Revue québécoise de linguistique

La structure des données et des algorithmes en Déredec

Pierre Plante

Volume 14, numéro 2, 1985

Linguistique et informatique

URI : https://id.erudit.org/iderudit/602541ar
DOI : https://doi.org/10.7202/602541ar

Aller au sommaire du numéro

Éditeur(s)

Université du Québec à Montréal

ISSN

0710-0167 (imprimé)
1705-4591 (numérique)

Découvrir la revue

Citer cet article

Plante, P. (1985). La structure des données et des algorithmes en Déredec. *Revue québécoise de linguistique*, *14*(2), 119–143. https://doi.org/10.7202/602541ar

LA STRUCTURE DES DONNÉES ET DES ALGORITHMES EN DÉREDEC*

Pierre Plante

1. Présentation du système Déredec

Le présent article est une introduction générale à la compréhension des structures de représentation des données et des programmes en Déredec. On y décrit les principaux principes de programmation propres à ce logiciel mais non la façon de mettre ceux-ci en action. Il ne faut donc pas chercher ici de réponses à des questions précises d'opérationalisation. Pour cela, il faudrait plutôt consulter le Manuel de Programmation. On peut aussi prendre note que la version micro-ordinateur du Déredec est munie d'un tutoriel interactif qui facilite l'apprentissage du système.

Le Déredec est un environnement de programmation permettant la simulation des modèles et la vérification des hypothèses linguistiques, l'analyse de contenu des textes ainsi que la mise au point de systèmes-experts en langage naturel.

Nous nous appliquerons ici à présenter les utilisations du système qui visent à satisfaire le premier objectif : simuler des modèles et/ou vérifier des hypothèses linguistiques.

Nous croyons que le Déredec est un environnement de programmation plus approprié aux questions d'analyse linguistique que ne le sont les langages de programmation utilisés en intelligence artificielle (tels PROLOG, LISP, LOGO...) qui, par leur généralité restent trop éloignés des objets spécifiques manipulés par les linguistes.

Le système permet de «stocker» de l'information aux différents niveaux caractéristiques du fonctionnement d'une langue : morphologie, syntaxe, sémantique, logique, pragmatique et structures de discours. Il

* Le présent document est en très grande partie composé d'extraits du Manuel de programmation du Déredec (Version 1.1, éditée par SAPIA, Montréal 1984).

permet aussi de manipuler ces niveaux, c'est-à-dire de les construire et de les explorer. Les constructions sont programmées en Déredec sous la forme d'«automates» et sont explorées avec des «modèles». Les machines Déredec chargées (par l'usager) de la construction des structures descriptives sont appelées «automates» parce qu'elles conservent de cette tradition computationnelle qui s'étend de Turing à Woods, l'idée d'un pointeur se promenant sur les données et d'une unité de contrôle qui se balade dans un circuit de règles et de conditions. La comparaison s'arrête ici puisque, contrairement à ces systèmes consacrés ou dédiés à une théorie linguistique précise, les automates Déredec sont des structures de programmation plus souples permettant, selon nous, la mise en algorithme de modèles ou théories linguistiques variées. Ces théories peuvent être axées sur différents principes de programmation des grammaires tels la présence ou l'absence de déterminisme (la présence ou l'absence de «retours en arrière»), l'ascendance ou la descendance ainsi que sur des principes algorithmiques originaux au Déredec telles les procédures PASC (Programmation Automatique Sensible au Contexte), les descriptions actives qui permettent les transferts automatiques de traits, une représentation originale des ambiguïtés, etc.

Le logiciel permet de multiples approches, et plusieurs expériences de programmation ont montré cette versatilité du système. Par exemple, une grammaire de surface du français (GDSF, Plante 1983) a été programmée en Déredec sur des principes ascendentistes et empiriques. Il s'agit d'une vaste ingénierie d'heuristiques (1800 règles) reposant sur l'indexation préalable d'une catégorie (partie du discours) à chaque lexème. Cette grammaire pratique une segmentation de la phrase en ses principaux constituants syntagmatiques. Elle réussit notamment, à ce niveau, à séparer les propositions entre elles et à dépister, pour toute proposition, le thème et le propos, des indications sur les compléments verbaux et plusieurs relations de détermination nominale. Cette grammaire est ascendante[1]; elle s'applique sur les mots catégorisés et tente, petit à petit, de composer des syntagmes, plutôt que de pratiquer le cheminement inverse qui partirait des syntagmes appréhendés et tenterait de rejoindre les items lexicaux par des dérivations successives. La GDSF pratique aussi une représentation originale des ambiguïtés syntaxiques; des relations orientées de dépendance

1. On a souvent «transféré», et c'est une erreur, les principes de programmation de cette grammaire GDSF sur les possibilités «plus générales» de programmation en Déredec.

contextuelle sont déposées dans la structure descriptive entre les mots ou syntagmes alternatifs. Nous préférons cette méthode à une autre qui fournirait à la sortie autant de structures qu'il y a d'interprétations à l'ambiguïté, bien que cette dernière méthode serait aussi programmable en Déredec. La GDSF n'utilise pas pour l'instant de composante morphologique. Lucie Dumas (du Centre d'Analyse de Textes par Ordinateur de l'Université du Québec à Montréal) développe, par ailleurs, en Déredec, un puissant analyseur morphologique du français qui lemmatise, caractérise en genre, nombre, personne, temps et mode, donne une catégorie du discours par analyse suffixale et regroupe les préfixes et suffixes par affinité sémantique. Une autre grammaire, programmée en Déredec, et en voie de parachèvement est principalement basée sur une analyse morphologique. Il s'agit de la Grammaire Déredec du Latin mise au point par Guy Allard et son équipe à l'Institut d'Études Médiévales de l'Université de Montréal. D'autres grammaires Déredec existantes usent quelque peu de morphologie, esquivent la syntaxe et plongent directement dans une description logico-sémantique. Ainsi en est-il des travaux d'Alain Lecomte (de l'Université de Grenoble 2) et de Catherine Péquégnat (du Centre de Sémiologie de Neuchâtel) qui programment des stratégies descriptives pour l'analyse de l'argumentation en langage naturel. De plus, les exercices de Jean-Marie Marandin et de Jacqueline Léon du Laboratoire Informatique et Sciences de l'Homme (CNRS Paris) mettent en valeur les possibilités de Déredec pour les grammaires inter-phrastiques. Plusieurs développements linguistiques récents exigent d'un logiciel des possibilités d'analyse extra et inter-phrastique (la levée des anaphores, diverses spécifications sémantiques...); le Déredec en offre plusieurs. Il est possible par exemple, de lever une ambiguïté syntaxique dans une phrase, en examinant un contexte global de type «structure de discours» dans lequel cette phrase se situe.

Le Déredec est donc un cadre computationnel général pour la programmation d'analyseurs ou simulateurs linguistiques basés sur des théories algorithmiques variées. Il offre aussi toutes sortes de facilités pour l'utilisation de ces analyseurs dans des structures d'application diverses telles les analyses de contenu des textes[2], les systèmes-experts en langage naturel utilisant les structures de raisonnement propres aux sémantiques

2. Ces analyses de contenu offrent des variations multiples selon les types de textes : textes à une signature, d'interviews ou de conversation à plusieurs, ou encore textes ayant des structures narratives complexes, etc.

naturelles, la mise au point des moteurs d'apprentissage visant l'amélioration automatique d'un algorithme de description, 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 nombre 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.

2. Les trois volets

Le Déredec est un système de programmation à trois volets.

2.1 *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, suites de phrases, textes, etc.) à partir du moment où ils sont déclarés admissibles au Déredec, jusqu'à la sortie des programmes de l'usager qui ont transformé ces objets 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.

Cette caractéristique d'unicité dans la formule de représentation des données rend possible la programmation de machines consacrées à

l'obtention de structures linguistiques variées (morphologiques, sémantiques, syntaxiques, pragmatiques,...) sans qu'il soit nécessaire d'avoir des interfaces entre ces diverses machines.

De plus, des Dictionnaires de catégories ou de réseaux sémantiques et des Lexiques (listes fréquentielles diverses) servent aussi à stocker de l'information.

2.2 *Une batterie de fonctions pour la manipulation des données*

2.2.1 Des fonctions descriptives ou fonctions de construction des EXFAD

Les suites d'EXFAD (ou DDT) sont obtenues par la programmation d'automates par l'usager. 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.

2.2.2. Des fonctions exploratrices

Les DDT ou suite d'EXFAD 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.

2.3 *Une structure de communication entre les volets 2.2.1 et 2.2.2*

Des fonctions transforment 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 considérées soit comme des données, soit comme des programmes.

3. Un premier scénario

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 appele ainsi un jeu d'automates Déredec susceptibles de plaquer, sur les séquences d'entrée, les structures descriptives désirées. Les automates 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 cette grammaire à son corpus. Ici le logiciel lui porte assistance de plusieurs façons : d'abord, il dépiste et diagnostique automatiquement pour l'usager, des erreurs de programmation; il permet aussi de suivre à la trace le comportement d'un automate; il autorise des interruptions dans le travail des automates afin de faire le point sur l'état de la description, de modifier la grammaire,... À 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.

L'étape suivante porte normalement sur l'exploration de ces résultats. L'usager s'adonne alors à la programmation de modèles d'exploration. Dans notre exemple, il pourrait, par un modèle d'exploration, rassembler tous les nominaux déterminés par un adjectif, ou, à l'aide d'un autre modèle, réunir tous les nominaux déterminés par une classe d'adjectifs d'abord regroupés sous une même catégorie, etc.

Notre usager peut alors procéder aux comparaisons inter-texutelles. Sur les différents fichiers contenant ces résultats d'exploration, il 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 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, en programmer l'exploration, analyser les résultats obtenus 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 facilitée par le fait que les expressions admissibles aux fonctions descriptives et aux fonctions exploratrices ont, tant à l'entrée qu'à la sortie, exactement la même syntaxe d'écriture du point de vue informatique.

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 alors 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. Une première version de ces modèles est donnée au point de départ par l'usager; il ne lui reste plus qu'à contrôler 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.

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, ce qui les rend composables avec ces dernières. C'est cette caractéristique qui est exprimée par 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 (APprenti-LECteur). APLEC associe de façon automatique un module de questions/réponses pour toute GDT soumise par un usager. Les questions du module sont formulées en langue naturelle et les réponses sont des segments dépistés dans le texte. Lorsque APLEC ne peut pas dépister de réponse, il diagnostique la difficulté, intervient auprès de l'usager, s'enquiert d'une solution et tente de la généraliser pour tout le corpus 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 représentant les structures indexées par la grammaire, l'usager ne fournit à APLEC que des paramètres très généraux sur les conditions de fouille et les conditions d'apprentissage.

Le module APLEC du Déredec est particulièrement bien adapté à la description du niveau «pragmatique» de fonctionnement des objets langagiers. Il permettra par exemple 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.

Les procédures PASC du Déredec (telle APLEC) autorisent, sur les données linguistiques, la mise au point de constructions structurantes dynamiques. 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 l'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.

4. Données et algorithmes

Trois objets Déredec retiendront 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.

4.1 *Les EXFAD*

Les EXFAD sont des structures arborescentes. Elles se construisent selon un schéma récursif simple représenté dans le tableau 1.

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 bien programmer leur construction par les automates et leur exploration par les modèles.

Dans le tableau 1, 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. Les embranchements signalent des alternatives dans le mécanisme d'écriture.

À l'encontre de ces composants descriptifs, les items du tableau 1 n'ayant pas de crochets peuvent se retrouver présents dans une EXFAD; il s'agit des parenthèses ouvrantes et fermantes, des EA (les Expressions Atomiques), des NCATEGORIE (Noms de CATÉGORIEs 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 être répété à volonté à cet endroit. Ainsi (EA EA...) signifie l'admissibilité, à cet emplacement, de (EA) mais aussi de (EA EA EA EA).

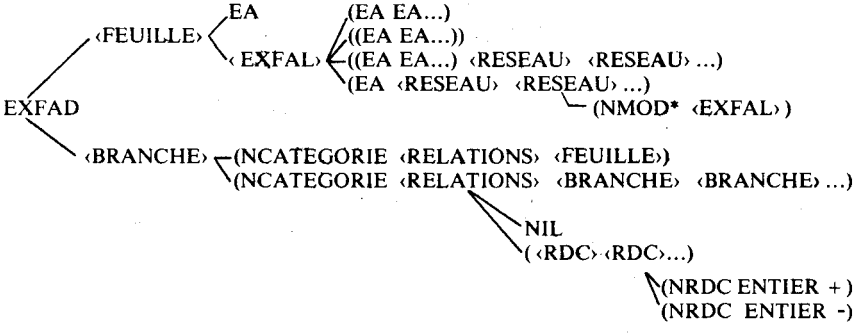


Tableau 1. Schéma de la construction des EXFAD.

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 comme le montre cet exemple d'une DDT (suite d'EXFAD) à ce niveau primitif de construction :

(1) «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.

Voici des exemples d'EXFAD se ramenant à une Feuille :

(2) «carte»
pour vérifier le caractère bien formé de cette EXFAD, suivre le cheminement suivant :
‹EXFAD› → ‹FEUILLE› → EA

(3) («jambon» «viande» «porc»)
cheminement :
‹EXFAD› → ‹FEUILLE› → ‹EXFAL› → (EA EA EA)

(4) («anticonceptionnel»
(RADICAL* («conception»))
(PREFIXE* («anti»))
(PLURIEL* («s»))
(FEMININ* («le»)))

cheminement :
‹EXFAD› → ‹EXFAL› →
(EA ‹RESEAU› ‹RESEAU› ‹RESEAU› ‹RESEAU›)

et, pour chacun de ces quatres réseaux :
(NMOD* ‹EXFAL›) → (NMOD* (EA))
... par exemple (RADICAL* («conception»))

(5) («Jean» (DEF* («agent» (TYPE* («humain»))))
(RELATION*
(«amant»
(QUI* («Marie»))
(OU* («exemples»
(TYPE* ((«linguistique»
«générative»))))))))

Dans une EXFAL, les noms des arcs qui relient entre elles les EA sont des noms de modèles d'exploration (NMOD* dans le tableau 1); ce sont des identificateurs choisis par l'usager, devant toujours se terminer par un *.

Les modèles d'exploration sont ces algorithmes Déredec programmés par l'usager pour parcourir des EXFAD et en dépister et extraire des sous-EXFAD. Ici, dans le cadre des EXFAL, ils ne servent qu'à nommer les arcs des réseaux. Ce double rôle des modèles d'exploration au sein du Déredec (algorithmes de fouille et appellations des arcs dans les EXFAL) n'est pas une erreur de conception. Lors d'une étape antérieure, les EXFAL pourront avoir été construites à l'aide de modèles d'exploration; l'utilisation du nom du modèle est alors une trace laissée au moment de cette construction automatique. Dans l'exemple (5), le fait que «Jean» soit un «agent» de TYPE* «humain» serait dépisté par l'exploration antérieure d'une DDT. C'est dans ce sens que l'on conseille d'utiliser les EXFAL pour stocker l'information paradigmatique d'une EA. Les EXFAL peuvent représenter l'«histoire» du comportement d'une EA dans une DDT. Les EXFAL pourront aussi être construites à la main (en interaction avec le Déredec) et déposées dans des dictionnaires susceptibles d'être projetés sur tout un texte.

Une EXFAL est donc une possibilité d'EXFAD. Elle peut de fait en contenir plus d'une, puisqu'une EXFAL contient des RESEAUX, eux-mêmes porteurs d'EXFAL. Les EXFAL sont des entités Déredec à la fois simples et complexes; simples par leur formule de construction et complexes par leur double rôle rétentif et procédural au sein du système.

Le niveau le plus simple d'une BRANCHE est celui d'une FEUILLE (EA ou EXFAL) catégorisée, n'ayant pas de RELATIONS (ce qui est exprimé par NIL); en voici des exemples :

(6) (NOM NIL «maison»)
cheminement :
‹EXFAD› →
(‹BRANCHE› ‹RELATIONS› ‹FEUILLE› →
(NCATEGORIE NIL EA) →
(NOM NIL «maison»)

(7) (ACTION NIL («manger» (FONCTION* («survivre»))))

La FEUILLE se ramène à une EA dans l'exemple (6) et à une EXFAL dans l'exemple (7). 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 plusieurs RELATIONS 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. Voici un exemple d'EXFAD où trois branches sont dominées par une seule catégorie :

(8) (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 l'exemple (8), le chiffre ENTIER 2 indique la distance entre les deux BRANCHES. Un ENTIER négatif ordonne un comptage vers le haut (vers 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 retrouver le partenaire d'une RDC dans les explorations. 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 illustrent les relations privilégiées qu'ont certaines BRANCHEs à l'intérieur d'un nœud commun; la distance séparant ces partenaires (c'est-à-dire le nombre de BRANCHEs déposées entre les deux) est alors reléguée au second plan. On appréciera la puissance des moyens combinés des RDC et des embranchements, tant dans la construction que dans l'exploration des DDT.

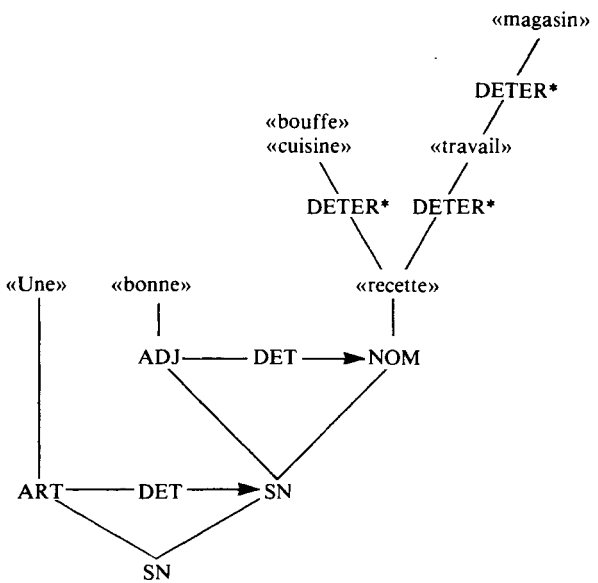
Considérons les exemples d'EXFAD suivants :

(9) (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»)))
(10) (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* est un nom de modèle d'exploration. DET symbolise une RDC, la relation de détermination.

Voici la représentation graphique de l'exemple (10) :



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 les 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. Voici des exemples de catégories suivies de leurs définitions :

(11) 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 ... (1N 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 par 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 catérogies et les RDC permettra une 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 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 de ses RDC; l'identité du mode de définition des valeurs des catégories et des RDC procurera à l'usager plus de flexibilité dans la mise au point de ses algorithmes.

4.2 *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. En d'autres termes, un modèle appliqué sur une EXFAD ne peut rapporter qu'une sous-section de cette EXFAD qui est 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.

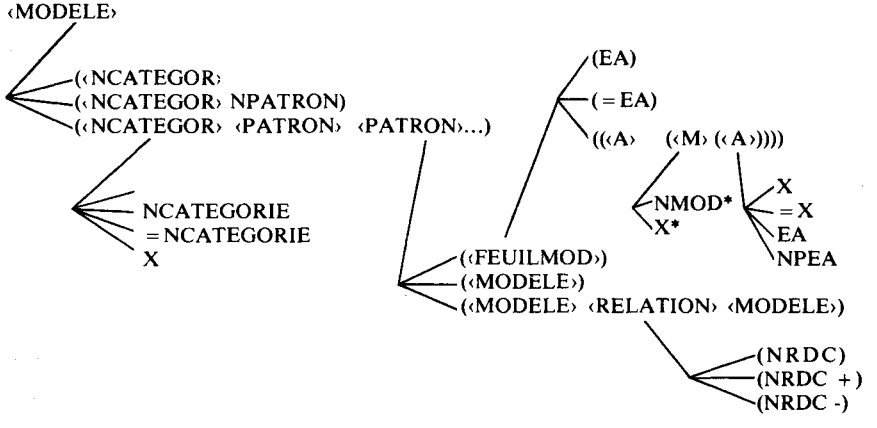


Tableau 2. Schéma de la construction des modèles d'exploration

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, lors de la construction d'un modèle, être remplis par les items équivalents des EXFAD construites par les automates de l'usager.

Le symbole X dans un modèle masque toute NCATEGORIE (d'une BRANCHE) ou toute EA (d'une FEUILLE). Les FEUILLEs des EXFAD 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 marque 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; on obtient ce modèle par le cheminement suivant :

(12) ‹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).

Lorsqu'un modèle d'exploration est évalué[3], le Déredec explore le contenu de certaines variables générales. Selon les résultats, un même modèle d'exploration peut, sur une même EXFAD, rapporter des sous-EXFAD différentes. Ces variables portent, par exemple, sur la hauteur des nœuds dépistés, ou sur le caractère conjoint ou disjoint des suites de PATRONs, ou encore sur l'aspect tronqué des expressions atomiques, etc.

4.3 *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 visonnement 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, toutes les opérations qui permettent de composer, avec un degré arbitraire de complexité, des structures arborescentes (des EXFAD) sur les éléments de la séquence analysée, peuvent être exécutées. D'autres opérations faciliteront l'indexation automatique d'EXFAL aux expressions atomiques. 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. À chacun de ces états se trouve associée une suite de règles. Chacune de ses règles a 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'unité de contrôle se dirige vers l'état nommé et, si désiré, le pointeur peut se déplacer d'une EXFAD sur la séquence en observant la direction indiquée par la règle.

3. On appelle «évaluation» le processus de computation propre aux interprètes comme LISP. BASIC, etc.

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

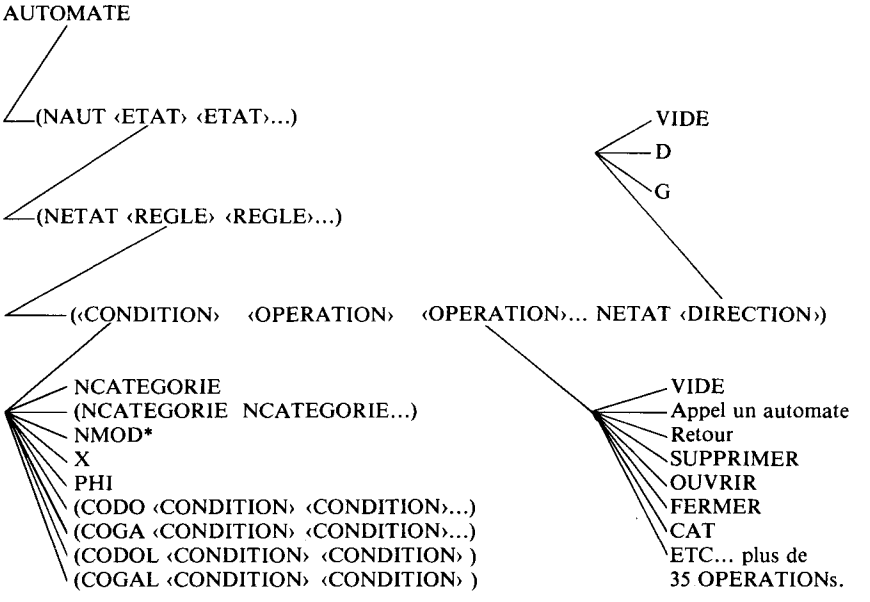


Tableau 3. Schéma de la construction des automates.

Dans le tableau 3, VIDE signifie que l'information n'est pas obligatoire à l'endroit indiqué; 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 (dans ce cas, le pointeur reste sur l'EXFAD pointée).

Un automate est toujours appliqué sur une ou plusieurs EXFAD. La fonction Déredec chargée de l'application des automates lit un fichier 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.

Les REGLES sont analysées dans leur ordre d'apparition dans l'ETAT. Une REGLE est composée d'une CONDITION, d'une série d'opérations qui seront exécutées si la condition est satisfaite, du nom du prochain ETAT où ira se loger l'unité de contrôle et enfin d'une DIRECTION (gauche, droite ou absente) pour indiquer le sens du déplacement du pointeur sur la séquence.

La CONDITION sera réalisée si NCATEGORIE, ou une NCATEGORIE dans 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.

Le symbole X est une CONDITION par défaut. Sa programmation permet de commander l'exécution d'opérations quel que soit le contenu de l'EXFAD pointée. Ce point est important puisqu'il «signe» l'autorisation de programmer des automates descendants en Déredec, permettant ainsi de suspendre ou de commencer l'observation de la surface qu'au moment jugé approprié dans l'algorithme.

CODO, COGA, CODOL et COGAL sont des CONDITIONS spéciales qui 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 catérogies ou un modèle d'exploration. CODOL et COGAL se programment dans la même position que CODO et COGA. CODOL sera évaluée positivement si, à la droite de l'EXFAD pointée, la première condition se réalise avant la seconde. Ainsi (CODOL NOM VERBE) sera positif si un NOM apparaît à la droite de

l'EXFAD pointée, avant un VERBE, quel que soit le nombre d'EXFAD séparant l'EXFAD pointée du NOM, ou le NOM du VERBE. COGAL fait le même travail pour la gauche de l'EXFAD pointée. Les arguments ‹CONDITION› de CODOL et COGAL ne sont assujetties à aucune restriction; il peut s'agir de catégories, de listes de catégories ou de modèles d'exploration[4].

Lorsqu'une CONDITION (y compris X, la CONDITION toujours vraie) est réalisée dans un ETAT, 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. La dernière information (DIRECTION) ordonne au pointeur le sens de sa promenade sur les EXFAD de la séquence. Ainsi, la séquence d'entrée peut être balayée dans les deux directions, ce qui n'empêche pas la programmation d'algorithmes uni-directionnels. Les CONDITIONS portent sur l'observation des éléments de la séquence, ce qui n'empêche pas la programmation d'algorithmes où une dérivation déterminée précède l'observation de la séquence.

Le Déredec permet un fonctionnement très «naturel» : les EXFAD de la séquence seront, au point de départ, les lexèmes des phrases analysées et, à la sortie de la grammaire, les structures descriptives des régularités linguistiques observées. Il n'y a donc qu'une seule structure rétentive et un seul pointeur. C'est par les OPERATIONS que sont programmées les constructions descriptives. Les automates se promènent sur la séquence, laissent différentes traces (des catégories, des regroupements syntagmatiques, des poussées d'EXFAL, des relations de dépendance contextuelle, etc.) que d'autres automates pourront explorer (avec des CONDITIONS) et modifier (sur-catégoriser, supprimer, abaisser, regrouper, etc.) avec des OPERATIONS.

Par ailleurs, certains algorithmes exigent la rétention de résultats intermédiaires dans des registres. Des OPERATIONS permettront de construire et de fouiller ces registres (toujours des EXFAD), de les insérer éventuellement dans la séquence ou de substituer ces registres à une séquence (ou à des éléments de celle-ci), etc. Il sera ainsi possible de traiter en parallèle plusieurs structures descriptives, de les empiler, puis de questionner cette pile dans des analyses de séquences utlérieures...

4. L'idée de «libérer» CODO et COGA en CODOL et COGAL est de Jean-Marie Marandin (de l'INALF, CNRS, France).

Par l'utilisation des registres ou de certaines OPERATIONS spéciales, l'analyse d'une séquence peut être contrainte à l'analyse des séquences antérieures ou postérieures. Tout est en place, dans Déredec, pour faciliter la programmation des algorithmes portant sur des questions comme la levée des anaphores ou la composition des structures narratives (l'avant et l'après des événements discursifs).

Pour la mise au point de ses grammaires, le Déredec offrira à l'usager toute une batterie de fonctions d'aide interactives, qui vérifient la syntaxe des programmes, fournissent de la documentation spécifique, diagnostiquent des causes, donnent des solutions, permettent de suivre à la trace le comportement d'un automate problématique ou de stopper la procédure en des endroits choisis afin d'observer l'état de la description, etc.

5. 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 composés du nom des catégories et des expressions atomiques, comme le montre l'exemple (13) :

(13) (NOM «maison»)
(VER2 «manger»)

Les dictionnaires d'EXFAL sont simplement des listes (en ordre alphabétique) d'EXFAL, tel qu'illustré par l'exemple (14) :

(14) («maison» (MDR2* («rue» (MOP* («quartier»)))
(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.

6. Les lexiques

Les lexiques Déredec sont des listes de couples structurés comme suit : expression atomique/fréquence d'apparition. Les lexiques sont construits

suite à l'application d'un modèle d'exploration susceptibles 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 de 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.

7. 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 description de texte dépistés par les fonctions exploratrices (qui commandent l'application des modèles d'exploration), sont des entités qui possèdent la même syntaxe de représentation du point de vue informatique.

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; ceci illustre 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. D'autre part, 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 elle-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 éléments entrant dans la composition des EXFAL, ils appartiennent aux structures descriptives indexées et comme arguments fournis aux fonctions exploratrices, ils feront partie de l'algorithme d'analyse. De surcroît, une opération du système 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.

	Entrée :	Sortie :	Programmes :
(FD	EXFAD	EXFAD	Automates)
(FE	EXFAD	EXFAD	Modèles d'exploration)
(FT	EXFAD		)

Tableau 4. Schéma des structures de communication

Le tableau représente les relations exprimées par les trois caractéristiques formelles de communication présentées. On y remarquera 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.

8. Les fonctions Déredec

Ces fonctions sont les véhicules de base, privilégiés pour 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 ces fonctions que sont obtenues les DDT provenant du travail des automates ou de celui des modèles d'exploration, ainsi que tous les autres objets Déredec tels les dictionnaires, les lexiques et autres fichiers-sorties des fonctions d'analyse du système.

Le Déredec est un logiciel actuellement opérationnel en NEW-UCI LISP (sur des machines DEC-10) et en IQ-LISP sur micro-ordinateur IBM-PC-XT-AT (et plusieurs compatibles). Un tutoriel interactif et un programme de documentation automatique est disponible sur cette dernière version. Une version pour VAX (Digital Equipment Corporation) en LISP NIL (M.I.T.) sera bientôt disponible.

Pierre Plante
Université du Québec à Montréal

Références

LECOMTE, A. (1985) *Des raisons sous les mots; approche algorithmique du raisonnement en langue naturelle*, GRAD, Université de Grenoble 11.

LECOMTE, A., J. Léon et J.M. Marandin (1984) «Analyse de discours : stratégies de description textuelle». *Mots* n° 8, Septembre, Paris.

PLANTE, P. (1979) *Le Déredec, logiciel pour le traitement linguistique et l'analyse de contenu des textes.* Thèse de Doctorat (Ph.D.), Université du Québec.

PLANTE, P. (1983) *GDSF, Une grammaire Déredec des structures de surface du français.* Service d'informatique de l'UQAM.

PLANTE, P. (1983) «Le système de programmation Déredec». *Mots*, n° 6, Paris.

PLANTE, P. (1985) *Déredec, logiciel de traitement linguistique, d'analyse de contenu des textes et de mise au point de systèmes-experts en langage naturel; Manuel de programmation. Version 1.1*, SAPIA, Montréal.