dans un ouvrage de R.B. Lees, *The Grammar of English Nominalizations,* publication de 1966, à laquelle il est exclu de ne pas faire allusion lorsqu'on traite des composés. Témoins les quelques linguistes contemporains qui, sans suivre Lees sur le plan théorique, ont pourtant été obligés de situer leur pensée par rapport à la sienne. Pour les composés français, il y a l'ouvrage de C. Rohrer (16) et pour ceux de l'allemand des pages tout à fait remarquables dans l'ouvrage de W. Wurzel (17). Mais l'analyse la plus originale, et qui a le rare mérite de rendre plus claire l'étude de Lees, tout en s'y opposant par bien des côtés, c'est certainement l'important ouvrage de R.P. Botha (18), qui traite entre autres des composés de l'Afrikaans mais déborde largement le cadre de cette seule langue. Afin de rendre compte de la direction que devrait prendre notre programme de bracketing et de simulation de composés néerlandais, il est utile de mettre en regard les théories de Lees et de Botha.

Pour Lees, la composition des éléments lexicaux peut être adéquatement décrite en termes de rapports syntaxiques entre les "formatifs" lexicaux et par la dérivation des structures composées superficielles à partir de la structure profonde. Cette structure profonde est présentée comme une phrase prédicative. Ainsi, pour appliquer le modèle de Lees à un exemple néerlandais, le composé ZOETWATERVIS (poisson d'eau douce) ne serait qu'une structure superficielle dérivable d'une phrase comprenant successivement des prédications telles que "een vis leeft (of woont) in water" (un poisson vit dans l'eau) et "water is

zoet" (l'eau est douce). Pour douter de l'adéquation de ces règles, Botha tire son argument principal de l'idiosyncrasie particulière des composés en Afrikaans (qui vaut en grande partie pour le néerlandais). C'est ainsi qu'en Afrikaans, les phonèmes de liaison semblent être tout à fait imprévisibles. Nous avons constaté, quant à nous, que malgré une certaine probabilité difficile à évaluer dans le cas des composés tricéphales, ce caractère aléatoire des phonèmes de liaison s'applique aussi au néerlandais. Il serait dès lors impossible de générer automatiquement des composés néerlandais acceptables (et d'exclure les composés agrammaticaux) sur la base du modèle *profond* de Lees, modèle qui n'admet pas le composé comme membre à part entière du lexique mais comme une structure superficielle dérivée. Si cela était exact nous serions obligés de prévoir une routine "ad hoc" réglant la casuistique des phonèmes de liaison.

La même remarque vaut pour l'idiosyncrasie des sens figurés ou métaphoriques des composés. Ici, la théorie de De Groot prend tout son sens. Les métaphores lexicales sont trop spécifiques, trop diverses et trop imprévisibles pour qu'une dérivation comprenant des marqueurs syntaxiques et sémantiques puisse jamais les simuler à partir d'une structure profonde suggérée par Lees. Comment subcatégoriser le formatif ROOD (rouge) quand on sait que, dans un cas, il devra déterminer BORST (rouge-gorge) pour que le composé puisse acquérir le signifié "oiseau", et que, dans l'autre, il devra déterminer KAPJE (chaperon rouge) pour signifier un personnage légendaire. La simulation automatisée à partir de structures profondes ne sera donc possible que si l'on exclut dès le départ les composés métaphoriques et ceux dont les phonèmes de liaison n'obéissent pas à un schéma strict et explicite.

Un avantage de la théorie de Lees, c'est qu'elle serait apte, le cas échéant, à rendre explicitement compte de certaines inversions de symboles terminaux des composés, ce qui est d'une importance capitale pour la simulation automatique en préparation. Ainsi, pour l'inversion d'un composé bicéphale tel que KERK-ORGEL (orgue d'église) $\Longrightarrow$ ORGELKERK (église à orgue), il ne s'agirait pas seulement d'un terminant et d'un déterminé qui changent de rôle, mais de deux dérivations différentes à partir de deux phrases : d'une part *"een orgel is in de kerk"* et d'autre part *"de kerk bezit een orgel"*. Cette inversion se conçoit également pour les composés tricéphales à condition de respecter la séparation primaire. Ainsi A//B/C pourrait être transformé en B/C//A mais non en C/A//B tout comme A/B//C pourrait l'être en C//A/B mais non en C//B/A. Un exemple :

(ZAKWOORDENBOEK) ZAK//WOORD/BOEK $\Rightarrow$ WOORD/BOEK//ZAK $\nRightarrow$ BOEK/ZAK//WOORD
(SPREEKWOORDENBOEK) SPREKEN/WOORD//BOEK $\Longrightarrow$ BOEK//SPREKEN/WOORD $\nRightarrow\Rightarrow$ WOORD//BOEK/SPREKEN

La question de savoir si de tels produits d'inversion font partie de la production linguistique, serait alors renvoyée à l'analyse de la performance. Pour les composés basés sur un rapport déterminant-déterminé (la deuxième catégorie de De Groot), cette inversion est toujours licite, bien que, souvent, le résultat ne soit pas (encore) adopté par les dictionnaires. Ainsi, lorsqu'on peut spécifier certains livres comme des *livres d'images*, il est également possible de spécifier certaines images comme des *images de livre* (PRENT... /BOEK $\Longrightarrow$ BOEK.../PRENT). La référence explicite à deux phrases différentes dans la structure profonde en rendrait adéquatement compte, sans que pour autant, comme nous l'avons souligné avec Botha,



Extrait de la Revue (R.E.L.O.)
VII, 1 à 4, 1971. C.I.P.L. - Université de Liège - Tous droits réservés.

les problèmes de l'idiosyncrasie des liaisons et des métaphores soient résolus.

6. *Reproduction du programme "AUTOMATIC COMPOUND WORD SPLIT".*

Dans les thèses annexes de son doctorat, H. Brandt-Corstius (19) affirme que l'explication d'un programme d'ordinateur n'a pas beaucoup de sens avant que le programme lui-même ne soit publié in extenso. Voici donc le programme, pour autant qu'il soit déjà opérationnel, avec un bref fragment de l'output.

** COMPOUND WORDS

DATUM 09/12/70 TIJD 03.31.33.990

DATUM 09/12/70 TIJD 03.31.34.210

DWERGNIJLPAARD

DWERG . /NIJL . /PAARD . /

DATUM 09/12/70 TIJD 03.31.34.650

HULPWERKWOORD

HULP . /WERK . /WOORD . /

DATUM 09/12/70 TIJD 03.31.35.410

GELOOFSINHOUD

** NO KERNEL HAS BEEN FOUND IN THE LEXIS
GELOOF .S /IN . /HOUDEN . /

DATUM 09/12/70 TIJD 03.31.36.290

ZEEWATERSCHADE

ZEE . /WATER . /SCHADE . /

DATUM 09/12/70 TIJD 03.31.36.730

DWARSDOORSNEDE

DWARS . /DOOR . /SNEDE . /

DATUM 09/12/70 TIJD 03.31.37.090

AFVALZIJDE

AF . /VAL . /ZIJDE . /

DATUM 09/12/70 TIJD 03.31.37.530

ZUIDWESTZIJDE

ZUID . /WEST . /ZIJDE . /

DATUM 09/12/70 TIJD 03.31.38.050

ZEVENTIENDUIZEND

ZEVEN . /TIEN . /DUIZEND . /

DATUM 09/12/70 TIJD 03.31.38.410

LANDBOUWHUISHOUDKUNDE

LAND . /BOUWEN . /HUIS . /HOUDEN
KUNDE . /

DATUM 09/12/70 TIJD 03.31.39.550

DAGDIENSTBODE

DAG . /DIENST . /BODE . /

NAJAARSSTORM

NA . /JAAR .S /STORM

DATUM 09/12/70 TIJD 03.31.46.210

ROVERHOOFDMAN

ROVER . /HOOFD . /MAN

DATUM 09/12/70 TIJD 03.31.46.650

MIDDELPUNTSTRALEN

** NO KERNEL HAS BEEN FOUND IN THE LEXIS
** NO LEFTWORD HAS BEEN FOUND IN THE LEXIS
MIDDEL . /PUNT .S /TRALEN . /

DATUM 09/12/70 TIJD 03.31.48.140

PIJPCELPOLIEP

PIJP . /CEL . /POLIEP . /

DATUM 09/12/70 TIJD 03.31.48.540

GLASBLAZERSZEEP

** NO LEFTWORD HAS BEEN FOUND IN THE LEXIS
GLAS . /BLAZEN . /ERS . /ZEEP

DATUM 09/12/70 TIJD 03.31.50.140

** NO LEFTWORD HAS BEEN FOUND IN THE LEXIS
STIKKEN . /STOF . /WATER .S /TOF
ZUUR . /

DATUM 09/12/70 TIJD 03.31.56.800

SLOKDARMKANKER

SLOK . /DARM . /KANKER . /

```
   DATUM  09/12/70            TIJD    03.31.57.260

HUWELIJKSVERMOGENSRECHT
                                HUWELIJK      .S   /VERMOGEN      .S   /RECHT        .   /
   DATUM  09/12/70            TIJD    03.31.57.520

KRIJGSMANSPLICHT
                                KRIJG         .S   /MAN           .S   /PLICHT      .   /
   DATUM  09/12/70            TIJD    03.31.57.900

AVONDWEERBERICHT
                                AVOND         .    /WEER         .    /BERICHT     .   /
   DATUM  09/12/70            TIJD    03.31.58.220

AVONDGODSDIENSTOEFNING
                                **    NO KERNEL HAS BEEN FOUND IN THE LEXIS
                                ** NO LEFTWORD  HAS BEEN FOUND IN THE LEXIS
                                AVOND         .    /GOD          .S   /DIENST      .   /OEFNING

   DATUM  09/12/70            TIJD    03.31.59.860

SPITSBOEVENGEZICHT
                                SPITS         .    /BOEF         .EN  /GEZICHT
   DATUM  09/12/70            TIJD    03.32.00.820

GEWELDDAAD
                                GEWELD        .    /DAAD         .    /
   DATUM  09/12/70            TIJD    03.32.01.080

LANGSMAAD
                                LANGS         .    /MAAD         .    /
   DATUM  09/12/70            TIJD    03.32.01.380

ZEERAAD
                                ZEE           .    /RAAD         .    /
   DATUM  09/12/70            TIJD    03.32.01.740
```

```
AAPDAPPELZAAD
                                        ** NO LEFTWORD  HAS BEEN FOUND IN THE LEXIS
                                      AAPDAPPEL      .    /ZAAD          .    /

 DATUM  02/12/70            TIJD    03.32.02.980


        /*   COMPOUND  WORDS  **  VAN OVERBEKE  **    */


                /*   COMPOUND  WORDS  **  VAN OVERBEKE  **    */
 1                PROG : PROCEDURE OPTIONS(MAIN);
 2                DECLARE
                   LEFTWORD(9) CHAR(40),    LEXWORD(500) CHAR(20),
                  LEXWORDC CHAR(20),
                  LEXLENGTH(500),
                        IPL(0:9),IPLFL(0:9),FLEX(9)   CHAR(4 ),
                    FORS(2) CHAR(1),VORZ(2) CHAR(1),  LEXWORDFLEX(2) CHAR(20),
                 COMPWORD CHAR(40),
                  VOWEL(4) CHAR(1),
                   KERNEL CHAR(20),
                     INFLECTS(7) CHAR(4),INFLECTLENGTH(7);
 3               FORS(1)='F'; FORS(2)='S'; VORZ(1)='V'; VORZ(2)='Z';
 7                INFLECTS(1)='IEN ';
 8                INFLECTS(2)='IE  ';
 9                INFLECTS(3)='ER  ';
10                INFLECTS(4)='S   ';
11                INFLECTS(5)='-   ';
12                INFLECTS(6)='EN  ';
13                INFLECTS(7)='E   ';
14                VOWEL(1)='A'; VOWEL(2)='E'; VOWEL(3)='O'; VOWEL(4)='U';
18               DO K=1 TO 7;
19               INFLECTLENGTH(K)=INDEX(INFLECTS(K),' ')-1;
20                 END ;
21                OPEN FILE (SYSPRINT) PAGESIZE(40);
22               PUT SKIP(3);
23               I=0;
24                 READ_LEXIS :
                  I=I+1;
25               GET EDIT(LEXWORD(I))(A(20));
26               GET SKIP ;
27               LEXLENGTH(I)=INDEX(LEXWORD(I),' ')-1;
28               PUT       EDIT(LEXWORD(I))                  (SKIP,A(20));
29                IF LEXWORD(I)='*' THEN GO TO END_LEXIS ;
31                GO TO READ_LEXIS ;
32                END_LEXIS :
                   NLEX=I-1;


        /*   COMPOUND  WORDS  **  VAN OVERBEKE  **    */


33                  ALPHA_LEX :
                    DO I=1 TO NLEX -1;
34                 DO J=I+1 TO NLEX ;
35                   IF LEXLENGTH(I) < LEXLENGTH(J)
                     |( LEXLENGTH(I) = LEXLENGTH(J)  &
36                     LEXWORD(I) > LEXWORD(J) ) THEN DO ;
37                    LEXWORDC=LEXWORD(I);
```

```
38                  LEXWORD(I)=LEXWORD(J);
39                   LEXWORD(J)=LEXWORDC;
40                 LEXLENGTHC=LEXLENGTH(I);
41                    LEXLENGTH(I)=LEXLENGTH(J);
42                    LEXLENGTH(J)=LEXLENGTHC;
43                  END ;  END;    END ;
46                  PUT EDIT('  **  SORTED_LEXIS')(PAGE,A);  PUT SKIP(3);
48                   DO I=1 TO NLEX ;
49                  PUT       EDIT(LEXWORD(I))                    (SKIP,A(20));
50                  END ;
51                 PUT EDIT('  **  COMPOUND WORDS')(PAGE,A);
52                 PUT SKIP(3);
53                READ_COMPOUND_WORD :
                     GET EDIT   (COMPWORD)(A(40));
54                   IF COMPWORD='*' THEN GO TO END_PROGRAM ;
56                  GET SKIP ;
57                  PUT EDIT(COMPWORD)(SKIP(2),A(40));
58                 ICOMPLENGTH=INDEX(COMPWORD,' ')-1;
59                 COMPARE_KERNEL :
                   DO I=1 TO NLEX ;
60                 IF ICOMPLENGTH < LEXLENGTH(I) THEN GO TO END_COMPARE_KERNEL ;
62                 IF SUBSTR(COMPWORD,ICOMPLENGTH-LEXLENGTH(I)+1,LEXLENGTH(I))=
63                  LEXWORD(I) THEN GO TO KERNEL_FOUND ;
64                 END_COMPARE_KERNEL : END COMPARE_KERNEL ;
65                  KERNEL_NOT_FOUND :
                     PUT EDIT((40)' ',' **    NO KERNEL HAS BEEN FOUND IN THE LEXIS')
                       (SKIP,A,A);
66                    IPLK=ICOMPLENGTH ;
67                    KERNEL=(20)' ';
68                 GO TO INITIAL_LEFT ;

        /*   COMPOUND  WORDS  **  VAN OVERBEKE  **     */

69                  KERNEL_FOUND:
                   J=LEXLENGTH(I);
70                  IPLK=ICOMPLENGTH-J;
71                  KERNEL=SUBSTR(COMPWORD,ICOMPLENGTH-J+1,J);
72                  INITIAL_LEFT :
                    IND=0;
73                 IPLFL(0)=0;
74                   SEQUENTIAL_LEFT :
75                 IF IPLFL(IND)=IPLK THEN GO TO  PRINT_POINT ;
76                  IND=IND+1;
77                  COMPARE_LEFT_WORD :
                   DO I=1 TO NLEX ;
78                 IF IPLK-IPLFL(IND-1) < LEXLENGTH(I) THEN GO TO END_COMPARE_LEFT_WORD
                     ;
80                  IF SUBSTR(COMPWORD,IPLFL(IND-1)+1,LEXLENGTH(I))= LEXWORD(I) THEN
81                 GO TO LEFT_WORD_FOUND ;
82                 END_COMPARE_LEFT_WORD :
                        END COMPARE_LEFT_WORD ;
83                   GO TO DELAY ;
84                  LEFT_WORD_FOUND :
                    J= LEXLENGTH(I);
```

```
 85                  IPL(IND)=IPLFL(IND-1)+J;
 86                  LEFTWORD(IND)=SUBSTR(COMPWORD,IPLFL(IND-1)+1,J);
 87                  IF IPL(IND)=IPLK THEN DO ;
 89                   IPLFL(IND)=IPL(IND);
 90                   FLEX(IND)='     '; GO TO PRINT_POINT ;
                        /*   EX  :   VERSREGEL                          */
 92                  END ;
 93                  DO K=1 TO 7;
 94                 IF SUBSTR(COMPWORD,IPL(IND)+1,INFLECTLENGTH(K))=INFLECTS(K)
 95                  THEN GO TO INFLECTS_FOUND ;
 96                 END ;
 97                  GO TO DOUBLE_CHAR_CYC ;
 98                  INFLECTS_FOUND :
                    FLEX(IND)=INFLECTS(K);
 99                 IPLFL(IND)=IPL(IND)+INFLECTLENGTH(K);
                        /*   EX  :   BOERENPSALM                        */

          /*   COMPOUND  WORDS  **  VAN OVERBEKE  **     */

100                  GO TO SEQUENTIAL_LEFT ;
101                  DOUBLE_CHAR_CYC :
                     DO K=6 TO 7;
102                 IF SUBSTR(COMPWORD,IPL(IND)+2,INFLECTLENGTH(K))=INFLECTS(K)
103                  THEN GO TO  TEST_DOUBLE_CHARACTER ;
104                   END ;
105                  GO TO TEST_MORE_GROUPS ;
106                  TEST_DOUBLE_CHARACTER :
                     IF SUBSTR(COMPWORD,IPL(IND)+1,1)=
107                   SUBSTR(COMPWORD,IPL(IND),1)  THEN DOUBLE_CHAR_FOUND : DO ;
108                  FLEX(IND)=INFLECTS(K);
109                  IPLFL(IND)=IPL(IND)+1+INFLECTLENGTH(K);
                        /*   EX  :   KATTEVEL                           */
110                 GO TO SEQUENTIAL_LEFT ;
111                 END ;
112                  TEST_MORE_GROUPS :
                    IPLFL(IND)=IPL(IND);
113                  FLEX(IND)='     ';
                        /*   EX  :   KOUDWATERBAD                       */
114                 GO TO SEQUENTIAL_LEFT ;
115                  DELAY :
                    DO I=1 TO NLEX ;
116                  LEXWORDC=LEXWORD(I);
117                  LEXLENGC=LEXLENGTH(I);
118                  IF LEXLENGC<3 THEN GO TO END_FLEX_LEXIS ;
120                  IFL=0;
121                  DO K=1 TO 4;
122                   IF SUBSTR(LEXWORDC,LEXLENGC-2,1)=VOWEL(K)   &
123                      SUBSTR(LEXWORDC,LEXLENGC-1,1)=VOWEL(K)   THEN  DO ;
124                    IFL=IFL+1;
125                  LEXWORDFLEX(IFL)=SUBSTR(LEXWORDC,1,LEXLENGC-2)  ||
126                      SUBSTR(LEXWORDC,LEXLENGC,1);      END ;    END;
                       /*   EX  ;   SOLDATENMANTEL                     */
128                 DO K=1 TO 2;
129                  IF SUBSTR(LEXWORDC,LEXLENGC,1)=FORS(K)  THEN DO ;
```

```
131                IFL=IFL+1;
132                 LEXWORDFLEX(IFL)=SUBSTR(LEXWORDC,1,LEXLENGC-IFL) || VORZ(K);

          /*   COMPOUND  WORDS  **  VAN OVERBEKE  **     */


133               END ; END ;
                      /*   EX  :   SCHROEVENDRAAIER                    */
                      /*   EX  :   SLAVENDRIJVER                       */
135                IFM=IFL ;
136               DO IFL=1 TO IFM ;
137              LENGWFLEX=INDEX(LEXWORDFLEX(IFL),' ')-1;
138              IF SUBSTR(COMPWORD,IPLFL(IND-1)+1,LENGWFLEX)=LEXWORDFLEX(IFL)
139                THEN GO TO   FLEX_WORD_FOUND ;
140               END ;
141               GO TO TEST_ON_EN ;
142                  FLEX_WORD_FOUND :
                   DO K=6 TO 7;
143               IF SUBSTR(COMPWORD,IPL(IND-1)+1+LENGWFLEX,INFLECTLENGTH(K))=
144                INFLECTS(K) THEN GO TO
                  FLEX_END ;
145               END ;
146                GO TO TEST_ON_EN ;
147                  FLEX_END :
                    IPL(IND)=IPLFL(IND-1)+LENGWFLEX ;
148              IPLFL(IND)=IPL(IND)+INFLECTLENGTH(K);
149                LEFTWORD(IND)=LEXWORDC;
150                              FLEX(IND)=INFLECTS(K);
151                GO TO SEQUENTIAL_LEFT ;
152                TEST_ON_EN :
153                IF SUBSTR(LEXWORDC,LEXLENGC-1,2)='EN'  THEN GO TO  ENDING_ON_EN ;
154                 ELSE GO TO   END_FLEX_LEXIS ;
155               ENDING_ON_EN :
156                IF LEXLENGC<4 THEN GO TO END_FLEX_LEXIS ;
157               IF SUBSTR(LEXWORDC,LEXLENGC-3,1)=
158                  SUBSTR(LEXWORDC,LEXLENGC-2,1)   THEN GO TO INV_DOUBLE ;
159                 ELSE GO TO TEST_VOWEL ;
160               INV_DOUBLE :
                   IF SUBSTR(COMPWORD,IPLFL(IND-1)+1,LEXLENGC-3)=
161                 SUBSTR(LEXWORDC,1,LEXLENGC-3)   THEN INV_DOUBLE_FOUND : DO ;
162                  IPL(IND)=IPLFL(IND-1)+LEXLENGC-3;
163                 IPLFL(IND)=IPL(IND);
164                 LEFTWORD(IND)=LEXWORDC;

          /*   COMPOUND  WORDS  **  VAN OVERBEKE  **     */


165                FLEX(IND)='    ';
166                GO TO INITIAL_LEFT;
                      /*   EX  :   SCHATPLICHT                         */
167               END  INV_DOUBLE_FOUND;
168                 TEST_VOWEL :
                    IGL=0;
169                 DO K=1 TO 4;
```

```
170                IF SUBSTR(LEXWORDC,LEXLENGC-3,1)=VOWEL(K)
171               THEN          DOUBLE :       DO ;  IGL=IGL+1;
173               LEXWORDFLEX(IGL)= SUBSTR(LEXWORDC,1,LEXLENGC-3) ||
                   SUBSTR(LEXWORDC,LEXLENGC-3,2);
                      /*   EX  :   BEHAAGZUCHT                          */
174                END;  END ;
176                 DO K=1 TO 2;
177                IF SUBSTR(LEXWORDC,LEXLENGC-2,1)=VORZ(K)  THEN       DO ;
179                   IGL=IGL+1;
180                 LEXWORDFLEX(IGL)= SUBSTR(LEXWORDC,1,LEXLENGC-3)  ||
                    SUBSTR(LEXWORDC,LEXLENGC-3,IGL-1)  ||  FLEXS(K);
                      /*   EX  :   DRIJFZAND                            */
                      /*   EX  :   SCHAAFWONDE                          */
181               END ;END ;
183                   IGM=IGL;
184               DO IGL=1 TO IGM ;
185              LENGWFLEX=INDEX(LEXWORDFLEX(IGL),' ')-1;
186              IF SUBSTR(COMPWORD,IPLFL(IND-1)+1,LENGWFLEX)=LEXWORDFLEX(IGL)
187                 THEN  GO TO INV_FLEX_FOUND ;
188               END ;
189              GO TO END_FLEX_LEXIS ;
190               INV_FLEX_FOUND :
                   IPL(IND)       =IPLFL(IND-1)+LENGWFLEX ;
191               IPLFL(IND)=IPL(IND)                        ;
192               LEFTWORD(IND)=LEXWORDC;
193                FLEX(IND)='     ';
194                GO TO SEQUENTIAL_LEFT ;
195               END_FLEX_LEXIS    :  END DELAY ;
196                 PUT EDIT((40)' ',' ** NO LEFTWORD  HAS BEEN FOUND IN THE LEXIS')
                    (SKIP,A(40),A);

        /*   COMPOUND  WORDS  **  VAN OVERBEKE  **    */

197                IPL(IND)=IPLK;
198                IPLFL(IND)=IPLK;
199                LEFTWORD(IND)=SUBSTR(COMPWORD,IPLFL(IND-1)+1,IPLK-IPLFL(IND-1));
200                FLEX(IND)='     ';
201                PRINT_POINT :
                    INDM=IND ;
202                IND1=IND+1;
203                 LEFTWORD(IND1)=KERNEL ;
204                 FLEX(IND1)='     ';
205               IPL(IND1)=ICOMPLENGTH ;
206                  IPLFL(IND1)=ICOMPLENGTH ;
207                PUT EDIT((40)' ')(SKIP,A(40));
208               PUT EDIT((SUBSTR(LEFTWORD(IND),1,14),'.',FLEX(IND),'/'
                  DO IND=1 TO IND1))(A(14),A(1),A(4),A(1));
209               GO TO READ_COMPOUND_WORD ;
210               END_PROGRAM :
                 PUT EDIT('  **  END OF CARDS HAS BEEN REACHED')(SKIP(5),A);
211                 END PROG ;
```

(*) Je tiens à remercier le mathématicien L. DE BUSSCHERE, pour sa collaboration indispensable à la réalisation de ce programme.

(1) J. Watson, *La double hélice,* Paris, 1970.

(2) Nous songeons entre autres à J. Friedman, *Applications of a Computer System for Transformational Grammar* et A. Querido, *Computer Implemetation of a Transformational Description of French* (deux communications au International Congress of Computational Linguistics, Sanga-Säby, sept., 1969). Voir également Y. Batori, *Disambiguating Verbs with Multiple Meaning in the MT-System of IBM-Germany,* ITL, vol. 5, 1969, p. 5-16.

(3) Cfr. A. Kraak, *Negatieve Zinnen. Een methodologische en grammatische analyse,* Hilversum, 1966, (tout le paragraphe 4).

(4) A. Kraak et W. Klooster, *Syntaxis.* Culemborg/Cologne, 1968, p. 15.

(5) W. Martin, *De verwerking van linguïstische opgaven door middel van een computer : de stand van zaken aan het ITL* (dans ce même recueil).

(6) J.G. Savard, *La valence lexicale,* Montréal, 1970, p. 27.

(7) W. Martin en R. Eeckhout, *Evolutie van de woordenschat in het huidige Nederlands,* Revue des langues vivantes, XXXV, 1969, 1, p. 72-73.

(8) C'est ce que nous avons tenté de faire pour un roman de Hugo Claus. Cfr. *Quelques applications du modèle ensembliste au contact entre les langues* (Recueil d'hommage à J.L. Pauwels, Louvain, 1971).

(9) A. Martinet, *Le français sans fard,* Paris, 1969, p. 59.

(10) Cfr. G.U. Yule, *The statistical study of literary vocabulary,* Cambridge, 1944, p. 59 : "All these are minor troubles, and difference of treatment would make very little difference in the final results".

(11) Voir au paragraphe 5 : *Profondeur et surface,* l'analyse de la théorie de R.B. Lees.

(12) Il y a deux sortes d'exceptions à cette règle. La première se présente lorsque, dans le cas d'un composé tricéphale (ABC), deux composés secondaires figurent dans le lexique, à savoir AB et BC; ainsi par ex. : SCHAAPHERDERSHOND (SCHAAPHERDER et HERDERSHOND existent tous deux) ou ZEEWATERSCHADE (ZEEWATER et WATERSCHADE existent tous deux). Dans ce cas, il y a ambiguïté formelle que la machine ne peut résoudre sans instructions supplémentaires. Le second cas est celui des nombres composés (ZEVENTIENDUIZEND, ACHTHONDERDMILIOEN). On sait

que le rapport arithmétique entre les composantes d'un tel nombre équivaut à la multiplication. Or, selon le principe de commutativité en vigueur en mathématiques, le produit de (AxB)xC est exactement le même que celui de Ax(BxC). Dans ces circonstances, même des références supplémentaires à un syntagme ou à une phrase plus "profondes" (cfr. Lees) ne suffiront pas à résoudre les problèmes du bracketing.

(13) A.W. de Groot, *Betekenis en betekenisstructuur* (Manuscrits publiés après la mort de l'auteur par Mme G.F. Bos et M.H. Roose), Groningue, 1966 et *Inleiding tot de algemene taalwetenschap,* Groningue, 1964², p. 239-242. A. Reichling, *Het woord,* Nimègue, 1967².

(14) H. Schultink, *De morfologische valentie van het ongelede adjectief in modern Nederlands,* La Haye, 1962, p. 10 : "De betekenis van composita is een vereniging van pre-existerende woordbetekenissen. Op grond daarvan wordt een compositum begrepen door ieder die de betreffende *simplicia* kent".

(15) Entre autres K. Heeroma, *Klemverschuiving bij samengestelde woorden,* Nieuwe Taalgids, 1949 et B. van den Berg, *De accentuatie van Nederlandse samenstellingen en afleidingen,* Nieuwe Taalgids, 1953.

(16) Ch. Rohrer, *Die Wortzusammensetzung im modernen französisch,* (Dissertation inaugurale), Tubingen, 1967.

(17) W.U. Wurzel, *Studien zur deutschen Lautstruktur,* Studia grammatica, 8, Berlin, 1970.

(18) R.P. Botha, *The Function of the Lexicon in Transformational Generative Grammar,* Janua Linguarum, Series major, La Haye, 1968.

(19) H. Brandt-Corstius, *Exercices in Computational Linguistics,* Amsterdam, 1970.

*SUMMARY*

In his introduction the author suggests that the computer be more and more employed by linguists as a simulation device rather than as a high-speed calculator or as a faultless classification-machine. In this scope a computer program intended for automatic split and bracketing of Dutch compounds is presented. Motives and needs that gave rise to this initiative are briefly cited, as well as wants that might be supplied by it. The program called AUTOMATIC COMPOUND WORD SPLIT is amply explained both with regard to the lemmatising rules and to the rewriting rules. Several limitations and ambiguities so far unsolved are treated. An attempt to derive bracketing instructions from the surface structure of compounds did not answer. Recent theories about generating compound lexical items by transformation rules applying to deep syntactic structures (such as Lees' and Botha's) are compared and elucidated. Finally, an outprint of the computer program in PL1 and some output sheets are added.