Publié dans Travaux du Centre de Recherches Sémiologiques 48, 1-32, 1985,
source qui doit être utilisée pour toute référence à ce travail

LA STRUCTURE DES DONNEES ET DES ALGORITHMES EN DEREDEC

Pierre Plante

1. IMPLEMENTATIONS

Le Déredec est un logiciel actuellement opérationnel en NEW-UCI LISP (sur des machines DEC-10), en INTERLISP (sur IBM, en version réduite) et en IQ-LISP sur micro-ordinateur IBM PC.

Une version pour VAX (Digital Equipment Corporation) en LISP NIL (M.I.T.) sera bientôt disponible.

2. VUE D'ENSEMBLE

Le Déredec est un logiciel de traitement linguistique, d'analyse de contenu des textes et de mise au point de systèmes-experts en langage naturel. Il s'agit d'un système de programmation exigeant peu de connaissances préalables en informatique. Tout chercheur en sciences humaines (philosophie, linguistique, psychologie, sociologie...) peut ainsi, à l'aide de cet outil, mettre au point et appliquer ses propres hypothèses de description de texte (description morphologique, syntaxique, sémantique, logique...), ses propres idées sur l'exploration des descriptions construites et l'analyse du contenu des résultats, ainsi qu'éventuellement son propre système de questions/réponses.

Le Déredec se définit comme un cadre computationnel général, simple et souple pour le traitement polyvalent des langues naturelles.

Le Déredec facilite la programmation d'analyses situées à d'autres niveaux que ceux auxquels nous ont habitués les logiciels con-

sacrés au dépistage des concordances et à l'analyse des cooccurrences.

Dans la production des concordances (segments d'un texte renfermant un mot clé donné), et dans l'analyse des cooccurrences (deux mots ont une relation de cooccurrence s'ils appartiennent au même segment), le segment, c'est-à-dire l'unité d'investigation contextuelle se définit habituellement dans les termes de certains traits d'édition (par exemple la "ligne" ou la "page") ou de bornes numériques (un certain nombre de mots avant ou après le mot étudié). Par la suite l'analyseur opère des regroupements statistiques sur les mots des segments dépistés.

Quelle que soit la sophistication des calculs effectués sur ces regroupements, l'interprétation finale des résultats reste toujours liée à la pauvreté initiale de l'investigation contextuelle.

C'est ce constat qui a motivé le développement du Déredec. Nous voulions répondre au besoin d'un logiciel général consacré à la programmation d'investigations contextuelles définies dans les termes d'une composition structurale dont le niveau de complexité soit arbitraire et dont le contenu puisse être associé à la solution de différents problèmes linguistiques.

Alors que dans l'analyse classique des cooccurrences, toutes les relations entre les mots dans une phrase sont aplaties à la relation de covoisinage, les analyseurs Déredec permettront, au moyen d'algorithmes d'indexation de symboles descripteurs, de privilégier certaines relations dans les phrases.

Le Déredec va permettre la programmation d'algorithmes où la notion de contexte peut se définir autrement que par des traits d'édition, ou par un simple décompte opéré sur la suite des mots du texte. En fait, la structure des expressions admissibles dans laquelle l'utilisateur sera amené à définir ses unités d'investigation est suffisamment sophistiquée pour traiter plusieurs des problèmes qui caractérisent la simulation des phénomènes de compréhension des langues naturelles: divers types de désambiguïsation syntaxico-sémantique, les simulations de dialogues et de processus cognitifs, la production de paraphrases, etc.

Les appellations "texte" et "description de texte" renvoient aux objets privilégiés d'analyse Déredec. On notera que le logiciel peut en fait s'appliquer à tous les objets ayant comme les textes la forme de séries d'événements et dont on imagine que les descripteurs

puissent avantageusement se structurer en arborescences. Retenons donc sous l'appellation "texte" l'idée très générale d'une concaténation d'éléments, ceux-ci pouvant être des mots dans des phrases, mais aussi des entités aussi diverses que des comportements dans une durée, des véhicules sur une route, des notes de musique sur une portée...

Le Déredec est un produit LISP. De l'intérieur du logiciel, l'usager a toujours accès aux fonctions de base de ce langage, à son éditeur et aux différentes structures de trace et de mise au point. Un usager averti peut ainsi augmenter à sa guise le Déredec de procédures particulières ou encore regrouper dans de nouvelles fonctions LISP des routines Déredec régulièrement appelées. Par ailleurs la connaissance de LISP n'est pas requise à l'utilisation stricte des fonctions Déredec.

Le Déredec est un système de programmation à trois volets. On y trouve:

a) Une structure de représentation (stockage) des données

Les "objets textuels" tels qu'on les identifie en début d'analyse ou tels qu'on les retrouve à la suite de manipulations linguistiques, possèdent toujours la même structure de formation. Celle-ci se nomme EXFAD pour EXpressions de Forme Admissible aux Descriptions. On appellera DDT (Description de Texte) toute suite d'EXFAD.

Tous les objets textuels (mots, phrases, suite de phrases, textes, etc.) dès qu'ils sont déclarés admissibles au Déredec, jusqu'à la sortie des programmes de l'usager ayant permis de les transformer en structures descriptives, sont toujours et uniquement des EXFAD. Une EXFAD peut donc être un objet très simple tel un mot, mais aussi un objet plus complexe tel un réseau sémantique ou une phrase munie de sa structure syntaxique.

C'est cette caractéristique d'unicité dans la formule de représentation des données qui rend possible en Déredec la programmation sans interfaces entre elles de machines consacrées à l'obtention de structures linguistiques variées: morphologique, sémantique, syntaxique, pragmatique...

Des Dictionnaires (de catégories ou de réseaux sémantiques) et des Lexiques (listes fréquentielles diverses) servent aussi en Déredec à stocker l'information.

b) Une batterie de fonctions pour la manipulation des données

- Des fonctions descriptives ou fonctions de constructions des EXFAD

Les suites d'EXFAD (ou DDT) sont obtenues par la programmation (par l'usager) d'automates à états finis spéciaux (subséquemment appelés automates Déredec). Ces automates analysent par un balayage contextuel les séquences d'EXFAD déposées sur un fichier d'entrée, et construisent sur celles-ci de nouvelles EXFAD descriptives des régularités linguistiques observées.

- Des fonctions exploratrices

Les DDT produites par les automates seront analysées par des fonctions exploratrices dont les arguments (appelés modèles d'exploration), fournis par l'usager, sont des structures de "pattern matching" ayant une syntaxe d'écriture simple et un pouvoir de discernement élevé. C'est par le biais des fonctions exploratrices que les grammaires descriptives de texte seront associées à des objectifs d'analyse de contenu.

c) Une structure de communication entre les volets a et b

Des fonctions Déredec transforment de différentes façons les structures de données en procédures de manipulation et certaines procédures de manipulation en structures de données. En fait, certaines entités Déredec n'ont pas de vocation définitive et peuvent, dépendant du point de vue où elles sont observées être tantôt considérées comme des données, tantôt comme des programmes.

Dans un scénario simple d'utilisation du logiciel, les fonctions du volet c ne seraient pas appliquées.

Imaginons un chercheur intéressé à comparer pour cinq textes différents tous les adjectifs situés dans les phrases en position de déterminants nominaux (la grosse pomme, les grands arbres, la table noire, etc.). Dans un premier temps il entrera ses textes sous autant de fichiers, à l'aide d'un simple éditeur de texte. Puis il programmera sa Grammaire de Texte (GDT). On appelle ainsi un jeu d'automates à états finis Déredec susceptibles de plaquer sur les séquences d'entrée les structures descriptives désirées. Les automates Déredec sont programmés par l'usager pour effectuer des balayages contextuels sur les éléments

des séquences et y laisser différentes traces susceptibles d'être explorées par la suite.

Une fois sa grammaire programmée, l'usager se trouve habituellement aux prises avec les différentes difficultés d'application de celle-ci sur son corpus.

Ici le logiciel lui porte assistance de plusieurs façons: d'abord il dépiste et diagnostique pour lui, et de façon automatique, des erreurs de programmation, il permet aussi de suivre à la trace le comportement d'un automate, de même il autorise des interruptions dans le travail des automates afin de questionner l'état de la description, de modifier la grammaire...

A la fin de ces séances de mise au point, l'usager aura obtenu ses DDT. Dans notre exemple, cela signifie que toutes les relations de déterminations adjectifs-nominaux auront été plaquées sur les séquences d'entrée des cinq textes.

La prochaine étape du projet porte normalement sur l'exploration de ces résultats. L'usager s'adonne alors à la programmation de modèles d'exploration. Celui de notre exemple pourrait par un modèle d'exploration rassembler tous les nominaux déterminés par une classe d'adjectifs d'abord regroupés sous une même catégorie, etc.

Puis suivent les comparaisons intertextuelles. Sur les différents fichiers contenant ces résultats d'exploration, notre usager commandera l'exécution de comparaisons diverses par l'entremise de fonctions d'analyse, afin d'étudier les différences de comportement de ses cinq textes eu égard à cette relation de détermination adjectivale.

Les séances de programmation en Déredec auront ainsi habituellement l'aspect d'un enchaînement de constructions d'automates, d'obtentions de DDT par l'application de ces automates sur le corpus, puis d'élaborations de modèles d'exploration, d'analyses des résultats éventuellement suivies de réexplorations ou de reconstructions de nouvelles descriptions. Programmer en Déredec signifie essentiellement programmer la production de DDT, puis programmer l'exploration de ces dernières, analyser les résultats et recommencer à l'une ou l'autre des étapes jusqu'à ce que ces résultats soient satisfaisants. L'ensemble de la démarche est hautement facilité par le fait que les expressions admissibles à l'entrée comme à la sortie, à la fois des fonctions descriptives et des fonctions ex-

ploratrices, ont, du point de vue informatique, exactement la même syntaxe d'écriture.

On peut penser que ce type d'expérimentation sera le lot de la majorité des usagers du logiciel. Mais l'usager très intéressé voudra sûrement goûter aux procédures PASC du Déredec, ces procédures de Programmation Automatique Sensibles au Contexte.

Il s'agit de retirer des mains du programmeur la majorité des opérations à effectuer au cours d'une séance de programmation, pour ne lui laisser que certaines décisions de haut niveau relatives à la planification générale des expériences.

Par exemple, certaines procédures PASC concernent la réapplication en chaîne de fonctions exploratrices sur une DDT; les résultats obtenus à chaque exploration servent à modifier le ou les modèles d'exploration utilisés, modèles dont une première version est donnée au point de départ par l'usager. Ce dernier contrôle la procédure PASC en fournissant certaines clés, certains paramètres qui guideront l'ensemble des opérations.

D'autres procédures PASC permettent d'enrichir progressivement une DDT en modifiant constamment un apparatus descriptif dont la structure générale est fournie au début, accompagnée là aussi de paramètres généraux sur la conduite du processus récursif.

On notera que les procédures de programmation automatique ont du point de vue de leur syntaxe informatique la même forme ou la même admissibilité que les autres fonctions ou opérations Déredec, de telle sorte qu'elles sont composables avec ces dernières. C'est cette caractéristique qui rend compte de l'étiquette "sensibles au contexte" dans l'appellation PASC: les procédures de programmation automatique sont logées dans des environnements susceptibles de fournir les paramètres pertinents à leur propre exécution.

La machinerie PASC qui de loin est la plus complexe et en quelque sorte la plus complète a pour nom APLEC.

APLEC (APprenti-LECteur) associe de façon automatique un module de questions/réponses pour toute GDT soumise par un usager, module où les questions sont formulées dans la langue naturelle du texte et où les réponses sont des segments dépistés dans celui-ci.

Lorsque APLEC ne peut pas dépister de réponse, il diagnostique la difficulté, intervient auprès de l'usager, s'enquiert d'une solu-

tion et tente de la généraliser pour tout le corpus, ceci, afin d'augmenter son pouvoir de fouille ultérieur et d'éviter le plus possible les interruptions.

En plus de la grammaire et de quelques modèles d'exploration (qu'il considère comme représentatifs des structures indexées par celle-ci), l'usager ne fournit à APLEC que des paramètres très généraux sur les conditions de fouille et les conditions d'apprentissage.

APLEC permet de distinguer formellement ce qui dans une entreprise sémiotique relève de la sémantique d'un groupe de textes, de la sémantique d'un texte particulier ou encore de celle d'un usage particulier d'un texte donné. Il permet aussi de délimiter les bornes sémantiques d'une GDT donnée et de faciliter par là son amélioration.

Les langues naturelles sont des objets, qui de façon évidente, résistent fortement aux techniques connues de formalisation.

On doit s'habituer à l'idée d'un système dont les règles changent selon évidemment la portion observée, mais aussi selon l'utilisation qu'on en fait. On peut toujours imaginer une sémantique particulière, fonctionnelle pour un univers langagier donné; plusieurs robots ont démontré qu'il est possible de simuler le fonctionnement d'une langue naturelle pour un univers sémantique restreint. Ces expériences sont intéressantes à titre illustratif, mais elles manquent l'essentiel de ce qui caractérise le comportement normal d'un locuteur: sa capacité de passer d'une sémantique à une autre en adaptant, voire en transformant au besoin les batteries de règles déjà édifiées. Nous devons disposer d'un logiciel qui facilite constamment la construction de nouveaux ensembles de primitifs, de nouveaux axiomes et de nouvelles règles d'inférence, ces éléments étant des variables des procédures programmées et non des constantes.

3. LES DONNEES ET LES ALGORITHMES

Trois objets Déredec retiendront constamment notre attention: les EXFAD, principales structures de rétention des données; les automates, ces machines programmables qui permettent de construire et de modifier les EXFAD et enfin les modèles d'exploration, algorithmes programmés par l'usager pour la fouille et l'extraction de sous-ensembles d'EXFAD. Programmer en Déredec consistera essentiellement à faire interagir ces objets entre eux à l'aide des fonctions Déredec.

3.1 Les EXFAD

Les EXFAD sont des structures arborescentes. Elles se construisent selon un schéma récursif simple représenté dans le tableau ci-bas.

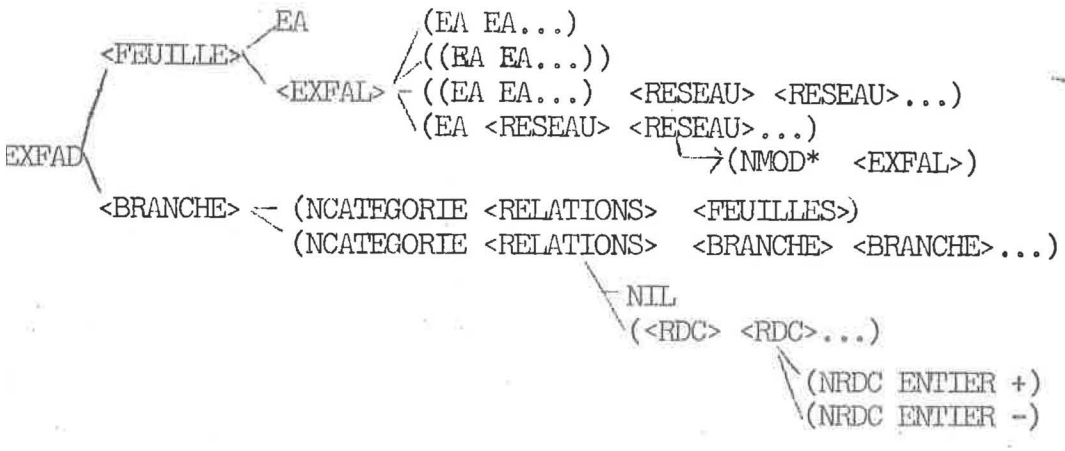
Il faut remarquer que l'usager est rarement appelé à construire lui-même ses EXFAD. Il attellera habituellement des automates Déredec à cette tâche.

Toutefois il doit quand même maîtriser la formule de structuration des EXFAD pour être habilité à bien programmer leur construction par les automates et leur exploration par les modèles.

Dans le tableau, des crochets entourent les concepts descriptifs des structures EXFAD. Certains de ces concepts sont récursifs, cela signifie que l'on devra "passer" plus d'une fois à ces endroits avant d'obtenir l'EXFAD. Dans la structure du tableau, les embranchements signalent des alternatives dans le mécanisme d'écriture.

A l'encontre de ces composants descriptifs, les items du tableau n'ayant pas de crochets peuvent se retrouver présents dans une EXFAD, il s'agit bien sûr des parenthèses ouvrantes et fermantes, des EA (les Expressions Atomiques), des NCATEGORIE (Noms de CATEGORIEs descriptives), des NRDC (Noms des Relations de Dépendance Contextuelle), des NMOD* (Noms de MODèles d'exploration) et des ENTIERS (chiffres).

Lorsqu'un élément est doublé et suivi de trois points, c'est qu'il peut à cet endroit être répété à volonté. Ainsi (EA EA...) signifie l'admissibilité à cet emplacement de (EA) mais aussi de (EA EA EA EA), etc.



On appelle Expression Atomique (EA) le niveau le plus simple d'une EXFAD. Une EA est une EXFAD et une EXFAD contient toujours une EA.

Tous les cheminements dans le tableau se terminent toujours par une EA. Les EA sont des chaînes de caractères mises entre guillemets. Exemple d'une DDT (suite d'EXFAD) à ce niveau primitif de construction:

"Il" "a" "fait" "une" "bonne" "recette" ".."

On ne peut imaginer d'EXFAD qui ne soit construite sur une EA. Une EXFAD pourra contenir plus d'une EA. On aura alors affaire soit à une EXFAL (EXpressions de Forme Atomique Liées) soit à une BRANCHE. L'usager utilisera habituellement les EXFAL pour stocker de l'information paradigmatique sur une EA (informations sémantiques ou morphologiques par exemple), alors qu'il utilisera les BRANCHEs pour décrire les relations syntagmatiques (syntaxiques, logiques...) qu'ont entre elles les EA d'une séquence.

Exemples d'EXFAD se ramenant à une feuille:

a) "carte"; pour vérifier le caractère bien formé de cette EXFAD, suivre le cheminement suivant dans le tableau:

<EXFAD> --> <FEUILLE"> --> EA

b) ("jambon" "viande" "porc") cheminement:

<EXFAD> --> <FEUILLES --> <EXFAL> --> (EA EA EA)

c)

```
("anticonceptionnel" (RADICAL*  ("conception"))
                     (PREFIXE*  ("anti"))
                     (PLURIEL*  ("s"))
                     (FEMININ*  ("le")))
```

cheminement:

<EXFAD> --> <EXFAL> --> (EA <RESEAU><RESEAU><RESEAU><RESEAU>)

et, pour chacun de ces quatre réseaux:

(NMOD* <EXFAL>) --> (NMOD* (EA))

...par exemple (RADICAL* ("conception"))

d)

```
("Jean" (DEF* ("agent" (TYPE ("humain"))
        (RELATION*
          ("amant"
            (QUI* ("Marie"))
            (OU* ("exemples"
                   (TYPE* (("linguistique"
                            "Chomsky"
                            "générative")))))))
```

b) (ACTION NIL ("manger" (FONCTION* ("survivre"))))

Dans le premier cas la FEUILLE se ramène à une EA et dans le deuxième cas à une EXFAL. NOM et ACTION sont les noms des catégories.

Tout comme c'est le cas pour les NMOD*, on doit distinguer le nom d'une catégorie de la valeur de cette catégorie. On verra plus bas l'utilité de cette distinction.

Deux branches peuvent recevoir une ou plus d'une RELATION si elles sont dominées par une même NCATEGORIE.

Chaque Relation de Dépendance Contextuelle (RDC) contient trois éléments: son nom (NRDC), un nombre entier positif ou négatif qui indique la distance entre les deux BRANCHES reliées par la RDC et enfin un signe + ou - qui indique le sens de la relation. Le + marque la BRANCHE d'où part la relation et le - la BRANCHE qui la reçoit.

Les RDC servent à marquer les relations orientées entre les BRANCHES dans les EXFAD. Exemple d'une EXFAD où trois branches sont dominées par une seule catégorie:

```
(GN NIL (NOM ((RELAT 2 -)) "envie")
        (PREP NIL "de")
        (INFINITIF ((RELAT -2`+)) "partir"))
```

Le nom "envie" reçoit une RDC nommée RELAT de l'INFINITIF "partir". Le nom de la RDC est choisi par l'usager; comme pour les catégories et les modèles d'exploration, on distinguera le nom d'une RDC de sa valeur.

Dans cet exemple, le chiffre ENTIER 2 indique la distance entre les deux BRANCHES. Un ENTIER négatif ordonne un comptage vers le haut (ou la gauche dans l'ordre de la séquence), tandis qu'un ENTIER positif signifie un comptage vers le bas. C'est par ce moyen que l'on pourra dans les explorations retrouver le partenaire d'une RDC. Par ailleurs les signes - et + situés en troisième position de la RDC indiquent respectivement le receveur et l'émetteur de la relation.

Les RDC relient des BRANCHEs de même niveau dans une EXFAD. Une RDC ne peut "traverser" une BRANCHE. Les RDC instruisent des relations privilégiées qu'ont certaines BRANCHEs à l'intérieur d'un noeud commun; la distance séparant ces partenaires (c'est-à-dire le nombre de BRANCHEs déposées entre les deux) étant reléguée à un second plan en

importance. On appréciera à la fois dans la construction et dans l'exploration des DDT, la puissance des moyens combinés RDC et embranchements.

Autres exemples d'EXFAD:

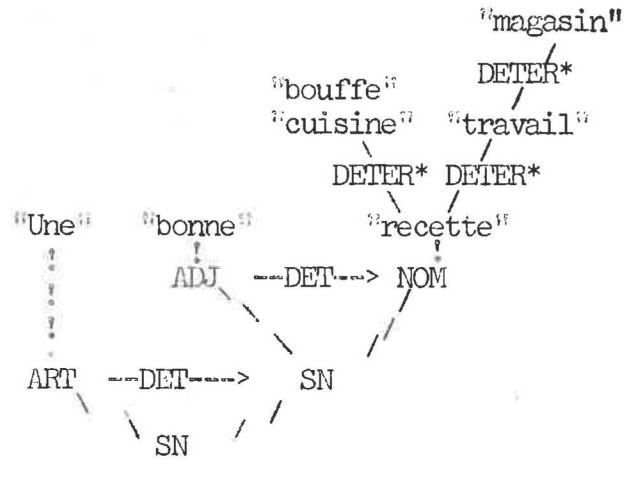
```
a) (SN NIL (SN ((DET 2 -))
               (ART ((DET 1 +)) "Les")
               (NOM ((DET -1 -)) "fondements"))
           (PREP NIL "de")
           (SN ((DET -2 +))
               (ART ((DET 1 +)) "sa")
               (NOM ((DET -1 -)) "métaphysique")))

b) (SN NIL (ART ((DET 1 +)) "Une")
           (SN ((DET -1 -))
               (ADJ ((DET 1 +)) "bonne")
               (NOM ((DET -1 -))
                ("recette" (DETER* (("cuisine" "bouffe")))
                           (DETER* ("travail"
                                       (DETER*
                                          ("magasin")))))))))
```

Dans ces exemples, SN, ART, NOM, ADJ, PREP sont des noms de catégories, et DETER* un nom de modèle d'exploration. DET symbolise une RDC, la relation de détermination.

On appellera couramment "catégorie dominante" d'une EXFAD, la catégorie d'ouverture de l'EXFAD (le SN le plus à gauche dans les deux derniers exemples).

De même, on appellera couramment "expressions atomiques dominantes" d'une EXFAD, toutes les expressions atomiques auxquelles ne sont pas attachées d'EXFAL. De plus, lorsqu'il y a EXFAL, les exepressions atomiques d'ouverture de ces EXFAL, dans l'exemple b), "Une", "bonne", et "recette" sont les expressions atomiques dominantes de cette EXFAD.

Voici la représentation graphique du dernier exemple



Pour que les EXFAD puissent être construites par les automates ou pour qu'elles puissent être fouillées par les modèles d'exploration, les catégories et les RDC devront recevoir une définition ou valeur. Les fonctions actives à l'intérieur des automates et des modèles d'exploration ne manipuleront jamais les noms de ces catégories et RDC mais uniquement leurs valeurs définies.

Les valeurs des catégories et des RDC ont la forme d'une liste ordonnée d'éléments. Ces derniers sont constitués d'un ou de plusieurs caractères. Ainsi la catégorie SN34S2 pourrait avoir différentes valeurs telles: (S N 3 4 S 2), ou (SN 3 4 S2), ou encore (S N 34 S 2), ou même (T RD 23) et ainsi de suite. Tous les tests d'identification d'une catégorie sont exécutés sur ces listes de valeurs. Dès qu'une liste plus petite se réalise dans une liste plus grande (à partir du début de la liste) la comparaison est positive. Ainsi par exemple les catégories suivantes et leurs définitions:

```
SN11 ...  (SN 1 1)
SN121 ... (SN 1 2 1)
SN122 ... (SN 1 2 2)
SN123 ... (SN 1 2 3)
SN2 ...   (SN 2)
SN ...    (SN)
SN1 ...   (SN 1)
SN12 ...  (SN 1 2)
```

Dans ces catégories, SN ramènerait toutes les valeurs, SN12 ramènerait SN12, SN121, SN122 et SN123. SN11 ne ramènera que SN11 ... ainsi de suite.

De plus, une position dans une définition peut être masquée par le caractère _ (le souligné); ainsi SN_2 (défini comme (SN _ 2) ramènera toutes les catégories commençant pas SN et ayant 2 comme troisième élément, quel que soit le contenu du deuxième élément.

Les RDC reçoivent leur valeur d'une façon absolument identique.

Cette façon générale de définir les catégories et les RDC permettra d'une part une très grande économie à la fois dans les batteries de catégories utilisées pour construire une grammaire et dans l'écriture des tests programmés dans les automates ou dans les modèles d'exploration. De plus la définition active d'une catégorie ou d'une RDC étant la dernière fournie (soit au terminal, soit de l'intérieur d'un programme) l'usager peut toujours changer ou faire changer (par ses programmes)

les valeurs de ses catégories ou RDC, ce qui lui procure une très grande flexibilité dans la mise au point de ses algorithmes.

Les NMOD* utilisés dans la construction des EXFAL doivent aussi recevoir une valeur. Ici on distinguera deux types de valeurs possibles, une valeur dite symbolique et une valeur effective. La valeur symbolique est toujours la suivante: (X); les modèles d'exploration à valeur symbolique ne sont utilisés dans les EXFAL qu'à titre de repères "symboliques" pour distinguer les différents arcs reliant les EA entre elles.

Par ailleurs les modèles d'exploration peuvent avoir une valeur effective algorithmique (présentée dans la prochaine section...) et leur insertion dans les EXFAL permettra de transformer alors ces dernières en procédures d'exploration et d'inférence.

3.2 Les modèles d'exploration

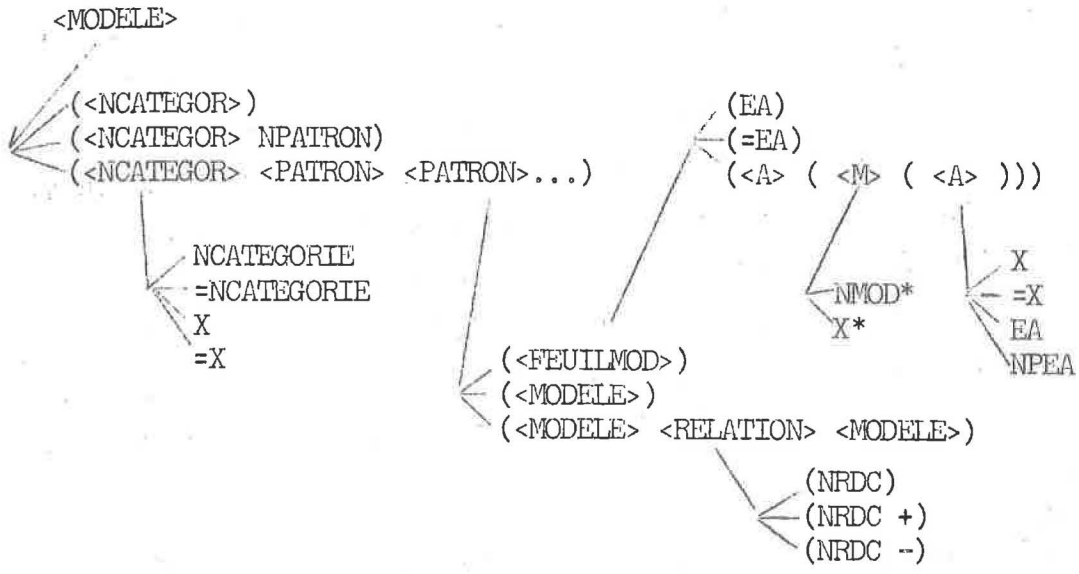
Toutes les EXFAD pourront être fouillées par ce que nous avons appelé les modèles d'exploration.

Ces modèles sont des patrons de fouille et de dépistage (pattern-matching) arbitrairement complexes qui ont pour fonction de rassembler sous des registres ou des fichiers des éléments terminaux ou non terminaux (des sections entières d'un arbre) des EXFAD.

La syntaxe des modèles d'exploration est très souple; elle permet de représenter élégamment les caractéristiques structurales des EXFAD et de dépister de façon très sélective les éléments de ces structures.

Tous les modèles d'exploration ramènent des EXFAD. C'est-à-dire qu'appliqué sur une EXFAD, un modèle ne peut rapporter qu'une sous-section de cette EXFAD qui soit elle-même une EXFAD.

Comme pour le schéma sur la construction des EXFAD, celui sur l'écriture des modèles contient des concepts descripteurs (mis entre crochets) et des éléments terminaux (sans crochets). Seuls ces derniers plus évidemment les parenthèses peuvent se retrouver dans des modèles programmés.



Les items NCATEGORIE (nom d'une catégorie), EA, NRDC (nom d'une RDC), NMOD* (nom d'un modèle d'exploration dans une EXFAL) renvoient directement aux éléments contenus dans une EXFAD. Ils doivent donc dans la construction d'un modèle être remplis par les items équivalents des EXFAD construites par les automates de l'usager.

X dans un modèle masque toute NCATEGORIE (d'une BRANCHE) ou toute EA (d'une FEUILLE). Les FEUILLEs sont spécifiquement explorées par la section FEUILMOD des modèles.

Le signe = lorsque composé avec X, avec une NCATEGORIE, ou avec le signe EA indique la section de l'EXFAD que l'on veut voir rapportée par le modèle d'exploration.

NPATRON (pour Nom d'une suite de PATRONs) et NPEA (pour Nom d'un Patron sur EA) sont des variables qui permettent de "passer" une suite de PATRONS (préalablement construite) au moment de l'évaluation du modèle.

Le contenu des variables occupant les positions NPATRON et NPEA est fourni automatiquement par l'évaluation de la fonction Déredec SOIT. Cette fonction transforme toute suite d'EXFAD en une liste de PATRONS.

Le système permet donc d'inclure à l'intérieur d'un modèle d'exploration des variables dont le contenu est fourni automatiquement au moment de l'évaluation du modèle.

Les RDC marquées dans les EXFAD peuvent être explorées par les modèles. Une RELATION est toujours programmée entre deux MODELES.

Elle est composée du nom de la RDC (NRDC) et accessoirement du signe + ou -. Si le signe + est programmé, le MODELE gauche renverra dans la fouille à la BRANCHE d'où part la RDC et le MODELE droit à celle qui reçoit la RDC. Le signe - renverse ces données et l'absence de signe signale l'indifférence quant au sens de la relation entre les deux BRANCHES. Par ailleurs, si l'un des MODELES doit contenir le signe =, il s'avère obligatoire de le mettre du côté droit de la RELATION.

Ainsi par exemple le modèle: (GP ((GN) (DET -) (=GN))) commande de rapporter tout GN déterminant un autre GN, le tout se passant dans un GP. L'on obtient ce modèle par le cheminement suivant dans le tableau:

```
<MODELE> -->
 (<NCATEGOR> <PATRON>) -->
   (NCATEGORIE (<MODELE> <RELATION> <MODELE>)) -->
          (GP ((<NCATEGOR>) (NRDC -) (<NCATEGOR>))) -->
             (GP ((<NCATEGORIE>) (DET -) (<NCATEGORIE>))) -->
                                  (GP ( (GN) (DET -) (=GN) ))
```

Le modèle cherchera une BRANCHE GP contenant à quelque part deux BRANCHEs GN reliées entre elles par une RDC DET. Quelque part signifie que le GP ne doit pas nécessairement dominer directement le niveau des GN.

Le modèle rapportera le GN d'où partait la relation (le déterminant).

Donnons d'autres exemples de modèles d'exploration "bien formés". Nous appliquerons ces modèles sur le dernier exemple d'EXFAD présenté à la section précédente (Une bonne recette).

a) Modèle: (=ADJ) ramène l'EXFAD:

   (ADJ ((DET 1 +)) "bonne")

b) Modèle: (ADJ ((=EA))) ramène l'EXFAD: "bonne"

c) Modèle: (ADJ ((=X))) ramènerait aussi l'EXFAD: "bonne"

d) Modèle: (SN ((SN ((=X))))) ramènerait les EXFAD:

   (ADJ ((DET 1 +)) "bonne") et

```
(NOM ((DET -1 -))
      ("recette" (DETER*
         (("cuisine" "bouffe")))
                  (DETER* ("travail" (DETER* ("magasin")))))))
```

Lorsqu'un modèle d'exploration est évalué (on appelle évaluation le processus de computation propre aux interprètes (comme LISP,

BASIC...)), le Déredec questionne le contenu de certaines variables générales. Selon les réponses trouvées, un même modèle d'exploration peut sur une même EXFAD rapporter des sous-EXFAD différentes.

L'usager peut modifier à volonté la valeur de ces variables. La dernière valeur donnée étant toujours active (le Déredec ne prend pas l'initiative de rechanger les valeurs...). Voyons brièvement le contenu de ces variables.

a) Les contraintes représentées par les patrons dans les suites de patrons peuvent être disjonctives, c'est-à-dire que la réalisation d'un seul patron de la suite suffira à évaluer positivement le modèle, ou conjonctives, tous les patrons devant être réalisés pour que le modèle soit évalué positivement. L'utilisateur indique son choix en donnant une valeur à la variable CONJONCTION: les contraintes seront conjointes si CONJONCTION est liée à T, et disjointes si cette variable a la valeur NIL. Exemple d'un modèle susceptible de dépister des EXFAD différentes selon la valeur accordée à CONJONCTION:

(=SN ((NOM)) ((ADJ))) Si CONJONCTION a la valeur T, le modèle rapportera toute EXFAD possédant deux BRANCHEs catégorisées NOM et ADJ. Si CONJONCTION a la valeur NIL, le modèle rapportera toute EXFAD possédant une BRANCHE NOM ou une BRANCHE ADJ.

b) Un même modèle peut se réaliser à plus d'un niveau dans une même EXFAD. Ainsi par exemple, le modèle:

(=SN (("chemin"))) et l'EXFAD:

```
(SN NIL (SN NIL (ART NIL "le") (NOM NIL "chemin"))
        (PREP NIL "de")
        (NOM NIL "terre"))
```

L'usager devra donner une valeur à la variable HAUTEUR, indiquant s'il préfère que le modèle dépiste l'EXFAD du plus haut niveau (HAUTEUR étant liée à NIL). Dans l'exemple, l'EXFAD du plus haut niveau est identique à l'ensemble de l'EXFAD, et celle du plus bas niveau à l'EXFAD emboîtée:

```
(SN NIL (ART NIL "le") (NOM NIL "chemin"))
```

c) Lorsqu'un modèle d'exploration contient quelque part une expression atomique, si EXPRES est lié à T, le modèle sera réalisé même si l'expression atomique se trouve enfouie dans une EXFAL. Si au con-

traire, EXPRES est lié à NIL, les tests s'arrêteront au niveau des expressions atomiques dominantes des EXFAD fouillées.

d) Si un modèle d'exploration commande une fouille à l'intérieur des EXFAL, par exemple: (X (((=X (DET* ("coco")))))), la liaison de la variable FOND à NIL signifiera le seul dépistage de l'expression atomique occupant la position =X; au contraire, la liaison préalable de FOND à T ordonnera le dépistage de toute la structure EXFAL dominée par =X.

e) Une autre variable permet de tronquer les fouilles effectuées sur les expressions atomiques. Lorsque LEMMA est liée à T, toutes les expressions atomiques qui contiennent minimalement l'expression argumentée dans le modèle seront dépistées. Ainsi (=X(("four"))) dépisterait "four" "fournaise" "fournir"...

De plus, il est alors possible (avec LEMMA = T) de masquer des caractères dans une expression atomique par l'utilisation du signe _ (souligné). Ainsi (=X (("p_r"))) dépisterait "par", "pire", "pure"...

f) Lorsque LEMMA est liée à T, l'indexation supplémentaire de la variable REVER à T (sa valeur initiale est NIL) permettra d'effectuer des fouilles tronquées à partir de la fin des expressions atomiques. Ainsi le modèle (=X (("ent"))) dépisterait alors toutes les expressions atomiques finissant par "ent".

g) Si dans une EXFAD le modèle d'exploration se réalise à plus d'un endroit, seule la première sous-EXFAD sera dépistée et envoyée sur le canal de SORTIE lorsque la variable PREMZ est liée à T, sa valeur initiale est NIL.

h) Lorsque dans une EXFAL, plus d'une expression atomique est alignée, par exemple: (N1 NIL ("attention" (DETER* (("chambre" "commune"))))), ces dernières sont considérées comme équivalentes par les procédures d'exploration.

Ainsi les modèles (=X (("chambre"))) et (=X (("commune"))) ramèneraient tous les deux l'EXFAD analysée. Si toutefois EQUISEUL est liée à T, seule la première expression atomique d'une suite (dans notre exemple: "chambre") sera dépistée. La valeur initiale de EQUISEUL est NIL.

i) Il est possible d'inverser complètement les résultats escomptés pour un modèle; il suffit pour cela de donner la valeur NIL à la variable POSITIF.

Seules les EXFAD de la DDT qui ne réalisent pas le modèle seront dépistées. Lorsque le Déredec est chargé en mémoire, POSITIF est à T.

j) Si la variable APLA est à T (autre valeur: NIL), le modèle ne rapportera que les expressions atomiques dominées par le noeud de l'EXFAD pointée par le signe =; toute la structure arborescente sous ce noeud se trouve "aplatie" à la suite des expressions atomiques.

3.3 Les automates Déredec

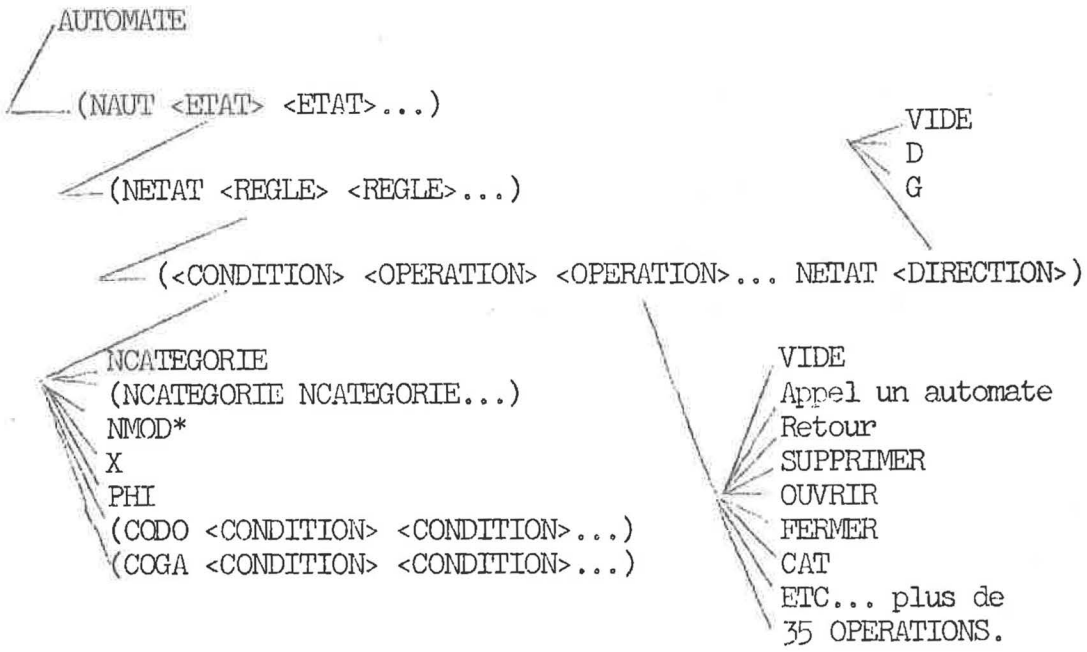
Les automates Déredec sont des machines que programme l'usager dans le but d'obtenir une description structurée de son texte. Ces machines balaient le contexte des EXFAD lues sur la séquence d'entrée et, à la suite de conditions portant sur le visionnement du contenu de ces EXFAD exécutent certaines opérations de construction de structures. Un premier groupe de ces opérations permet de catégoriser l'EXFAD pointée, de la relier à d'autres EXFAD de la séquence par des RDC, de rassembler différentes EXFAD d'une même séquence en BRANCHES, une BRANCHE devenant par la suite une seule EXFAD pouvant elle-même être catégorisée, reliée et composée dans une autre BRANCHE. Bref, des opérations qui permettent de composer avec un degré arbitraire de complexité des structures arborescentes sur les éléments de la séquence analysée.

D'autres opérations faciliteront l'indexation automatique d'EXFAL aux expressions atomiques, et enfin un dernier groupe d'opérations élargira à la grandeur du texte les possibilités d'analyse contextuelle.

Les automates Déredec sont des machines à états finis que l'on imagine munies d'un pointeur pouvant parcourir dans les deux directions une séquence ordonnée d'EXFAD et d'une unité de contrôle se déplaçant dans les différents états d'un réseau. A chacun de ces états se trouve associée une suite de règles, chacune ayant la forme d'un quadruplet condition/suite d'opérations/nom d'un état/direction. Lorsque, dans un état donné, une condition est satisfaite par l'EXFAD pointée sur la séquence, les opérations qui lui sont associées sont exécutées, l'uni-

té de contrôle se dirige vers l'état nommé et le pointeur se déplace éventuellement (il peut aussi rester sur place) d'une EXFAD sur la séquence, observant la direction indiquée par la règle.

Comme pour les EXFAD et pour les modèles d'exploration, un tableau nous servira à représenter les principes de construction des automates.



NAUT est le nom de l'automate, il s'agit d'un identificateur choisi par l'usager. NETAT est le nom d'un état; les NETAT sont composés de la lettre S suivie d'un chiffre; exemples: S1 S2 S3... S45. Le premier état d'un automate doit obligatoirement se nommer S1; pour les autres le choix du numéro est laissé à la discrétion de l'usager. Quoiqu'il soit recommandé de le faire, les états ne doivent pas nécessairement être en ordre numérique croissant.

Dans notre tableau, VIDE signifie qu'en l'endroit indiqué l'information n'est pas obligatoire (il peut ne pas y avoir d'OPERATION programmée à la suite d'une CONDITION; et il peut aussi ne pas y avoir de DIRECTION pour le pointeur à la fin d'une règle (auquel cas le pointeur reste sur l'EXFAD pointée).

Un automate est toujours appliqué sur une ou plus d'une EXFAD. La fonction DESCRIP chargée de l'application des automates lit un fi-

chier d'EXFAD séquence par séquence. Ces séquences sont découpées dans la DDT d'entrée à l'aide d'une variable nommée LIMITE dont la valeur est fournie par l'usager. Cette valeur peut être numérique ou catégorielle. Dans le premier cas, les séquences auront le nombre d'EXFAD égal à la valeur de LIMITE. Dans le deuxième cas les séquences comprendront toutes les EXFAD incluses entre deux réalisations d'une CATEGORIE, celle dont la valeur est donnée à LIMITE. Par exemple si des EXFAD d'une DDT ont reçu la CATEGORIE CONNEC, les séquences traitées par les automates (si LIMITE = CONNEC) incluront toutes les EXFAD d'un CONNEC à l'autre, le dernier CONNEC faisant partie de la séquence.

Toute séquence construite (par DESCRIP) contient comme première EXFAD une BRANCHE inventée et déposée à cet endroit comme repère gauche au balayage des EXFAD sur la séquence. Cette BRANCHE a pour NCATEGORIE: PHI. On veillera à ne pas inclure cette branche dans la construction d'EXFAD nouvelles sur la séquence. Il s'agit d'une EXFAD dégénérée ne possédant pas d'EA en feuille.

Lorsque l'automate est appelé (par DESCRIP) pour être appliqué sur une séquence le pointeur se trouve sur cette première EXFAD catégorisée PHI et l'unité de contrôle dans le premier ETAT de l'automate.

Les REGLES sont analysées dans leur ordre d'apparition dans l'ETAT.

Une REGLE est composée d'une CONDITION portant sur le contenu de l'EXFAD pointée.

La CONDITION sera réalisée si NCATEGORIE ou l'une d'une suite de NCATEGORIEs est la CATEGORIE dominante de l'EXFAD pointée. Ce test compare la valeur de la catégorie de l'EXFAD à la valeur de la catégorie dont le nom est argumenté en CONDITION. Une CONDITION peut aussi se ramener à l'application d'un modèle d'exploration sur l'EXFAD pointée. Dans ce cas la CONDITION est satisfaite si le modèle dépiste quelque chose sur cette EXFAD.

PHI comme on l'a vu est la catégorie de la première EXFAD de la séquence et X est une CONDITION par défaut. Comme les règles sont testées dans l'ordre, et que X est toujours une CONDITION réalisable, on mettra normalement une règle comprenant X comme CONDITION en fin de liste des règles d'un ETAT.

CODO et COGA permettent de programmer en quelque sorte des raccourcis dans l'examen des contextes droits ou gauches de l'EXFAD pointée.

Ainsi par exemple (CODO GN C22 V1 C1) sera considéré comme une CONDITION réalisée si les EXFAD suivant l'EXFAD pointée ont dans l'ordre les catégories dominantes GN, C22, V1 et C1; c'est-à-dire que l'EXFAD suivant immédiatement par la droite l'EXFAD pointée a la catégorie GN, la suivante encore à droite a la catégorie C22, etc.

COGA fonctionne de la même façon mais par la gauche. Son premier argument concerne l'EXFAD immédiatement à gauche, son deuxième argument celle à gauche de cette dernière, etc.

Toute condition peut être programmée comme argument à CODO ou COGA: une catégorie, une liste de catégories ou un modèle d'exploration. Exemples:

```
(CODO GN (C22 C1 C32) GV1 MOD33*)
(COGA MOD01* (GN GV))
```

On doit noter par ailleurs que si CODO ou COGA rencontre la limite droite ou gauche (PHI) de la séquence, la condition (CODO ou COGA) ne se trouve pas réalisée mais le système n'envoie pas de message d'erreur et ne met pas l'automate en mode d'arrêt non programmé.

Lorsque dans un ETAT une CONDITION est réalisée les opérations éventuelles (il peut ne pas y en avoir: VIDE) sont exécutées et l'ordre est donné à l'unité de contrôle de se diriger dans l'ETAT dont le nom apparaît à la suite des OPERATIONS et DIRECTION. Cette dernière information (DIRECTION) ordonne au pointeur le sens de sa promenade sur les EXFAD de la séquence.

Prenons en guise d'illustration du fonctionnement général des automates Déredec un problème simple de syntaxe: la formation d'un certain type de syntagmes nominaux. Il s'agirait d'obtenir la description de texte (b), à partir de la description de texte (a):

a)

```
(PRON NIL "il")
(VER NIL "habite")
(ART NIL "la")
(ADV NIL "très")
(ADJ NIL "grande")
(NOM NIL "maison")
```

b)

```
(ART NIL "il")
(VER NIL "habite")
(SN NIL
 (ART ((DET 1 +)) "la")
 (SN ((DET -1 -)) (SADJ ((DET 1 +)) (ADV ((DET 1 +)) "très")
                                    (ADJ ((DET -1 -)) "grande"))
                  (NOM ((DET -1 -)) "maison")))
```

Deux automates AUTO1 et AUTO2 permettront la construction de la nouvelle DDT. On suppose comme le montre la description en (a), qu'au moment de l'appel à AUTO1, chaque élément de la séquence d'entrée a reçu une catégorie descriptive.

```
(AUTO1 (S1 (CONNEC (RETURN))
           (NOM S2 G)
           (X S1 D))
       (S2 (ADJ (E AUTO2 S1 G)
                (R DET NOM D)
                (O G)
                S3 D)
           (ART (R DET (NOM SN) (O G) S3 D)
           (X S4 D))
       (S3 ((NOM SN) (F D SN) S2 G)
       (S4 ((NOM SN) S1 D)))

(AUTO2 (S1 ((ADV ADJ) (R DET (ADJ SADJ) D)
                      (O G) S2 D)
           (X S3 D))
       (S2 (ADJ (F D SADJ) S1 G))
       (S3 (X (RETURN))))
```

Lorsque AUTO1 est appelé, le pointeur est au début de la séquence et l'unité de contrôle est logée au premier état S1. Trois règles composent cet état. Elles ont pour effet de faire avancer le pointeur jusqu'au premier NOM rencontré sur la séquence, ainsi que de signaler la fin de l'exécution de AUTO1 à la rencontre de la catégorie CONNEC. Dans ces trois règles, la condition devant être réalisée est réduite au test le plus simple: celui de l'identité entre la catégorie dominante de l'expression pointée (EXFAD) sur la séquence et celle constitutive de la règle. X est une condition par défaut et se trouve toujours satisfaite; on doit aussi noter qu'une condition peut par ailleurs être beaucoup plus complexe et se trouver associée à la réalisation d'un modèle d'exploration sur l'EXFAD pointée.

Ainsi donc, dans S1, au point de départ, le pointeur est sur la première EXFAD de la séquence; cette EXFAD n'est ni un CONNEC (connecteur), ni un NOM, il s'agit donc d'un X; la troisième règle est appliquée. Ici aucune opération n'est programmée, et l'unité de contrôle ne fait que poursuivre sa course en S1, ordonnant au pointeur, à chaque boucle de se déplacer vers la droite d'une EXFAD. Lorsqu'un NOM est rencontré, la deuxième règle est exécutée: ici non plus aucune opération n'avait été programmée, mais l'unité de contrôle poursuit cette fois sa course en S2, le pointeur se déplaçant d'une EXFAD vers la gauche.

En l'état S2 trois règles sont programmées. La première signale que si l'EXFAD pointée est un ADJ, trois opérations devront être appliquées. La première ordonne l'exécution de l'automate AUTO2 (E AUTO2 S1 G), en spécifiant que l'unité de contrôle devra se diriger en l'état S1 et que le pointeur devra se déplacer d'une EXFAD vers la gauche. AUTO2 est chargé de la construction des syntagmes adjectivaux (SADJ). Cette tâche est confiée à un automate autonome puisqu'on peut penser dans la planification plus générale de cette grammaire, qu'il soit nécessaire de l'exécuter à partir d'un autre environnement que celui de la construction des SN (par exemple la construction des SV: il est très grand). Il n'y a pas de limite (à part celle de la mémoire de l'ordinateur) à l'auto-emboîtement et à la récursion des automates entre eux (AUTO2 pourrait lui-même appeler d'autres automates...); les opérations d'appel et de retour peuvent se loger n'importe où en position d'opération à l'intérieur des automates.

En S1 de AUTO2, si l'EXFAD pointée est un ADJ ou un ADV, deux opérations seront exécutées; la première (R DET (ADJ SADJ) D) ordonne de lier l'EXFAD pointée au premier ADJ ou SADJ rencontré par la droite sur la séquence, par une relation DET (pour symboliser détermination; il s'agit évidemment d'un identificateur choisi par l'usager); la deuxième opération ordonne d'ouvrir une parenthèse au côté gauche de l'EXFAD pointée. A la suite de ces opérations, l'unité de contrôle se dirige en S2, le pointeur se déplaçant vers la droite. L'adjectif est alors de nouveau rencontré, le syntagme est fermé à sa droite et il est nommé SADJ. L'unité de contrôle retournera en S1, lisant vers la gauche pour permettre le traitement récursif de structures du type: la très grande... ou la belle grande... etc. Finalement le pointeur se repositionnera sur

sur SADJ ou ADJ et le contrôle repassera à l'automate appelant (en S2 de AUTO1), où seront exécutées les deux opérations restantes. La première de celles-ci permettra de relier l'ADJ ou le SADJ nouvellement formé au NOM; la seconde d'ouvrir un syntagme qui se trouvera refermé et nommé en SN en S3. Le traitement se poursuivra par la gauche permettant d'inclure les articles dans les syntagmes nominaux du type: la maison... la belle maison... la très belle maison... la très belle grande maison... L'exemple laisse de côté de nombreux problèmes syntaxiques (par exemple la conjonction), sa présentation n'avait pour but que l'illustration du fonctionnement général des automates et de quelques opérations primitives offertes par le logiciel. Pour avoir une vue plus exhaustive, examinons le résumé suivant de l'ensemble des opérations programmables dans les automates.

Dans une règle, les opérations pouvant être exécutées à la suite de la réalisation d'une condition se rapportent à des procédures:

a) de catégorisation des EXFAD; lorsque l'EXFAD pointée est réduite à une expression atomique, le système permet de projeter la catégorie proposée pour toutes les expressions atomiques de la suite du texte qui ont la même composition (pour tous les "tokens" du même "type"); opérations: CAT, SUBCAT et PROCAT;

b) de regroupement syntagmatique opéré sur les EXFAD d'une même séquence; opérations: O et F; la procédure permet d'ouvrir ou de fermer des parenthèses arbitrairement à gauche ou à droite de l'expression pointée;

c) de dépistage de RDC entre des EXFAD d'une même séquence; opération R;

d) d'indexation de catégories à la suite de fouilles spéciales (par exemple la combinaison d'une fouille réalisée et d'une autre non réalisée) sur l'EXFAD pointée; opérations FOUDEC1 et FOUDEC2;

e) de modification de la structure environnante des expressions atomiques de l'EXFAD pointée, soit en associant des EXFAL à ces dernières, soit en les catégorisant par la base (c'est-à-dire malgré l'existence de structures syntagmatiques déjà composées); opérations LEXIDEC, TRANEXA et TRAXINTER;

f) de fouilles arbitrairement complexes sur les EXFAD pointées; opération FOUIL;

g) de suppression d'une EXFAD de la séquence; opération SUPPRI= MER; ou de rabaissement général des structures; opération DEFAIRE; aussi d'ajout d'expressions; opération AJOUTER;

h) de transformation automatique d'une EXFAD ou d'une suite d'EXFAD (résultat d'une fouille ou d'une description de texte) en un argument pour une fouille à venir; opération SOIT;

i) de projection automatique des contraintes exprimées dans les EXFAL (par les modèles d'exploration); opération EXPOZ;

j) de transition entre les automates; à cet égard la procédure est très souple, elle permet de suspendre le travail d'un automate et de faire exécuter un ou une série d'automates auxiliaires; on doit noter que les opérations d'ALLER et de RETOUR peuvent s'effectuer dans n'importe quel état d'un automate. Lorsqu'une opération RETOUR est programmée, l'unité de contrôle vient se loger dans l'automate appelant, puis y exécute les opérations restantes. Si l'automate appelant est le premier automate de la série, l'analyse de la séquence est terminée;

k) de communication interactive automate/usager; l'utilisateur peut programmer des arrêts de la computation. Lors d'un tel arrêt, la procédure permet de visionner la séquence et l'EXFAD pointées, de connaître l'automate et l'état où se trouve l'unité de contrôle, de modifier des automates, ou des registres, et enfin de faire exécuter directement l'une ou l'autre des opérations ou fonctions Déredec; opérations ARRET, ARRET2, ECOUTE, VOIRP, VOIRD, ALLER, CONTINUER, SUSPENDRE. Des arrêts non programmés peuvent aussi survenir, l'automate ne pouvant compléter la description d'une séquence; la machine Déredec fournit alors un diagnostic de l'erreur (aucune condition n'est satisfaite dans un état, l'unité de contrôle boucle indéfiniment...) et une description de son contexte d'apparition;

l) d'analyse contextuelle hors-séquence; opérations PRODEC, PONDEC et ECART; la bande sur laquelle sont inscrites les EXFAD et sur laquelle se déplace le pointeur ne peut être, pour des questions de gestion mémorielle, de longueur indéfinie, et c'est séquence par séquence

qu'une description de texte sera construite. La limite de ces séquences est déterminée par l'utilisateur selon des critères numériques ou grammaticaux. A ce type séquentiel d'analyse contextuelle s'ajoute une possibilité d'analyse contextuelle élargie sur soit l'ensemble de la description de texte, soit le segment de la description de texte qui précède la séquence pointée (depuis le début de la DDT), soit le segment de la description qui suit la séquence pointée (jusqu'à la fin de la DDT), ou soit encore une description de texte totalement différente de celle où s'inscrit l'EXFAD pointée (un autre texte).

Les résultats de ces analyses hors-séquence s'indexeront de deux manières à l'EXFAD pointée. D'une première façon en associant une EXFAL, à une expression atomique de l'EXFAD; EXFAL composée automatiquement par une exploration du contexte élargi en question (opération LEXIDEC). D'une autre façon, des catégories descriptives seront composées et indexées à l'EXFAD pointée (opération PRODEC, ECART).

Dans ce dernier cas, lorsque l'analyse porte sur le segment de la description de texte qui suit la séquence courante, l'automate génère une catégorie descriptive de l'ATTENTE qu'exerce l'EXFAD pointée eu égard au phénomène que veut décrire la grammaire de l'utilisateur; et d'autre part, lorsque l'analyse porte sur le segment de la description qui précède la séquence courante, le traitement est associé à la RESOLUTION du phénomène étudié. Ces catégories peuvent être pondérées (opération PONDEC), et de plus, à la suite d'une mesure programmée (opération ECART) de l'écart entre l'ATTENTE et la RESOLUTION qu'exerce pour un phénomène donné une même EXFAD, l'automate indexera cette dernière de l'une des deux catégories PREMISSE ou DENOUEMENT selon que dans cette EXFAD l'ATTENTE prédomine sur la RESOLUTION (PREMISSE), ou que la RESOLUTION prédomine sur l'ATTENTE (DENOUEMENT).

Ainsi, de même que les analyses contextuelles séquentielles permettent de programmer des grammaires récursives de type phrastique, les analyses contextuelles élargies vont permettre de programmer des grammaires récursives d'un type nouveau, grammaires hautement sensibles au contexte, qu'il est permis de qualifier de proprement textuelles.

Cette possibilité d'analyser chaque événement du texte, en tenant compte de la structure narrative dans laquelle il se situe, c'est-à-

dire en marquant à chaque fois l'ATTENTE spécifique qu'il crée et la RESOLUTION qu'il exerce, offre à l'utilisateur l'occasion d'effectuer sur les descriptions de texte qu'il obtient des analyses de contenu où la cohésion textuelle n'est plus représentée comme une résultante unique, à l'image d'un texte qui ne posséderait qu'un seul point de gravitation, mais comme une courbe où s'exprime la polarisation active et en quelque sorte changeante des forces d'attraction dans le texte.

On rappellera que l'usager peut en fait programmer toute fonction LISP en position d'opération Déredec, ce qui lui permet:

a) tout emmagasinage d'information dans des registres arbitrairement nommés, toute modification ou tout test sur le contenu de ces registres;

b) et règle générale, l'emboîtement des opérations dans des tests arbitrairement complexes.

Ainsi, pour nous résumer quelque peu, disons que les opérations Déredec ont trait à des procédures de construction, de modification et de dépistage d'éléments structuraux associés aux expressions de la séquence analysée. Ces opérations sont exécutées dans les états des automates à la suite de la réalisation de conditions portant elles-mêmes sur la composition de ces structures.

3.4 Les dictionnaires

Les dictionnaires Déredec sont de deux types: dictionnaires de catégories et dictionnaires d'EXFAL. Dans le premier type il s'agit d'une liste en ordre alphabétique de couples noms de catégories/expressions atomiques; ex:

```
(NOM "maison")
(VER2 "manger")
```

Les dictionnaires d'EXFAL sont simplement des listes (en ordre alphabétique) d'EXFAL;

```
ex:    ("maison" (MDR2* ("rue" (MOP* ("jamais")))
                        (MDR1* ("ville")))
       ("manger" (MOP* ("facilement")))
```

Lorsque les dictionnaires sont "parachutés" sur une DDT, les expressions atomiques de cette dernière pour lesquelles une entrée existe dans le dictionnaire reçoivent le contenu (catégorie ou EXFAL) associé à cette entrée.

Les dictionnaires peuvent être construits à la main, c'est-à-dire en interaction avec le logiciel ou de façon complètement automatique par l'application de modèles d'exploration sur des DDT.

3.5 Les lexiques

Les lexiques Déredec sont des listes de couples expression atomique/fréquence d'apparition. Les lexiques sont construits suite à l'application d'un modèle d'exploration susceptible de ramener d'une DDT les expressions atomiques qui satisfont un jeu de contraintes. La fréquence associée à chaque expression atomique du lexique indique le nombre de fois où cette dernière a été rapportée par le modèle d'exploration. Les lexiques se présentent en ordre alphabétique ou en ordre fréquentiel croissant.

Les lexiques Déredec servent à la présentation de résultats; ils sont aussi les fichiers d'entrée d'autres fonctions Déredec telles les fonctions d'élaboration de dictionnaires et les fonctions de comparaison de résultats. On obtient un dictionnaire à partir d'un lexique soit de façon manuelle interactive, soit de façon automatique par des fonctions de transformation.

3.6 Communications données-programmes

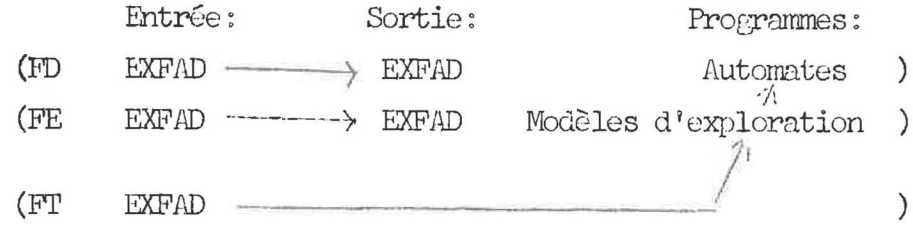
Le texte analysé (tel qu'il se présente à son entrée sur support magnétique), les descriptions de texte obtenues par la programmation des fonctions descriptives (qui commandent l'application des automates), de même que les sous-ensembles de descriptions de texte dépistés par les fonctions exploratrices (qui commandent l'application des modèles d'exploration), sont des entités qui, du point de vue informatique, possèdent la même syntaxe de représentation.

Cette structure d'information, commune à tous les objets textuels manipulés en Déredec, va permettre la plus haute composition entre les fonctions descriptives et exploratrices du système; les unes étant applicables sur les résultats des autres et vice-versa. Ici se trouve indi-

quée une première structure de communication entre des programmes Déredec.

Une seconde voie de communication vient des liens spéciaux tissés entre les EXFAD et les modèles d'exploration. D'une part, il s'agit de noter que toute EXFAD (qu'elle soit le produit d'une description ou d'une exploration) peut automatiquement et en contexte (c'est-à-dire sans connaître à l'avance son contenu) devenir une contrainte dans la constitution d'un modèle d'exploration; et d'autre part, que les symboles dont l'usager se servira pour nommer ses modèles d'exploration sont ceux-là mêmes qui parmi d'autres symboles peuvent servir à représenter les arcs des réseaux EXFAL constitutifs des EXFAD. Ainsi retenons que les EXFAD sont des structures à double rôle: elles sont d'abord des supports pour la rétention des descriptions textuelles, et elles ont de surcroît une fonction procédurale, puisqu'elles peuvent elles-mêmes devenir des arguments pour des fonctions exploratrices, ou plus directement des algorithmes pour la poursuite de séries d'explorations.

Un troisième élément de communication interne autorise au moment de la construction d'une description de texte, que la réalisation positive d'une fouille (une exploration) soit une condition (parmi d'autres) à la poursuite éventuelle de la description. C'est dire ainsi que les fonctions d'exploration peuvent être appelées de l'intérieur des fonctions descriptives et y jouer le rôle de contraintes. On notera donc le double statut de ces modèles d'exploration; comme élément entrant dans la composition des EXFAL, ils appartiennent aux structures descriptives indexées; comme arguments fournis aux fonctions exploratrices, ils feront partie de l'algorithme d'analyse. De surcroît, une opération du système: EXPOZ, permet même de passer automatiquement d'un statut à l'autre, c'est-à-dire de considérer les modèles d'exploration contenus dans une EXFAL comme autant de contraintes réelles pour un travail de fouille et d'indexation à venir.



Le tableau représente les relations exprimées par les trois caractéristiques formelles de communication présentées. On y souligne d'une part que les fonctions descriptives (FD) et les fonctions exploratrices (FE) prennent toujours à l'entrée et fournissent toujours à la sortie des EXFAD, c'est-à-dire des objets textuels ayant la même syntaxe, d'autre part qu'une fonction de traduction (FT) peut transformer toute EXFAD en une contrainte dans un modèle d'exploration, et enfin, qu'un tel modèle d'exploration peut lui-même agir comme contrainte dans l'exercice d'un automate Déredec.

Les relations de contrainte régissent la circulation formelle entre les objets textuels et les programmes; on notera que cette circulation est non déterministe en ce sens que les nouvelles EXFAD produites à la suite d'une traduction EXFAD-->programmes--> EXFAD ne conservent pas nécessairement la marque de cette traduction.

3.7 Les fonctions Déredec

Les fonctions Déredec sont les véhicules directs et premiers de la programmation en Déredec. Elles ont toutes une syntaxe d'écriture simple, directement commandée au clavier de l'ordinateur. Les fonctions Déredec permettent de définir les automates Déredec, les variables Déredec telles les catégories descriptives, les relations de dépendance contextuelles (RDC) et les modèles d'exploration ainsi que d'éventuelles nouvelles fonctions LISP. Elles facilitent aussi la gestion du contenu de la mémoire, les diverses relations entre celle-ci et l'espace-disque. C'est par l'application de fonctions Déredec que sont obtenues les DDT provenant du travail des automates ou de celui des modèles d'exploration, et tous les autres objets Déredec tels les dictionnaires, les lexiques et autres fichiers-sorties des fonctions d'analyse du système. Elles sont directement programmables au moniteur Déredec, ce qui les distinguent des opérations Déredec programmées de l'intérieur des automates.

Pour donner un bref aperçu des fonctions Déredec, mentionnons:

a) l'application des automates sur des DDT (fonctions: DESCRIP, SIMULECART, USAGER, TRADES, IMPRI);

b) la construction (automatique ou assistée) et la manipulation des lexiques et des dictionnaires (fonctions: TRIER, LEXIQUE,

LEXINDI, COLEXO, COLEXI, FONDREX, LEXIFORME, MULTILEX, TRAFIC, PREPARE, SURPROJEC, COMPTER, INSERCAT, INSEREXFAL);

c) l'exploration des DDT. Les fonctions exploratrices analysent le contenu des informations rassemblées dans les descriptions de texte; elles permettent d'observer la composition des structures dépistées, et de construire pour une description de texte donnée, divers profils distributionnels des structures descriptives indexées (fonctions exploratrices/ PROJEC, PROLEX, EXFAMOD, EXPAF, EXPA, PROFIL);

d) la traduction d'EXFAD en argument d'exploration (fonction SOIT);

e) des manipulations diverses sur les fichiers utilisés (fonctions: COPIER, TYPE, EMBELLIR).