Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique : définition d'un système prototype

Christian Boitet

N° D'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir le grade de

DOCTEUR ES SCIENCES MATHEMATIQUES

par

CHRISTIAN **BOITET**

UN ESSAI DE REPONSE A QUELQUES QUESTIONS THEORIQUES ET PRATIQUES LIEES A LA TRADUCTION AUTOMATIQUE DEFINITION D'UN SYSTEME PROTOTYPE

Thèse soutenue le 16 avril 1976 devant la commission d'examen :

Monsieur	J. KUNTZMANN	Président
Messieurs	C. BENZAKEN	
	A. LENTIN	
	B. VAUQUOIS	Examinateurs
	G. WERNER	
	J.M. ZEMB	
	D. HERAULT	Invité

N° D'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir le grade de

DOCTEUR ES SCIENCES MATHEMATIQUES

par

CHRISTIAN **BOITET**

UN ESSAI DE REPONSE A QUELQUES QUESTIONS THEORIQUES ET PRATIQUES LIEES A LA TRADUCTION AUTOMATIQUE DEFINITION D'UN SYSTEME PROTOTYPE

Thèse soutenue le 16 avril 1976 devant la commission d'examen :

Monsieur	J. KUNTZMANN	Président
Messieurs	C. BENZAKEN	
	A. LENTIN	
	B. VAUQUOIS	Examinateurs
	G. WERNER	
	J.M. ZEMB	
	D. HERAULT	Invité

UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

Monsieur Michel SOUTIF : Président

Monsieur Gabriel CAU : Vice-Président

MEMBRES DU CORPS ENSEIGNANTS DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des Fluides
	ARNAUD Paul	Chimie
	AUBERT Guy	Physique
	AYANT Yves	Physique Approfondie
Mme	BARBIER Marie-Jeanne	Electrochimie
MM.	BARBIER Jean-Claude	Physique Expérimentale
	BARBIER Reynold	Géologie Appliquée
	BARJON Robert	Physique Nucléaire
	BARNOUD Fernand	Biosynthèse de la Cellulose
	BARRA Jean-René	Statistiques
	BARRIE Joseph	Clinique Chirurgicale
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie Chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (I.U.T. B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique Ophtalmologique
	BONNET-EYMARD Joseph	Clinique Gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie Générale
MM.	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique Rhumatologique et Hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie Végétale
	CARRAZ Gilbert	Biologie Animale et Pharmacodynamie
	CAU Gabriel	Médecine Légale et Toxicologie
	CAUQUIS Georges	Chimie Organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-Rhino-Laryngologique
	CHATEAU Robert	Clinique de Neurologie
	CHIBON Pierre	Biologie Animale
	COEUR André	Pharmacie Chimique et Chimie Analytique
	CONTAMIN Robert	Clinique Gynécologique
	COUDERC Pierre	Anatomie Pathologique
	CRAYA Antoine	Mécanique
Mme	DEBELMAS Anne-Marie	Matière Médicale
MM.	DEBELMAS Jacques	Géologie Générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumo-Phtisiologie
	DEPORTES Charles	Chimie Minérale
	DESRE Pierre	Métallurgie

MM.	DESSAUX Georges	Physiologie Animale
	DODU Jacques	Mécanique Appliquée (I.U.T. A)
	DOLIQUE Jean-Michel	Physique des Plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	GAGNAIRE Didier	Chimie Physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse Numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique Générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie Végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie Générale
	LATURAZE Jean	Biochimie Pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique Médicale B
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences Nucléaires
	LONGEQUEUE Jean-Pierre	Physique Nucléaires
	LOUP Jean	Géographie
Melle	LUTZ Elisabeth	Mathématiques Pures
	MALGRANGE Bernard	Mathématiques Pures
	BOUTET DE MONVEL Louis	Mathématiques Pures
	MALINAS Yves	Clinique Obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MAZARE Yves	Clinique Médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique Maladies Infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie Nucléaire
	MULLER Jean-Michel	Thérapeutique (Néphrologie)
	NEEL Louis	Physique du solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie Systématique
	RENARD Michel	Thermodynamique
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-Chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SIBILLE Robert	Construction Mécanique (I.U.T. A)
	SOUTIF Michel	Physique Générale
	TANCHE Maurice	Physiologie

MM.	TRAYNARD Philippe	Chimie Générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
	VAUQUOIS Bernard	Calcul Electronique
Mme	VERAIN Alice	Pharmacie Galénique
MM.	VERAIN André	Physique
	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique Nucléaire Théorique

PROFESSEURS ASSOCIES

MM.	CLARK Gilbert	Spectrométrie Physique
	CRABBE Pierre	CERMO
	ENGLMAN Robert	Spectrométrie Physique
	HOLTZBERG Frédéric	Basses Températures
	ROST Ernest	Sciences Nucléaires

PROFESSEURS SANS CHAIRE

Melle	AGNIUS-DELORD Claudine	Physique Pharmaceutique
	ALARY Josette	Chimie Analytique
MM.	AMBROISE-THOMAS Pierre	Parasitologie
	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques Appliquées
	BIAREZ Jean-Pierre	Mécanique
	BILLET Jean	Géographie
	BOUCHET Yves	Anatomie
	BRUGEL Lucien	Energétique (I.U.T. A)
	BUISSON René	Physique (I.U.T. A)
	CONTE René	Physique (I.U.T. A)
	DEPASSEL Roger	Mécanique des Fluides
	GAUTHIER Yves	Sciences Biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	GROULADE Joseph	Biochimie Médicale
	HACQUES Gérard	Calcul Numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Médecine Préventive
	IDELMAN Simon	Physiologie Animale
	JOLY Jean-René	Mathématiques Pures
	JULLIEN Pierre	Mathématiques Appliquées
Mme	KAHANE Josette	Physique
MM.	KUHN Gérard	Physique (I.U.T. A)
	LE ROY Philippe	Mécanique (I.U.T. A)
	LUU DUC Cuong	Chimie Organique
	MAYNARD Roger	Physique du Solide
	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PSISTER Jean-Claude	Physique du Solide
Melle	PIERY Yvette	Physiologie Animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie Générale
	RICHARD Lucien	Biologie Végétale
Mme	RINAUDO Marguerite	Chimie Macromoléculaire
MM.	ROBERT André	Chimie Papetière

MM.	SARRAZIN Roger	Anatomie et Chirurgie
	SARROT-REYNAUD Jean	Géologie
	SIROT Louis	Chirurgie Générale
Mme	SOUTIF Jeanne	Physique Générale
MM.	STREGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie (I.U.T. A)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro chirurgie
	BARJOLLE Michel	MIAG
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (C.U.S.)
MM.	BRODEAU François	Mathématiques (I.U.T. B)
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et Organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie Papetière
	CHIAVERINA Jean	Biologie Appliquée (EFP)
	COHEN-ADDAD Jean-Pierre	Spectrométrie Physique
	COLOMB Maurice	Biochimie Médicale
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie Végétale
	DUSSAUD René	Mathématiques (C.U.S.)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine Légale
	FAURE Gilbert	Urologie
	FONTAINE Jean-Marc	Mathématiques Pures
	GAUTIER Robert	Chirurgie Générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GROS Yves	Physique (I.U.T. A)
	GUITTON Jacques	Chimie
	HICTER Pierre	Chimie
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	KOLODIE Lucien	Hématologie
	KRAKOWIAK Sacha	Mathématiques Appliquées
	LE NOC Pierre	Bactériologie-virologie
	LEROY Philippe	I.U.T. A
	MACHE Régis	Physiologie Végétale
	MAGNIN Robert	Hygiène et Médecine Préventive
	MALLION Jean-Michel	Médecine du Travail
	MARECHAL Jean	Mécanique (I.U.T. A)
	MARTIN-BOUYER Michel	Chimie (C.U.S.)

M.	MICHOULIER Jean	Physique (I.U.T. A)
Mme	MINIER Colette	Physique (I.U.T. A)
MM.	NEGRE Robert	Mécanique (I.U.T. A)
	NEMOZ Alain	Thermodynamique
	NOUGARET Marcel	Automatique (I.U.T.A)
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (I.U.T. B)
	PEFFEN René	Métallurgie (I.U.T. A)
	PERRET Jean	Neurologie
	PERRIER Guy	Géophysique - Glaciologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine Interne
	RACINET Claude	Gynécologie et Obstétrique
	RAMBAUD André	Hygiène et Hydrologie
	RAMBAUD Pierre	Pédiatrie
Mme	RENAUDET Jacqueline	Bactériologie
MM.	ROBERT Jean-Bernard	Chimie-Physique
	ROMIER Guy	Mathématiques (I.U.T. B)
	SHOM Jean-Claude	Chimie générale
	STOEBNER Pierre	Anatomie pathologique
	VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M.	COLE Antony	Sciences Nucléaires

CHARGE DE FONCTIONS DE MAITRE DE CONFERENCES

M.	JUNIEN-LAVILLAVROY Paul	O.R.L.

Fait à SAINT MARTIN D'HERES,
DECEMBRE 1975.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. NEEL Louis

Vice-Présidents : M. BENOIT Jean

M. BONNETAIN Lucien

PROFESSEURS TITULAIRES

MM.	BENOIT Jean	Radioélectricité
	BESSON Jean	Electrochimie
	BLOCH Daniel	Physique du solide
	BONNETAIN Lucien	Chimie Minérale
	BONNIER Etienne	Electrochimie et Electrométallurgie
	BRISSONNEAU Pierre	Physique du solide
	BUYLE-BODIN Maurice	Electronique
	COUMES André	Radioélectricité
	FELICI Noël	Electrostatique
	LESPINARD Georges	Mécanique
	MOREAU René	Mécanique
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du solide
	PERRET René	Servomécanismes
	POLOUJADOFF Michel	Electrotechnique
	SILBERT Robert	Mécanique des Fluides

PROFESSEURS ASSOCIES

MM.	RUPPERSBERG Albert, Henner	Chimie
	ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM.	BLIMAN Samuel	Electronique
	BOUVARD Maurice	Génie Mécanique
	COHEN Joseph	Electrotechnique
	DURAND Francis	Métallurgie
	FOULARD Claude	Automatique
	LACOUME Jean-Louis	Géophysique
	LANCIA Roland	Electronique
	VEILLON Gérard	Informatique Fondamentale & Appliquée
	ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM.	ANCEAU François	Mathématiques Appliquées
	BOUDOURIS Georges	Radioélectricité
	CHARTIER Germain	Electronique
	GUYOT Pierre	Chimie Minérale
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du solide
	MORET Roger	Electrotechnique Nucléaire
	PIERRARD Jean-Marie	Hydraulique
	ROBERT François	Analyse Numérique
	SABONNADIERE Jean-Claude	Informatique Fondamentale & Appliquée
Mme	SAUCIER Gabrièle	Informatique Fondamentale & Appliquée

MAITRE DE CONFERENCES ASSOCIE

M.	LANDAU Ioan	Automatique

CHERCHEURS DU C.N.R.S. (Directeurs et Maîtres de Recherche)

MM.	FRUCHART Robert	Directeur de Recherche
	ANSARA Ibrahim	Maître de Recherche
	CARRE René	Maître de Recherche
	DRIOLE Jean	Maître de Recherche
	MATHIEU Jean-Claude	Maître de Recherche
	MUNIER Jacques	Maître de Recherche

Il est d'usage de commencer une thèse en affirmant sa gratitude à tous et à chacun. Je voudrais que cette traditionnelle "page de remerciements" n'ait pas l'effet d'un ronronnement conventionnel et vaguement réconfortant : il est certain que j'ai une dette envers tous ceux qui vont être mentionnés, et d'ailleurs envers bien d'autres qui, je l'espère, ne me tiendront pas rigueur de ne pas se retrouver ici.

C'est au sein du Groupe d'Etudes pour la Traduction Automatique (GETA) que j'ai poursuivi les recherches qui font l'objet de cette thèse. Je tiens à remercier M. VAUQUOIS, Professeur à l'Université Scientifique et Médicale de Grenoble (USMG) et Directeur du GETA, qui m'a introduit à la Traduction Automatique et a bien voulu diriger et suivre ces travaux, ainsi que tous les membres du GETA, qui m'ont aidé directement ou grâce à leur travail dans le groupe.

Monsieur KUNTZMANN, Professeur à l'USMG, m'a fait l'honneur de présider le jury, ce dont je le remercie, et plus encore de l'intérêt qu'il a toujours marqué à mon endroit.

J'ai également des raisons particulières de remercier les autres membres du jury. C'est Monsieur WERNER, Maître de Conférences à l'IUT de Lille, qui m'a introduit au royaume de la récursivité. Il n'a cessé par la suite de me stimuler dans ce domaine et s'est intéressé de très près à une grande partie de ce travail. Monsieur LENTIN, Professeur à l'Université René Descartes à Paris, a bien voulu lire et critiquer en détail la rédaction provisoire de cette thèse, et m'a certainement aidé à préciser plusieurs points importants. Monsieur BENZAKEN, Professeur à l'USMG, m'a également aidé de remarques et de conseils précis, et c'est à lui que je dois un intérêt certain pour les études algébriques en théorie des automates. Monsieur HERAULT, Directeur du Centre de Linguistique Quantitative de L'Université Pierre et Marie Curie à Paris, a bien voulu parrainer ces recherches et me donner des indications très utiles sur ce qu'on peut attendre des méthodes statistiques en linguistique. "Last but not least", Monsieur ZEMB, Professeur à l'Université Sorbonne Nouvelle à Paris, m'a aidé à mieux m'orienter dans les méandres de la linguistique et de la grammaire modernes, au cours de discussions passionnantes, parfois surprenantes et toujours enthousiastes !

Mesdemoiselles CURINIER et ALLOSIO ont dactylographié ces pages. Un œil exercé ne manquera pas de remarquer la difficulté de la tâche, dûe à la multiplicité des signes et des caractères, et d'apprécier la qualité du résultat.

Je ne saurais oublier les membres du service de reproduction du CICG qui ont terminé la réalisation matérielle de ce travail.

Cette thèse est dédiée à la mémoire de Jean TRAIN, professeur de russe à l'Ecole des Langues Orientales, qui fut un ardent défenseur de la Traduction Automatique dès ses débuts et, par son enseignement et ses conseils, m'engagea dans cette voie.

In fact there is good reason to believe that there may be similarities and a kinship between creativity and madness, at least in an obsessive singleness of purpose and involvement, and in a fresh, if not bizarre, way of looking at things.

L. UHR

INDICATIONS GENERALES

Afin de faciliter la lecture de ce travail, j'ai essayé de rendre les différentes parties relativement indépendantes, et de clarifier la rédaction en donnant l'idée générale des démonstrations avant les preuves formelles et en expliquant les définitions un peu "barbares". Ceci ne va pas sans quelques redites ou lourdeurs, dont on voudra bien m'excuser.

J'ai souvent tenté de situer les résultats que j'ai obtenus par rapport à ceux de la littérature. Quand une démonstration ou une définition est due à un auteur, je l'indique entre parenthèses avant l'énoncé de la propriété ou de la définition. Si elle est classique, mais anonyme, ce nom est remplacé par une étoile. Les autres sont originales, et n'engagent donc que moi.

D'autre part, j'ai adopté une double pagination : en haut d'une page, on trouve au milieu la pagination dans le chapitre, et à droite la pagination globale. Les références éventuelles se rapportent à la pagination interne. De même, les définitions, théorèmes, propositions ou lemmes sont numérotés à l'intérieur de chaque chapitre.

Enfin, les références bibliographiques sont directes ou indirectes. Dans le premier cas, le nom de l'auteur est suivi, entre parenthèses, de l'année et éventuellement d'une minuscule, par exemple Blum (1967a). Cette référence renvoie à la bibliographie d'ensemble, placée à la fin du volume et présentée par ordre alphabétique. Dans le second cas, un numéro entre crochets renvoie à une table située à la fin de chaque chapitre ; elle donne la référence directe correspondante et sa position dans le chapitre est rappelée en haut et à gauche de chaque page.

ABREVIATIONS

IA	intelligence artificielle
TEAHQ	traduction entièrement automatique de haute qualité (FAHQT en anglais)
MT	machine de Turing
r.e.	récursivement énumérable
EAO	effectivement approximativement ordonné
EQO	effectivement quasi-ordonné
PEC	à production effectivement contrôlée
EF	états finis
HC	hors-contexte
SC	sous-contexte
AFA	approche faible
AFO	approche forte
APP	approche
IDA	identification approximative
IDAF	identification approximative en temps fini
IDAX	identification approximative en temps fixé
IFA	identification faible
IFO	identification (forte)
RT	réseau de transitions
RTR	réseau de transitions et de ruptions
ssi	si et seulement si
p.p.	presque partout
i.s.	infiniment souvent
i.e.	"id est" - c'est-à-dire

NOTATIONS USUELLES

Ensembles de base

$\mathbb{N}$	entiers naturels
$\mathbb{Z}$, $\mathbb{Z}_+$, $\mathbb{Z}_-$	entiers relatifs (positifs, négatifs)
$\mathbb{Q}$, $\mathbb{Q}_+$, $\mathbb{Q}_-$	rationnels (positifs, négatifs)
$\mathbb{C}$	complexes

Cortèges ou familles

$(x_o, \ldots, x_n)$	cortège d'éléments
$\langle x_o,\ldots,x_n \rangle$	code d'un cortège
$(x_i)_{i \in I}$	famille indexée par I (suite si $I = \mathbb{N}$)
$\{x_i\}_{i \in I}$	ensemble des éléments de $(x_i)_{i \in I}$

Relations ensemblistes

$A = \{x \mid P(x)\}$ ou $A = \{x : P(x)\}$	ensemble des éléments possédant la propriété P
$\overline{A}$	complémentaire de A dans un surensemble fixé (souvent $\mathbb{N}$)
$A-B$	différence simple $(A \cap \overline{B})$
$A \oplus B$	somme directe
$A \ominus B$ ou $A \Delta B$	différence symétrique $((A-B) \cup (B-A))$
χ_A	fonction caractéristique
$\lvert A \rvert$	cardinal
A^*	monoïde engendré par A
$\lvert x \rvert$	longueur d'une chaîne $x \in A^*$
Λ	neutre (mot vide)
A^+	$A^* - \{\Lambda\}$
A^B	ensemble des fonctions de B dans A
$A^{(B)_a}$	sous-ensemble des fonctions prenant presque partout (au sens d'une mesure sur B) la valeur a
$\mathrm{Dom}\,\varphi$	domaine de définition ($\varphi \in A^B$)
$\mathrm{Im}\,\varphi$	image
2^A	ensemble des parties de A
$[\Phi\ (A)\ ]$	pour $\Phi \in (X^k)^X$ et $A \subset X$: ensemble des éléments des images, soit $\bigcup_{x \in A} \{\Phi(x)\}$
$[\Phi^{n+1}(A)]$	$[\ \Phi\ ([\Phi^{n}(A)]\ )\ ]$

Notations empruntées à la récursivité

P^n	fonctions récursives partielles à n arguments
R^n	fonctions récursives (totales) à n arguments
PR^n	fonctions récursives primitives à n arguments
$\mathcal{C}[A]$	dans une classe de fonctions, sous-ensemble formé par celles qui prennent leurs valeurs dans A.
$\varphi_x^{(n)}$	fonction partielle récursive à n arguments, d'indice x dans une gödélisation acceptable.
$W_x = \mathrm{Dom}\varphi_x$	ensemble récursivement énumérable (r.e.) d'indice r.e. x.
$C_x = \varphi_x^{-1}(1)$	ensemble récursif d'indice caractéristique $x\,(\varphi_x \in R^1_{[0,1]})$
$D_x = \{x_o,\ldots,x_n\}$	ensemble fini d'indice canonique $x = \sum_{i=0}^{n} 2^{x_i}$
$\varphi(x)\downarrow$	$\varphi(x)$ converge $(x \in \mathrm{Dom}\,\varphi)$
$\varphi(x)\uparrow$	$\varphi(x)$ diverge $(x \notin \mathrm{Dom}\,\varphi)$
$K = \{x \mid \varphi_x(x)\downarrow\}$	(ensemble r.e. non récursif)
$\lambda y[f(x,y,z)]$	fonction prise par rapport à y
$\mu x \leqslant n[P(x)]$ ou $\mathop{\mathfrak{M}}_{x \leqslant n}[P(x)]$	le plus petit $x \leqslant n$ tel que P(x)
$\mathrm{Max}\ x \leqslant n\ [P(x)]$	le plus grand $x \leqslant n$ tel que P(x)
$\mu x\ [P(x)]$	le plus petit x tel que P(x)
$\mathrm{Max}\ x\ [P(x)]$	le plus grand x tel que P(x)
$\lceil x \rceil$	le plus petit entier supérieur à x
$\lfloor x \rfloor$	le plus grand entier inférieur à x
$x \dot{-} y$	Max (0, x-y)
$(\overset{\infty}{\forall} x)\ [P(x)]$	Pour presque tous les x, P(x) ; ou : "P(x) presque partout" (p.p.)
$(\overset{\infty}{\exists} x)\ [P(x)]$	Pour une infinité d'x, P(x) ; ou : "P(x) infiniment souvent" (i.s.)

SOMMAIRE

CHAPITRE O

HISTORIQUE ET PROBLEMES ACTUELS

"За прошедшие с тех пор /1955 г./ полтора десять лет данная область науки бурно развивалась; она знала свои спады и подъемы, в ней сложились разнообразные направления, разрабатываемые многочисленными коллективами; относительно ее значения и перспектив существуют также весьма разнообразные, зачастую противоречивые точки зрения."

Кулагина и Мельчук /1971/

HISTORIQUE ET PROBLEMES ACTUELS

La traduction automatique n'est pas une branche des sciences comme les mathématiques, la physique ou la logique. C'est plutôt un problème dont la (ou les) solution doit emprunter des méthodes à divers domaines, ou y susciter des développements qui restent intéressants en eux-mêmes. Le but de ce travail consiste, en partant de l'expérience acquise, à dégager quelques principes directeurs pour la construction de "meilleurs" systèmes de TA, à les expliciter dans un cadre suffisamment formel et enfin à les appliquer effectivement à la définition d'un système prototype. Avant de commencer, il semble donc utile de retracer l'histoire relativement brève de la TA et des problèmes auxquels elle s'est heurtée.

1 - UN PEU D'HISTOIRE

L'histoire de la TA est brève mais déjà passablement mouvementée. Le mémorandum de Warren Weaver, en 1949, est sans doute le premier document qui précise le problème et envisage sa solution au moyen des calculateurs généraux, qui commencent juste à être disponibles. On peut distinguer trois périodes : de 1949 à 1961, de 1961 à 1967, après 1967.

Les débuts de la TA ont été très rapides : l'ouvrage bibliographique de Mel'tchuk et Ravitch [25] ne contient pas moins de 1430 références ! Dès 1949, le MIT, avec Yngve et Bar-Hillel [38] , se lance dans la course, le premier symposium a lieu en 1952, on édite des publications spéciales §, puis des ouvrages de vulgarisation§§, et on réalise les premières expériences aux USA§§§ et en URSS§§§§. On trouvera dans [34] une étude détaillée de cette période dont la fin voit la création de très nombreux groupes de recherche.

Suit un second souffle, pendant lequel les groupes existants améliorent et développent leurs systèmes, tandis que de nouveaux venus (CETA en France, Université de Kyoto au Japon, ...) en construisent sur de nouvelles bases en utilisant le nouvel appareil de la théorie des langages formels et de la compilation. A la fin de cette période, on a obtenu des résultats très encourageants sur des masses de textes déjà considérables§§§§§. Mais le rapport ALPAC [40] , sur la base des travaux de la première période, conteste l'intérêt et l'utilité des recherches dans ce domaine et entraîne, aux USA tout au moins, la quasi-disparition de ce thème de recherche, en faveur de "l'intelligence artificielle" et de la "communication homme-machine". On trouvera une analyse (très) critique du rapport ALPAC dans le contre-rapport de Pankowicz [42] .

Depuis, de nombreux groupes ont connu des difficultés, surtout en URSS. Mais la plupart de ceux qui continuent à s'intéresser à la TA cherchent à construire des systèmes plus efficaces parce que plus "intelligents", tandis que, surtout aux USA, le traitement automatique de langues naturelles reste une pierre de touche pour l'IA, de sorte que ces deux voies se rapprochent. Quoi qu'on en ait dit, automatiser la traduction, toujours dans une certaine mesure, sera bientôt, plus qu'un problème abstrait, une nécessité économique. Depuis 1967, on peut remarquer le développement de systèmes informatiques généraux §§§§§§permettant d'écrire tout ou partie d'une chaîne de TA pour un couple de langues arbitraire, et de recherches concernant l'utilisation de la "sémantique" dans les systèmes de TA. De plus, les recherches entreprises à propose de la TA ont eu des retombées dans d'autres domaines, comme l'analyse [14] ou la synthèse [32] de la parole, ou encore les grammaires transformationnelles [5] .

---

§ A partir de 1954, "Mechanical Translation" (MIT)
A partir de 1958, "Научно-Техническая Информация" (Moscou), traitant aussi d'autres sujets
A partir de 1960, "La Traduction Automatique" (Paris), devenue "TA-informations" en 1965.

§§ Panov (1960), Delavenay (1960).

§§§ En 1954, L. Dostert présente une première expérience de traduction russe-anglais.
En 1961, ce système est opérationnel et Brown commence à l'appliquer à la traduction français-anglais .

§§§§ En 1956-57, O.S. Kulagina et I.A. Mel'tchuk font un premier essai de TA français-russe, et le groupe de Leningrad (Tseitin, Leikina, Fitialov, ...) réalise une expérience sur la TA anglais-russe vers 1962, dans laquelle on sépare déjà complètement algorithmes et données linguistiques [29]

§§§§§ Voir [28] pour l'expérience de Kyoto sur la TA anglais-japonais. Au CETA, les essais portent sur un corpus de près de 400000 mots, traduits à raison d'environ 1,2 mots/s par un système, programmé sur IBM 7044, indépendant des langues traitées et appliqué principalement à la TA russe-français, mais aussi à l'analyse de l'allemand et du japonais.

§§§§§§ Systèmes-Q au projet TAUM [44] , systèmes ATEF et CETA au GETA [6,7] .

2 - DIFFERENTES APPROCHES

Le terme "traduction automatique" est vague, et recouvre en fait tout une classe de problèmes différents, que des personnes de formations variées ont tenté de résoudre au moyen de techniques diverses : autant de raisons pour expliquer la multiplicité des solutions proposées.

Chercher à réaliser un système efficace dans des délais relativement courts impose de modérer ses ambitions, tandis que l'étude de systèmes complets et adéquats à une certaine théorie[§] du langage et du processus de traduction est une entreprise de très longue haleine. De même, la traduction à partir d'un texte oral n'a encore été abordée dans aucun système existant, bien qu'il y ait eu beaucoup d'études consacrées à la perception de la parole[§§]. Il semble maintenant que toutes les réalisations entreprises utilisent des calculateurs généraux digitaux, bien qu'on ait tenté dans le passé d'utiliser des machines spéciales[§§§]. Enfin, les méthodes employées ont été fort différentes selon que la formation des chercheurs était de type informatique, linguistique ou mathématique.

Les toutes premières expériences ont consisté à consulter des "dictionnaires automatiques" pour réaliser des traductions mot-à-mot, puis, devant leur mauvaise qualité, à introduire dans les programmes des renseignements particuliers fournis par les linguistes sous forme de tables ou d'organigrammes : on programme directement les algorithmes imaginés par le linguiste[§§§§]. L'étape suivante consiste à structurer l'ensemble des programmes de traitement une fois pour toutes, en fonction par exemple de niveaux issus de la théorie linguistique ; tous les renseignements particuliers sont alors introduits comme des tables ou des procédures particulières, à chaque niveau[§§§§§]. On pourrait dire que c'est une étape de semi-formalisation. Enfin, on s'est de plus en plus orienté vers des "systèmes intégrés" [34] où la formalisation est totale en ce sens que les modèles (ensembles des programmes traitant un niveau particulier) sont complètement séparés des données linguistiques (grammaires, dictionnaires, ...) qui sont écrites dans des langages artificiels[§§§§§§]. Ces systèmes peuvent donc utiliser des algorithmes efficaces développés dans le cadre de la théorie des langages formels et des automates, et dont on connait les propriétés (complexité place-temps, décidabilité,...).

---

§ Voir Schreider (1973) pour une discussion de l'emploi de ce terme .

§§ Voir Frey (1970), Tubach (1970), ainsi que leurs références et les travaux du CNET à Lannion.

§§§ Machine à disque photoscopique de King (1959), machine de Ceccato.

§§§§ Voir le système "Fulcrum" de P. Garvin (1967), ou celui de l'Université de Saskatoon [4] .

§§§§§ Le programme de Georgetown [41] considérait par exemple les opérations suivantes : consultation de dictionnaires, fabrication de syntagmes de bas niveau, puis de haut niveau, réarrangement, consultation de dictionnaires de transfert, synthèse (des terminaux).
Les premiers essais de O.S. Kulagina et I.A. Mel'tchuk [19] étaient semble-t-il, eux aussi de ce type, et réalisaient une analyse en dépendance.

§§§§§§ Ce type de systèmes a été proposé par Yngve [39] et une première implémentation de ce type a été réalisée par le groupe du Pr. Tseitin [29] dès 1962-63 sur une machine de taille et de puissance réduites (BESM 3M-4, 8 K mots de 45 bits, cycle = 0,5 µs, tambour de ... 4K, bandes magnétiques et cartes perforées). On peut ensuite citer l'Université de Kyoto et le CETA, où la plus grande partie des modèles est de ce type. Les systèmes-Q du projet TAUM utilisent un unique modèle, tandis que le GETA achève actuellement l'écriture d'un système totalement intégré comprenant plusieurs modèles associés à différents niveaux de traitement.

A ces différences dans la conception de la réalisation informatique s'ajoutent celles dues au degré d'adéquation linguistique. Très tôt, on a distingué les niveaux morphologique, syntaxique et sémantique. Mais ce qu'on y met dépend à la fois des écoles linguistiques et des contraintes imposées par les modèles. Comme le note Mel'tchuk [23] , la "linguistique pour TA" doit expliciter et compléter les connaissances fournies par la linguistique "classique" pour qu'un procédé automatique puisse les utiliser. Il faut aussi, dans une certaine mesure, les adapter au modèle considéré, et, si celui-ci a été conçu a priori, le codage de ces renseignements peut ne plus avoir qu'un rapport lointain avec la théorie initiale[§]. Il semble que cette réaction de l'algorithme de traitement sur la présentation des données linguistiques soit générale, et, à tout prendre, inévitable, sauf peut-être dans les méthodes à "filtres", où l'algorithme n'est pas précisé : on sait seulement qu'il extrait toutes les solutions vérifiant un certain ensemble de contraintes (qui constitue justement la grammaire) [29,30] . Inversement, on peut construire des modèles informatiques exactement adaptés à un modèle formalisé issu de recherches linguistiques : c'est le cas du système de J. Friedman [12] pour la simulation de grammaires transformationnelles de Chomsky. D'autre part, la stratégie de traduction peut être plus ou moins représentative de l'hypothèses linguistique souvent avancée selon laquelle il existerait un niveau profond de représentation du discours, indépendant de la langue utilisée [8, 24] certains systèmes effectuent la traduction en opérant un transfert lexical et les transformations sur les structures de surface [28] , d'autres passent par un "langage pivot", dont la structure est indépendante des langues utilisées, mais qui utilise leur lexique (systèmes du CETA, de TAUM). Enfin, le degré d'adéquation linguistique d'un système dépend de ce que l'on en attend : une traduction de haute qualité, une traduction automatique aidée par l'homme, ou une traduction humaine aidée par la machine ? Bar-Hillel, qui fut d'ailleurs le premier chercheur à plein temps dans ce domaine, a affirmé très tôt [1] qu'une traduction entièrement automatique de haute qualité (TEAHQ) était impossible, essentiellement parce qu'elle supposait une simulation du processus humain de traduction, et donc l'utilisation "intelligente" d'une énorme connaissance de base[§§]. Ceci reste vrai à l'heure actuelle, mais Bar-Hillel lui-même n'exclut pas de lier l'appréciation de la qualité à un but plus modeste que la TEAHQ [2] . La traduction humaine aidée par la machine, prônée par M. Mastermann [22] , ne semble pas répondre aux exigences des utilisateurs éventuels : pour être vraiment utile à un bureau de traduction, un système automatique doit déjà fournir une traduction, éventuellement soumise à révision. On s'oriente donc presqu'uniquement vers la TA aidée par l'homme. Il est évident que ce terme couvre encore une large gamme de possibilités, depuis la traduction mot-à-mot à la traduction la plus "intelligente possible", en passant par le transcodage [37, 38] , le "transformulage" [15] (si l'on ose écrire ce terme en français ...), la traduction dans un domaine restreint de textes à syntaxe prédéfinie [1] , la traduction de textes quelconques à l'aide d'un système interactif ou non, etc. C'est ce dernier type de traduction que cherche à réaliser l'équipe du Pr. Vauquois (GETA), au sein de laquelle j'ai eu la chance de travailler ces dernières années.

---

§ Dans le système du CETA, les règles de la grammaire d'analyse syntaxique devaient être écrites dans une forme normale de Chomsky généralisée, liée à un certain algorithme d'analyse [35] , et on était amené à définir certaines classes syntaxiques sans réelle interprétation linguistique, mais qui permettaient de réduire le nombre des ambiguïtés syntaxiques.
De même, le programme de Georgetown impose une présentation en trois niveaux, morphologique, syntagmatique et syntaxique.

§§ "To sum this part up, I would say there is no prospect whatsoever that the employment of electronic digital computers in the field of translation will lead to any revolutionary changes. A complete automation of the activity is wholly utopian, since the fact that book and papers are usually written for readers with a certain background knowledge and an ability for logical deduction and plausible reasoning cannot be over-ridden by even the cleverest utilization of all formal features of a discourse" [1,p. 183].

3 - LES "GENERATIONS"

On parle souvent des systèmes de TA de 1ère, 2ème ... génération. Cette notion de génération est liée aux méthodes employées, mais il en existe au moins deux définitions non équivalentes.

O.S. Kulagina et I.A. Mel'tchuk ont proposé dans [20] une classification "positive". Les systèmes de "première génération" sont caractérisés par la binarité, la non-séparation et l'unicité, c'est-à-dire qu'ils sont construits pour un couple de langues donné, que les renseignements linguistiques sont intégrés aux programmes et qu'en cas d'ambiguïté à un moment quelconque du traitement, on choisit plus ou moins arbitrairement une seule possibilité. Lors de leurs premiers essais, ces auteurs ont utilisé [19] ce type de méthode. Dans les systèmes de "deuxième génération", on s'efforce d'écrire des programmes valables pour une large classe de couples de langues, de séparer les données des programmes et de conserver les solutions multiples pour choisir le plus tard et, on l'espère, le mieux possible. C'est le stade où l'on s'intéresse énormément aux procédés d'analyse syntaxique. P. Garvin [14] à Washington, S. Sugita [28] à Tokyo, O.S. Kulagina [18] à Moscou ont réalisé des systèmes de ce type. Entre la première et la seconde génération, il en existe une autre, "hybride", ou "1 1/2" [20] . Les systèmes de ce type, initialement de première génération, en ont gardé les traits principaux, mais certaines parties du traitement, en particulier l'analyse syntaxique, y ont été remaniées en fonction des résultats théoriques sur les langages formels. Un bon exemple est le système de Georgetown dans sa forme finale [41] . Enfin, les systèmes de "troisième génération" comportent une partie "sémantique", au sens où ils passent par l'intermédiaire d'une représentation "profonde" du texte de type logique et utilisent certains renseignements qui ne sont pas uniquement des traits associés aux unités de la langue. D'après [20] , le système du CETA [33,34] est la réalisation qui s'en rapproche le plus.

B. Vauquois [34] donne une caractérisation "négative". Les systèmes de première génération n'utilisent ni la théorie des langages formels, ni celle des automates, ni celle de la compilation. Ceux de seconde génération utilisent les théories précédentes, mais pas encore de sémantique "déductive", et ne peuvent offrir ni différents niveaux de qualité, ni adaptation automatique des données linguistiques à un corpus, ni possibilité de simulation de stratégies de traduction. On pourrait les appeler "intégrés", et le système du CETA serait donc un système intégré. La "troisième génération", abordée dans ce travail, jouirait des avantages précédents mais serait encore très éloignée de l'étape, que Bar-Hillel considère d'ailleurs comme utopique, et peut-être à juste titre, où un système de TA serait à même de simuler vraiment les processus humains de compréhension et de traduction. On s'oriente donc vers des systèmes "plus intelligents", et on cherche alors naturellement à adapter des méthodes suggérées par certaines recherches en "intelligence artificielle" (IA) [36] .

On peut peut-être proposer une classification plus fine en distinguant trois niveaux pour trois critères différents : le contenu linguistique du système, l'organisation informatique de chaque composant et l'organisation générale de ces composants.

D'abord, on peut considérer trois niveaux de base [10] dans le langage : L1 est le niveau grammatical, où l'on ne s'occupe que des propriétés formelles des éléments ; L2 est le niveau conceptuel, où l'on dispose d'une représentation formalisée de certaines relations entre éléments provenant de l'existence d'un univers de référence (trait "gnosto-encyclopédique" de Mel'tchuk) ; L3 est le niveau pragmatique de l'usage du langage pour communiquer dans une situation donnée.

Ensuite, les données linguistiques peuvent être intégrées dans les programmes, comme dans les premiers programmes de TA, ou complètement séparées, ou encore séparées, mais avec des possibilités de contrôle sur un algorithme général, qu'on peut d'ailleurs interpréter comme un dispositif de génération pour toute une classe d'algorithmes.

Enfin, l'enchaînement des composants peut être incorporé aux différents programmes, déterminé par un programme moniteur ou encore défini dans un métalangage, au même titre que les données linguistiques.

On pourrait alors dire que le système de Georgetown [41] est (1,1,2), tandis que celui du CETA [34] est (1,2,2) et celui du GETA [6,7] (1,3,2).

4 - UN CONFLUENT

On peut, pour conclure, insister encore sur la tridisciplinarité de la TA [§]. Linguistique d'abord, et l'expérience passée a montré qu'il était utile de décomposer le traitement d'un texte en plusieurs phases, correspondant au passage d'un niveau de description (déterminé par une certaine théorie linguistique) à un autre, et de définir des langages formels pour l'écriture des données linguistiques [§§]. On souhaiterait également, à la fois pour accroître la qualité des traductions et progresser dans la connaissance du langage, introduire un niveau de représentation du sens [24,36] . Et, pour répondre à un souhait du Pr. Vinay, il serait intéressant de disposer d'un outil permettant de simuler différents types de traduction. On rapprochera également ces idées de l'importante étude réalisée à l'Université du Texas [43] .

Informatique ensuite : là aussi l'expérience montre qu'il faut chercher à réaliser des systèmes informatiques conçus pour un traitement de masse, aussi bien en ce qui concerne les données (variables, classes grammaires, dictionnaires..) que les textes. La séparation totale entre algorithme et données linguistiques n'est pas très pratique, c'est pourquoi on construit plutôt des systèmes dans lesquels les données peuvent contenir certains contrôles sur l'algorithme principal de traitement. Il semblerait intéressant de définir des systèmes qui permettent une interaction entre certaines phases du traitement, avec l'utilisateur et ... avec le texte.

En effet, si la distinction entre niveaux linguistiques successifs semble admise[§§§], rien n'impose a priori de faire passer toute une unité de traduction (phrase, paragraphe, texte) globalement de niveau en niveau, sans retour possible ; cette façon de faire impose d'ailleurs souvent à une phase de défaire une partie de ce qu'a fait la précédente. D'autre part, les systèmes conversationnels se répandent de plus en plus, et on pourrait ne pas limiter l'emploi de cette possibilité à la seule préparation des données : un système de TA, dans un certain mode de fonctionnement, devrait permettre à l'utilisateur d'influer sur le cours du traitement, en lui posant des questions ; et on rejoint là l'approche "man-aided machine translation". Enfin, l'expérience a montré combien il était pénible de mettre au point les données linguistiques sur un corpus de taille déjà respectable, et de les modifier pour en traiter un autre ; on verra plus loin comment essayer d'automatiser une partie de cette tâche.

Mathématique enfin, bien que de façon moins visible. Il importe en effet de ne pas écrire des programmes "au petit bonheur la chance", mais d'étudier les opérations à effectuer sur la représentation formelle considérée et de dégager des algorithmes généraux[§§§§] qui, à partir de données formalisées, construisent les algorithmes effectifs de traitement, dont la classe doit posséder certaines propriétés. En particulier la décidabilité : il faut que ces algorithmes calculent des fonctions totales, sinon, et des expériences fâcheuses l'ont prouvé, on s'expose à "boucler" dans des cas particuliers. De plus, ce type d'études recherche des modèles (automates abstraits) dont la complexité soit la plus faible possible, compte tenu de la puissance désirée.

---

§ Voir aussi à ce propos B. VAUQUOIS (1968).

§§ Et, selon Mel'tchuk [20,23] , cela impose aux linguistes d'être "exacts et exhaustifs".

§§§ Même si la définition de ces niveaux donne lieu à nombre de controverses.

§§§§ Qui sont plutôt des algorithmes "universels" au sens de la théorie de la récursivité.

5 - PLAN

J'ai essayé d'apporter une contribution personnelle dans ces trois domaines, en analysant les propriétés "désirables" d'un système de TA, en cherchant comment y introduire plus de "sémantique" sans diminuer trop son efficacité, et en définissant un prototype qui utilise comme "modules" plusieurs systèmes réalisés au GETA (systèmes ATEF et CETA, le premier légèrement modifié) ainsi qu'un "moniteur" programmable (système MONIT) et un module permettant l'utilisation d"algogrammaires" (système ALGOG).

Les idées initiales dérivent des expériences dont j'ai eu connaissance par la littérature et de l'expérience directe de différents systèmes. Elles consistent essentiellement à revenir à plus de souplesse sans perdre l'acquis des systèmes de deuxième génération. Je propose donc que l'organisation d'un système de TA ne soit plus fixée une fois pour toutes : comme cela a déjà été fait pour différents modules de systèmes existants (analyseurs morphologiques, syntaxiques, grammaires transformationnelles, ...), on définit dans le "moniteur" du système une partie algorithmique et une partie linguistique, dans laquelle on décrit l'organisation souhaitée dans un langage formel adéquat. On peut y faire appel aux différentes opérations élémentaires des "modules" (ATEF, CETA, ALGOG, INTER-interface avec l'utilisateur, module de génération de formes). D'autre part, je cherche à rendre le système le plus souple possible, en permettant une exécution conversationnelle dans certains modules, en en munissant certains de propriétés adaptatives et en permettant de concevoir le processus de traduction comme un processus heuristique.

Je suis donc parti d'idées très concrètes, qui m'ont amené à étudier des questions plus abstraites. Mais il serait difficile de procéder ici par abstractions successives sans augmenter sensiblement la taille de l'exposé. C'est pourquoi j'ai adopté l'ordre inverse.

Dans le premier chapitre, j'ai cherché à préciser les notions de "machine abstraite" (déterministe ou non), d"heuristique" et de "contrôle", qui sont à la base du système de TA proposé.

Le chapitre II provient de la question suivante : dans quelle mesure et comment un système de TA peut-il "apprendre" ? J'y présente quelques résultats originaux (sur la densité dans R^1 et $R^1_{[0,1]}$, sur l'identification faible ou approximative, sur les enveloppes récursives de r.e.) tout en tentant une synthèse des différents courants qui se manifestent dans la littérature sur le sujet.

Le chapitre III est consacré aux "algogrammaires" et aux heuristiques sur ces algogrammaires. Une algogrammaire est analogue à un réseau de transitions augmenté [37] , avec essentiellement les différences suivantes : elle est destinée à analyser non pas une chaîne, mais tout un "graphe de chaînes" (différence mineure), et elle ne suppose pas l'existence d'un algorithme d'analyse particulier (différence majeure). Il existe un jeu d'opérations élémentaires au moyen desquelles on peut construire toute une classe d'heuristiques.

Au chapitre IV, je précise alors la définition de ce système prototype, et plus particulièrement les parties plus originales, en donnant la syntaxe et la sémantique (de la façon la plus intuitive possible) des langages formels ALGOG et MONIT. Le lecteur rebuté par les parties théoriques pourrait se contenter de lire les introductions et/ou conclusions de ces parties avant de passer à ce chapitre.

Bien que les problèmes liés à la sémantique soient extrêmement importants, je ne les aborde qu'au dernier chapitre, parce que je n'ai pas réalisé les propositions que j'y fais, et qu'il faudrait certainement améliorer. En particulier, il serait sans doute très intéressant d'étudier un système qui permette de traiter les structures très complexes qui, me semble-t-il, sont inévitables quand on veut utiliser de la sémantique de niveau élevé.

TABLE BIBLIOGRAPHIQUE

[1] Bar-Hillel (1964)
[2] Bar-Hillel (1971)
[3] Boitet (1972)
[4] Booth, ed. (1967)
[5] Chauché (1974)
[6] Chauché (1975)
[7] Chauché & al (1973)
[8] Chomsky (1972)
[9] Delavenay (1960)
[10] Droste (1973)
[11] Ducrot (1973)
[12] Friedman (1969)
[13] Fry (1970)
[14] Garvin (1967)
[15] Harris (1971)
[16] Harris & Hofmann (1970)
[17] King (1969)
[18] Kulagina (1973)
[19] Kulagina & Mel'tchuk (1956)
[20] Kulagina & Mel'tchuk (1971)
[21] Ljudskanov (1968)
[22] Masterman (1971)
[23] Mel'tchuk (1973b)
[24] Mel'tchuk (1974)
[25] Mel'tchuk & Ravitch (1967)
[26] Panov (1960)
[27] Schreider (1973)
[28] Sugita (1968)
[29] Tseitin (1966)
[30] Tseitin (1971)
[31] Tubach (1970)
[32] Vaissière (1971)
[33] Vauquois (1966)
[34] Vauquois (1975)
[35] Veillon (1970)
[36] Wilks (1972)
[37] Woods (1970)
[38] Yngve (1961)
[39] Yngve (1964)
[40] * ALPAC report (1966)
[41] * GEORGETOWN report (1963)
[42] * Pankowicz (1973)
[43] * RADC final report (1971)
[44] * TAUM71 (1971)

CHAPITRE I

MACHINES ABSTRAITES, HEURISTIQUES, SYSTEMES DE MACHINES

In the theory of abstract automata, we are less concerned with the design of automata to do specific tasks, and more concerned with understanding the capabilities and limitations of whole classes of automata. One might say, then, that "Automata theory [...] is the pure mathematics of computer science".

Arbib (1969)

INTRODUCTION

Comme on l'a vu dans l'introduction, on s'efforce maintenant de réaliser des systèmes de TA dont les différents modules sont des implémentations d'automates abstraits aux propriétés connues. En particulier, ces automates abstraits sont toujours "non-déterministes", puisqu'il se trouve qu'à tous les niveaux les phénomènes linguistiques présentent des ambiguïtés. Cependant, aucun système de calcul, de par sa nature même, ne peut ni ne doit être "non-déterministe". Cette contradiction s'explique si l'on remarque que les automates non-déterministes ... ne sont pas des automates. En effet, on ne définit jamais leur "fonctionnement", comme on peut le faire pour les automates abstraits déterministes. Au contraire, on considère toujours qu'ils définissent tout un ensemble de calculs, ou "arborescence des choix", auquel on associe un critère particulier pour définir une fonction ou une multifonction dont on dit, par abus de langage, qu'elle est calculée par l'automate en question. Il existe d'ailleurs des critères simples tels que les fonctions ainsi définies puissent ne pas être calculables. Par contre, d'autres critères permettent de ne définir que des fonctions calculables. Calculer une telle fonction (ou multifonction) revient alors à parcourir l'arborescence des choix tout en vérifiant le critère de définition. La plupart du temps, on utilise un parcours "canonique" ("rétrogression" ("back-track") si l'arborescence est finie, énumération des branches finies issues de la racine sinon), ce qui explique qu'on confonde les notions de définition et d'effectivité. Pour marquer cette différence, je parlerai de "systèmes définitionnels" et réserverai le terme de "machine algorithmique" pour les dispositifs déterministes. Une "heuristique d'énumération" (plus simplement "heuristique") sera alors une méthode générale de parcours des arborescences des choix associées aux différentes entrées possibles d'un système définitionnel.

Dans une première partie, je reviendrai sur la notion de machine algorithmique pour la généraliser de la façon suivante : on autorisera des "structures" (analogues aux "bandes" d'une machine de Turing) munies d'un ensemble fini de "fonctions de déplacement" (analogues aux mouvements vers la droite ou vers la gauche sur une bande), en exigeant seulement qu'on puisse coder effectivement ces structures sur $\mathbb{N}$ et que, moyennant ce codage, les fonctions associées soient récursives. On pourra alors généraliser la notion de "simulation" due à Fisher et montrer l'équivalence avec le modèle des "automates à ballon" dû à Hopcroft & Ullman.

La seconde partie précisera les notions d'arborescence des choix et de critère de définition. On donnera des exemples où les fonctions définies ne sont pas toujours calculables, puis on précisera la notion de "recherche" et on examinera certaines contraintes qu'on peut imposer pour obtenir des recherches finies quand l'arborescence des choix l'est. On définira enfin une "heuristique" (d'énumération) comme une "technique d'énumération" (au sens du Young) "constructible" en un certain sens.

Dans la troisième partie, on verra au moyen de quel type de fonctions (sur l'arborescence des choix) on peut obtenir des heuristiques réalisables au moyen d'un système de machines algorithmiques déduit du système définitionnel initial et d'une définition constructive de l'heuristique. Il faudra donc préciser ce qu'est un système de machines algorithmiques. On verra dans la suite de ce travail comment ces idées apparaissent dans la définition d'un système de TA, proposée pour illustrer ces idées.

I - MACHINES ALGORITHMIQUES

1.1. RETOUR SUR LA NOTION D'ALGORITHME

1.1.1. GENERALITES

La notion d'algorithme est aujourd'hui fondamentale tant par son importance purement théorique que par son utilisation constante en informatique. Cependant, son étude systématique est relativement récente, car elle est difficile à cerner avec rigueur[§]. De manière intuitive, on entend par "algorithme" une méthode générale pour résoudre toute une classe de problèmes, qu'on exprime au moyen d'un ensemble fini d'instructions assez détaillées pour qu'on puisse appliquer la méthode sans la comprendre. Il peut bien sûr exister plusieurs algorithmes pour résoudre le même problème (par exemple, l'extraction d'une racine carrée).

On peut préciser la notion d'algorithme en disant qu'elle suppose les propriétés informelles suivantes [48,39] :

-1- Un algorithme est donné par un ensemble I d'instructions de taille finie.

-2- Il y a un "agent de calcul" A (humain, mécanique, ou idéalisé) qui, étant donnée une entrée, peut réagir à ces instructions et effectuer les calculs.

-3- Il y a un "espace de travail" M ou l'on peut mettre ou prendre de l'information.

-4- Tout calcul est "discret", c'est-à-dire se réalise par "pas de calcul".

-5- A réagit à I de manière déterministe : une méthode aléatoire n'est pas en général algorithmique.

-6- On ne borne a priori ni la taille de I, ni celle de la mémoire, ni celle des entrées, bien que toute réalisation effective impose de telles bornes.

-7- Au contraire, l'agent de calcul A peut avoir une capacité très réduite, consistant à examiner un symbole (d'un alphabet fini) dans M, et, selon le contenu d'une "mémoire à court terme" finie et d'un ensemble fini de règles simples, à se déplacer d'un symbole au plus dans I et dans M après avoir éventuellement écrit un symbole dans M à la place du symbole examiné.

-8- Enfin, on ne borne pas a priori la longueur d'un calcul (en fonction par exemple de I et de l'entrée) : un algorithme donne un résultat sur une entrée si A s'arrête après un nombre fini de pas de calcul.

1.1.2. MACHINES DE POST, DE TURING

Pour préciser ces notions plutôt vagues, on a été amené à les formaliser au moyen de celles de "machines abstraites" (Turing (1936), Post (1936)).
Une machine de Post est constituée par une unité de contrôle à nombre fini d'états E et une tête de lecture-écriture pointant sur une bande de travail divisée en cellules portant chacune un symbole d'un alphabet fini Σ, dit "de bande", et toutes, sauf un nombre fini, un symbole de Σ distingué, le "blanc". On exprime d'habitude les règles de fonctionnement d'une telle machine au moyen d'un ensemble fini de quadruplets de l'une des quatre formes suivantes [15] :

(1) $q_i\ S_j\ S_k\ q_l$

(2) $q_i\ S_j\ R\ q_l$

(3) $q_i\ S_j\ L\ q_l$

(4) $q_i\ S_j\ q_k\ q_l$

où q désigne un état, S un symbole de bande, R et L les déplacements d'une cellule à droite et à gauche. On interprète les trois premiers types en disant que la machine M, si elle est placée dans l'état q_i et lit le symbole S_j, passe dans l'état q_l et écrit le symbole S_k ou déplace sa tête à droite ou à gauche. Le quatrième type permet d'introduire la notion d'"oracle" ou de "calcul P-relatif" : dans les mêmes conditions, M passe dans l'état q_k ou q_l selon que l'expression de bande vérifie ou non la condition P. On exige enfin qu'il n'y ait pas deux quadruplets dont les deux premières composantes soient égales, imposant ainsi le déterminisme. Ainsi, une machine de Post vérifie bien les conditions -1- à -8-. Une configuration de M est une expression $c \in \Sigma^* \times E \times \Sigma^+$[§§], elle donne donc en même temps l'expression de bande, la position de la tête et l'état. Un calcul consiste en une suite de configurations dont la première est dite initiale et toute autre est obtenue à partir de la précédente, soit $c=\sigma q_i S_j \tau$, en utilisant le (seul) quadruplet de M commençant par $q_i S_j$. Il se termine quand on arrive à une configuration terminale, c'est-à-dire sur laquelle aucun quadruplet n'est applicable.

---

§ - Brauer & Indermarck (1968) donnent un très bon survol historique du développement de cette branche des mathématiques.

§§ - Comme d'habitude, Σ^* est le monoïde libre, de neutre Λ, engendré par Σ, et $\Sigma^+=\Sigma^*-\{\Lambda\}$.

Bien d'autres modèles de machines abstraites ont été étudiés dans la littérature. Le plus célèbre est celui de Turing [59], qui diffère un peu du prédécent, et je suivrai désormais la tradition établie par Davis en parlant de "machines de Turing" pour tous les modèles analogues, qu'on obtient en modifiant la définition du "pas de calcul", en augmentant le nombre de bandes ou en généralisant leur structure [3,19,24,28].
D'autre part, on a développé des "systèmes de réécriture" [36,58] qui formalisent aussi la notion d'algorithme. Enfin, la théorie des fonctions récursives et de la λ-calculabilité ([10,54] cf. aussi [15,48] et leurs références) a précisé la notion d'effectivité pour des fonctions d'entiers. Il est remarquable de constater que tous ces modèles sont deux-à-deux équivalents, au sens où tout algorithme écrit dans l'un peut être "traduit" -de façon algorithmique d'ailleurs- dans l'autre.
On montre donc, en même temps que les classes des fonctions "calculées" par l'un et l'autre sont identiques : c'est la classe des fonctions partielles récursives. Ainsi, les notions de calculabilité, d'effectivité et de récursivité coïncident, et on admet, c'est la "thèse de Church", qu'elles formalisent exactement la notion d'algorithme.

On vient de parler des fonctions "calculées" par une machine abstraite. Ceci n'est évidemment possible que grâce à deux codages transformant un entier (ou un cortège d'entiers) en une configuration et une configuration en un entier ou un cortège d'entiers. Les plus célèbres se trouvent dans Davis (1958) : on appelle S_0=B le "blanc" et S_1=| le "un" et on code tout cortège $m=(m_1,\dots,m_k)\in\mathbb{N}^k$ par $\bar{m}=\bar{m}_1B\bar{m}_2B\dots B\bar{m}_k$

avec $\bar{m}_i=\underbrace{|\,|\,|\dots|}_{m_i+1}=1^{m_i+1}$

Inversement, si c est une expression, <c> est le nombre de "|" de c.
Bien d'autres codages peuvent être utilisés. On exige seulement que ce soient des bijections algorithmiques. Ici par exemple, on voit facilement qu'il existe deux machines de Turing (MT) réalisant ces transformations, en ayant par exemple choisi la représentation des entiers en base 1. L'idée essentielle est donc qu'on peut arithmétiser chacun de ces dispositifs abstraits, et, finalement, on peut même directement définir ce genre de "machines" à partir des entiers.

1.1.3. MACHINES CODEES

Pour illustrer ce point et introduire les définitions qui vont suivre, on peut reprendre la présentation de Lacombe (1960).

Définition 1 (Lacombe)

Une machine *est un couple* (C,Φ) *où* C *est l'ensemble des configurations et* $\Phi\in C^C$ *est le* programme. $C^*=\{x\mid\Phi(x)=x\}$ *est l'ensemble des* configurations stables. *La* fonction de marche $\Psi:C\times\mathbb{N}\to C$ *est définie récursivement par* $\begin{cases}\Psi(x,0)=x\\ \Psi(x,n+1)=\Phi(\Psi(x,n))\end{cases}$

Une machine codée M *est alors un quintuplet* $(C,\Phi,\theta_0,C_1,\theta_1)$, *où* (C,Φ) *est une machine et* (θ_0,C_1,θ_1) *un* code, *c'est-à-dire que l'on a :*

$$\begin{cases}\theta_1:\mathbb{N}\to C\\ \theta_1:C_1\to\mathbb{N}\\ C_1\subset C^*\end{cases}$$

$\theta_0(\mathbb{N})$ *est l'ensemble des* entrées *et* C_1 *celui des* sorties. *On dit que la machine est* régulière *si, pour toute entrée, on ne peut atteindre une configuration stable que dans* C_1 *et jamais dans* C^*-C_1. $D=\{m\mid M \text{ atteint un } y\in C_1\}$ *est le domaine de la* fonction partielle calculée par M, *et on posera* $\varphi(m)=\theta_1(y)$ *si* $m\in D$.

Ce formalisme très général permet de présenter les MT différemment. Avec les mêmes notations que précédemment, on posera $C=E\times F\times\mathbb{Z}$, ou $\mathbb{Z}$ est l'ensemble des entiers rationnels et $F=\Sigma^{(\mathbb{Z})}S_0$ désigne l'ensemble des applications de Z dans Σ prenant presque partout (c'est-à-dire sauf sur un nombre fini de points) la valeur "blanc" (S_0). Une configuration est donc donnée par une expression de bande, un état et une position sur la bande ($\mathbb{Z}$). Les "règles" (ou actions possibles) sont données par le contrôle fini $\alpha:E\times\Sigma\to E\times\Sigma\times\mathbb{Z}$ de composantes $\alpha_1,\alpha_2,\alpha_3$.

La fonction de transition $\Phi:(e,\xi,z)\to(e',\xi',z')$ est alors définie par :

$$e'=\alpha_1(e,\xi(z))$$

$$\xi'=\alpha_2(e,\xi(z))\ \&\ [i\neq z\Longrightarrow\xi'(i)=\xi(i)]$$

$$z'=z+\alpha_3(e,\xi(z))$$

Cette formalisation permet de séparer clairement les "ressources" de la machine : lisant un certain symbole dans un certain état, elle peut produire un symbole, changer d'état et modifier la position de la tête, en utilisant ce que nous appellerons une "fonction de déplacement" récursive. Ici, il s'agit de l'addition, et il suffit d'exiger $|\alpha_3| \leq 1$ pour se ramener à la MT classique.

1.2. TROIS TYPES DE GENERALISATION

On a défini beaucoup de types de machines abstraites. Cette prolifération n'est pas gratuite. En effet, pour un type donné d'application (théorique ou pratique), on cherche à définir un type d'automate adapté. Par exemple, une MT est définie pour posséder la structure la plus simple possible, tout en recouvrant complètement la notion intuitive d'algorithme. Mais personne ne songe à programmer directement un algorithme sous cette forme.

1.2.1. EXTENSIONS DE "MACHINES"

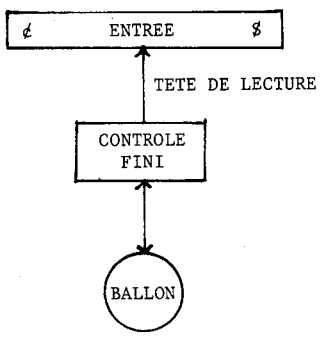
Pour faciliter certaines preuves, on a par exemple introduit les MT multibandes, chaque bande contenant éventuellement plusieurs pistes ([28] ch. VI, [3] ch. IV, [49]).
Ou bien on remplace les bandes par des "hyperbandes" de dimensions arbitraires, ou certaines par des demi-bandes, ou encore on limite les actions possibles (automates à compteurs, à piles, à stacks,...).

Un modèle beaucoup plus proche des calculateurs réels est dû à Minsky (1961 et 1967 ch. XI). Son principal avantage est "d'autoriser la description de processus très compliqués sans avoir à manipuler des séquences de symboles absurdement longues et difficiles à comprendre". Un "programme" d'une machine de Minsky est très proche d'un programme réel écrit en code machine, à la différence qu'il n'y a que très peu d'instructions (4) et que la machine dispose d'un nombre fini de "registres" de capacité infinie, au lieu d'un très grand nombre de registres de capacité finie. On trouve également toute une collection de machines analogues dans [55].

1.2.2. AUTOMATES A BALLON

Une machine de Minsky dispose de trois fonctions récursives très simples ($\lambda x[0]$, $\lambda x[x+1]$ et $\lambda x[x\dot{-}1]$). Poursuivant cette idée en la combinant avec le modèle habituel des MT, Hopcroft & Ullman (1967) ont défini les automates à ballon, dont on va rappeler la définition précise, car elle sera utilisée au paragraphe suivant. Il n'est pas inutile de commencer par une description informelle.

Un automate à ballon est constitué par : une bande d'entrée munie de deux symboles délimiteurs, "¢" et "$", un contrôle fini, une tête de lecture qui peut se déplacer dans les deux sens, et une mémoire infinie dénombrable à structure non spécifiée, le ballon. On suppose que les états du ballon sont représentés par les entiers ($\mathbb{N}$). Ceci est essentiel et permet d'utiliser, pour "abréger" la description d'un tel automate, trois fonctions récursives éventuellement très complexes.



Un mouvement de l'automate se réalise en trois temps. On commence par utiliser une fonction récursive h pour extraire du ballon une information de taille finie, analogue à la combinaison des symboles pointés par une MT multibande. Puis, en fonction de l'information obtenue, de l'état du contrôle fini et du symbole lu par la tête de lecture, on détermine (de façon nécessairement récursive) le nouvel état du contrôle et la direction de déplacement de la tête. Enfin, à partir du nouvel état du contrôle et de l'état du ballon, une fonction récursive f détermine le nouvel état du ballon. Certains états du contrôle sont finals, et une entrée est acceptée si l'automate arrive dans un état final à partir de cette entrée.

Définition 2 (Hopcroft & Ullman)

Un automate à ballon *est un 8-uplet* $A=(S,I,M,f,g,h,q_0,F)$, *où :*

- S *est* l'ensemble *fini* des états
- I *est* l'alphabet *fini* d'entrée. *Il contient* ¢ *et* $.
- $M \subset \mathbb{N}$ *est un ensemble fini d'entiers :* l'information du ballon
- $h:\mathbb{N}\to M$ *est récursive totale* ($\in R^1$)
- $g: S\times I\times M \to S\times\{-1,0,+1\}$ *non nécessairement totale ni surjective*
- $f: S\times\mathbb{N}\to\mathbb{N}$ *est partielle récursive* ($\in P^2$, *en rappelant que* $S\subset\mathbb{N}$)
- $F\subset S$ *est* l'ensemble des états finals
- $q_0\in S$ *est* l'état initial

Une configuration *de A est un quadruplet* (q,w,j,i), *où :*

- $q\in S$ *est l'état du contrôle.*
- $w\in I^*$ *s'écrit* $w=¢a_1\ldots a_n\$$, $n\geq 0$ *et* $a_k\in I-\{¢,\$\}$ *pour* $1\leq k\leq n$.
- $j\in[0,n+1]$ *est la position de la tête.*
- $i\in\mathbb{N}$ *est l'état du ballon.*

En imposant certaines contraintes à f, g et h, on peut alors définir des classes d'automates à ballon, et on retrouve par exemple les automates à piles, à stack, à stack sans effacement, imbriqués à stack, à compteur, finis, ainsi enfin que les MT. Par contre les automates linéaires bornés ne forment pas une classe. La raison en est que la définition de ce type d'automates fait intervenir une limitation globale sur la mémoire en fonction de l'entrée, et non pas seulement des contraintes locales sur les manipulations autorisées dans la mémoire. Nous retrouverons une distinction analogue à propos des critères de définition des systèmes définitionnels (2.1.).

1.2.3. DEFINITIONS ALGEBRIQUES

Pour clore ce bref tour d'horizon, rappelons enfin qu'il existe des définitions qui ne font plus du tout intervenir la notion de fonctionnement. Par exemple, Eilenberg et Wright (1970) définissent les "T-automates" (A,t), qui sont des couples formés d'une T-algèbre finie A et d'un sous-ensemble t de A. (Une "théorie" est une catégorie particulière et la notion de T-algèbre est alors très voisine de celle de "modèle" au sens de Tarski).

Dans un ouvrage plus récent [16], Eilenberg précise ce formalisme en s'abstenant volontairement d'utiliser un langage trop "catégoriel".

Définition 3 (Eilenberg)

Soit Σ *un alphabet fini. Un* Σ-automate *A est la donnée de :*

- Q, *ensemble fini des états*
- $I\subset Q$, *ensemble des états initiaux*
- $T\subset Q$, *ensemble des états terminaux*
- $E\subset Q\times\Sigma\times Q$, *ensemble des arcs*

Soient X un ensemble quelconque et Φ *une famille de relations* $\varphi: X\to X$ (*c'est-à-dire de fonctions* $\varphi: X\to 2^X$). *Une* X-machine de type Φ *M est un* Φ_1*-automate* (Q,I,T), *avec* Φ_1 *sous-ensemble fini de* Φ.

Si $c: q_0 \xrightarrow{\varphi_1} q_1 \cdots \xrightarrow{\varphi_n} q_n$ *est un* chemin *de M, on lui associe la relation composée* $|c| = \varphi_1 \ldots \varphi_n : X\to X$ (*Eilenberg note la composition dans cet ordre). Le* comportement *de M est compris comme la relation globale induite par M, soit* $|M| = \bigcup_{c\in\mathcal{C}(M)} |c|$.

Enfin, la notion de calcul *implique celle de* code *: soient* α *et* ω *les codes d'entrée et de sortie, avec* $\alpha: Y\to X$ *et* $\omega: X\to Z$ *deux relations, la relation composée* $f_M = \alpha|M|\omega$ *est dite* calculée *par M via* α *et* ω.

Ainsi, le type d'une machine, Φ[§], correspond à la classe des opérations élémentaires dans une machine abstraite classique. Remarquons ici que l'ensemble X a la forme $X=\Gamma^*\times M\times\Sigma^*$, où $Z=\Gamma^*$ et $Y=\Sigma^*$, et où M, la mémoire, peut avoir une structure arbitraire. D'autre part, la définition générale d'une machine n'impose plus le déterminisme, qui est une propriété particulière de certaines familles Φ: une X-machine de type Φ n'est pas toujours une machine algorithmique.

---

§ - Dans [16], Eilenberg donne une explication intuitive qui rejoint celles qu'on a quand on définit (cf. infra) des "systèmes" de machines : "Intuitively, one should think about Φ as a collection of components or devices each available in an unlimited supply and each capable of performing some (usually relatively simple) function. The machine $\mathcal{M}$ is then a "hook-up" of a finite number of such components.

1.3. MACHINES ALGORITHMIQUES GENERALES

Comme nous venons de le voir, on peut présenter les machines abstraites de façon plus ou moins "machinale" ou "algébrique", "concrète" ou "abstraite". Ces présentations sont équivalentes en ce sens qu'elles recouvrent bien la même notion d'algorithme. On prouve souvent ce genre d'équivalence en montrant que toute machine d'un certain type est simulable par une machine d'un autre type, et inversement. Cependant, la complexité de description et de calcul d'un algorithme n'est en général pas conservée par simulation[§], et c'est pourquoi il reste utile d'introduire différents modèles dans différents contextes où l'on cherche soit la simplicité de fonctionnement, soit l'économie de taille, soit encore l'économie de temps ou de place dans les calculs.

1.3.1. SIMULATION ET COMPLEXITE

Toutes les machines et automates abstraits (déterministes) introduits dans la littérature évoluent d'une façon discrète d'une configuration à une autre au moyen d'une fonction (relation) de marche. On utilisera ici la notation de Lacombe M=(C,Φ). Soient donc deux machines M'=(C',Φ') et M"=(C",Φ"). Quand dira-t-on que M" simule M' ? Il semble raisonnable de dire que, partant d'une configuration initiale de M', on lui associe une configuration de M", puis que M" calcule en passant par certaines configurations qui "correspondent" à celles du calcul de M', et qu'enfin on associe à la configuration finale de M" la configuration finale de M' sur cette même entrée.

Plus précisément, soient :

- $p:C"\to\{0,1\}$ une fonction caractéristique
- $h:C"\to C'$ une surjection
- $f:C'\to C"$ une injection

Notons d'autre part C(I,t) la configuration d'une machine M à l'instant t si sa configuration initiale est I=C(I,0). Soit enfin $g:C'\times\mathbb{N}\to\mathbb{N}$ définie par :

$$g(I,0)=\mu z[p(C"(f(I),z))=1]$$

$$g(I,n+1)=\mu z[z>g(t) \;\&\; p(C"(f(I),z))=1]$$

Définition 4 (Fischer)

M" simule M' relativement à une classe de fonctions *si et seulement si pour toute* $I\in C'$ *et pour tout* $t\geq 0$

(1) $C'(I,t)=h(C"(f(I),g(I,t))$

(2) $C'(I,t)\in C'_1$ *(finale)* $\Longleftrightarrow C"(f(I),g(I,t))$ *finale*

(3) $f,h,p\in \mathcal{C}$

Remarque

Cette définition est celle de Fischer (1965), légèrement modifiée. Elle suppose d'avoir déjà défini une arithmétisation des machines, puisqu'on assimile les configurations à des entiers. Cette assimilation est en fait un codage, lui-même algorithmique, qu'on peut choisir le plus simple possible, par exemple en écrivant une configuration comme un mot sur un alphabet Σ de k symboles et en lui associant l'entier décrit en base k par cette même suite[§§].

D'autre part, via par exemple le codage de Cantor, on peut associer à g une fonction à un seul argument, et $\mathcal{C}$ peut être prise comme une classe "simple" de fonctions calculables à un argument. Fischer suggère de prendre la classe des fonctions élémentaires de Kalmár.

Avant de donner quelques résultats connus, obtenus en analysant les simulations de certaines machines par d'autres, on peut, pour fixer les idées, rappeler les définitions modernes de la complexité d'un algorithme, au sens de la taille de la description ou de la difficulté d'un calcul sur une entrée. Pour plus de détails, on pourra se reporter à [5,6,21 ou 68]. Ces définitions supposent qu'on ait une énumération standard $\lambda i\varphi_i$ des fonctions récursives partielles. Cette "gödelisation"

---

§ - Voir par exemple [11,12,37], qui étudient la complexité de fonctions exprimables dans une hiérarchie de langages formels analogues aux langages de programmation.

§§ - $n(\Lambda)=0$ et $n(s)=\sum_{i=0}^{p} a_i k^i$ si $s=a_p\ldots a_0\neq\Lambda$

doit être "acceptable" au sens de Rogers (1959 et 1967 p. 41), c'est-à-dire qu'elle vérifie les théorèmes d'énumération et d'itération (s-m-n), et par suite le théorème de récursion. Ces propriétés et les axiomes suivants permettent de dériver tous les résultats standards en théorie de la complexité "abstraite" (voir aussi Werner (1973)).
Intuitivement, φ_i est la fonction calculée, dans un certain formalisme, par le i-ième programme, soit P_i. On parle donc souvent du "programme i".

Définition 5 (Blum)

Soit $\lambda i \Phi_i$ une énumération de fonctions partielles récursives ($\in P^1$). C'est une mesure de complexité *si :*

(1) $\forall i$ Dom φ_i = Dom Φ_i

(2) $\lambda ixyM(i,x,y)$ est récursive totale ($\in R^3$), où

$$M(i,x,y) = \begin{cases} 1 \text{ si } \Phi_i(x)=y \\ 0 \text{ sinon} \end{cases}$$

Cet axiome signifie que $\lambda i \Phi_i$ est une famille mesurée *de fonctions.*

D'autre part, une fonction récursive totale ($\in R^1$), notée $||$, est une mesure de taille *si :*

(3) il existe une procédure effective pour, étant donné n, énumérer l'ensemble fini des programmes j de taille n,

soit : $\exists f \in R^1 : D_{f(n)} = \{j : |j| = n\}$

Remarque

L'axiome (3) impose donc les conditions intuitives suivantes : on peut déterminer effectivement la taille de tout programme, il n'existe qu'un nombre fini de programmes de taille donnée et on peut effectivement en donner la liste. Il est surprenant de constater qu'on peut (Pager 1969, 1974) ne pas exiger l'effectivité de f et obtenir des résultats profonds : ainsi, on ne peut pas trouver effectivement un programme de taille minimale pour une table de décision à plus d'une entrée, et, toujours quelle que soit la mesure de taille choisie, il existe deux ensembles finis de programmes et une entrée $E = \{x,y\}$ ($x \in \mathbf{N}$, $y \in \{0,1\}$) telle que la table de décision obtenue en ajoutant une entrée $\{x',y'\}$ à E ait un programme de taille minimale dans l'un de ces deux ensembles sans qu'on puisse effectivement trouver lequel !

Définition 6 (Young)

Un système de programmation mesuré *est un quadruplet $(P,\varphi,\Phi,||)$, où P est une énumération de programmes, φ une gödelisation acceptable, Φ une mesure de complexité et $||$ une mesure de taille.*

Exemples

(1) Si P_i est la i-ième machine de Turing d'un certain type, on peut prendre

$\Phi_i(n)$ = nombre de pas du calcul de $\varphi_i(n)$ (complexité en temps), ou encore
$\Phi_i(n)$ = nombre de cellules utilisées dans le calcul de $\varphi_i(n)$ (complexité en place) à condition [21] d'exiger que $\Phi_i(n)$ soit indéterminé si $\varphi_i(n)$ diverge.

(2) $|i|$ peut être le nombre d'instructions d'une machine de Minsky, ou encore le nombre de caractères d'un programme Fortran. Par contre, on ne peut prendre comme mesure de taille le nombre d'instructions d'un programme Fortran [68].

(3) Si Z est une machine de Turing, on peut par exemple prendre $|Z| = nm$, où n est le nombre d'états et m le nombre de symboles de la description de Z.

1.3.2. QUELQUES RESULTATS

Rappelons qu'une machine de Turing peut à la fois écrire un symbole, déplacer sa tête de lecture et changer d'état. Une machine de Post ne peut exécuter les deux premières actions, une "D-machine" les deux extrêmes, et une "S-machine" les deux dernières en même temps. Enfin, une U-machine ne peut en exécuter qu'une à la fois. Donnons sous forme de tableau les résultats démontrés par Fischer (1965) sur la simulation d'une machine de Turing à m symboles (d'alphabet) et n états.

type	T	P	P	B	D	S	S	U
symboles	m	m	m(n+1)	m	2	$\leqslant$3m	m+1	$\leqslant$3m
états	n	$\leqslant$3n	n+1	(m+1)n	m(4n+7)	n	(m+1)n	2(m+2)n

Shannon (1956) a montré qu'il existe une MT (T-machine) universelle à deux états, Anderaa et Fischer (1967) ont d'autre part pu prouver que le problème de l'arrêt est décidable dans la classe des machines de Post à deux états. Ainsi, il ne peut exister de P-machine universelle à deux états, mais le résultat précédent montre qu'on peut en trouver une à trois états, et qu'il ne peut exister de T-machine universelle à un état. Si on prenait la mesure de taille $|Z| = nm$, on voit enfin qu'elle augmente toujours, le plus souvent de façon polynômiale, dans les simulations précédentes. Ainsi, on vérifie, ce qui est intuitivement évident, qu'il est plus facile de programmer un algorithme avec une machine dont les pas de calcul sont plus complexes.

D'autre part, Hartmanis, Lewis et Stearns [22,33,25] ont établi toute une série de résultats concernant la variation de complexité de calcul par simulation. Les mesures choisies sont $\lambda n L(n)$ et $\lambda n T(n)$, la place et le temps, où n est la taille de l'entrée (ou, de façon équivalente, l'entrée elle-même, si on a arithmétisé les modèles).
Résumons les plus caractéristiques dans le tableau suivant :

simulé par / Type	MT à k bandes (k>2)	MT à 2 bandes	MT à 1 bande	MT à 1/2 bande	Automate à stack déterministe
MT à k bandes (k⩾1)	cL(n), pour tout c>0		L(n)		
	cT(n), pour tout c>0, si $\inf_{n\to\infty} \frac{T(n)}{n} = \infty$ et si k>1	T(n) log T(n)	$T^2(n)$		
MT à 1 bande			cT(n), pour tout c>0 si $\inf_{n\to\infty} \frac{T(n)}{n^2} = \infty$	T(n)	si L(n) = n log n
MT à 1/2 bande					
Automate à stack déterministe sans effacement			L(n)=n log n		

1.3.3. MACHINES ALGORITHMIQUES GENERALES

Les différents types de MT qui apparaissent dans la littérature se distinguent par leurs capacités opératoires et par le nombre de leurs bandes. De même, les définitions les plus "algébriques" [16] ne font intervenir que des monoïdes pour structurer la "mémoire", ce qui revient à n'utiliser que la structure $\{Z, \lambda x[x+1], \lambda x[x-1]\}$. Arbib (1969) reprend dans son livre une extension, les MT à bandes multidimensionnelles, chacune pointée par une ou plusieurs têtes de lecture-écriture.

Théorème 1 (Arbib)

Soit Z une MT généralisée, à n bandes multidimensionnelles, la j-ième étant de dimension l_j et portant h_j têtes. On peut la simuler au moyen d'une MT Z' à une bande (unidimensionnelle) et à une tête.

Pour chaque bande, la simulation augmente la complexité en temps de calcul de façon linéaire en h_j et quadratique[§] si $(\exists j)[l_j > 1]$.

---

§ - Si on ne réduit pas le nombre de bandes, Hartmanis & Stearns (1965, th. 7) ont montré qu'on peut effectuer la simulation en restant dans la même classe de complexité (de temps de calcul). Si on ne fait que réduire la dimension de chaque bande, on peut en fait [25] effectuer la simulation en n log n.

Définition 7

Une "structure de travail" [§] *est un triplet* (S,F,γ) *où :*

(1) S *est un ensemble dénombrable de* "cellules"

(2) $F \subset S^S$ *est un ensemble fini d'applications, les* "fonctions de déplacement", *qui rend* S *fortement connexe* [§§] *et contient l'identité.*

(3) $\gamma : S \to \mathbb{N}$ *est un* codage *T-calculable tel que* [§§§] $\gamma_i(F_i) \subset R^1$. *C'est dire qu'il existe une arithmétisation rendant les déplacements récursifs. C'est évidemment une propriété récursivement invariante.*

Une structure (S,F,γ) *est*

- "à retour" *si* G *est un ensemble d'injections*
- "à rappel" *si* $(\exists s_0 \in S)(\exists f_0 \in F)(\forall s \in S)[s_0 \in \bigcup_{n\in\mathbb{N}} \{f_0^n(s)\}]$ [§§§§]

Dans une structure à retour, tout déplacement (composé) est inversible, dans une structure à rappel, on peut revenir à un "centre" en itérant la fonction de rappel.

Exemples de structures de travail

-1- $(\mathbb{Z}, \{-1,0,+1\})$. C'est la structure habituelle d'une bande.

-2- $(\mathbb{N}, \{-1,0,+1\})$. C'est la structure habituelle d'une demi-bande.

-3- $(\mathbb{N}^*, \{f,s,m\})$, avec
$$\begin{cases} f(x) = x.0 \\ s(x.n) = x.n+1 \\ m(x) = \begin{cases} \Lambda \text{ si } x = \Lambda \\ y \text{ si } x = y.n \end{cases} \end{cases}$$

C'est l'arborescence discrète générale ("function tree"). Elle n'est pas à retour, mais est à rappel (avec le couple (Λ,m)).

-4- $(\mathbb{N}^*, \{f,s\})$ n'est pas une structure de travail, car $\{f,s\}$ ne rend pas $\mathbb{N}^*$ connexe (on ne peut par exemple aller de 0.x à 1.y via $\{f,s\}$.

-5- $(\mathbb{Z}^2, \{h,b,d,g\})$. C'est la structure de bande à deux dimensions, avec

$$\begin{cases} h(x,y) = (x,y+1) \text{ "haut"} \\ b(x,y) = (x,y-1) \text{ "bas"} \\ d(x,y) = (x+1,y) \text{ "droite"} \\ g(x,y) = (x-1,y) \text{ "gauche"} \end{cases}$$

Cette structure est évidemment à retour, mais pas à rappel, puisqu'il faut utiliser au moins deux fonctions différentes pour joindre deux points qui ne sont ni sur la même ligne ni sur la même colonne.

-6- $(2^{(\mathbb{N})}, \{f,g,h\})$. $2^{(\mathbb{N})}$ désigne l'ensemble des parties finies de $\mathbb{N}$. On rappelle que, si $A=\{x_1,\ldots,x_n\} \subset 2^{(\mathbb{N})}$, $x=\sum_{i=1}^{n} 2^{x_i}$ est l'indice canonique de A, et qu'on note $A=D_x$, avec la convention que $\emptyset=D_0$. Alors :

$$\begin{cases} f(x) = x-2^{\min(x)} \\ g(x) = x+2^{\max(x)+1} \\ h(x) = f(x)+2^{\min\, y\leqslant x[\mathrm{app}(x,y)=0]} \end{cases} \qquad \text{avec} \qquad \begin{cases} \min(x) = \max\, y\leqslant x[2^y \mid x] \\ \max(x) = \max\, y\leqslant x[f^y(x)\neq 0] \\ \mathrm{app}(x,y) = \begin{cases} 1 \text{ si } 2\mid [x\mid 2^{y\dot{-}1}] \\ 0 \text{ sinon} \end{cases} \end{cases}$$

$f(0)=h(0)=0$ $\qquad$ $(\mathrm{app}(x,y)=1 \iff y\in D_x)$

---

§ - La notion habituelle de structure (avec lois internes et relations) est donc simplifiée pour obtenir une notion opératoire.

§§ - C'est-à-dire que le monoïde engendré (par composition) opère transitivement sur S_i.

§§§ - $\gamma_i(F_i)$ est une notation abrégée (fonctorielle) pour $\gamma_i F_i \gamma_i^{-1}$. On l'utilisera par la suite.

§§§§ - C'est moins que d'exiger $s_0=\lim_{n\to\infty} f_0^n(s)$, ce qui rendrait la structure tout-à-fait analogue à un ensemble "retraçable" (cf. définition 26).

A partir de D_x, on obtient donc $D_{f(x)}$, en supprimant le plus petit élément, $D_{g(x)}$ en ajoutant le successeur du plus grand élément de D_x, et $D_{h(x)}$ en remplaçant le plus petit élément par son premier successeur hors de D_x. Cette structure n'est pas à retour, mais à rappel (avec le couple $(\emptyset,f)$). De plus, tout D_x est constructible au moyen de $\{h,g\}$ à partir de $\emptyset$, ce qui assure la connexité : si $A = \{x_1,\ldots,x_n\} = D_x$, où $i>j \Longrightarrow x_i>x_j$, $x = h^{x_n-(x_{n-1}+1)} g \ldots h^{x_2-(x_1+1)} g\, h(0)$

-7- Citons enfin la structure de "stack (auto-)enchâssé" introduite par Aho (1969).

Définition 8

Une machine algorithmique générale *(G-machine) est un couple $M=(C,\Phi)$ où :*

(1) $C=E\times\prod_{i\in I}(B_i\times S_i^{h_i})$ *est l'ensemble des* configurations, *avec*

- *E fini ensemble des* états du contrôle
- *I fini indexant les* structures de travail *et leurs* alphabets *: pour tout i,*
  - A_i *est un alphabet fini, d'élément distingué* a_i°, *le* "blanc".
  - (S_i,F_i,γ_i) *est une structure de travail. On posera* $F_i=\{f_{ij}\}_{j\in}$.
  - h_i *fini est le* nombre de têtes *sur* S_i *et* $B_i=A_i^{(S_i)}\cdot a_i^\circ$. B_i *est donc l'ensemble des* "expressions de structures", *p.p. blanches.*

(2) $\Phi:C\to C$ *est la* fonction de transition *; elle est déterminée de manière unique par le* contrôle fini $\alpha:E\times\prod_{i\in I}A_i^{h_i}\to E\times\prod_{i\in I}A_i^{h_i}\times\prod_{i\in I}F_i^{h_i}$ *de composantes* $\alpha_1,\alpha_2,\alpha_3$. *Si* $c=(e,\prod_{i\in I}(b_i,s_i))$, $\Phi(c)=c'=(e',\prod_{i\in I}(b'_i,s'_i))$, *où* $s_i=\prod_{j=1}^{h_i}s_{ij}$, *avec :*

$$\begin{cases} e' = \alpha_1\left(e,\ \prod_{i\in I}\prod_{j=1}^{h_i} b_i(s_{ij})\right) \\ \prod_{i\in I}\prod_{j=1}^{h_i} b'_i(s_{ij}) = \alpha_2\left(e,\ \prod_{i\in I}\prod_{j=1}^{h_i} b_i(s_{ij})\right) \\ b'_i(t_{ij}) = b_i(t_{ij}) \text{ si } t_{ij}\neq s_{ij} \\ \prod_{i\in I}\prod_{j=1}^{h_i} s'_{ij} = \alpha_3\left(e,\ \prod_{i\in I}\prod_{j=1}^{h_i} b_i(s_{ij})\right)\left(\prod_{i\in I}\prod_{j=1}^{h_i} s_{ij}\right) \end{cases}$$

(3) On appellera $\theta_0:\mathbb{N}\to C$ *le codage déductible des* $(\gamma_i)_{i\in I}$.

Une G-machine est "à retour" *(resp.* "à rappel"*) si toutes ses structures le sont.*

Une G-machine est "simple" *si elle a exactement une tête par structure.*

Une G-machine est "normale" *si chacune de ses structures est à rappel ou à retour.*

Pour (3), tout A_i s'identifie (via une fonction η_i) à une partie finie de $\mathbb{N}$, ainsi que E, et on a le diagramme : $B_i=A_i^{(S_i)}\cdot a_i^\circ \xrightarrow{(\gamma_i,\eta_i)} \eta_i(A_i)^{(\gamma_i(S_i))}\cdot\eta_i(a_i^\circ) \xrightarrow{\mu_i} \mathbb{N}$, d'où $B_i \xrightarrow{\lambda_i} \mathbb{N}$. Si $|A_i|=k_i$, μ_i est une énumération standard de $k_i^{(\mathbb{N})}$.

Alors $\theta_0=\tilde{\theta}_0^{-1}$, avec $\tilde{\theta}_0(c)=\langle e,\langle(\langle\lambda_i(b_i),\gamma_i(S_i)\rangle)_{i\in I}\rangle\rangle$

Ainsi, le fonctionnement d'une G-machine est tout-à-fait analogue à celui d'une MT : dans un certain état de contrôle fini, M lit h_i symboles sur sa i-ième structure de travail et détermine l'état suivant du contrôle, les symboles à écrire aux emplacements pointés par les têtes et le mouvement des têtes. La complexité d'une action élémentaire est cependant supérieure en général à celle d'une action élémentaire d'une MT, puisque les $\gamma_i(f_{ij})$ sont des fonctions récursives éventuellement complexes. En ce sens, on inclut dans le modèle la complexité du codage des structures de données choisies, mais, par contraste avec le modèle des automates-ballons, on limite la complexité de chaque manipulation élémentaire de la mémoire.

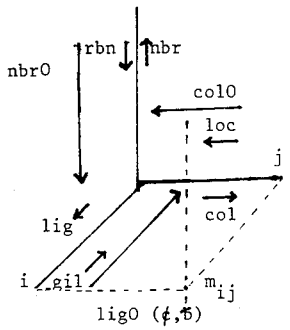
Cette définition semble bien formaliser la notion de "module" dans un système informatique de TA "intégré"[§]. En effet, on programme seulement les opérations élémentaires du module (les f_{ij} et les écritures de symboles a_{ij}), et la table de transitions (Φ) est fournie à partir de la compilation de données linguistiques écrites dans un certain langage formel au moyen d'un "automate-constructeur" [9] (l'exact analogue d'une fonction s_n^m - [15], p. 146).
D'autre part, il est plus agréable de pouvoir utiliser des structures de travail analogues aux structures algébriques qui interviennent dans les problèmes traités ("arborescences de choix", "graphes de chaînes", "multigraphes" particuliers,...).

Exemples de G-machines

-1- Les restrictions usuelles (sur α) permettent de définir les automates à piles, à stacks, à compteurs, etc. comme des G-machines à bandes ou demi-bandes.

-2- Donnons une G-machine qui calcule le produit de deux matrices d'entiers. On utilisera

- six états $E = \{e_0, e_1, e_2, e_3, e_4, e_f\}$
- trois structures identiques $S_1=S_2=S_3=\mathbb{N}^3 x \{lig, lig0, gil, col, col0, loc, nbr, nbr0, rbn, 0\}^2$



avec les notations évidentes :

$$\begin{cases} lig(x,y,z)=(x+1,y,z) & gil(x,y,z)=(x\dot{-}1,y,z) & lig0(x,y,z)=(0,y,z) \\ col(x,y,z)=(x,y+1,z) & loc(x,y,z)=(x,y\dot{-}1,z) & col0(x,y,z)=(x,0,z) \\ nbr(x,y,x)=(x,y,z+1) & rbn(x,y,z)=(x,y,z\dot{-}1) & nbr0(x,y,z)=(x,y,0) \end{cases}$$

L'ensemble F consistera en ces fonctions, leurs compositions binaires et l'identité (notée 0).

- trois alphabets identiques $A_1=A_2=A_3=\{0,1,ƀ,ɸ\}x\{1,b\}$. $(ƀ,ƀ)$ est le blanc.

Chaque élément d'une matrice $\mathfrak{M}=(m_{ij})$ sera codé dans cette structure de la façon suivante : $b(i,j,0)=(ɸ,ƀ)$.

La colonne (i,j) porte la représentation binaire de m_{ij}, dans l'ordre des poids croissants : $m_{ij}=\sum_{k\geq 1} \pi_1(b(i,j,k))2^{k-1}$. La seconde composante de l'alphabet sert à marquer la position initiale de la tête quand on fait une addition (on prend le produit classique $p_{ij}=\sum_k m_{ik}n_{kj}$, et on arrête le calcul dès qu'un des termes est non défini (blanc), de sorte qu'on calcule en fait le produit des deux plus grandes matrices composables contenues (à gauche) dans $\mathfrak{M}$ et $\mathfrak{N}$).
On abrégera l'écriture en ne notant que la première composante des symboles sur S_1 et S_2, la seconde valant toujours ƀ.

D'où la fonction de transition :

§ - cf. ch. 0.

$E \times (A_1 x A_2 x A_3)$	$E \times (A_1 x A_2 x A_3) x (F_1 x F_2 x F_3)$	Commentaires
$e_0 \begin{pmatrix} \mathfrak{b} \\ \mathfrak{b} \\ x \end{pmatrix}$	$e_f \begin{pmatrix} \mathfrak{b} \\ \mathfrak{b} \\ x \end{pmatrix} \begin{pmatrix} col0 \circ lig0 \\ col0 \circ lig0 \\ col0 \circ lig0 \end{pmatrix}$	
$e_0 \begin{pmatrix} ¢ \\ ¢ \\ \mathfrak{b}\,\mathfrak{b} \end{pmatrix}$	$e_1 \begin{pmatrix} ¢ \\ ¢ \\ ¢\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} nbr \\ nbr \\ nbr \end{pmatrix}$	Début du calcul d'un p_{ij}
$e_0 \begin{pmatrix} x \\ \mathfrak{b} \\ \mathfrak{b}\,\mathfrak{b} \end{pmatrix}$	$e_0 \begin{pmatrix} x \\ \mathfrak{b} \\ \mathfrak{b}\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} lig \circ lig0 \\ col0 \\ lig \circ lig0 \end{pmatrix}$ avec $x \in \{0,1\} x \{\mathfrak{b}\}$	Début du calcul de $p_{i+1\ 1}$
$e_0 \begin{pmatrix} \mathfrak{b} \\ x \\ \mathfrak{b}\,\mathfrak{b} \end{pmatrix}$	$e_f \begin{pmatrix} \mathfrak{b} \\ x \\ \mathfrak{b}\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} col0 \circ lig0 \\ col0 \circ lig0 \\ col0 \circ lig0 \end{pmatrix}$ " "	
$e_1 \begin{pmatrix} 0 \\ x \\ y\,\mathfrak{b} \end{pmatrix}$	$e_1 \begin{pmatrix} 0 \\ x \\ y \end{pmatrix} \begin{pmatrix} nbr \\ 0 \\ nbr \end{pmatrix}$ " "	Décalage dans le calcul de $m_{ik}\ n_{kj}$
$e_1 \begin{pmatrix} 1 \\ x \\ y\,\mathfrak{b} \end{pmatrix}$	$e_2 \begin{pmatrix} 1 \\ x \\ x+y\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ nbr \\ nbr \end{pmatrix}$ avec $xy=0$	Début de l'addition $n_{kj}+p_{ij}$
$e_1 \begin{pmatrix} 1 \\ 1 \\ 1\,\mathfrak{b} \end{pmatrix}$	$e_3 \begin{pmatrix} 1 \\ 1 \\ 0\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ nbr \\ nbr \end{pmatrix}$	Avec retenue
$e_2 \begin{pmatrix} 1 \\ x \\ y\,\mathfrak{b} \end{pmatrix}$	$e_2 \begin{pmatrix} 1 \\ x \\ x+y\ \mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ nbr \\ nbr \end{pmatrix}$ avec $xy=0$	
$e_2 \begin{pmatrix} 1 \\ 1 \\ 1\,\mathfrak{b} \end{pmatrix}$	$e_3 \begin{pmatrix} 1 \\ 1 \\ 0\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ nbr \\ nbr \end{pmatrix}$	
$e_2 \begin{pmatrix} 1 \\ \mathfrak{b} \\ x\,\mathfrak{b} \end{pmatrix}$	$e_4 \begin{pmatrix} 1 \\ b \\ x\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ nbr0 \\ nbr0 \end{pmatrix}$	Fin de l'addition $n_{kj}+p_{ij}$
$e_3 \begin{pmatrix} 1 \\ \mathfrak{b} \\ x\,\mathfrak{b} \end{pmatrix}$	$e_4 \begin{pmatrix} 1 \\ \mathfrak{b} \\ 1\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ nbr0 \\ nbr0 \end{pmatrix}$ avec $x \in \{0,\mathfrak{b}\}$	Fin de l'addition $n_{kj}+p_{ij}$
$e_3 \begin{pmatrix} 1 \\ \mathfrak{b} \\ 1\,\mathfrak{b} \end{pmatrix}$	$e_3 \begin{pmatrix} 1 \\ \mathfrak{b} \\ 0\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ nbr0 \\ nbr0 \end{pmatrix}$	Propagation de la dernière retenue
$e_4 \begin{pmatrix} 1 \\ ¢ \\ y\,\mathfrak{b} \end{pmatrix}$	$e_4 \begin{pmatrix} 1 \\ ¢ \\ y\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ rbn \end{pmatrix}$	Retour pour trouver le début de l'addition
$e_4 \begin{pmatrix} 1 \\ ¢ \\ y\,1 \end{pmatrix}$	$e_1 \begin{pmatrix} 1 \\ ¢ \\ y\,0 \end{pmatrix} \begin{pmatrix} 0 \\ nbr \\ nbr \end{pmatrix}$	
$e_1 \begin{pmatrix} \mathfrak{b} \\ x \\ y\,\mathfrak{b} \end{pmatrix}$	$e_1 \begin{pmatrix} \mathfrak{b} \\ x \\ y\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} nbr0 \\ nbr0 \\ nbr0 \end{pmatrix}$	Fin du calcul de $m_{ik}\ n_{kj}$
$e_1 \begin{pmatrix} ¢ \\ ¢ \\ ¢\,\mathfrak{b} \end{pmatrix}$	$e_1 \begin{pmatrix} ¢ \\ ¢ \\ ¢\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} col \\ lig \\ 0 \end{pmatrix}$	Début du calcul de $m_{ik+1}\ n_{k+1\ j}$
$e_1 \begin{pmatrix} x \\ y \\ ¢\,\mathfrak{b} \end{pmatrix}$	$e_0 \begin{pmatrix} x \\ y \\ ¢\,\mathfrak{b} \end{pmatrix} \begin{pmatrix} col0 \\ col \circ lig0 \\ col \end{pmatrix}$ avec x ou y = $\mathfrak{b}$	Passage au calcul de $p_{i\ j+1}$

Remarque

Comme dans le cas des MT généralisées, il peut y avoir conflit, pour une G-machine non simple, si deux têtes placées au même endroit doivent écrire deux symboles différents. Dans ce cas, on conviendra de donner la priorité à la tête du plus petit numéro.

Remarque

On utilisera effectivement plusieurs têtes par structure dans le module associé à l'"algogrammaire" (cf III). Les lemmes suivants montreront qu'on peut simuler une G-machine normale par une machine simple, mais de façon déjà complexe. Ainsi, l'utilisation de plusieurs têtes peut réduire la complexité d'une G-machine réalisant un certain algorithme.

Remarque

Dans un travail récent, Savitch (1973) a introduit les "automates reconnaisseurs de labyrinthes" ou MRA (maze recognizing automaton). Un MRA a comme entrée (donnée) une structure qui est un graphe binaire fini (tout point est relié à exactement deux autres points). Même dans ce cas simple, les propriétés d'un MRA dépendent "de façon critique" d'une propriété analogue à la connexité : il faut qu'un MRA puisse parcourir le labyrinthe entier, indépendamment des arcs présents. Ceci correspond à l'introduction d'une nouvelle fonction sur la structure, dans ce cas précis à une permutation circulaire sur les noeuds.

Lemme 1 (Réduction des têtes)

Toute G-machine à retour est simulable par une G-machine simple et à retour, donc normale.

La démonstration est analogue à celle d'Arbib (1969) pour le cas des MT multitêtes. Il suffit évidemment de montrer qu'on peut simuler une G-machine M à h têtes sur S par une G-machine M' possédant en plus une demi-bande S', où S et S' sont simples pour M'. L'alphabet de S sera $Ax[1,2^h]$ et celui de S' sera $F\cup\{\Lambda\}$, où $\Lambda\in F$. On distinguera un point particulier $s_0\in S$, et toute configuration c de M sera codée par une configuration de M' où l'unique tête de S pointera sur s_0, tandis que l'alphabet choisi codera les informations sur le contenu (A) de la structure et la position des h têtes[§]. M' disposera dans son contrôle fini de h registres destinés à contenir les symboles et les fonctions associés aux h têtes de M sur S et d'un registre ρ à 5 valeurs correspondant aux phases de la simulation. Ainsi, $E'=Ex(A\cup\{*\}xF\cup\{\Lambda\})^h x\rho$, où $*\notin A$.

La simulation d'un pas du calcul de M s'effectue de la manière suivante :

- Phase 1 :

$\rho=1$ et M' utilise S' pour localiser les têtes sur la configuration associée de M, en utilisant S' pour énumérer tous les chemins possibles dans S issus de s_0, par longueur croissante. Un chemin de longueur k est en effet représentable par un élément de F^k, et inversement. L'essai des chemins en utilisant cette énumération est facilité par l'injectivité des f_{ij}. M' remplit ses registres avec les symboles associés aux têtes de M. Quand tous les registres sont remplis, M' a toute l'information nécessaire pour passer à la

- Phase 2 :

$\rho=2$ et M' retourne en s_0 (en effaçant S'). M' passe à la

- Phase 3 :

$\rho=3$ et M' commence par mettre dans ses registres les nouveaux symboles et les déplacements produits par le contrôle fini de M. M' énumère de nouveau les chemins issus de s_0 pour "distribuer" ce résultat. Quand elle rencontre un symbole qui code la présence d'une ou plusieurs têtes de M, elle écrit le symbole (de A) du registre associé à la tête du plus petit numéro, soit k, sur la cellule de S et dans les registres associés aux autres têtes codées sur cette cellule. Elle empile alors sur S' le déplacement f_j contenu dans le k-ième registre et passe à la phase 4, qui est plutôt une sous-phase. Quand les h registres contiennent $(*,\Lambda)$, elle passe à la phase 5.

---

§ - En effet, l'alphabet $[1,2^h]$ permet de coder $\mathcal{P}(\{t_1,\ldots,t_h\})$. La deuxième composante indique donc quelles têtes pointaient sur la cellule considérée.

- Phase 4 :

$\rho=4$ et M' exécute le déplacement f_j lu sur S', modifie le symbole de la cellule (de S) atteinte pour coder la présence de la k-ième tête et revient à son point de départ en dépilant f_j, en mettant $(*,\Lambda)$ dans son k-ième registre et en passant à la phase 3.

- Phase 5 :

$\rho=5$ et M' revient en s_0 (en effaçant S'), avant de repasser à la phase 1. A ce moment, M' se trouve dans la configuration qui code $\Phi_M(c)$.

On pourrait bien sûr donner une démonstration entièrement formelle. On en trouvera une pour le lemme suivant, qui est un peu plus compliqué.

En itérant cette transformation pour les n structures de M, on obtient finalement une machine $M^{(n)}$ à 2n structures, toutes simples et à retour, puisque les demi-bandes introduites le sont. ⊠

Lemme 2

Toute G-machine à rappel est simulable par une G-machine simple à rappel, donc normale.

La démonstration est analogue à celle du lemme 1. Soit (S,F) une structure à rappel (via(s_0,f_0)). M' contient :

- S munie de l'alphabet $A'=Ax[1,2^h]x\{\mathfrak{b},\mathfrak{c}\}$ $(a'_0=(a_0,1,\mathfrak{b}))$
- S', demi-bande simple d'alphabet $A''=F\cup\{\Lambda\}$, $\Lambda\notin F$.
- $E'=Ex(A\cup\{*,+\}xF\cup\{\Lambda\})^h x\rho$, où $+,*\notin A$ et $\rho=\{n,v,r,s\}x\{1,2,3,4,5\}$.

La première composante de ρ indique le "mode de fonctionnement" de M' (neutre, vidage ou remplissage des registres, simulation d'un pas), et la seconde indique le "mode de parcours" de S'. Nous allons effectivement construire M' en donnant sa fonction de transition α, et en utilisant quelques abréviations :

- R_i est le i-ième registre, et donc $E'=Ex\prod_{i=1}^{h} R_i x\rho$
- "R plein" $\equiv (\forall i\in[1,h])[R_i\neq(*,\Lambda)]$
- O désigne les neutres de F et F', et D (G) les mouvements à droite ou à gauche sur S'. De plus, on suppose que $F=\{f_0,\ldots,f_n\}$.
- $Symb(R)=\prod_{i=1}^{h}\pi_1(R_i)$
- $Sim(e,R) \equiv (\forall i\in[1,h])[R_i:=(\pi_i\circ\alpha_2^S(e,Symb(R)),\pi_i\circ\alpha_3^S(e,Symb(R)))]$
  (α^S désigne la restriction de α à S)
- On n'explicitera pas la correspondance biunivoque $[1,2^h]\to\mathcal{P}([1,h])$ et on notera directement l'ensemble des têtes codées par t ou son développement $t=\{t_1,\ldots,t_{m_t}\}$.
- $Min(t)=Max(0,\mu t_i[t_i\in t \;\&\; \pi_2(R_{t_i})\neq\Lambda])$
- $Plus(t)=\mu t_i[t_i\in t \;\&\; \pi_1(R_{t_i})=+]$
- $Symb(t)=\pi_1(R_{Min(t)})$
- $Remp(t,a)\equiv(\forall t_i\in t)[\pi_2(R_{t_i})\neq\Lambda \Longrightarrow \pi_1(R_{t_i}):=a \;\&\; \pi_1(R_{Min(t)}):=*]$

Une configuration de M est donc codée dans M' par une configuration où la tête de S pointe sur s_0 et celle de S' sur la première cellule, où S' est entièrement blanche et S ne contient que des symboles du type $(a,t,\mathfrak{b})$, enfin où l'état est $e'=(e,(*,\Lambda)^h,(n,1))$. Avec ces conventions, il est aisé de voir que M' simule bien M, en définissant la fonction de transition α' au moyen de la table suivante. D'autre part, il est évident que cette simulation vérifie bien la définition 4, avec des fonctions p,h,f très simples.

E'	A'xA"	E'	A'xA"	FxF'	Commentaires
e R vide n 1	a t ƀ ƀ	e Remp(t,a) r 2	a t ¢ Λ	O D	Marquage de s_0 et de la première cellule de S'.
e R non plein r 2	a t ƀ f	e Remp(t,a) r 2	a t ƀ f	f D	M' suit le chemin indiqué sur S'.
e R non plein r 2	a t ƀ ƀ	e Remp(t,a) n 3	a t ƀ ƀ	O G	M' arrive au bout du chemin noté sur S'.
e R non plein n 3	a t ƀ a"	e R v 3	a t ƀ a"	O O	M' n'a pas trouvé toutes les têtes de M sur ce chemin.
e R non plein v 3 ou 4	a t ƀ $f_i (i<n)$	e R v 5	a t ƀ f_{i+1}	O G	M fabrique le chemin suivant.
e R non plein v 3 ou 4	a t ƀ f_n	e R v 4	a t ƀ f_0	O G	Transmission de la "retenue" (pour le chemin suivant)
e R non plein v 4	a t ƀ Λ	e R v 2	a t ƀ Λ	O D	
e R non plein v 2	a t ƀ f_0	e R v 2	a t ƀ f_0	O D	Propagation vers la droite de S'
e R non plein v 2	a t ƀ ƀ	e R v 5	a t ƀ f_0	O G	Premier chemin de longueur supérieure.
e R non plein $\begin{cases} v \\ s \end{cases}$ 5	a t ƀ f	e R $\begin{cases} v \\ s \end{cases}$ 5	a t ƀ f	O G	Retour à la première cellule de S'
e R non plein $\begin{cases} v \\ s \end{cases}$ 5	a t ƀ Λ	e R $\begin{cases} v \\ s \end{cases}$ 5	a t ƀ Λ	f_0 O	Retour à s_0
e R non plein v 5	a t ¢ Λ	α_1^S(e, Symb(R)) Sim(e,R) s 2	a t ¢ Λ	O D	Chargement dans R des actions à effectuer.
e R non vide s 2	a t x a"	e Remp(t,Symb(t)) s 3	Symb(t) t x a"	$\pi_2(R_{Min(t)})$ O	Si Min(t)≠0 : M' simule la première tête.
e R non vide s 2	a t x f	e R s 2	a t x f	f O	Si Min(t)=0 : M' cherche la prochaine tête à simuler.
e R s 3	a t x a"	e $R_{Plus(t)}$:=(*,Λ) s 5	a tUR a"Plus(t)x	O O	Simulation du mouvement d'une tête.
e R non vide s 5	a t ¢ Λ	e R s 2	a t ¢ Λ	O D	Début de la simulation d'une autre tête.
e R vide s 5	a t ¢ Λ	e R n 2	a t ¢ Λ	O D	Début d'effacement de S'.
e R vide n 2	a t ¢ f	e R n 2	a t ¢ ƀ	O D	Effacement de S'.
e R vide n 2	a t ¢ ƀ	e R n 5	a t ¢ ƀ	O G	Retour à la première cellule de S'.
E R vide n 5	a t ¢ Λ	e R n 1	a t ¢ ƀ	O O	Fin de la simulation d'un pas de M.

Théorème 2

Toute G-machine normale est simulable par une G-machine simple et normale.

C'est une conclusion immédiate des deux démonstrations précédentes. ⊠

Théorème 3

Toute G-machine est simulable par une machine de Post.

Je n'ai pas pu trouver de démonstration directe pour ce cas plus général. Il semble d'ailleurs que toutes les preuves directes de simulation de machines à structures généralisées (bandes multidimensionnelles ou labyrinthes) utilise le fait qu'on peut revenir à un point distingué de la structure [3,23]. On fera donc un détour en déduisant ce résultat du théorème 4 suivant, dont il est une conséquence immédiate. ⊠

Théorème 4

Toute G-machine est simulable par un automate à ballon et inversement.

Montrons d'abord le

Lemme 3

Tout automate à ballon est simulable par une MT à trois bandes, et donc par une MT à une bande (cf 1.3.2 ou le lemme précédent).

Reprenons les notations du 1.2.2. pour l'automate à ballon $\mathcal{B}$ à simuler, avec une exception : Q désignera l'ensemble des états de $\mathcal{B}$. La machine $\mathcal{A}$ utilisera trois bandes :

S_1 correspond à la bande d'entrée de $\mathcal{B}$.

S_2 sert de bande de travail.

S_3 contiendra l'information finie obtenue (par h) à partir du ballon.

On prendra pour alphabets $A_1=I$ et $A_2=A_3=M=[1,m]$. Les états de $\mathcal{A}$ sont les éléments de $E=Q\cup(Q_h\times Q)\cup(Q_f\times Q)$, où Q_h est l'ensemble des états d'une MT à deux bandes qui calcule h, soit $\mathcal{H}$, et Q_f celui d'une MT à une bande, soit $\mathcal{F}$, qui calcule f. $\mathcal{H}$ a S_2 pour bande d'entrée et s'arrête, pour une configuration initiale donnée sur S_2, avec un seul symbole non blanc, pointé par t_3, sur S_3. $Q_h^t\subset Q_h$ est l'ensemble des états terminaux de $\mathcal{H}$ et $Q_f^t\subset Q_f$ celui de $\mathcal{F}$; q_h^o et q_f^o désignent les états initiaux de ces machines.

La fonction de transition α de $\mathcal{A}$ est alors donnée de manière évidente par :

$$\alpha\left(e,\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix}\right)=\begin{cases}\left(\left(q_h^o,e\right),\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix},\begin{pmatrix}0\\0\\0\end{pmatrix}\right) & \text{si } e\in Q\\ \left(\left(q_h^j,e'\right),\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix},\begin{pmatrix}0\\0\\D\end{pmatrix}\right) & \text{si } e=(q_h^i,e')\in(Q_h-Q_h^t)\times Q \text{ et } q_h^i a_3 a'_3 q_h^j D\in\mathcal{H}\\ \left(\left(q_f^o,e''\right),\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix},\begin{pmatrix}D\\0\\0\end{pmatrix}\right) & \text{si } e=(q_h^t,e')\in Q_h^t\times Q \text{ et } g(e',a_1,a_3)=(e'',D)\\ \left(e',\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix},\begin{pmatrix}0\\D\\0\end{pmatrix}\right) & \text{si } e\in Q_f-Q_f^t \text{ et } ea_2a'_2e'D\in\mathcal{F}\\ \left(e',\begin{pmatrix}a_1\\a_2\\a_3\end{pmatrix},\begin{pmatrix}0\\0\\0\end{pmatrix}\right) & \text{si } e=(q,e')\in Q_f^t\times Q\end{cases}$$

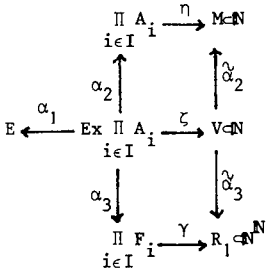
Enfin, une MT à trois bandes est évidemment une G-machine, ce qui démontre la seconde partie du lemme. ⊠

Pour l'autre sens, il suffit de prouver le

Lemme 4

Toute G-machine est simulable par un automate à ballon.

On prendra M=[1,m], avec $m=\left|\prod_{i\in I} A_i\right|$ et M simple. La démonstration se généralise de manière évidente, mais plus lourde, au cas M non-simple. On considérera que les applications η,ζ,γ du diagramme suivant sont les codages liés à $\mathcal{A}$.



- α_2 et α_3 sont toujours les deux dernières composantes du contrôle fini.
- η correspond à la numérotation triviale de $\prod_{i\in I} A_i$.
- $\zeta=\lambda xy[<x,\eta(y)>]$ où $\lambda xy[<x,y>]$ est un codage (récursif primitif) de $\mathbb{N}^2$ sur ℕ, d'inverse (Π_1,Π_2), et dont on exige que $0=<0,0>$. Par exemple, $<x,y>=2^x(2y+1)-1$.

La simulation d'un calcul de $\mathcal{A}$ à partir d'une configuration initiale $c\in C$ se fera de la façon suivante. Il existe un codage récursif de C sur ℕ, soit δ. $\mathcal{B}$ commencera son calcul avec ¢ $\underbrace{|\,|\ldots|\,|}_{\delta(c)+1}$ \$ sur sa bande d'entrée, et 0 dans le ballon. $\mathcal{B}$ mettra alors δ(c) en réserve dans le ballon).
Pour ceci, un entier du ballon codera (par <>) un couple d'entiers, dont le premier est nul pendant la transmission de δ(c) et vaut 1 ensuite, tandis que le second contiendra le code d'une configuration de $\mathcal{A}$. $\mathcal{B}$ peut ensuite simuler chaque pas du calcul de $\mathcal{A}$ en extrayant (par h) l'information nécessaire du code de la configuration de $\mathcal{A}$, puis (par g) en déterminant l'état suivant de $\mathcal{A}$, tout en se souvenant de l'état actuel, et enfin en réalisant (par f) la transformation du ballon donnant le code de la nouvelle configuration de $\mathcal{A}$, où les symboles ont été écrits et les têtes bougées. Il faut maintenant construire l'ensemble Q des états et les fonctions f,g,h.

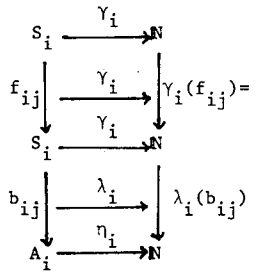
Donnons d'abord quelques notations pour les codages utilisés, en nous aidant des diagrammes ci-dessous. Si $\mathbb{N}^\infty=\bigcup_{n\in\mathbb{N}}\mathbb{N}^n$, $\tau^*:\mathbb{N}^\infty\to\mathbb{N}$ est un codage récursif primitif défini par :

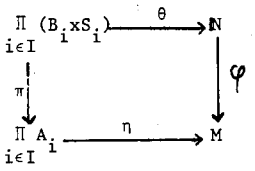
$\tau^*(\emptyset)=0$

$\tau^*(n_1,\ldots,n_k)=<k-1,<n_1<\ldots,n_k>>\ldots>+1$.

On notera encore $<n_1,\ldots,n_k>=\tau^*(n_1,\ldots,n_k)$.

Rappelons que $B_i=A_i^{(S_i)}a_i^o$





On posera donc, de manière évidente :

- si $b_{ij}=(s_{ijk},a_{ijk})_{k\in\mathbb{N}} \in B_i$, $\lambda_i(b_{ij})=\tau^*((<\gamma_i(s_{ijk}),\eta_i(a_{ijk})>)_{\{k\in\mathbb{N}\mid a_{ijk}\neq a_i^\circ\}})$

(on code la suite finie des éléments non blancs de S_i).

- si $(b_{ij},s_{ij})_{i\in I} \in \prod_{i\in I}(B_i\times S_i)$, $\theta((b_{ij},s_{ij})_{i\in I})=\tau^*((<\lambda_i(b_{ij}),\gamma_i(s_{ij})>)_{i\in I})$

(on code à la fois la position des têtes et les expressions dans chaque structure).

- si $(b_{ij},s_{ij})_{i\in I} \in \prod_{i\in I}(B_i\times S_i)$, $\pi((b_{ij},s_{ij})_{i\in I})=(b_{ij}(s_{ij}))_{i\in I}$. C'est l'ensemble des symboles pointés.

- $\varphi=\eta\pi\theta^{-1}$, $\gamma_i(f_{ij})=\gamma_i\, f_{ij}\, \gamma_i^{-1}$ et $\lambda_i(b_{ij})=\eta_i b_{ij}\gamma_i^{-1}$

Définition de Q,h,g et f

On prendra $Q=\{q_0,q_1,q_2\}\cup E^2\cup F$, avec

$F=\{(e,m)\mid \alpha(e,\eta^{-1}(m))\downarrow\}\subset E\times M$ l'ensemble des états finals, et

$$h(x)=\begin{cases}0 \text{ si } \pi_1(x)=0\\ \varphi\pi_2(x) \text{ si } \pi_1(x)\neq 0\end{cases}$$

La table des transitions est donnée par :

$$\left.\begin{array}{l}g(s_0,\not\in,0)=(s_1,+1)\\ g(s_1,1,m)=(s_1,+1)\\ g(s_1,\$,m)=(s_2,0)\end{array}\right\} \quad \text{transmission de la configuration d'entrée} \quad \text{avec} \quad \left\{\begin{array}{l}f(s_0,x)=0\\ f(s_1,x)=<0,\pi_2(x)+1>\\ f(s_2,x)=<1,\pi_2(x)>\end{array}\right.$$

$g(s_2,\$,m)=((e_0,e_0),0)$

$$g((e_1,e_2),\$,m)=\begin{cases}(\alpha_1(e_1,\eta^{-1}(m)),e_1),0) & \text{si } \alpha(e_1,\eta^{-1}(m))\downarrow\\ ((e_1,\eta^{-1}(m)),0) & \text{sinon}\end{cases}$$

Il reste à compléter la définition de $f((e_1,e_2),x)$ si $(e_1,e_2)\in E^2$, c'est-à-dire f simule alors les changements dans la configuration de $\mathcal{A}$. Il est clair qu'il faut transformer le code de la configuration en deux temps, correspondant au changement de symboles et au mouvement des têtes. On a besoin d'états dans E^2 à cause de la définition de la marche d'un automate ballon, puisque le nouvel état est déterminé avant l'application de f. Un couple (e_1,e_2) permet de "garder en réserve" l'état précédent (e_2).

On est amené à définir les fonctions récursives (primitives) suivantes :

- $\mu(x)$="la composante k du code de b_{ij} qui correspond au point s_{ij} de S_i, si $x=<\lambda_i(b_{ij}),\gamma_i(S_{ij})>$"

Soit :

$\mu(x)=\mathcal{M}_{k\leq|\pi_1(x)|}[\pi_1\pi_k\pi_1(x)=\pi_2(x)]$, où $\mathcal{M}$ est l'opérateur minimum borné et où $|x|=sg(x)[\pi_1(x\dot{-}1)+1]$ est la longueur du cortège codé par x.

- $\rho'(x,y)$="le code de l'expression b_{ij},s_{ij} quand on remplace $b_{ij}(s_{ij})$ par $y=\eta_i(a_{ij})$, si $x=<\lambda_i(b_{ij}),\gamma_i(s_{ij})>$"

Soit la fonction auxiliaire (remplacement d'une composante) :

$\sigma(x,y,j)=\tau^*(\pi_1(x),\ldots,\pi_{j-1}(x),y,\pi_{j+1}(x),\ldots,\pi_{|x|}(x))$

Alors :

$\rho'(x,y)=<\sigma(\pi_1(x),y,\mu(x)),\pi_2(x)>$

- $\rho(x,y)$="le code de la configuration de code x où les symboles pointés sont remplacés par $y=\eta(a_1,a_2,\ldots,a_n)$".

Soit :

$$\rho(x,y)=\tau^*((\rho'(\pi_i(x),\pi_i(y)))_{i\in I})$$

Il reste encore à simuler le mouvement des têtes. Pour ceci, on suppose que $\{\ \}_{n\in\mathbb{N}}$ est une gödelisation acceptable arbitraire et que

$$\begin{cases}\gamma_i(f_{ij})=\varphi_{\tilde{\gamma}_i(f_{ij})}\\ \gamma((f_{ij_i})_{i\in I})=\tau^*((\tilde{\gamma}_i(f_{ij_i})_{i\in I})\end{cases}$$

- $\nu'(x,y)$="le code de l'expression b_{ij},s_{ij} quand on remplace s_{ij} par $\varphi_y(s_{ij})$, où $y=\tilde{\gamma}_i(f_{ij})$ pour $f_{ij}\in F_i$".

Soit :

$$\nu'(x,y)=\langle\pi_1(x),\varphi_y(\pi_2(x))\rangle$$

- $\nu(x,y)$="le code de la configuration de code x si on lui applique les fonctions codées par y".

Soit :

$$\nu(x,y)=\tau^*((\nu'(\pi_i(x),\pi_i(y)))_{i\in I})$$

Finalement,

$$f((e_1,e_2),x)=\nu(\rho(x,\tilde{\alpha}_2(\langle e_2,\varphi(x)\rangle),\tilde{\alpha}_3(\langle e_2,\varphi(x)\rangle))$$

On a donc exhibé tous les composants de $\mathcal{B}$, qui, par construction, simule $\mathcal{A}$ pas-à-pas dès qu'il passe dans l'état initial (e_0,e_0). Le théorème est donc démontré. ✉

II- SYSTEMES DEFINITIONNELS ET ALGORITHMES

Nous avons rappelé que la notion d'algorithme implique le déterminisme, et presque toutes les machines abstraites mentionnées plus haut étaient déterministes. Cependant, on n'obtient cette propriété qu'en imposant une restriction au formalisme de description (par exemple, deux quadruplets différents ne peuvent commencer par le même couple), et il est tentant de la lever. Quand on le fait, on parle donc de "MT non-déterministes", ou même d'"algorithmes non-déterministes" (Floyd (1967) et ses successeurs [14,30]. Mais ceci est un abus de langage[§]. En effet, de telles machines abstraites (ou même certains ensembles d'équations de récursion) peuvent définir des fonctions ou des processus non Türing-calculables, c'est-à-dire non algorithmiques. Je parlerai donc plutôt de "systèmes définitionnels" pour ce type de description. Dans tous les cas, ils définissent une "arborescence des calculs possibles", et on leur associe, implicitement ou explicitement, un "critère de définition" destiné à choisir dans cette arborescence le ou les points significatifs (terminaux). Selon le critère choisi, il peut ou non y avoir un algorithme qui "réalise" le processus ou la fonction ainsi défini. Dans les cas classiques, il s'agit souvent de calculer des multifonctions (à image dans $\mathbb{N}^*$) définies par un système dont toutes les arborescences de calcul sont finies, et il suffit de faire un "back-tracking" pour trouver toutes les solutions. Ou encore on cherche le premier calcul qui converge dans une arborescence des calculs d'une MT non-déterministe, pour, par exemple la "simuler" par une MT déterministe (et ceci est encore un abus de langage[§§]).

2.1. ARBORESCENCES DES CALCULS ET CRITERES DE DEFINITION

Nous allons donner quelques exemples de systèmes définitionnels algorithmiques et non-algorithmiques. On peut en donner une définition très générale, fondée sur la notion de système de transitions.

Définition 9

(1) Un **système de transitions** *est un couple $T=(C,\Phi)$, où C est un ensemble dénombrable de* **configurations** *défini comme pour les G-machines, et Φ, le* **programme** *de T, un élément de $(\bigcup_{i=1}^{r} C^i)^C$, tels qu'il existe une bijection algorithmique $\theta_0:\mathbb{N}\to C$ qui rende Φ récursif.*

Φ est défini par le **contrôle fini** *α, multifonction de $E \times \prod_{i\in I} A_i^{h_i}$ dans $E \times \prod_{i\in I} A_i^{h_i} \times \prod_{i\in I} F_i$, de multiplicité au plus égale à r. Considéré comme une relation, α est un ensemble fini de couples, les* **transitions**.

Pour toute configuration c, $\Phi^-(c)=\bigcup_{n\in\mathbb{N}} [\Phi^n(c)]$ est l'ensemble[§§§] des configurations accessibles à partit de c, *et $\Phi^*(c)=\bigcup_{n\in\mathbb{N}}\{c_0,c_1\ldots c_n \mid c=c_0 \ \&\ (\forall p<n)[c_{p+1}\in\{\Phi(c_p)\}]\}$ est* l'arborescence des calculs issus de c. L'ensemble des calculs terminaux issus de c *sera noté $Max\Phi^*(c)=\{x\in\Phi^*(c) \mid (\forall y,z\in C^*)[y=xz \ \&\ z\in C^+ \Longrightarrow y\notin\Phi^*(c)]\}$.*

On définit enfin une **fonction de sortie** *récursive (via θ_0) $\theta_1:C\to\mathbb{N}$, qu'on prolonge trivialement à C^+ en posant $(\forall x\in C^*)(\forall c\in C)[\theta_1(xc)=\theta_1(c)]$.*

(2) Une **fonction d'évaluation** *pour un système de transition T est une fonction récursive (via θ_0) $\varepsilon : \Phi^*(C) \to \mathbb{N}$ ou $\mathbb{Q}$, qui associe un entier ou un rationnel à tout calcul issu de c, pour tout c*[§§§§].

Remarques

- La multiplicité de Φ est nécessairement bornée par
$$\Gamma_0=|E|.\prod_{i\in I}|A_i|^{h_i}.\prod_{i\in I}|F_i|$$

- ε est "récursive via θ_0" si et seulement si $\varepsilon\theta_0$ l'est.

- on exige que toute configuration ait un nombre borné (par r) de successeurs, ce qui est nécessaire si Φ est donné par un moyen fini constructif. D'autre part, $\Phi^-(c)$ n'est pas en général une arborescence. Dans Vuillemin (1974 a,b), où on s'intéresse à un ensemble d'équations de récursion, c'est un treillis. Dans le cas général (Savitch, 1970), c'est un multigraphe[§§§§§].

---

§ - Floyd écrit d'ailleurs : "Non deterministic algorithms are conceptual devices to simplify the design of back-tracking algorithms by allowing considerations of program bookkeeping required for back-tracking to be ignored".

§§ - Chauché (1974, I-17) contient une remarque analogue sur ce point.

§§§ - $[\Phi^n(c)]$ est bien un ensemble d'éléments (cf les notations au début du volume).

§§§§ - cf. encore Floyd : "all points of termination are labeled as successes or failure" - pour lui donc, Im $\varepsilon \subset \{0,1\}$. D'autre part, il est bien connu que $\mathbb{Q}$ est isomorphe à $\mathbb{N}\times(\mathbb{N}-\{0\})$, et on peut encore considérer $\varepsilon':Max\Phi^*(C)\to\mathbb{N}$ et prendre $\varepsilon(c)=(\Pi,\varepsilon'(c),\Pi_2\varepsilon'(c))$.

§§§§§ - De façon imagée, W. Savitch parle de "labyrinthes" ("mazes"), "ensemble de pièces reliées par des corridors à sens unique", où on distingue une "pièce de départ" et des "pièces d'arrivée".

- enfin, on demande d'évaluer tous les calculs, terminaux ou non. La plupart du temps, on construit cette évaluation de proche en proche : $\varepsilon(x.c)=\varepsilon_1(\varepsilon(x),\theta_0^{-1}(c))$. Dans les cas les plus simples, on peut restreindre ε à $Max\Phi^*(C)$, l'ensemble des calculs terminaux.

Pour pouvoir définir un ensemble de résultats d'un système de transitions sur une configuration initiale, avec quelque chance de pouvoir les obtenir algorithmiquement, on n'associe de résultat qu'à un calcul terminal, et il faut disposer d'une énumération effective des calculs terminaux (ou de leurs valeurs) et de leurs évaluations, pour pouvoir appliquer un certain test. C'est cette idée que nous allons préciser.

Proposition 1

Pour tout système de transitions $T=(C,\Phi)$, il existe une infinité de fonctions d'énumération $\tilde{\eta}\in R^1$ *telles que :*

(1) $(\forall n)[\varphi_{\tilde{\eta}(n)}(\mathbb{N})=\theta_0^{-1}\Phi^*\theta_0(n)]^{\S}$

(2) $(\forall n)[W_{\tilde{\eta}(n)}$ *est une section commençante de* $\mathbb{N}]$.

A toute fonction d'évaluation ε et à toute fonction d'énumération $\tilde{\eta}$, on peut associer de manière unique une fonction de résultat $\eta\in R^1$ *qui énumère les couples évaluation-valeurs des calculs terminaux, dans l'ordre induit par $\tilde{\eta}$, et telle que :*

(1) $(\forall n)[\varphi_{\eta(n)}(\mathbb{N})=\{<\varepsilon(c),\theta_1(c)>\}_{c\in Max\Phi^*\theta_0(n)}]$

(2) $(\forall n)[W_{\eta(n)}$ *est une section commençante de* $\mathbb{N}]$.

Comme la multiplicité de Φ est bornée, on peut uniformément énumérer $\Phi^*(c)$ par longueur croissante, d'où une fonction $\tilde{\eta}$, et on en obtient aisément une infinité d'autres en introduisant des répétitions, ou, si $W_{\tilde{\eta}(n)}$ est infini, en définissant $\tilde{\eta}'$ par $\varphi_{\tilde{\eta}'(n)}=\varphi_{\tilde{\eta}(n)}\cdot\psi^{-1}$ pour toute permutation récursive ψ. η s'obtient alors aisément en posant :

$$\begin{cases}\hat{\eta}(n,0)=\mu k[\Phi\varphi_{\tilde{\eta}(n)}(k)=\emptyset]\\ \hat{\eta}(n),p+1)=\mu k[\Phi\varphi_{\tilde{\eta}(n)}(k)=\emptyset\ \&\ k>\hat{\eta}(n,p)]\end{cases}\quad\text{et}\quad\varphi_{\eta(n)}(p)=<\varepsilon\varphi_{\tilde{\eta}(n)}\hat{\eta}(n,p),\theta_1\varphi_{\tilde{\eta}(n)}\hat{\eta}(n,p)>$$

Définition 10

Un critère de définition *pour un système de transitions $T=(C,\Phi)$ est un quadruplet $\delta=(\varepsilon,\tilde{\eta},\gamma,Q)$ formé d'une fonction d'évaluation, d'une fonction d'énumération, d'un prédicat récursif $\gamma\in R^3$ et d'un quantificateur $Q\in\{\forall,\exists\}$.*

Si $\eta'\in R^2$ est définie par $\begin{cases}\eta'(n,0)=0 & \text{pour } n\in\mathbb{N}\\ D_{\eta'(n,p+1)}=D_{\eta'(n,p)}\cup\{\varphi_{\eta(n)}(p)\} & \text{pour } n\in\mathbb{N}\text{ et }p\in W_{\eta(n)}\end{cases}$

L'ensemble des valeurs de T sur n relativement à δ *est alors :*

$$T_\delta(n)=\{m\in\pi_2\varphi_{\eta(n)}(\mathbb{N})\mid(Qp)[\gamma(m,\eta'(n,p),p)]\}$$

$D_{\eta'(n,p)}$ est l'ensemble des (codes des) couples évaluation-valeur déductibles de la configuration de code n et énumérés par $\varphi_{\eta(n)}$ en moins de p pas (il peut donc y avoir eu des répétitions). $T_\delta(n)$ est un ensemble de valeurs (d'où π_2), et γ porte donc sur les ensembles finis de valeurs produites par $\varphi_{\eta(n)}$. Cette définition est très générale, de façon à recouvrir la plupart des critères qu'on rencontre dans la littérature.

Exemples

(1) $\gamma(m,\eta'(n,p),p)\equiv[p=1\ \&\ m=\pi_1D_{\eta'(n,1)}]$, $Q\equiv\exists$: on prend la première valeur rencontrée (via l'énumération $\tilde{\eta}$ choisie).

(2) $\gamma(m,\eta'(n,p),p)\equiv[\sum_{x\in X(m,n,p)}\pi_1(x)>\lambda]$ avec $X(m,n,p)=D_{\eta'(n,p)}\cap\pi_2^{-1}(m)$, $Q\equiv\exists$ et λ réel dans $[0,1]$. ε s'interprète alors comme une probabilité : on attend un certain seuil avant de "produire" m.

(3) $\gamma(m,\eta'(n,p),p)\equiv[m=\mathrm{Min}\,\pi_2D_{\eta'(n,p)}]$, $Q\equiv\forall$: $T_\delta(n)$ est la plus petite valeur associée à un calcul terminal.

---

§ - En posant $\theta_0^{-1}(c_1\ldots c_n)=<\theta_0^{-1}(c_1),\ldots,\theta_0^{-1}(c_n)>$.

Définition 11

Un **système définitionnel** *est le couple D=(T,δ) d'un système de transitions T=(C,Φ) et d'un critère de définition* $\delta=(\varepsilon,\tilde{n},\gamma,Q)$.

Un système définitionnel est **réalisable** *si*
$(\exists \rho\in R^2)(\forall n)[T_\delta(n)=\bigcup_{p\in N} D_{\rho(n,p)}]$

Un système définitionnel est **finiment réalisable** *si*
$(\exists \rho\in R^1)(\forall n)[T_\delta(n)=D_{\rho(n)}]$

Un système de transitions (ou un système définitionnel) est à **calculs effectivement finis** *si*
$(\exists \xi\in R^1)(\forall n)[\theta_0^{-1}\Phi^*\theta_0(n)=D_{\xi(n)}]$

Un système définitionnel est réalisable si on peut effectivement énumérer ses valeurs sur chaque argument : à un certain moment, on est sûr que $m\in T_\delta(n)$, et m ne peut plus être remis en cause. Quand D est finiment réalisable, on sait a priori que $T_\delta(n)$ est fini pour tout n, et on sait prévoir sa taille. C'est dire qu'on peut décider quand il faut arrêter de chercher de nouvelles valeurs. Ces notions ne coïncident pas, même dans le cas où $T_\delta(n)$ est fini pour tout n, car le problème de savoir si $|\mathrm{Im}\varphi_{\eta(n)}|>p$ est en général indécidable.

Exemples

(1) Avec le critère δ défini à l'aide de l'exemple (1) ci-dessus, ε et $\tilde{n}$ étant arbitraires, toute G-machine M=(C,Φ) est un système définitionnel D=(M,δ). (On peut prendre $\theta_1=\theta_0^{-1}$, ou toute fonction calculable).

(2) Les grammaires probabilistes ou pondérées [50] constituent également des systèmes de transitions, sur lesquels on considère souvent deux§ critères de définition, correspondant aux exemples (3) et (2) ci-dessus : si G_w est une grammaire pondérée, $\lambda>0$, et ε correspond à la probabilité ou au poids d'une dérivation. Désignons par $\mathfrak{D}(x)$ l'ensemble des dérivations d'une chaîne x dans G. Les interprétations "minimum" et "somme" définissent deux langages associés à G :

$$L_m(G_w,\lambda)=\{x\mid(\exists D)[D\in\mathfrak{D}(x)\ \&\ \varepsilon(D)>\lambda]\}$$
$$L_s(G_w,\lambda)=\{x\mid \sum_{D\in\mathfrak{D}(x)}\varepsilon(D)>\lambda\}$$

Proposition 2

(1) Tout système définitionnel où Q=∃ est réalisable.

(2) Tout système définitionnel à calculs effectivement finis est finiment réalisable.

(3) Pour tout système définitionnel réalisable, $(\exists \sigma\in R^1)(\forall n)[T_\delta(n)=W_{\sigma(n)}]$, *i.e. les ensembles de valeurs sont uniformément r.e..*

(1) Il suffit de définir ρ par :

$$\begin{cases}\rho(n,0)=0 \text{ pour tout } n\\ D_{\rho(n,p+1)}=D_{\rho(n,p)}\cup D_{\nu(n,p)} \text{ pour tout } p\in \mathrm{Dom}\,\varphi_{\eta(n)},\ \text{avec}\end{cases}$$

$$D_{\nu(n,p)}=\{m\in\pi_2 D_{\eta'(n,p)}\mid\gamma(m,\eta'(n,p),p)\}$$

(2) D'abord $(\exists \xi'\in R^1)[D_{\xi'(n)}=\theta_0^{-1}\mathrm{Max}\Phi^*\theta_0(n)]$, puisque la propriété d'être terminal est récursive. Donc $(\exists\beta\in R^1)[|D_{\xi'(n)}|=\beta(n)]$. Par suite $\zeta(n)=\mu p[|D_{\eta'(n,p)}|=\beta(n)]$ est récursive d'après la définition de η et η'.

Alors, si Q=∃, $D_{\rho(n)}=\bigcup_{p=0}^{\zeta(n)} D_{\nu(n,p)}$ et ρ est récursive.

Si Q=∀ $D_{\rho(n)}=\{m\in\pi_2 D_{\xi'(n)}\mid\bigwedge_{p=0}^{\zeta(n)}\gamma(m,\eta'(n,p),p)\}$ et ρ est récursive.

(3) est immédiat d'après les théorèmes de projection et le théorème s-m-n [48]. ∎

§ - Ces critères sont bien différents, comme le montrent les théorèmes d'inclusion stricte (10,11,12) de Salomaa.

On peut facilement concevoir des systèmes non finiment réalisables, si par exemple $|T_\delta(n)| = \infty$ pour tout n.

Exemple

Soit la machine de Post non déterministe T donnée par

$q_0\ 1\ R\ q_0$

$q_0\ B\ 1\ q_0$

$q_0\ B\ L\ q_1$

$q_1\ 1\ L\ q_1$

$q_1\ B\ R\ q_2$

On prend pour θ_0 et θ_1 les fonctions habituelles [15], $\varepsilon(c)=1$ pour tout c, γ est le prédicat trivial vrai partout, η est l'énumération canonique d'une arborescence et $Q=\mathbb{Z}$.

Alors $(\forall n)[T_\delta(n)=\{m \mid m>n\}]$.

2.2. DES SYSTEMES REALISABLES

Proposition 3

Tout système définitionnel à calculs de longueur finie est finiment réalisable.

Tout d'abord, $\Phi^*(c)$ est fini pour tout c, puisque chaque configuration a au plus r successeurs (Endlichkeitslemma) On sait alors construire une énumération récursive de $\Phi^*(c)$ par rétrogression (back-track), qui permet de déterminer ξ. Soit M le nombre de lignes de la table de α, appelons Φ_k la fonction de transition définie par la k-ième ligne de α, et posons $\overset{\gamma}{\Phi}_1=\theta_0\Phi_1\theta_0^{-1}$. Soit alors f, définie par :

$$\begin{cases} f(e,0) = <\theta_0(e),0,0> & \text{Soit } f(e,n)=<c,u,k> \\ f(e,n+1)= \begin{cases} <\overset{\gamma}{\Phi}_1(c),<c,u,1>,0> & \text{si } 1=\mu j[k<j\leqslant M \ \&\ \overset{\gamma}{\Phi}_j(c)\neq c] \\ u & \text{sinon} \end{cases} \end{cases}$$

La fonction $\beta=\lambda e[\mu n(f(e,n)=0)]$ est alors récursive et

$D_{\xi(e)}=\{\theta_0^{-1}\pi_1 f(e,n) \mid n<\beta(e)\}$ ✉ Les calculs finals sont tels que $\pi_3 f(e,n)=0$.

Ce cas est important dans la pratique, et correspond aux problèmes où on cherche à construire un certain nombre, fini, de solutions en parcourant une "arborescence des choix" dont on sait a priori qu'elle est finie : analyse morphologique d'un mot, analyse syntaxique d'une phrase, problèmes combinatoires [35], etc. Dans ces cas, il est souvent commode et moins coûteux d'utiliser la rétrogression introduite dans la démonstration.

Proposition 4

Tout système définitionnel $D=(T,\delta)$ de critère $\delta=(\varepsilon,\overset{\gamma}{\Pi},\gamma,\mathbb{Z})$ tel que $\gamma(m,\eta'(n,p),p)\equiv[\sum_{x\in X(m,n,p)}\pi_1(x)>\lambda]$ avec $X(m,n,p)=D_{\eta'(n,p)}\cap\pi_2^{-1}(m)$ et $\lambda\in Q$ est réalisable, mais non nécessairement finiment réalisable.

Il est réalisable puisque $Q=\mathbb{Z}$. Mais, si $\Phi^*(c)$ est infini, on ne peut jamais être sûr d'avoir trouvé tous les éléments de $T_\delta(n)$. Remarquons que prendre $\lambda\in\mathbb{R}$ n'ajoute rien, puisque $\mathbb{Q}$ est dense dans $\mathbb{R}$. ✉

C'est souvent ce critère qu'on emploie quand on parle de MT non déterministes (avec $\varepsilon=1$) ou des grammaires probabilistes [50]. C'est pourquoi on a pu dire, par abus de langage, que "toute MT non-déterministe est simulable par une MT déterministe"§ : ceci n'est pas vrai pour tous les critères, comme nous allons le voir.

---

§ - [28] p. 96

2.3. DES SYSTEMES IRREALISABLES

Les systèmes réalisables sont assez naturels, au sens où on peut produire progressivement les résultats en tenant compte d'une information "cumulative" ou simplement locale. Si par contre, comme dans le cas des MT "probabilistes", Q=∀ et le critère porte sur tout $\Phi^*(c)$, éventuellement infini, la situation est plus délicate.

Un autre cas intéressant et plus subtil se présente quand on peut associer une (multi) fonction à la description (structure) d'un tel système, alors qu'il ne peut la "calculer", ni même en calculer aucune pour aucun δ, comme ceci se produit pour certains systèmes d'équations de récursion.

2.3.1. MACHINES PROBABILISTES ET FONCTIONS NON CALCULABLES

Ce paragraphe reprend une démonstration dûe à Santos (1969), avec de légères modifications.

Définition 12 (*)

Une **"G-machine probabiliste"** *est le couple d'un système de transitions* $T=(C,\Phi)$ *et d'une fonction* $q:C^2\to[0,1]$, *segment de la droite réelle, avec*

$$(\forall c\in C)\Big[\sum_{c'\in\Phi(c)} q(c,c')=1\Big]$$

$p(c,c')$ est interprété comme la probabilité de transition de c à c'.

Une **"machine de Post probabiliste"** *est une G-machine probabiliste simple à une bande donnée par une fonction* $p:E\times A\times A'\times E\to[0,1]$ *avec* $A'=A\cup\{L,R,0\}$ *et*

(1) $$(\forall (e,a)\in E\times A)\Big[\sum_{(a',e')\in A'\times E} p(e,a,a',e')=1\Big]$$

(2) $$(\forall a\in A)(\forall e,e'\in E)[e\neq e' \Longrightarrow p(e,a,0,e')=0]$$

On notera une telle machine $Z=(E,A,p)$.

On retrouve bien la machine de Post classique si on exige que Im p ⊂ {0,1}. D'autre part, une machine de Post probabiliste est une G-machine probabiliste, de probabilité de transition q_z donnée par :

$q_z(c,c') = p(e,a,a',e')$ si c = x e a y , c' = x e' a' y , a'∈A

$= p(e,a,R,e')$ si c = x e a a' y , c' = x a e' a' y , a'∈A
ou si c = x e a , c' = x a e' a_0 ,

$= p(e,a,L,e')$ si c = x a e a' y , c' = x e a a' y , a'∈A
ou si c = e a y , c' = e a_0 a y ,

$= 0$ sinon,

avec $x,y\in A^*$.

Notons que (2) impose que l'arrêt provienne uniquement des quadruplets du type e,a,0,e. Ces états e sont les états finals de Z.

La probabilité de transition q_z permet de définir une fonction d'évaluation sur $\Phi^*(C)$ tout entier, en posant :

$$\begin{cases}\varepsilon_z(c) = 1\\ \varepsilon_z(u.c.c') = \varepsilon_z(u.c)\ .\ q_z(c,c')\end{cases}$$

Elle permet aussi de définir la probabilité de dérivation en n pas

$$\begin{cases} q_z^{(0)}(c,c') = \begin{cases}1 \text{ si } c=c'\\ 0 \text{ sinon}\end{cases}\\ q_z^{(n)}(c,c') = \sum_{c''\in C} q_z^{(n-1)}(c,c'')\ .\ q_z(c'',c)\end{cases}$$

Il est immédiat que :

$$(\forall c\in C)(\forall n\in\mathbb{N})\Big[\sum_{c'\in C} q_z^{(n)}(c,c')\leqslant 1\Big]$$

On peut encore définir la probabilité pour arriver de c à une configuration terminale c' en n pas :

$$t_z^{(n)}(c,c') = p(e,a,0,e)\; q_z^{(n-1)}(c,c') \qquad \text{avec } c' = x\ e\ a\ y$$

et $t_z(c,c') = \sum_{n=1}^{\infty} t_z^{(n)}(c,c')$

$t_z(c,c')$ est une série convergente pour tout couple (c,c'), puisque

$$\sum_{c'\in C}\Big[q_z^{(n)}(c,c') + \sum_{k=1}^{n} t_z^{(k)}(c,c')\Big] = 1$$

Définition 13 (Santos)

Une fonction aléatoire k-aire *est une fonction* $\psi:\mathbb{N}^{k+1}\to[0,1]\subset\mathbb{R}$ *telle que :*

$$(\forall(m_1,\ldots,m_k)\in\mathbb{N}^k)\Big[\sum_{m\in\mathbb{N}} \psi(m_1,\ldots,m_k,m)\leqslant 1\Big]$$

On va maintenant construire le critère de définition. Tout d'abord, on définit pour chaque k>0 une fonction k-aire $\psi_z^{(k)}$ par :

$$\psi_z^{(k)}(m_1,m_2,\ldots,m_k,m) = \sum_{\theta_1(c')=m} t_z(\theta_0(m_1,\ldots,m_k),c')$$

Rappelons que $\theta_1(c)=\langle c\rangle$ est le nombre de "|" (a_1) de c et que $\theta_0(m_1,\ldots,m_k)=e_0(\overline{m_1,\ldots,m_k})=e_0\underbrace{|\ldots|}_{m_1+1}\,B\,\ldots\,B\,\underbrace{|\ldots|}_{m_k+1}$

Dans les deux définitions qui suivent, la terminologie de Santos est modifiée pour la cohérence avec l'ensemble du chapitre.

Définition 14 (Santos)

Une fonction aléatoire k-aire est P-concevable *s'il existe une machine de Post probabiliste Z telle que* $\psi=\psi_Z^{(k)}$.

Soit $\lambda\in[0,1[$. *Une fonction k-aire* $f:\mathbb{N}^k\to\mathbb{N}$ *est* engendrée par une fonction aléatoire k-aire ψ avec seuil λ *si*

(1) $(\forall(m_1,\ldots,m_k)\notin \mathrm{Dom}\ f)(\forall m\in\mathbb{N})[\psi(m_1,\ldots,m_k,m)\leqslant\lambda]$

(2) $(\forall(m_1,\ldots,m_k)\in \mathrm{Dom}\ f)\ [f(m_1,\ldots,m_k)=m \Longrightarrow$

$\psi(m_1,\ldots,m_k,m) = \sup_{m'\in\mathbb{N}}\{\psi(m_1,\ldots,m_k,m')\}>\lambda]$

Remarquons que, si $F(\psi,\lambda)$ est l'ensemble des fonctions engendrées par ψ avec seuil λ, $F(\psi,\lambda)\neq\emptyset$ pour tout λ. En effet, $\sum_{m\in\mathbb{N}} \psi(m_1,\ldots,m_k,m)\leqslant 1$ et donc le sup est atteint. De plus, si $\lambda\geqslant\frac{1}{2}$, $F(\psi,\lambda)$ contient exactement une fonction.

Définition 15 (Santos)

Une fonction k-aire $f:\mathbb{N}^k\to\mathbb{N}$ *est* P-définissable avec seuil λ *s'il existe une machine de Post probabiliste Z telle que f soit engendrée par* $\psi_Z^{(k)}$ *avec seuil* λ. *Elle est* P-définissable *si elle est P-définissable avec seuil* λ *pour au moins un* $\lambda\in[0,1]$.

Ceci définit une famille de critères de définition, où $Q=\forall$, $\varepsilon=t_z$, $\tilde{\eta}$ est arbitraire et γ est donné par :

$$\gamma(m,\eta'(n,p),p)\equiv(\forall m')(\exists q)\Big[\sum_{x\in X(m',n,p)}\pi_1(x)\leqslant\sum_{x\in X(m,n,q)}\pi_1(x)\ \&\ \sum_{x\in X(m,n,q)}\pi_1(x)>\lambda\Big],\ \text{avec } X(m,n,p)=D_{\eta'(n,p)}\cap\pi_2^{-1}(m).$$

Nous allons voir que c'est un critère trop "global" pour que ce système soit réalisable. Plus précisément, on va montrer qu'il définit des fonctions non-calculables.

Notations

Si $S=(E,A,p)$, $Z^{(n)}$ est obtenue à partir de Z en remplaçant chaque e_i par e_{i+n}.

Si $Z_1=(E_1,A_1,p_1)$ et $Z_2=(E_2,A_2,p_2)$, avec $E_1 \cap E_2=\{\tilde{e}\}$, $Z_1 \to Z_2$ est une nouvelle machine $(E_1 \cup E_2, A_1 \cup A_2, p)$, avec

$p(e,a,a',e') = p_1(e,a,a',e')$ si $e,e' \in E_1$, $a \in A_1, a' \in A_1'$ et $e \neq \tilde{e}$

$= p_2(e,a,a',e')$ si $e,e' \in E_2$, $a \in A_2, a' \in A_2'$

$= 1$ si $e=e' \in E_1$, $a \notin A_1, a'=0$

ou si $e=e' \in E_2$, $a \notin A_2, a'=0$

$= 0$ sinon

On a donc une composition linéaire via l'état $\tilde{e}$, et $Z_1 \to Z_2$ est encore probabiliste.

La démonstration se fait en utilisant les expansions binaires des nombres rationnels. Donnons d'abord une machine de Post qui transforme un nombre (donné en base 1 via θ_0) en son expansion binaire. On prendra R=(E,A,p) une machine de Post (déterministe), avec : $A=\{0,1,a,b,c\}$, $E=\{e_0,\dots,e_{12}\}$ et $C_p=p^{-1}(1)$ donné par :

$(e_0\ 1\ R\ e_0)$ $(e_0\ 0\ b\ e_0)$ $(e_0\ b\ L\ e_1)$ $(e_1\ 1\ 0\ e_1)$

$(e_1\ 0\ L\ e_2)$ $(e_2\ 1\ c\ e_3)$ $(e_3\ c\ R\ e_4)$ $(e_4\ 0\ R\ e_4)$

$(e_4\ 1\ R\ e_4)$ $(e_4\ b\ L\ e_5)$ $(e_5\ 0\ 1\ e_6)$ $(e_6\ 1\ L\ e_6)$

$(e_6\ 0\ L\ e_6)$ $(e_6\ c\ 0\ e_7)$ $(e_7\ 0\ L\ e_2)$ $(e_5\ 1\ 0\ e_8)$

$(e_8\ 0\ L\ e_9)$ $(e_9\ 0\ 1\ e_6)$ $(e_9\ 1\ 0\ e_2)$ $(e_2\ 0\ R\ e_{10})$

$(e_{10}\ 0\ R\ e_{10})$ $(e_{10}\ 1\ L\ e_{11})$ $(e_{11}\ 0\ a\ e_{12})$

Si $c=e_0\bar{m}$, $t_R(c,c')=1 \Longrightarrow c'=ae_{12}xb$, où x est l'expansion binaire de m.

On construit maintenant une machine probabiliste telle que chaque entier soit transformé en son expansion binaire, avec une probabilité qui dépend de cette expansion. On utilisera la machine

Q=(E,A,p) avec $A=\{0,1,a,b\}$, $E=\{e_0,e_1\}$ et p donnée par :

$P_{0,R}=\begin{pmatrix}1 & 0\\ 1/2 & 1/2\end{pmatrix}$, $P_{1,R}=\begin{pmatrix}1/2 & 1/2\\ 0 & 1\end{pmatrix}$ pour les transitions $e_i\ 0\ R\ e_j$ et $e_i\ 1\ R\ e_j$,

et $p(e_0\ a\ R\ e_0)=1$ $p(e_1\ a\ 0\ e_1)=1$

$p(e_0\ b\ b\ e_0)=1$ $p(e_1\ b\ 0\ e_1)=1$, $p(e\ a\ a'\ e')=0$ sinon.

Par construction, il est évident que l'état final est e_1. D'autre part, on a le

Lemme 5

Si $c=a\ e_0\ x\ b$, avec $x=i_1\ i_2 \dots i_n$, $i_k \in \{0,1\}$ pour $1 \leq k \leq n$,

$$t_Q(c,c')=\begin{cases}0 & \text{si } c' \neq a\ x\ e_2\ b\\ 0,\ i_n\ i_{n-1} \dots i_1 & \text{si } c'=a\ x\ e_2\ b\end{cases}$$

$0,\ i_n \dots i_1$ désigne comme d'habitude $\sum_{k=1}^{n} i_k 2^{-k}$.

En effet, si $c' \neq a\ x\ e_2\ b$ la propriété est évidente. Sinon, $t_Q(c,c')$ est la probabilité de passage de e_0 à e_1 en lisant l'entrée x, et donc

$t_Q(c,c')=q$, si $P_{i_1R} \dots P_{i_nR}=\begin{pmatrix}p & q\\ r & s\end{pmatrix}$.

La démonstration se fait par récurrence sur n. Si n=1 et i_1=0, q=0 dans $P_{0,R}$; si n=1 et i_1=1, q=1 dans $P_{1,R}$. Si maintenant la propriété est vraie au rang n, alors :

- si i_{n+1}=0, $P_{i_1R} \cdots P_{i_nR} P_{OR} = \begin{pmatrix} p & q \\ r & s \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1/2 & 1/2 \end{pmatrix} = \begin{pmatrix} p+q/2 & q/2 \\ r+s/2 & s/2 \end{pmatrix}$, et

$q/2 = \frac{1}{2}.\ 0,i_n \cdots i_1 = 0,0i_n \cdots i_1$

- si i_{n+1}=1, $P_{i_1R} \cdots P_{i_nR} P_{0,R} = \begin{pmatrix} p & q \\ r & s \end{pmatrix} \begin{pmatrix} 1/2 & 1/2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} p/2 & p/2+q \\ r/2 & r/2+s \end{pmatrix}$, et

comme p+q=1, $p/2 + q = q/2 + 1/2 = 0,1i_n \cdots i_1$, d'où la propriété. ▰

Notations

Si l'expansion binaire de m est $i_1 \cdots i_n$, on notera $b(m) = 0,i_n \cdots i_1$ et $g(m) = {}_1(i_1 \cdots i_n)$.

Lemme 6

Soit $Z = \tilde{R} \dot{+} Q^{(12)}$, *alors*

$$\Psi_Z^{(1)}(m_1,m) = \begin{cases} b(m_1) & \text{si } m = g(m_1) \\ 0 & \text{sinon} \end{cases}$$

Soit $\lambda \in [0,1[$. *Il existe exactement une fonction* f_λ *engendrée par* $\Psi_Z^{(1)}$ *avec seuil* λ, *et*

$$f_\lambda(m) = \begin{cases} g(m) & \text{si } b(m) > \lambda \\ \uparrow & \text{si } b(m) \leqslant \lambda \end{cases}$$

La démonstration est immédiate. ▰ On a finalement le résultat annoncé.

Théorème 5 (Santos)

La classe des fonctions P-définissables a la puissance du continu, et donc le système définitionnel des machines de Post non déterministes n'est pas réalisable.

En effet, $f_{\lambda_1} = f_{\lambda_2} \Longleftrightarrow \lambda_1 = \lambda_2$, car il est clair que $\lambda_1 < \lambda_2 \Longrightarrow$ Dom $f_{\lambda_2} \subset$ Dom f_{λ_1}, et, comme b(m) est dense dans $[0,1[$, $(\exists m_0)[\lambda_1 < b(m_0) < \lambda_2]$ et donc $m_0 \in$ Dom f_{λ_1} - Dom f_{λ_2}. Comme les fonctions récursives sont dénombrables, $\{f_\lambda\}_{\lambda \in [0,1[}$ contient nécessairement des fonctions non récursives, donc non calculables, et le système n'est pas réalisable. ▰

2.3.2. SYSTEMES D'EQUATIONS DE RECURSION ET FONCTIONS NON CALCULABLES

Parmi les différentes approches de la notion de calculabilité, celle de Kleene est une des plus connues [31]. Elle consiste à étendre la méthode de définition usuelle des fonctions récursives primitives, qui consiste à donner un ensemble de fonctions de base (successeur, nulle, choix d'arguments) et des opérations de clôture (composition et schéma de récursion primitive), exprimées au moyen d'un ensemble d'équations, dites "de récursion". Le formalisme de Kleene, dérivé de celui d'Herbrand et Gödel, consiste à utiliser un système d'équations pour définir une fonction "générale récursive", et la notion de calculabilité coïncide alors avec celle de dérivabilité dans un système logique [31, p. 265 et suivantes].

Deux écueils au moins apparaissent : d'abord, il se peut qu'un tel système d'équations soit satisfait par plusieurs, voire même une infinité de fonctions (calculables ou non) distinctes (exemple trivial : f(x)=f(x+1)-1). Et ce n'est pas un hasard si Davis ([15] p. 47) insiste sur le fait que le schéma de récursion primitive définit bien une et une seule fonction totale à partir de deux fonctions totales. D'autre part, il se peut que le système définisse (en tant que système d'équations, et non pas en tant que système formel de réécriture) une ou plusieurs fonctions totales et qu'aucune ne soit calculable, comme dans l'exemple qui va suivre.

La relation de dérivabilité (pas substitution et instantiation) permet de considérer qu'un système d'équations de récursion définit un système de transitions. Le critère de définition adopté par Kleene est global et restreint la classe des systèmes d'équations possibles : toutes les "équations finales" déduites à partir d'un même argument doivent être égales, et définissent la valeur de la fonction associée sur cet argument. Cette supposition apparaît d'ailleurs de façon implicite dans l'article de Vuillemin (1974) sur les stratégies d'implémentation de la récursion : en fait, une "règle de calcul" définit un parcours du "treillis de calcul" ($\Phi^{-}(C)$), et par suite les "valeurs principales" de l'arborescence des calculs.

Exemple

Soit $g \in R^3$ définie par :

$$g(x,y,z) = \begin{cases} 1 & \text{si } \varphi_x(x) \text{ converge en moins de } y \text{ pas} \\ 2z & \text{sinon} \end{cases}$$

g est récursive (primitive), et il existe donc ([31] p. 326) un système d'équations de récursion (E) qui la définit. Soit :

(E')=(E)$\cup\{f(x,y)=g(x,y,f(x,y+1))\}$, où f est un nouveau symbole de fonction, et où l'équation additionnelle est l'équation principale. Rappelons que $K=\{x \mid \varphi_x(x)\downarrow\}$. Alors :

$$x \in \bar{K} \Longrightarrow \forall y\ f(x,y)=2f(x,y+1) \Longrightarrow f(x,y)=0$$

$$x \in K \Longrightarrow \exists z\ \varphi_x(x) \text{ en } z \text{ pas} \Longrightarrow f(x,y)=\begin{cases} 1 & \text{si } y \geq z \\ 2^{z-y} & \text{si } y<z \end{cases}$$

f est donc totale, et c'est l'unique fonction définie par (E'). Si elle était récursive, k définie par

$$k(x) = \begin{cases} 1 & \text{si } f(x,0)>0 \\ 0 & \text{sinon} \end{cases}$$

serait une fonction récursive caractéristique de K, d'où la contradiction.

III - HEURISTIQUES ET TECHNIQUES D'ENUMERATION, SYSTEMES DE G-MACHINES

Nous avons vu que les critères définitionnels des systèmes réalisables font intervenir certaines énumérations de $\Phi^*(c)$ pour tout c. Une "bonne" énumération doit être compatible avec la structure de $\Phi^*(c)$, au sens où on n'énumère pas un calcul sans avoir énuméré ses prédécesseurs. Je dirai qu'une telle énumération est une recherche (dans les calculs possibles). Cette notion est très générale, et on cherchera dans quelle mesure des propriétés de densité, de contrôle ou de constructivité sont appropriées à décrire des recherches intéressantes dans la pratique. Cette pratique va consister à associer à tout système définitionnel réalisable un système de G-machines dont un module permettra de réaliser la recherche ou la classe de recherches (heuristique) choisie.

3.1. ENUMERATIONS, RECHERCHES ET HEURISTIQUES

Définition 16

Soit S un ensemble (partiellement) ordonné par $<_S$. *On dira qu'une* énumération $s_0 s_1 \ldots s_n$ *de S est* compatible *avec* $<_S$ *si :*

$(\forall m,n\in\mathbb{N})[s_m <_S s_n \Longrightarrow (\exists p\leq n)[s_p = s_m]]$

Soit T un système de transitions. Une recherche *(effective) est une fonction récursive* σ *telle que :*

(1) $(\forall n)$ *Dom* $\varphi_{\sigma(n)}$ *est une section commençante de* $\mathbb{N}$.

(2) $(\forall n)$ $\theta_0 \varphi_{\sigma(n)}^{(\mathbb{N})} \subset \Phi^* \theta_0(n)$ *est une énumération de l'arborescence des calculs compatible avec la relation préfixe.*

Intuitivement, une "bonne" recherche n'énumère pas "trop souvent" les mêmes points, et est "assez facile" à réaliser. Pour qu'elle puisse être intéressante, on ne peut exiger qu'elle soit sans répétition. La première notion peut se formaliser en termes de "densité d'énumération" de chaque point énuméré, ou, plus globalement, en termes de "contrôle de production" de $\varphi_{\sigma(n)}$. La seconde fera intervenir des fonctions qui structurent $\Phi^*(c)$ pour tout c et en font l'homologue exact d'une structure de travail d'une G-machine.

Les énumérations qui nous intéressent sont des énumérations de calculs de systèmes de transitions. Cependant, avec les définitions choisies, chaque arborescence des calculs est isomorphe (récursivement) à $\mathbb{N}^*$, et donc à $\mathbb{N}$: nous nous ramenons donc pour le moment à des énumérations d'entiers.

3.1.1. DENSITE

Définition 17 (*)

Une énumération *est une fonction f d'une section commençante de* $\mathbb{N}$ *dans* $\mathbb{N}$.

Elle est récursive *si f est récursive, et* localement finie *(finite-one) si l'image réciproque de tout point est finie.*

Ainsi, dans une énumération localement finie, aucun terme ne peut apparaître une infinité de fois. C'est évidemment le cas si f est à domaine fini, mais, en général, on peut avoir à énumérer un ensemble infini. Dans ce cas, si l'énumération correspond intuitivement à une "recherche", on peut être amené à revenir une infinité de fois sur certains points, voire sur tous (cf. par exemple l'énumération des calculs possibles d'une MT non-déterministe). Pour cette raison, on peut penser à imposer une contrainte plus souple, à savoir que la densité de l'énumération soit nulle en tout point.

Définition 18 (Tennenbaum)

Soient $A\subset\mathbb{N}$ *et* $f:\mathbb{N}\to\mathbb{N}$. *La* densité de l'ensemble A relativement à la fonction f *est définie par :*

$$d(A,f) = \lim_{n\to\infty} \inf\left(\frac{\alpha_n}{n}\right), \text{ avec}$$

$$\alpha_n = |f^{-1}(A) \cap \{0,\ldots,n\}|$$

Exemples

(1) Notons $\langle x,y\rangle = 2^x(2y+1)-1$. $\pi_1(\langle x,y\rangle) = x$ est la première composante de l'inverse. Alors, pour π_1,

$$\frac{\alpha_n^x}{n} = \frac{p+1}{n} \leqslant \frac{p+1}{2^x(2p+1)-1} \quad \text{si } 2^x(2p+1)-1 \leqslant n < 2^x(2p+3)-1, \text{ et donc } (\forall x)\ d(\{x\},\pi_1) = \frac{1}{2^{x+1}} \neq 0.$$

Par contre, pour π_2, on aura

$$\frac{\alpha_n^y}{n} = \frac{p+1}{n} \leqslant \frac{p+1}{2^p(2y+1)-1} \quad \text{si } 2^p(2y+1)-1 \leqslant n < 2^{p+1}(2y+1)-1, \text{ et donc } (\forall x)\ d(\{x\},\pi_2)=0.$$

(2) Soit maintenant τ le codage de Cantor (récursivement isomorphe), défini par $\tau(x,y)=y+\frac{1}{2}(x+y)(x+y+1)$. Alors, pour π_1, $\frac{\alpha_n^x}{n} = \frac{p+1}{n} \leqslant 2.\ \frac{p+1}{(x+p)(x+p+1)+2p}$ si $\tau(x,p) \leqslant n < \tau(x,p)+x+p+2$, et donc $(\forall x)\ d(\{x\},\pi_1)=0$. De même, $(\forall y)\ d(\{y\},\pi_1)=0$.

Définition 19

Soit f une énumération, la densité d'énumération *de x par f est :*

$$\delta_f(x) = d(\{x\},f)$$

Une énumération *est* de densité nulle *si tout point a une densité d'énumération nulle. Si de plus*

$(\exists k \in R^2)(\forall n)(\forall x)[\frac{\alpha_n^x}{n} \leqslant \frac{1}{k(n,x)}]$, *où k est non nulle, croissante en n et non bornée, on dira que l'énumération est* de densité effectivement nulle. *Et si*

$(\exists k' \in R^1)(\forall n)(\forall x)[\frac{\alpha_n^x}{n} \leqslant \frac{1}{k'(n)}]$, *où k' est non nulle, croissante et non bornée, on dira que l'énumération est* de densité uniformément effectivement nulle.

Les exemples précédents montrent qu'avoir une densité nulle n'est pas une propriété récursivement invariante. On verra au chapitre II d'autres notions de densité, au sens de topologies sur des ensembles de fonctions, et non plus sur leurs ensembles de valeurs.

Proposition 5

Les trois propriétés de densité nulle, effectivement nulle et uniformément effectivement nulle sont distinctes.

(1) Dans les exemples précédents, les densités nulles l'étaient uniformément effectivement. Il est facile cependant de construire une énumération récursive de densité effectivement, mais non uniformément, nulle. On prendra par exemple

$$\begin{cases} f(0)=0 \\ f(n+1)=p \quad \text{si } 2^{p-1} \leqslant n+1 < 2^p, \end{cases}$$

et on peut prendre $k(n,x)=\max(1,\lfloor n/2^{x-1} \rfloor)$. Mais on ne peut trouver de k' uniforme, car

$$(\forall x)[\frac{\alpha_{2^x-1}^x}{2^x-1} = \frac{1}{2}]$$

(2) Pour construire une énumération de densité nulle, mais non effectivement, donnons-nous une énumération récursive sans répétition de K, soit $\{k_0,k_1,\ldots k_n,\ldots\}$. L'énumération f prendra la valeur 0 une infinité de fois, et chaque valeur de $\mathbb{N}-\{0\}$ une seule fois, avec f(0)=0. Elle sera telle qu'en une suite de points m_n, on ait exactement $\alpha^0_{m_n}=\frac{1}{2+k_n}$. Si alors il existe une fonction de majoration k(x,n), on doit avoir : $(\forall n)(\forall p)[p\geq n \Longrightarrow k(0,m_n)<k_p]$. Mais alors on pourrait en déduire une énumération strictement croissante de K, qui serait donc récursif, d'où la contradiction. Il reste à exhiber l'énumération f. On utilisera une fonction récursive auxiliaire g, et, pour chaque valeur, on calculera d'abord g, puis f.

On posera g(0)=f(0)=0, et, pour n>0, on aura :

$$g(n)=\begin{cases} g(n-1) & \text{si } \alpha^0_{n-1}\neq\frac{n-1}{2+k_{g(n-1)}} \\ g(n-1)+1 & \text{sinon}\end{cases} \quad \text{et} \quad f(n)=\begin{cases} n & \text{si } \alpha^0_{n-1}>\frac{n-1}{2+k_{g(n)}} \\ 0 & \text{si } \alpha^0_{n-1}\leq\frac{n-1}{2+k_{g(n)}}\end{cases}$$

Remarquons que le cas d'égalité dans la définition de f est impossible par définition de g. ⊠

Rappelons deux définitions classiques :

Définition 20 (*)

Un ensemble $A=\{a_0,a_1,\ldots,a_n,\ldots\}$ *infini, donné par ordre croissant, est* majoré par une fonction h *si* $(\forall n)[h(n)\geq a_n]$.

Un ensemble est hyperimmune *s'il n'est majoré par aucune fonction récursive.*

Proposition 6 (*)

Tout ensemble hyperimmune est de densité nulle par rapport à toute injection récursive.

En effet, $\alpha_n=|\{f(0),\ldots,f(n)\}\cap A|$ dans ce cas. Si d(A,f)>0, soit d(A,f)=δ, et donc $(\forall\varepsilon\in\mathbb{R})(\exists p\in\mathbb{N})[n\geq p \Longrightarrow \delta-\varepsilon<\frac{\alpha_n}{n}]$. Posons $s=\mu k[\frac{1}{2^k}<\delta]$ et prenons $\varepsilon=\delta-\frac{1}{2^s}$, on en déduit que $n\geq p \Longrightarrow \frac{1}{2^s}<\frac{\alpha_n}{n}$ ce qui fournit une majoration f' de A, qui ne peut donc être hyperimmune. f' est évidemment définie par :

$f'(n)=f(\max(p,2^s)+n\ 2^s)$ ⊠

Remarque

Ainsi, les notions de densité d'un ensemble par rapport à une fonction et de densité d'énumération sont bien différentes, malgré leurs formulations voisines. En particulier, tous les ensembles que nous considérerons sont récursivement énumérables (r.e.), alors qu'un ensemble hyperimmune ne l'est jamais. Ceci veut dire qu'on ne s'intéresse pas aux ensembles énumérés, mais aux procédés d'énumération eux-mêmes.

Proposition 7

(1) Toute énumération à domaine fini est de densité nulle.

(2) Toute énumération d'image finie et de domaine infini n'est pas de densité nulle.

La démonstration est triviale. ⊠ Ceci correspond à deux éventualités pratiques. Dans la première, on effectue une recherche qui s'arrête après un nombre fini de pas. Dans la seconde, on "boucle". Les contraintes à apporter sur les recherches ne peuvent cependant pas s'appuyer sur cette seule notion de densité pour imposer l'arrêt d'une recherche, car cette notion est asymptotique. D'autre part, il se peut qu'une recherche intéressante énumère certains points avec une densité non nulle, et on ne peut donc se borner à imposer une borne de nullité uniforme.

3.1.2. CONTROLE

Pour obtenir des stratégies intéressantes, il faut qu'elles "progressent" assez. Nous avons vu que la notion de densité nulle était assez mal adaptée. Une autre voie d'approche peut consister à exiger de déterminer effectivement à partir de quand l'énumération doit produire de nouveaux points. On peut formaliser ceci de plusieurs façons : les deux premières ont été introduites à l'occasion de travaux sur l'apprentissage (en théorie de la récursivité) par Feldman (1972) et Wharton (1974).

Définition 21

Une énumération f est effectivement approximativement ordonnée *(EAO) s'il existe une fonction récursive $\tilde{f}$ telle que :*

$(\forall n)(\forall m)[m \geq \tilde{f}(n) \Longrightarrow f(m) > f(n)]$

Une énumération f est effectivement quasi-ordonnée *(EQO) s'il existe une fonction récursive $\tilde{f}$ telle que :*

$(\forall n)(\forall m)[m \geq \tilde{f}(n) \Longrightarrow [f(m) > f(n)$ *ou* $(\exists p)[p \leq f(n) \ \& \ f(p)=f(m)]]]$

Une énumération f est à production effectivement contrôlée *(PEC) s'il existe une fonction récursive $\hat{f}$ telle que :*

$(\forall n)(\forall m) \ m \geq \hat{f}(n) \Longrightarrow |\{f(0),\ldots,f(m)\}| > n|$

Remarque

Les définitions de Feldman et Wharton sont légèrement différentes, car elles concernent des suites $(y_0,y_1,\ldots)$ EAO ou EQO par une fonction. Les définitions données ici correspondent donc au cas particulier de la suite $(0,1,\ldots)$.

Proposition 8

On a les relations :

(1) f EAO $\Longrightarrow$ f EQO
(2) f EAO $\Longrightarrow$ f PEC
(3) f récursive $\Longrightarrow$ f PEC
(4) f EQO $\not\Longrightarrow$ f PEC

(1) et (2) sont évidents. Pour le (4), prenons un contre-exemple. Soit $A=\{a_0,a_1,\ldots\}$ un ensemble hyperimmune. On considère l'énumération (non récursive) f définie par $f(n)=p+1$ si $a_p \leq n < a_{p+1}$ et $f(0)=0$ si $n<a_p$.
Comme elle est croissante, elle est EQO par l'identité, mais n'est PEC par aucune fonction récursive $\hat{f}$, car on a $\hat{f}(n) \geq a_n$ pour tout n, et $\hat{f}$ majore donc A. Pour démontrer le (3), construisons d'abord la fonction récursive h telle que :
$D_{h(n)}=\{f(0),\ldots,f(n)\}$. h est récursive puisque f l'est.
$\hat{f}$ s'obtient alors facilement :

$$\begin{cases} \hat{f}(0)=0 \\ \hat{f}(n+1)=\mu m[h(m)>h(n)] \end{cases} \qquad \boxtimes$$

En pratique, les énumérations considérées doivent être récursives. Ceci limite l'intérêt des énumérations PEC. Mais on peut s'intéresser à une famille d'énumérations à production contrôlée par une même fonction. La proposition suivante montre qu'on n'atteint plus toutes les énumérations récursives.

Proposition 9

Soit f une fonction récursive "de temps", c'est-à-dire strictement croissante et supérieure à l'identité§. Alors l'ensemble des (indices des) énumérations PEC par f est productif.

Soit donc $A=\{x \mid \varphi_x \in R^1 \ \& \ \varphi_x$ PEC par f$\}$. Prenons un sous-ensemble r.e. quelconque W_a de A. On sait qu'il existe une fonction primitive récursive g telle que $W_a=\mathrm{Im}\,\varphi_{g(a)}$ et que $\varphi_{g(a)}$ soit totale dès que $W_a \neq \emptyset$. Définissons une nouvelle énumération h par :

$$h(n) = \begin{cases} n \text{ si } \varphi^{(0)}_{g(a)} \text{ ne converge pas en moins de n pas (donc } h(0)=0) \\ \min[D_{k\varphi^{(n-z_o)}_{g(a)}(n-z_o),f(n+1))} - \{h(0),\ldots,h(n-1)\} - \{\varphi^{(n)}_{g(a)}(n-z_o)\}] \text{ si } \varphi^{(0)}_{g(a)} \downarrow \text{ en } z_o \text{ pas.} \end{cases}$$

---

§ - On trouve parfois le même terme pour une fonction simplement croissante et supérieure à l'identité.

où k est définie par $D_{k(i,n)}=\{\varphi_i(0),\ldots,\varphi_i(n)\}$, soit

$$k(i,n) = \sum_{j=0}^{n} 2^{\varphi_i(j)}$$

et est donc partielle récursive, et totale sur les indices i de fonctions totales. Il est clair (thèse de Church et théorème s-m-n) que h est récursive et qu'on peut lui trouver un indice effectivement en fonction de a, soit $h=\varphi_{l(a)}$. l est la fonction productive cherchée. En effet, si $W_a=\emptyset$, $h=\lambda x[x]$ est certainement PEC par f, et sinon on a produit une énumération PEC par $\lambda x[x]$, donc par f, égale à l'identité tant que $\varphi^{(0)}_{g(a)}$ ne converge pas, et prenant ensuite ses valeurs dans chacune des énumérations sans être égale à aucune d'elles (diagonalisation). ⊠

Remarquons qu'on aurait pu obtenir une fonction h de façon un peu plus simple en n'exigeant pas qu'elle prenne en tout point $n\geq z_0$ une valeur de $\varphi_{\varphi_{g(a)}}(n-z_0)$. D'autre part, il ne semble pas possible de généraliser cette démonstration au au cas où f est seulement une fonction de temps partielle récursive définie sur une section commençante de IN.

On peut également établir certaines relations entre les contraintes de densité et de contrôle.

Proposition 10

(1) f EA0 $\Longrightarrow$ f de densité nulle
(2) f EQ0 $\not\Longrightarrow$ f de densité nulle
(3) f de densité effectivement uniformément nulle $\not\Longrightarrow$ f EQ0

(1) est évident, car chaque valeur ne peut être atteinte qu'un nombre fini de fois.

Pour (2), il suffit de considérer le premier exemple (π_1) suivant la définition 18. Il est clair que π_1 est EQ0 par $\lambda x[2^x+1]$, et n'est de densité nulle en aucun point.

Pour (3), considérons encore une énumération récursive sans répétition de K, soit $\{k_0,k_1,\ldots,k_n,\ldots\}$. Elle est évidemment de densité uniformément effectivement nulle (avec k(n)=n+1), mais n'est pas EQ0, sinon, comme on l'a déjà vu, K serait récursif. ⊠

Il existe certaines énumérations qui ont la propriété de se "contenir" strictement elles-mêmes, donc de se répéter.

Définition 22

Une énumération f est auto-enchâssée *s'il existe une fonction de temps $\tilde{f}$ telle que $(\forall n)[f(n)=f\tilde{f}(n)]$.*
Si $\tilde{f}$ est récursive, f est récursivement auto-enchâssée.

Remarquons qu'une énumération auto-enchâsée l'est une infinité de fois, via $\{\tilde{f}^n\}_{n>0}$. Intuitivement, on peut donc penser que l'auto-enchâssement a un rapport direct avec la propriété de densité nulle. En fait,

Proposition 11

Il existe des énumérations récursives auto-enchâssées de densité nulle et d'autres de densité non nulle en tout point.

-1- Considérons l'énumération f définie par (cf [48])

$$\begin{cases} f(2n)=n^2 \\ f(2n+1)=f(n) \end{cases}$$

Il est clair que f est auto-imbriquée, avec $\tilde{f}=\lambda n[2n+1]$. On peut le représenter par un schéma où apparaissent les imbrications :

n	0	1	2	3	4	5	6	7	8	9	10	11
f	0	0	1	0	4	1	9	0	16	4	25	1
$f\tilde{f}$		0		0		1		0		4		1
$f\tilde{f}^2$				0				0				1

De plus, on calcule facilement par récurrence sur p le numéro de la p-ième image réciproque de n^2, soit $\beta_p^{n^2} = 2^p(2n+1)-1 = \langle p,n\rangle$. On en déduit immédiatement que $\alpha^{n^2}_{\langle p,n\rangle} = \frac{p}{\langle p,n\rangle} < \frac{p}{\langle p,0\rangle} \xrightarrow[p\to\infty]{} 0$, et donc f est de densité uniformément effectivement nulle (les points $\langle p,n\rangle$ correspondent aux maxima locaux de α_q^{n2}, $q\in\mathbb{N}$).

-2- Soit maintenant l'énumération f définie par :

f(n) = le nombre de "0" consécutifs suivant immédiatement le premier "1" dans l'expansion binaire de n.

f est récursive primitive. Pour le voir, soit g définie par

$$\begin{cases} g(0)=0 \\ g(n)=\max x\leqslant n\ [2^x\leqslant n] \text{ si } n>0 \end{cases}$$

g est récursive primitive (voir par exemple [35]), et

$$f(n)=g(n) \dot{-} \min(1, g(n-2^{g(n)}))$$

D'autre part, f est auto-imbriquée, avec encore $\tilde{f}=\lambda n[2n+1]$, puisque passer de n à 2n+1 revient à ajouter un "1" à la fin de l'expansion binaire de n. On aura le schéma :

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
f	0	0	1	0	2	1	0	0	3	2	1	1	0	0	0	0	4	3	2	2	1
$f\tilde{f}$		0		0		1		0		2		1		0		0		3		2	
$f\tilde{f}^2$				0				0				1				0				2	

Enfin, $(\forall x)[\delta_f(x)=\frac{2}{1+2^{x+2}} \neq 0]$. Pour le voir, on peut raisonner sur les longueurs l(n) des expansions binaires.

D'abord $l(n)\leqslant x+1 \Longrightarrow n\leqslant 2^{x+1}-1$, donc $|\{n \mid l(n)\leqslant x+1\}|=2^{x+1}$, et le seul n tel que f(n)=x dans cet ensemble est 2^x.

Ensuite, si l(n)=x+p+1 avec p>0, les expansions binaires de n s'écrivent $1\,i_0\,i_1\,\ldots\,i_x\,u$, où $u\in\{0,1\}^{p-1}$. Les seules expansions telles que f(n)=x sont celles où $i_0=\ldots=i_{x-1}=0$ et $i_x=1$, il y en a donc $\frac{1}{2^{x+1}}$.

Finalement, on a donc $(\forall p)\left[\frac{\alpha^x_{2^{x+p+1}-1}}{2^{x+p+1}-1} = \frac{1}{2^{x+1}}\right]$. Ceci n'implique pas $\delta_f(x)=\frac{1}{2^{x+1}}$. En effet, dans chaque segment $\{2^{x+p+1},\ldots,2^{x+p+2}-1\}$, correspondant à une longueur x+p+1, il y a 2^{p-1} images réciproques de x, consécutives et constituant le second sous-segment de longueur 2^{p-1}, soit $\{2^{x+p+1}+2^{p-1},\ldots,2^{x+p+1}+2^p-1\}$. La densité minimale dans le segment est donc obtenue au point $2^{x+p+1}+2^{p-1}-1$, et on a

$$\frac{\alpha^x_{2^{x+p+1}+2^{p-1}-1}}{2^{x+p+1}+2^{p-1}-1} = \frac{2}{1+2^{x+2}-\frac{1}{2^{p-1}}} \xrightarrow[p\to\infty]{} \frac{2}{1+2^{x+2}} \qquad \boxtimes$$

Remarquons que $\limsup_{n\to\infty} \frac{\alpha^x_n}{n} = \frac{2}{1+2^{x+1}}$

On peut donc provisoirement conclure que les contraintes "globales" à apporter à des énumérations "intéressantes" ne concerneront pas l'auto-enchâssement, mais peut-être, et de façon indépendante, les critères de densité et de contrôle. En particulier, on pourra assez aisément tester qu'une énumération est PEC par une fonction donnée, mais plus difficilement qu'elle est EAO, EQO, ou de densité effectivement nulle, toujours avec une fonction de contrôle donnée à l'avance.

3.1.3. HEURISTIQUES ET CONSTRUCTIVITE

Quand on se donne à l'avance une fonction de contrôle, les critères précédents déterminent une certaine classe d'énumérations possibles. Young (1969) a introduit une notion voisine, celle de "technique d'énumération"[§]. Nous allons voir que les procédés d'énumération utilisables, c'est-à-dire constructibles au moyen d'un ensemble fini de fonctions de base (n'oublions pas que le but ultime est de parcourir certaines arborescences de calculs), ne sont pas des techniques d'énumération au sens de Young. On appellera heuristique (d'énumération) un procédé qui permet de définir une classe de recherches constructibles et soumises à d'éventuelles contraintes du type précédent. Le choix de ce terme est un peu arbitraire, mais semble correspondre à un usage récent (surtout en IA) qui l'assimile à une règle générale plutôt qu'à un procédé ad hoc (tactique). D'autre part, on peut réserver le mot "stratégie" pour "organisation et principes généraux".

Définition 23 (Young)

Une technique d'énumération *est une fonction récursive* $E \in R^2$ *telle que, pour tout ensemble r.e. W, il existe un entier e tel que* $W = \bigcup_n D_{E(e,n)}$. *On écrit alors* $W = W_e$ *et on dit que* e est un indice de W relativement à E.

Si E est une technique d'énumération, on appelle E' la technique d'énumération associée définie par :

$$(\forall e)(\forall n)[D_{E'(e,n)} = \bigcup_{j \leq n} D_{E(e,n)}]$$

Ainsi, e code la "structure" d'un dispositif qui énumère à chaque pas n un nombre fini, éventuellement nul, d'éléments de W_e. Remarquons que la technique E qui définit l'indexation standard (W_e = Dom φ_e) est celle qu'on emploie dans les preuves en "queue d'aronde" ("dove-tailing", [48] p. 60).

Une heuristique doit être un moyen de définir une recherche pour chaque cas particulier, et est donc l'analogue d'une technique d'énumération. Mais nous allons voir qu'elle ne peut énumérer tous les r.e. si elle doit "construire" les recherches $\{\lambda n E(e,n)\}_{e \in \mathbb{N}}$ à partir d'un ensemble fixe de fonctions de base.

Définition 24

Une énumération g est constructible par rapport à un ensemble $F = \{f_j\}_{j \in J}$ *de fonctions (d'une variable) si*

$$(\forall n)[g(n+1) \neq g(n) \Longrightarrow (\exists j \in J)[g(n+1) = f_j\, g(n)]]$$

Elle est effectivement constructible par rapport à F *s'il existe une fonction récursive* $\tilde{g}$ *telle que*

$$(\forall n)[g(n+1) = f_{\tilde{g}(n)} g(n)],$$

avec la convention que f_o *est l'identité, sans perte de généralité, et que J est une section commençante de* $\mathbb{N}$.

Si $F \subset R^1$, une énumération effectivement constructible par rapport à F est récursive. Les cas pratiques sont bien sûr ceux où F est fini.

Définition 25

Un sous-ensemble A de $\mathbb{N}$ *est une* F-chaîne *s'il existe une énumération F-constructible de A.*

Un sous-ensemble A de $\mathbb{N}$ *est* F-libre *si*

$$(\forall x, y \in A)[x \neq y \Longrightarrow (\forall i, j \in J)[x \neq f_i(y) \ \&\ y \neq f_j(x)]]$$

---

§ - Arbib (1969, ch. 7) a condensé l'essentiel de cet article de Young.

Théorème 6

Pour tout ensemble fini $F=\{f_0,\ldots,f_k\}$ de fonctions récursives,

(1) il existe des ensembles F-libres arbitrairement grands, et il existe même une technique d'énumération E qui, à tout code $e=\tau^*(x_0,\ldots,x_k)$ d'indices des f_j, associe l'énumération d'un ensemble W r.e. ne contenant que des F-chaînes finies et telle que $\{D_{E(e,n)}\}$ soit une famille d'ensembles F-libres à n éléments.

(2) aucun ensemble hyperimmune ne contient de F-chaîne infinie.

Ainsi, un ensemble F-libre doit être "assez rare" par rapport à F, ce qui est le cas de tout hyperimmune "presque partout". Un problème ouvert consiste à obtenir effectivement un r.e. F-libre infini, s'il existe.

Remarquons une certaine analogie entre le (1) et un des résultats de [46] sur l'existence d'une classe r.e. d'ensembles finis "difficile à énumérer" (il est impossible de l'énumérer sans répétition).

Avant de démontrer ce théorème, on peut donner un résultat un peu plus précis dans le cas où F ne contient qu'une seule fonction f non égale à l'identité.

Proposition 12

Soit f une fonction récursive quelconque.

(1) si f est d'image finie, $\overline{Imf}$ est récursif, f-libre et infini, mais on ne peut trouver effectivement son indice en fonction d'un indice de f.

(2) si f a une image infinie, il existe un ensemble f-libre r.e. infini dont on peut trouver effectivement un indice à partir d'un indice de f.

(1) Il est évident que $\overline{Imf}$ est f-libre. Savoir trouver effectivement son indice voudrait dire que

$$(\exists h\in R^1)(\forall z)[\mathrm{Im}\,\varphi_z \text{ finie} \Longrightarrow W_{h(z)}=\overline{\mathrm{Im}\,\varphi_z}]$$

Soit ψ définie par

$$\psi(x,y) = \begin{cases} 1 & \text{si } y=0 \text{ et } x\in K \\ \uparrow & \text{sinon} \end{cases}$$

$\psi\in P^2$ par Church, et donc (par s-m-n) il existe $g\in R^1$ telle que

$$W_{g(x)} = \begin{cases} \{0\} & \text{si } x\in K \\ \emptyset & \text{si } x\in\overline{K} \end{cases}$$

Alors $W_{g(x)}$ est toujours fini, et donc, par hypothèse sur h,

$$W_{hg(x)} = \begin{cases} \mathbb{N}-\{0\} & \text{si } x\in K \\ \mathbb{N} & \text{si } x\in\overline{K} \end{cases}$$

et donc $\overline{K}=\{x\,|\,\varphi_{hg(x)}(0)\downarrow\} = \{x\,|\,(\exists y)[\varphi_{hg(x)}(0)\downarrow \text{ en moins de y pas}]\}$ serait r.e. par projection, d'où une contradiction.

(2) On va construire W comme l'union croissante d'ensembles f-libres A_n. On aura :

$$A_0 = \emptyset$$

$$A_{n+1} = A_n \cup \{\mu x[x>f(A_n) \;\&\; f(x)\notin A_n]\}$$

Pour tout n, on peut trouver un tel x, sans quoi on aurait $Imf \subset A_n \cup f(A_n)$, une contradiction. ⊠

Démonstration du théorème

Ces deux cas (image finie ou infinie) doivent être réunis pour produire effectivement l'ensemble W du théorème sans savoir a priori -ce qui est indécidable- si Imf est finie ou non. Donnons d'abord une démonstration pour le cas simple qu'on vient d'envisager.

(1) a-

On construit encore une suite A_i d'ensembles finis tels que $|A_i|=i$. Pour ceci, supposons qu'on a disposé $\mathbb{N}$ en une liste verticale et qu'on peut associer à chaque entier deux marques éventuelles : un "+" pour indiquer qu'il est dans A_n, et un "marqueur" $\boxed{m}$ parmi l'infinité $\boxed{0}$, $\boxed{1}$,...

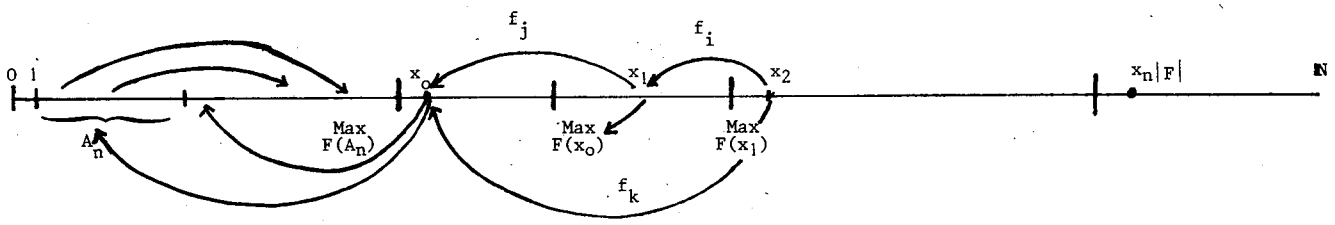
On prendra encore, au pas 0, $A_0=\emptyset$, et aucun "+" ni aucun marqueur n'est utilisé. Au pas n+1, on descend dans la liste jusqu'à max $f(A_n)+1$ et on examine successivement les entiers $x(>f(A_n))$:

- si $f(x)\notin A_n$, on met un "+" à x $(A_{n+1})=A_n\cup\{x\})$, on efface tous les marqueurs et on va au pas n+2.
- si $f(x)\in A_n$, ne porte pas de marqueur, on associe le marqueur $\boxed{x}$ à f(x) et on continue l'examen (avec x+1).
- si f(x) A_n porte le marqueur $\boxed{y}$, on efface le "+" de f(x), on en met un à y et à x $(A_{n+1}=A_n+\{x,y\}-\{f(x)\})$, on efface les marqueurs et on va au pas n+2.

Le procédé dépend évidemment effectivement de f.

(1) b-

Pour le cas général, il faut tenir compte du fait qu'un point x peut avoir plusieurs images $f_j(x)$ dans A_n. L'idée de la démonstration est, pour construire A_{n+1}, de fabriquer un ensemble fini assez grand pour qu'il contienne un élément non relié à A_n ou n+1 éléments F-libres entre eux, mais peut-être reliés à A_n. On pourra considérer le schéma suivant :



Au pas 0, on prend encore $A_0=\emptyset$. Au pas n+1, on pose $A'_n=\bigcup_{i=0}^{n} A_i$, $t=\mathrm{Max}(A'_n\cup F(A'_n))$, et $C_n=\emptyset$, puis, tant que $|C_n|\leq n|F|$, on fait $C_n:=C_n\cup\{t+1\}$, et si $F(t+1)\cap A'_n=\emptyset$, on pose $A_{n+1}=A_n\cup\{t+1\}$ et on va au pas n+2.
Sinon, on fait $t:=\mathrm{Max}(t+1,F(t+1))$.

Si on arrive à un ensemble $C_n=\{x_0,\ldots,x_{n|F|}\}$, on pose $C'_n=C_n$, $B_n=\{x_{n|F|}\}$, et, tant que $|B_n|\neq n+1$, on modifie C'_n et B_n par : $C'_n:=C'_n-F(\min B_n)$, $B_n:=B_n\cup\{\max C'_n\}$ (f_0 est l'identité).
Tous les éléments $x_0,\ldots,x_{n|F|}$ ont au moins une image dans A'_n, donc au plus $|F|-1$ dans C_n, et donc $|B_n|=n+1$ est obtenu en moins de n sous-pas. On prend alors $A_{n+1}=B_n$ et on va au pas n+2.

L'énumération des A_n est bien effective en F, d'où l'existence de $E\in R^2$, avec $(\forall n)[A_n=D_{E(e,n)}]$.

Aucune F-chaîne de W ne peut être infinie puisque, par construction,

$(\forall x,y\in W)(\forall i\in J)[f_i(x)=y \Longrightarrow y<x]$.

(2) Supposons que A contienne une F-chaîne infinie. Alors il existerait une énumération F-constructible infinie, non nécessairement récursive, $\{a_0,a_1,\ldots\}$. Soit alors h définie par $h(p)=\text{Max } F^p(a_0)$, avec la convention que $F^0(a_0)=\{a_0\}$. h est récursive puisque F ne contient que des fonctions récursives, et h majore A, qui ne peut donc être hyperimmune. ⊠

On retrouve à l'occasion de ce théorème une propriété analogue à la retraçabilité. Par contraste, on peut rappeler quelques résultats sur les ensembles retraçables, qui ont quelque analogie avec les structures à "rappel" (définition 8).

Définition 26 (*)

Un ensemble A⊂ℕ est retraçable *si*

$$(\exists \psi\in P^1)(\forall x)[x\in A \Longrightarrow \psi(x)\downarrow \ \&\ \psi(x)=\begin{cases} x \text{ si } x=\min A \\ \max y<x[y\in A] \text{ sinon}\end{cases}]$$

Remarquons que les ensembles W du théorème précédent peuvent contenir des sous-ensembles infinis retraçables (par des fonctions récursives). Cette notion se rapproche de celle de "fonction de rappel" d'une structure de travail.

Proposition 13 (*)

(1) A retraçable ⟹ A récursif ou immune
(2) A retraçable et $\bar{A}$ r.e. ⟹ A récursif ou hyperimmune
(3) Il existe des ensembles retraçables ni récursifs ni hyperimmunes
(4) Il existe des ensembles retraçables hyperimmunes
(5) Si A est infini et $\bar{A}$ r.e., A hyperhyperimmune ⟺ A n'a pas de sous-ensemble infini retraçable.

N'ayant pu trouver les démonstrations dans la littérature, je les inclus pour complétude.

(1) D'abord, si A est retraçable et B⊂A infini, A est B-récursif. En effet,

$h(x)=\mu y[y>x \ \&\ y\in B]$ est totale et B-récursive, et

$$\chi_A(x)=\begin{cases}1 \text{ si } (\exists p\leq h(x))[\psi^p h(x)=x] \\ 0 \text{ sinon}\end{cases} \quad \text{est B-récursive.}$$

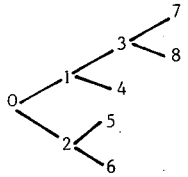
Ensuite, si A est retraçable, r.e. et infini, A est récursif. En effet, soit $f\in R^1$ telle que A=Imf, $h(x)=f(\mu y[x<f(y)])$ est récursive, et χ_A, avec la même définition, l'est encore.

Si donc A est retraçable et non-immune, A contient B r.e. infini, donc A, B-récursif, est r.e. et infini, donc récursif.

(2) Si A n'est pas hyperimmune, il est majoré par une fonction $f\in R^1$.

Pour tester si $x\in A$, on calcule f(x) et on effectue en parallèle les calculs de $\psi(0),\ldots,\psi f(x)$ en même temps qu'on énumère $\bar{A}$. Si $y\in\bar{A}$, on arrête le calcul de $\psi(y)$ et on ôte y de la liste l initialement égale à $\{0,\ldots,f(x)\}$. Au bout d'un nombre de pas fini, l est constituée d'une unique chaîne φ-retraçable $\{a_0,\ldots,a_z\}$, avec $x<z\leq a_z\leq f(x)$. Il suffit alors de tester si $x\in l$.

(3) Soit l'arbre binaire étiqueté de la façon suivante :



i.e., on prend $\psi(n)=\left\lfloor\frac{n+1}{2}\right\rfloor$. Toutes les branches sont φ-retraçables et aucune n'est hyperimmune, puisqu'elles sont toutes majorées par $\lambda x[2^x]$. On prendra pour A l'ensemble $\{a_0,a_1,\ldots\}$ défini par :

$$\begin{cases} a_0 = 0 \\ a_{n+1} = \begin{cases} 2a_n+1 & \text{si } n\in K \\ 2a_n+2 & \text{si } n\in K\end{cases}\end{cases}$$

A n'est pas récursif, sinon, pour tester $n\in K$, il suffit de trouver (effectivement) $a_{n+1}=\mu x[2^{n+1}-1\leq x\leq 2^{n+2}-2 \ \&\ x\in A]$, et $n\in K \Longleftrightarrow a_{n+1}$ impair.

(4) Soit $B=\{b_0,\ldots,b_n,\ldots\}$ un ensemble hyperimmune. On peut toujours supposer que $b_0=0$. On prendra alors $A=\{a_0,a_1,\ldots\}$ avec $a_0=b_0=0$ et $a_{n+1}=<b_n,a_n>$. $\lambda xy[<x,y>]$ est croissante en x et en y, A est retraçable via π_2, $a_{n+1}\geq b_n$, et donc si f majorait A, $\lambda_n[f(n+1)]$ majorerait B, une contradiction.

(5) Cette caractérisation a été trouvée par Yates (1962), on trouvera sa démonstration dans Rogers (1967) p. 177. Comme on vient de voir qu'un ensemble hyperimmune peut être retraçable, on a une propriété plus forte. Remarquons enfin que le (4) indique qu'on ne peut améliorer le (2) du théorème 6 : il existe des hyperimmunes non F-libres. ⊠

Il reste maintenant à donner une définition plus précise d'une heuristique.

Définition 27

Une heuristique (d'énumération) *est une fonction récursive* $H\in R^2$ *telle que* $(\forall e,n)[|D_{H(e,n)}|=1]$. *Il existe donc* $n\in R^2$ *telle que* $(\forall e,n)[D_{H(e,n)}=\{h(e,n)\}]$.

Soit $F=\{f_j\}_{j\in I}$ *un ensemble de fonctions, avec les conventions précédentes. Une heuristique H est* F-constructible *s'il existe une fonction récursive* $d\in R^2$ *telle que* $(\forall e,n)[h(e,n+1)=f_{d(e,n)}h(e,n)]$.

Conclusions

Dans la pratique, on sera amené à définir des heuristiques constructibles sur un ensemble fini F de fonctions récursives. Le théorème 6 indique que la classe des recherches engendrées (pour e variant) par une telle heuristique n'est jamais complète, et même qu'une heuristique n'est jamais une technique d'énumération.

Proposition 14

(1) Si F est un ensemble fini de fonctions récursives, aucune heuristique F-constructible n'est une technique d'énumération.

(2) Pour toute heuristique H, la propriété, pour un r.e., d'être énuméré par une recherche engendrée par H est indécidable.

(3) Par contre, cette propriété est dans Σ_1.

(1) découle immédiatement du théorème 6, puisque l'ensemble W construit est r.e. et non F-constructible.

(2) provient du théorème de Rice. En effet, reprenons l'indexation standard ($W_x=\mathrm{Dom}\,\varphi_x$). Alors $B=\{x|(\exists e)[W_x=\bigcup_{n\in\mathbb{N}} D_{H(e,n)}$ ou $(\exists p)[W_x=\bigcup_{n\leq p} D_{H(e,n)}]]\}$ n'est pas trivial ($\neq\emptyset$ et $\neq\mathbb{N}$) et n'est donc pas récursif[§].

(3) est une conséquence directe du théorème de Rice et Shapiro ([48 , p. 324) en prenant pour fonction d'énumération d'ensembles finis[§§] $f=\lambda u[H(\pi_1(u),\pi_2(u))]$. ⊠

Nous allons maintenant revenir aux systèmes définitionnels et aux G-machines pour voir comment réaliser un système définitionnel muni d'une heuristique au moyen d'un système de G-machines. Intuitivement, le non-déterminisme se traduira par l'existence d'un module du système comportant une structure de travail arborescente, et les fonctions qui serviront à construire les heuristiques utilisées seront justement des fonctions de déplacement dans cette arborescence. Le fait que chaque noeud a un nombre borné de fils garantira enfin l'arrêt si on utilise certains contrôles de densité ou de production sur des arborescences finies.

---

§ - Voir Rogers (1967) ou Young (1969) pour la démonstration de ce théorème.

§§ - $B\in\Sigma_1$ ssi la classe de r.e. correspondante $\zeta=\{A|(\exists x)[x\in B \;\&\; W_x=A]\}$ est telle que $\zeta=\emptyset$ ou $(\exists f\in R^1)[\zeta=\{A|A \text{ r.e. } \&\; (\exists u)[D_{f(u)}\subset A]\}]$.

3.2. SYSTEMES DE G-MACHINES

On peut combiner des machines abstraites en utilisant des types de composition différents. Hermes (1965) introduit des diagrammes pour définir des MT à partir d'autres MT, toutes sur le même alphabet A : les noeuds sont étiquetés par les MT de base et les arcs par $A\cup\{\varepsilon\}$, où $\varepsilon\notin A$. Cette composition ne comporte pas d'autre contrainte que le déterminisme. Chauché (1974) a encore défini des compositions "linéaire", "ultralinéaire" et "récursive" sur les transducteurs, cette dernière opération permet de conserver certaines propriétés (de décidabilité ou de complexité) en augmentant la puissance du système obtenu. D'autres types de composition (en série, en parallèle, en cascade) ont été définis par Krohne et Rhodes[§]. Enfin, les modèles de "réseaux de transitions" (Conway 1963, Woods 1969, Lomet 1973) sont très proches des "systèmes d'automates finis" de l'Ecole de Vienne (Ollongren 1974). Et sans doute pourrait-on encore citer d'autres approches.

Certains de ces modèles sont plus "logiciels", et d'autres plus "matériels". Les premiers composent des sous-machines éventuellement complexes, et cette composition peut elle-même nécessiter un algorithme spécial qui travaille avec une pile (ce sera le "moniteur" d'un système informatique). Les seconds cherchent à décomposer une machine complexe en des éléments très simples tirés d'une collection finie et composés de manière simple. Les théorèmes dûs à Krohne et Rhodes montrent par exemple qu'on ne peut réaliser tous les automates finis avec un jeu fini d'automates élémentaires si la composition ne comporte pas de boucle.

On pourrait définir pour les G-machines une composition sans contrainte analogue à celle d'Hermes. Mais, si elle est agréable pour les preuves de calculabilité des fonctions récursives, elle est trop générale en pratique : on pourrait dire qu'elle correspond à une programmation "non structurée". La composition sera obtenue au moyen de l'appel récursif entre sous-machines du système, qui partagent une pile de contrôle. Dans le cas des systèmes hiérarchisés, l'appel récursif induit un ordre partiel strict, la taille de la pile de contrôle est bornée et la fonction d'organisation du système peut être effectuée par un automate d'états finis.

Définition 28

Une G-machine à pile de contrôle *est une G-machine munie d'une structure de travail particulière, la pile* $P=(\mathbb{N},\{-1,0,+1\})$, *sur laquelle elle travaille indépendamment des autres structures. Plus précisément, si* $P=(S_o,F_o)$, $a_o^o=b\in A_o$ *et* $I'=I-\{0\}$, *et si on note* $\Sigma_{I'}=\prod_{i\in I'} A_i^{h_i}$, α *est telle que :*

$$(\forall e\in E)(\forall a,a'\in A_o)(\forall x,x'\in\Sigma_{I'},$$

$$[[a=a'=b \;\&\; \pi_o\alpha_2(e,(a,x))=a' \Longrightarrow \pi_o\alpha_3(e,(a,x))=0]$$

$$\&\; [a=b \;\&\; a'\neq b \;\&\; \pi_o\alpha_2(e,(a,x))=a' \Longrightarrow \pi_o\alpha_3(e,(a,x))=+1 \;\&\; (\forall i\in I')[\pi_i\alpha_3(e,(a,x))=0] \;\&\; \alpha(e,(a,x))=\alpha(e,(a,x'))]$$

$$\&\; [a\neq b \;\&\; a'=b \;\&\; \pi_o\alpha_2(e,(a,x))=a' \Longrightarrow \pi_o\alpha_3(e,(a,x))=-1 \;\&\; (\forall i\in I')[\pi_i\alpha_3(e,(a,x))=0] \;\&\; \alpha(e,(a,x))=\alpha(e,(a,x'))]]$$

Cette définition et les suivantes généralisent celles de Lomet (1973) sur les DPDA imbriqués. Remarquons que, dans le cas général, la machine peut commencer un calcul avec la pile vide et le finir sans qu'elle le soit. Pour le système de sous-machines qu'on va définir, ceci correspond à pouvoir "sortir" à n'importe quel niveau de récursion.

Définition 29

Une G-machine à pile de contrôle strict *transforme de plus toute configuration où la pile est vide en une configuration finale où elle l'est encore.*

On peut classer les états d'une G-machine à pile de contrôle suivant leur action sur la pile.

§ - Voir Arbib (1969) ch. 8.

Définition 30

Si $(\exists x\in\Sigma_I')(\exists a'\in A_o)[\pi_o\alpha_2(e,(\flat,x))=a'\neq\flat \ \&\ \alpha_1(e,(\flat,x))=e']$, *e est un* état d'appel *et e' un* état d'entrée.

Si $(\exists x\in\Sigma_I')(\exists a\in A_o)[a\neq\flat \ \&\ \pi_o\alpha_2(e,(a,x))=\flat \ \&\ \alpha_1(e,(a,x))=e']$, *e est un* état de sortie *et e' un* état de retour.

Un état e_k est connecté à un état e_l *si l'une des trois conditions suivantes est réalisée, et on notera alors* $e_k \overset{*}{\underset{M}{\vdash\!\!-\!\!-}} e_l$.

(1) $(\exists x\in\Sigma_I')[\pi_o\alpha_2(e_k,(\flat,x))=\flat \ \&\ \alpha_1(e_k,(\flat,x))=e_l]$. e_k *est alors* simplement connecté *à* e_l. *On notera* $e_k \underset{M}{\vdash\!\!-\!\!-} e_n$.

(2) $(\exists a\in A_o)(\exists e_m,e_n\in E)(\exists x\in\Sigma_I')[e_m \overset{*}{\underset{M}{\vdash\!\!-\!\!-}} e_n \ \&\ a\neq\flat \ \&\ \pi_o\alpha_2(e_k,(\flat,x))=a \ \&\ \alpha_1(e_k,(\flat,x))=e_m \ \&\ \pi_o\alpha_2(e_n,(a,x))=\flat \ \&\ \alpha_1(e_n,(a,x))=e_l]$.

e_k *est* connecté par appel *à* e_l. *On notera* $e_k \overset{o}{\underset{M}{\vdash\!\!-\!\!-}} e_l$ *et on dira que* e_k appelle e_m *et* e_n retourne e_l via a.

(3) Il existe une suite finie $e_{n_1}=e_k,\ldots,e_{n_p}=e_l$ *tels que, pour* $1\leq i<p$, e_{n_i} *soit connecté à* $e_{n_{i+1}}$ *simplement ou par appel.*

La méthode de Lomet permet alors de décomposer une A-machine à pile de contrôle strict en sous-machines correspondant aux automates finis d'un système de transitions [13,65] ou d'un système d'automates [41].

Définition 31

Une sous-machine *d'une G-machine à pile de contrôle strict M est une G-machine M' de structures de travail* $(S_i,F'_i)_{i\in I'}$, *définie par :*

- *un état d'entrée* e'_o *de M, qui définit l'ensemble des états :*

$$E'=\{e'_o\}\cup\{e\in E \mid e'_o \overset{*}{\underset{M}{\vdash\!\!-\!\!-}} e\}$$

- $(\forall i\in I')[F'_i=F_i\cup\{f_i^{[k,l]} \mid e_k$ *est un état d'entrée de M,* e_l *un état de sortie,*

$e_k \overset{*}{\underset{M}{\vdash\!\!-\!\!-}} e_l$ *et* $(\exists e_m,e_n\in E')(\exists a\in A_o-\{\flat\})[e_m$ *appelle* e_k *et* e_l *retourne* e_l, *via a*$]\}]$.

Intuitivement, les $f_i^{[k,l]}$ *sont des noms pour d'autres sous-machines* $[k,l]$, *et sont interprétés comme l'identité* f_{io}.

- $\alpha'(e,x)=\alpha(e,(\flat,x))$, *restreint à* $(S_i,F'_i)_{i\in I'}$, *si* $\pi_o\alpha_2(e,(\flat,x))=\flat$, $e\in E'$ *et* $x\in\Sigma_{I'}$.

$\alpha'(e_m,x)=(e_n,x,\prod_{i\in I'} f_i^{[k,l]})$, *pour tout* $x\in\Sigma_{I'}$, *si, via* $a\in A_o-\{\flat\}$, e_m *appelle* e_k, e_l *retourne* e_n *et* $e_k \overset{*}{\underset{M}{\vdash\!\!-\!\!-}} e_l$.

Une sous-machine M' appelle une sous-machine M" *si* $(\exists e\in E')(\exists a\in A_o-\{\flat\})[E'$ *appelle* e''_o *via a*$]$. *Cet appel est* gauche *si* $e=e'_o$, droit *si tous les états de sortie de E" retournent, via a, des états de sortie de E'.*
Sinon, l'appel est interne. *Il est* strictement interne *si aucun des états de sortie de E" ne retourne, via a, d'état de sortie de E'.*

Ainsi, aucune sous-machine n'utilise la bande de contrôle, et

- il y a une manipulation sur les structures de travail avant et après un appel strictement interne.
- il n'y a pas de telle manipulation avant un appel gauche.
- il n'y en a pas après un appel droit.
- il y en a avant et parfois après un appel interne.

Définition 32

Une sous-machine M' *de M* n'utilise pas la structure (S_j,F'_j) *si*

$(\forall(e,x) \in Dom\ \alpha')[\pi_j\alpha'_2(e,x)=\pi_j(x)\ \&\ \pi_j\alpha'_3(e,x) \notin F'_j - \{f_{jo}\}]$

Si deux sous-machines M' et M" utilisent la même structure, elles la partagent.

La relation de domination *entre sous-machines est l'extension transitive de la relation d'appel.*

On dira que M' est :

- récursive *si elle se domine elle-même*
- auto-récursive *si elle s'appelle elle-même*
- élémentaire *si elle ne comporte pas d'appel*
- initiale *si* $e'_o=e_o$
- irréductible *si elle est initiale ou auto-récursive à appels uniquement internes.*

On aboutit maintenant à un théorème de décomposition.

Définition 33

Une G-machine à pile de contrôle strict est sous forme standard si :

- *toutes les sous-machines sont irréductibles*
- *leurs états sont disjoints*
- *le vocabulaire de pile caractérise les états d'appels :*

$(\forall a\in A_o)(\exists!e\in E)(\exists e'\in E)$[*e appelle e' via a*].

Théorème 7

Pour toute G-machine à pile de contrôle strict M, on peut trouver effectivement une machine $\hat{M}$ *sous forme standard qui la simule.*

La démonstration de Lomet pour les automates à pile est immédiatement généralisable, à cause de l'indépendance des actions sur la pile et sur les autres structures. ⊠

Remarquons que ce théorème permet d'obtenir une forme réduite pour des réseaux de transitions non augmentés, de façon à éliminer en particulier la récursivité à gauche. Pour les G-machines, on pourrait affaiblir la notion d'irréductibilité en n'exigeant pas que les appels soient internes, et on obtiendrait évidemment un résultat analogue.

Remarquons encore que les sous-machines obtenues n'ont pas le même fonctionnement seules ou considérées comme parties de M, puisque dans le premier cas les $f_i^{[k,1]}$, appels d'autres sous-machines, sont interprétés comme l'identité. Cependant, on peut construire une machine équivalente à M en ajoutant à cette collection un "moniteur" qui opère sur elle.

Définition 34

Soit $\mathcal{M}=\{M_1,\ldots,M_n\}$ *une collection de G-machines, dont les ensembles d'états sont disjoints, et soit* ρ *une fonction de* $\bigcup_{k=1}^{n} \prod_{i\in I_k} F_i$ *dans* $\bigcup_{k=1}^{n} (\{M_k\}\times E_k)$, *telle que, si* $f=\{f_{ij}\}_{ij\in I_k\times J_i}$ *est dans le domaine de* ρ, *chaque* f_{ij} *est égale à l'identité* f_{io}. Le système $S_{\mathcal{M}}$ des G-machines $\mathcal{M}$ *est la G-machine* M_o *déduite des précédentes en prenant :*

- $E_o = \bigcup_{k=1}^{n} E_k \cup \{\neq\} \cup \{\flat\} \times \bigcup_{k=1}^{n} E_k \times Im\ \rho$
- *les structures de travail* (S_i,F'_i) *et une pile* $(S_o,F_o)=(\mathbb{N},\{-1,0,+1\})$, *avec* $F'_i=F_i - Dom\ \rho$ *et* $A_o = \{\flat\} \cup \bigcup_{k=1}^{n} E_k$.

- *des transitions déterminées, si* $e\in E_k$, $x\in \prod_{I_k} A_i^{h_i}$ *et* $z\in A_o$, *par :*

$$\alpha^o\left(e,\begin{pmatrix}z\\x\end{pmatrix}\right)=\left(\alpha_1^k(e,x),\left(\alpha_2^k{}^{z}(e,x)\right),\left(\alpha_3^k{}^{0}(e,x)\right)\right) \text{ si } \alpha_3^k(e,x)\cap Dom\ \rho=\emptyset.$$

$$\alpha^o\left(e,\begin{pmatrix}z\\x\end{pmatrix}\right)=\left(\begin{pmatrix}\flat\\ \rho\alpha_3^k(e,x)\end{pmatrix},\begin{pmatrix}z\\x\end{pmatrix},0\right) \text{ sinon}$$

$$\alpha^o\left(e,\begin{pmatrix}z\\x\end{pmatrix}\right)=\left(\neq,\begin{pmatrix}z\\x\end{pmatrix},0\right) \text{ si } (e,x)\notin Dom\ \alpha^k$$

$$\alpha^o\left(\begin{pmatrix}\flat\\ e,M_k,e'\end{pmatrix},\begin{pmatrix}\flat\\x\end{pmatrix}\right)=\left(e',\begin{pmatrix}e\\x\end{pmatrix},\begin{pmatrix}+1\\0\end{pmatrix}\right) \text{ pour } (M_k,e')\in Im\ \rho$$

$$\alpha^o\left(\neq,\begin{pmatrix}e\\x\end{pmatrix}\right)=\left(e,\begin{pmatrix}\flat\\x\end{pmatrix},\begin{pmatrix}-1\\0\end{pmatrix}\right)$$

Les M_k $(1\leq k\leq n)$ *seront appelées* modules *du système* $S_{\mathcal{M}}$.

Proposition 15

(1) Tout système de G-machines est une G-machine à pile de contrôle strict.

(2) Soit M une G-machine à pile de contrôle strict, le système M' dont les modules sont les sous-machines de M simule M.

(3) Si la relation de domination induite par ρ *sur* $\mathcal{M}$ *est une hiérarchie,* $S_{\mathcal{M}}$ *peut être simulé par une G-machine possédant les mêmes structures de travail et n'utilisant pas la pile. On dira que le contrôle est d'états finis.*

(1) est immédiat d'après les définitions, (2) par construction, et (3) en remplaçant l'ensemble fini des expressions de pile par autant d'états supplémentaires et en modifiant α^o en conséquence, pour ne plus avoir ni mouvement ni écriture sur la pile. ⊠

3.3. SYSTEME DE G-MACHINES REALISANT UNE HEURISTIQUE ASSOCIEE A UN SYSTEME DEFINITIONNEL

Dans tout système définitionnel, rappelons que, pour toute configuration c, $\Phi^*(c)$ peut être identifié à un sous-ensemble de l'arborescence générale $\mathbb{N}^*$, qui, munie des fonctions F = {f,s,m,O}, est une structure à rappel (cf. Déf. 7). La définition précède une explication plus intuitive.

Définition 35

Une heuristique associée à un système définitionnel *D=(T,δ), avec* $\delta=(\varepsilon,\tilde{\eta},\gamma,Q)$ *est un triplet* $\mathcal{H}=(H,P,\chi)$, *où :*

(1) H est une heuristique d'énumération F-constructible engendrant des recherches compatibles avec l'ordre des calculs de D, et la fonction h associée vérifie :

$$(\forall e\in\mathbb{N})[\{\theta_o h(e,n)\}_{n\in\mathbb{N}}\cap\{c\mid\theta_1(c)\in T_\delta(e)\}\neq\emptyset\ \&\ \{\theta_o h(e,n)\}_{n\in\mathbb{N}}\subset\Phi^*\theta_o(e)]$$

(2) $P\in R^2$ *et* $\chi\in R^1$ *sont un* poids *et un* critère, *tels que, si p,* $\omega\in R^2$ *sont définis comme suit,*

$$(\forall e)[\bigcup_{n\in\mathbb{N}} D_{\omega(e,n)}=T_\delta(e)]$$

$$\begin{cases}p(e,0)=0 \text{ pour tout } e\\ p(e,n+1)=P(p(e,n),\ \varepsilon\theta_o h(e,n)) \text{ pour tout } n\end{cases}$$

$$\begin{cases}\omega(e,0)=0 \text{ pour tout } e\\ D_{\omega(e,n+1)}=\begin{cases}D_{\omega(e,n)}\cup\{\theta_1\theta_o h(e,n)\} \text{ si } [\chi p(e,n)=1\ \&\ \theta_o h(e,n)\in Max\ \Phi^*(e)]\\ D_{\omega(e,n)} \text{ sinon.}\end{cases}\end{cases}$$

$\mathcal{H}$ *est* L-décidable $(L\subset\mathbb{N})$ *si la fonction* $\lambda n[\mu n(\psi(e,n)=1)]$ *est récursive, pour* $e\in L$, *avec*

$$\psi(e,n)=\begin{cases}1 \text{ si } (\forall m\geq n)[D_{\omega(e,m)}=D_{\omega(e,n)}]\\ 0 \text{ sinon}\end{cases}$$

Ainsi, (1), H énumère des codes de calculs issus de e d'une manière compatible avec la structure de $\Phi^*\theta_o(e)$, et, pour chaque valeur de D sur e, H donne au moins un calcul terminal ayant cette valeur. Notons que H ne donne pas nécessairement une sous-énumération de $\tilde{\eta}$: le critère définitionnel peut utiliser un "parcours" et l'heuristique choisie un autre. D'autre part, p permet d'avoir une information "cumulative" qui utilise ε, mais n'est plus fixe pour tout point de $\Phi^*(e)$: elle dépend de tout le parcours suivi pour y arriver, et est donc en principe plus "globale".

Enfin, H est (ℕ)-décidable si, dans le cas où $T_\delta(e)$ est fini pour tout e, on sait décider quand on l'a atteint : en pratique, ceci revient à dire qu'on peut arrêter l'énumération par H, car elle ne produira rien de nouveau.

A titre d'exemples d'heuristiques, on peut citer toutes les méthodes de parcours des arborescences de choix qui apparaissent dans les jeux (procédures minimax, moyenne-max, α-β, γ,... [40,56,57]).

Définition 36

Tout contrôle d'énumération ζ (EAO, PEC,...) définit pour chaque heuristique $\mathcal{H}$ un sous-ensemble formé des stratégies contrôlées engendrées par $\mathcal{H}$, *qu'on notera $\mathcal{H}_\zeta$.*

La proposition suivante est alors immédiate :

Proposition 16

(1) Si ζ est un contrôle de production défini par une même fonction, toute heuristique $\mathcal{H}$ associée à un système définitionnel est $\mathcal{H}_\zeta$-décidable.

(2) Tout système définitionnel qui admet une heuristique associée (décidable) est (finiment) réalisable.

L'inverse de (2) peut ne pas être vrai, puisqu'on a vu plus haut des r.e. non F-constructibles. Ainsi, la notion d'heuristique associée est à la fois plus restreinte et plus large que celle de critère de définition : celui-ci peut définir une énumération non F-constructible, mais doit permettre de parcourir tout $\Phi^*(c)$, pour chaque c. Au contraire, une heuristique peut "élaguer" l'arbre des calculs : on retrouve le fameux "tree pruning" de l'IA [40]. ✉

Nous arrivons maintenant au résultat qu'on utilisera effectivement au III pour construire un système d'analyse prédictive incluant des heuristiques.

Théorème 8

Pour tout système définitionnel D=(T,δ) et toute heuristique associée $\mathcal{H}$ définie par (F,d,P,χ), on peut construire uniformément un système hiérarchisé de G-machines $\mathcal{M} = R(T,\varepsilon,d,P,\chi)$, qui réalise D, et dont les structures sont fixées dès que T l'est.

Les structures de travail de $\mathcal{M}$ seront celles de T, cinq bandes pour $\theta_1,\varepsilon,d,P,\chi$ (on peut prendre des MT classiques pour les calculer, et, pour simplifier, utiliser un alphabet de taille supérieure à |F|) et une arborescence générale $\mathcal{A}$ pouvant porter des couples d'entiers (codes de configurations de T et estimations), isomorphe à $\mathbb{N}^*\times\mathbb{N}^2$, avec comme fonctions de déplacement $F \cup G_1 \cup G_2$, où $G_1 = G_2 = \{-1,0,+1\}$[§], par exemple. Le système est bâti à partir des "modules" $\theta_o,\theta_1,\varepsilon,d,P,\chi,T'$ et M_o. T' est obtenu à partir de T en modifiant le contrôle fini de la façon suivante : E est remplacé par Ex{1,...,r} et α par la fonction α' de $Ex\{1,\dots,r\} \times \prod_{i\in I} A_i^{h_i}$ dans $Ex\{1\}\times \prod_{i\in I} A_i^{h_i} \times F_i$, qui coïncide avec α si on la restreint à $E \times \prod_{i\in I} A_i^{h_i}$. On a ainsi défini canoniquement (i.e. à partir de la table de α) un ordre des choix, qui ordonne totalement $\Phi^*(c)$ pour tout c. M_o, le "moniteur", a pour rôle de simuler chaque pas de H. Pour ceci, chaque configuration de T sera représentée dans $\mathcal{M}$ par la même configuration sur les structures de T', et M_o conservera sur les points de $\mathcal{A}$ les codes et les estimations des configurations déjà rencontrées. Un pas de la réalisation de $\mathcal{H}$ consiste donc à coder (si ce n'est fait) la configuration de T' sur le point de $\mathcal{A}$ où se trouve M_o, à calculer ε,P,χ, éventuellement (si χ=1) à "sortir" $\theta_1(c)$, puis à exécuter un pas de H en calculant d : si le point x de $\mathcal{A}$ désigné a déjà été parcouru, on le décode (via θ_o), sinon M_o appelle T' avec l'état (e,k), où e est l'état de la configuration de T' codée sur le père de x, et k correspond au choix de transition

§ - Voir l'exemple sur les matrices donné au 1.3.3.

déterminé par d et par l'énumération obtenue jusque là (par exemple, on prend la première transition non encore essayée à partir du "père" de x).

Il doit être clair que ce procédé $\mathcal{R}$ est effectif et fournit bien $\mathcal{M}$ uniformément en fonction de ses arguments. On trouvera au chapitre III la construction détaillée de $\mathcal{M}$ dans un cas plus particulier. Remarquons à ce propos qu'on a pris ici une méthode grossière de réalisation, puisqu'on code toute une configuration sur chaque point de $\mathcal{A}$. Il peut être possible, comme au chapitre III, de trouver des codages plus complexes qui permettent de gagner une place et un temps importants dans la pratique. ◻

On peut remarquer que la construction précédente donne un M_o fixe et d'états finis pour tous les couples $(D,\mathcal{H})$, puisque le système résultant est hiérarchisé. Les différents modules apparaissent de manière indépendante, et on peut donc réaliser toute une classe de "F-heuristiques" en écrivant une fois pour toutes des modules (sous-modules de M_o) qui réalisent les éléments de F sur $\mathcal{A}$. C'est cette possibilité que nous utiliserons au chapitre IV pour la définition d'un système informatique permettant de programmer une telle classe, pour un F fixé. Quant à F, il est clair qu'on peut l'étendre en y admettant des "abréviations", i.e. des composées d'éléments de F, ou même de nouvelles fonctions (de retour à des points de $\mathcal{A}$ déjà énumérés, par exemple), pour augmenter la classe d'heuristiques atteintes, ou simplement la commodité d'écriture.

COMPLEXITE

A ce propos, on peut revenir sur les notions de complexité introduites au début de ce chapitre. Il est clair que la notion intéressante est celle de système de programmation mesuré.

Supposons que $\mathcal{M} = \{M_i\}_{1\leqslant i\leqslant k}$ soit un système de G-machines de tailles $|M_i|$ et de complexités $\mathbf{C}_i$. On peut prendre $|\mathcal{M}| = \sum_{i=I}^{k} |M_i|$ ou $|\mathcal{M}| = k \max_{i=I}^{k} |M_i|$ comme mesure de taille de $\mathcal{M}$. Ces deux définitions satisfont les axiomes de Young, mais peut-être la seconde est-elle plus intuitive si on interprète $|\mathcal{M}|$ non pas seulement comme la longueur de la description de $\mathcal{M}$, mais comme la complexité de sa structure. C'est d'ailleurs une mesure analogue à celle de Shannon et Fischer (nombre d'états x nombre de symboles) pour les MT. Il est alors clair, et la pratique le confirme, qu'on a intérêt, pour une même fonction, à utiliser un système de G-machines ne comprenant ni trop ni trop peu de modules. En ce qui concerne $\mathbf{C}$, la mesure de complexité du système, un lemme bien connu de Blum (1967a) indique qu'elle est récursivement reliée aux $\{\mathbf{C}_i\}_{1\leqslant i\leqslant k}$ dans le cas général. Dans une optique plus pratique, on peut admettre, comme dans Pager (1970) que chaque $\mathbf{C}_i$ est une fonction monotone croissante du "temps" et de l'"espace", où le temps est le nombre de pas de calcul et l'espace le maximum de l'espace utilisé dans chaque structure (comme pour les MT à k bandes [14]). Dans ce cas, $\mathbf{C}$ est reliée aux $\mathbf{C}_i$ en prenant la somme des composantes "temporelles" et le maximum des composantes "spatiales", pour avoir une interprétation cohérente : il ne suffit pas pour améliorer un algorithme de l'écrire de façon structurée (i.e. de décomposer une G-machine qui le calcule en un système équivalent de G-machines), ceci ne peut que simplifier sa description, ce qui est déjà beaucoup, sans l'accélérer§. Et on sait bien qu'il n'existe pas de méthode effective et générale d'accélération [7].

Une autre notion intéressante a été introduite par Pager (1970). Il s'agit de définir la complexité d'un algorithme f de façon générale, et non pas sur chaque argument. Pour ceci, il faut supposer qu'on dispose d'une mesure de complexité $\mathbf{C}$ et d'une distribution de probabilité π sur les arguments, bien sûr telle que $\mathfrak{D} = \pi^{-1}(0) \subset \text{Dom } f$. Alors $\mathbf{C}_\pi(f) = \sum_{x\in\mathfrak{D}} \pi(x)\, \mathbf{C}(x)$. Il est remarquable de constater que, dans la pratique, on utilise toujours (implicitement) un tel π, puisqu'on borne toujours le domaine par une limitation physique, par exemple par une taille maximum pour tout argument. Les résultats théoriques que Pager obtient indiquent que, dans le cas (présent) où $\mathbf{C}$ a une composante spatiale, il existe une procédure effective uniforme d'accélération χ qui produit l'algorithme optimal pour f (sur $\mathfrak{D}$) au bout d'un nombre de pas fini, mais inconnu : si $\varphi_\pi(\tilde{f})$ est un tel programme optimal ($\tilde{f}$ est un indice de f), $(\overset{\infty}{\forall} t)[\chi_\pi(\tilde{f},t) = \varphi_\pi(\tilde{f})]$. Et $\varphi_\pi(\tilde{f})$ existe dès que $\mathfrak{D}$ est fini, ce qui est le cas.

---

§ - Constable (1971) a même prouvé le résultat remarquable suivant : si on dispose d'un formalisme permettant d'exprimer des fonctions subrécursives (une sous-classe de R^1) et muni d'une mesure de taille, il existe toujours une fonction récursive f de la classe dont le programme le plus court dans ce formalisme est arbitrairement inefficient (peut être arbitrairement accéléré).

Ces résultats théoriques donnent l'espoir de pouvoir accélérer un algorithme (donné ici au moyen d'un système de G-machines), bien que l'on ne connaisse ni π ni ϑ a priori. Mais on connait un surensemble de Dom f dès qu'on limite la taille des arguments. χ n'étant pas donné par un procédé effectif, on se contentera d'une approximation, qui consiste très simplement à :

- utiliser les bornes pour définir des codages plus compacts des configurations (on codera ainsi a en utilisant un stack contenant des pointeurs).

- modifier l'ordre des choix, dans le cas d'une réalisation de système définitionnel, en utilisant les complexités effectives des calculs déjà effectués.

- pondérer le contrôle fini et, si ε ou p utilise cette pondération, à la modifier au fur et à mesure des calculs. Ceci revient en fait à donner non plus le système T, mais une classe de tels systèmes.

Nous verrons enfin au chapitre II pourquoi, semble-t-il, il faut se contenter d'approximations de ce type : il n'existe pas de procédé uniforme et effectif d'optimisation ou d'appentissage, même dans des cas très simples.
C'est pourquoi nous parlerons plutôt d'adaptation ou d'amélioration.

TABLE BIBLIOGRAPHIQUE

[1] Aho (1969)
[2] Anderaa & Fischer (1967)
[3] Arbib (1969)
[4] Barbault & Desclés (1970)
[5] Blum (1967a)
[6] Blum (1967b)
[7] Blum (1971)
[8] Brauer & Indermark (1968)
[9] Chauché (1974)
[10] Church (1944-1956)
[11] Constable (1971)
[12] Constable & Borodin (1972)
[13] Conway (1963)
[14] Cook (1973)
[15] Davis (1958)
[16] Eilenberg (1974)
[17] Eilenberg & Wright (1970)
[18] Feldman (1972)
[19] Fischer (1965)
[20] Floyd (1967)
[21] Hartmanis & Hopcroft (1971)
[22] Hartmanis, Lewis & Stearns (1965)
[23] Hartmanis & Stearns (1965)
[24] Hennie (1966)
[25] Hennie & Stearns (1966)
[26] Hermes (1965)
[27] Hopcroft & Ullman (1967)
[28] Hopcroft & Ullman (1969)
[29] Kain (1972)
[30] Karp (1972)
[31] Kleene (1952)
[32] Lacombe (1960)
[33] Lewis, Stearn & Hartmanis (1965)
[34] Lomet (1973)
[35] Mal'cev (1970)
[36] Markov (1962)
[37] Meyer (1972)
[38] Minsky (1961)
[39] Minsky (1967)
[40] Nilsson (1971)
[41] Ollongren (1974)
[42] Pager (1969)
[43] Pager (1970)
[44] Pager (1974)
[45] Post (1936)
[46] Pour-El & Putnam (1962)
[47] Rogers (1959)
[48] Rogers (1967)
[49] Salomaa (1969a)
[50] Salomaa (1969b)
[51] Santos (1969)
[52] Savitch (1970)
[53] Savitch (1973)
[54] Shannon (1956)
[55] Shepherdson & Sturgis (1963)
[56] Slagle (1970)
[57] Slagle & Lee (1971)
[58] Thue (1914)
[59] Turing (1936-37)
[60] Vauquois (1968a)
[61] Vuillemin (1974a)
[62] Vuillemin (1974b)
[63] Werner (1973)
[64] Wharton (1974)
[65] Woods (1969)
[66] Yates (1962)
[67] Young (1969)
[68] Young (1971)

CHAPITRE II

APPRENTISSAGE

"Although it is possible to give a formal description of certain aspects of language and although several mathematical models of the learning process have been developed, the intersection of these two theoretical endeavours remains disconcertingly vacant".

Chomsky & Miller (1963)

INTRODUCTION

Le terme d'"apprentissage" apparaît fréquemment dans bien des domaines liés à la TA : en linguistique, en psycho-linguistique, en théorie des automates et des langages formels, en automatique. C'est relativement récemment que Gold [7] a introduit en théorie de la récursivité la notion de récursivité, ou calculabilité " à la limite", pour formaliser un certain type d'apprentissage, connu depuis sous le nom d'"inférence grammaticale". Car il y a bien des types d'apprentissage différents : enrichissement d'un ensemble de données, fabrication effective d'une représentation équivalente plus simple, découverte d'un objet d'une certaine classe (grammaire, fonction, boîte noire"...) par observation de son comportement externe ou interne, recherche enfin des paramètres optimaux d'un système paramétré.

Le dénominateur commun de toutes ces "situations d'apprentissage"[§] est le suivant : on fait une expérience, qui donne des résultats de façon discrète, à chaque unité de temps, et on cherche à extrapoler, s'il existe, un objet d'une classe Γ donnée a priori qui fournisse des résultats comparables à ceux de l'expérience : identiques, identiques à partir d'un certain moment, ou assez proches[§§]. On pourra distinguer entre les méthodes de type énumératoire qui utilisent une énumération des objets de Γ, avec éventuellement un critère d'adéquation (complexité minimale, probabilité maximale) [7,10,14] et les méthodes "structurales" qui utilisent une définition de Γ à partir de la structure de ses objets : il s'agit des méthodes constructives [2,3,4,23,39], paramétriques [1,6,25,26,31,37,38] ou simplificatoires [35].

Après avoir rappelé les principaux résultats dans ces domaines et complété certains d'entre eux, j'examinerai les méthodes envisageables dans le cadre d'un système de TA : il semble qu'il faille se limiter pour l'instant à une "adaptation" consistant en un enrichissement et un ajustement de paramètres, les autres approches étant soit inutiles soit impossibles, en principe ou à cause de l'ampleur astronomique des énumérations nécessaires.

---

§ - à part l'apprentissage par "simplification", où on part d'une représentation déjà correcte.

§§ - au sens d'une métrique donnée sur la classe Γ.

I - METHODES ENUMERATOIRES

Dans "Aspects"[§], Chomsky donne les critères qu'une théorie doit vérifier pour rendre compte de l'apprentissage du langage : il faut qu'elle donne

(1) une énumération de la classe $s_1, s_2, \ldots$ des phrases possibles
(2) une énumération de la classe $SD_1, SD_2, \ldots$, des descriptions structurales possibles
(3) une énumération de la classe $G_1, G_2, \ldots$ des grammaires génératives possibles
(4) une fonction universelle f telle que $SD_{f(i,j)}$ soit la description structurale associée à s_i par G_j
(5) une fonction m telle que m(j) soit la "valeur" de la grammaire G_j.

Cette formulation permet d'imaginer un algorithme d'apprentissage de la façon suivante : l'expérience donne à chaque instant t un échantillon E_t de phrases contenues dans le langage L considéré (et de leurs SD) ou dans son complément $\bar{L}$, et l'algorithme d'apprentissage "essaie" la première grammaire G_t compatible avec E_t et SD qu'elle contient, et de valeur maximale (ce qui limite alors le nombre de grammaires qu'on peut essayer à chaque pas[§§]). Cet algorithme peut converger ou non selon qu'on choisit une classe $\{G_i\}$ ou une autre, qu'on utilise ou non le critère et que l'expérience énumère toutes les phrases possibles ("informant") ou seulement toutes celles de L ("texte"). Il peut y avoir différentes notions de convergence, plus ou moins contraignantes, d'où toute une famille de résultats sur une hiérarchie de classes de langages, partant des langages finis pour arriver aux langages récursivement énumérables. Retenons donc que l'apprentissabilité est une propriété de classe d'objets qui dépend d'une situation d'apprentissage particulière.

1.1. APPRENTISSAGE DE GRAMMAIRES

1.1.1. MODELES D'APPRENTISSAGE

On suppose fixé un alphabet fini non vide A. Tous les langages considérés seront des sous-ensembles du monoïde libre A^* engendré par A, c'est-à-dire des éléments de $\Lambda=2^{A^*}$ [§§§].

Définition 1 (Gold)

Un modèle d'apprentissage de langages *est la donnée :*
- *d'une* relation d'étiquetage *qui associe des* noms *aux langages*
- *d'une* méthode de présentation de l'information
- *d'une* définition de l'apprentissabilité *(critère de convergence)*

La relation d'étiquetage pourra par exemple associer à un langage L les indices de sa fonction caractéristique (et il peut ne pas y en avoir si L n'est pas récursif), ou ceux de ses fonctions d'énumération, ou encore de ses grammaires hors-contexte, etc. Il s'agit toujours de fonctions calculables, et les noms de L peuvent donc être considérés comme des indices de machines de Türing (MT), qu'on appellera accepteurs (testeurs) dans le premier cas, générateurs dans le second.

Définition 2 (Gold)

Une séquence d'information pour un langage $L\in\Lambda$ *est une suite* $I=(x_t)$ *d'éléments de l'ensemble* $\{+y:y\in L\}\cup\{-y:y\in\bar{L}\}$. *On notera* $x_t=\pm y_t$ *et* $S_t(I)=\{x_r\}_{r\leq t}$ *est l'*échantillon à l'instant t.

Une séquence d'information de $L, I_+(L)$, *est* positive *si elle ne contient que des éléments de forme +y. C'est un* texte *si elle contient* tous les éléments de L. *Une séquence d'information positive et négative qui contient* tous les éléments de A^* *est un* informant.

---

§ - [7], p. 31.

§§ - Pour ceci, m doit vérifier les axiomes d'une mesure de taille (ou complexité intrinsèque), cf. chapitre I.

§§§ - Certains auteurs demandent que les langages considérés ne contiennent pas le mot vide ("ε-free"), ce qui ne change rien aux résultats obtenus.

D'où plusieurs méthodes de présentation de l'information :

(1) Texte arbitraire : x_t *est une fonction quelconque de t (qui atteint tout L).*

(2) Texte EQO par la longueur l : *il existe une fonction récursive* τ_l *telle que* $(\forall n)(\forall t)[t>\tau_l(n) \Rightarrow [l(x_t)>n$ *ou* $(\exists r)[r\leqslant\tau_l(n) \ \& \ x_r = x_t]]]$

(3) Texte récursif : x_t *est une fonction récursive de t.*

(4) Texte primitif récursif : x_t *est une fonction primitive récursive de t.*

(5) Informant arbitraire : y_t *est une fonction quelconque de t (qui atteint tout* A^**).*

(6) Informant méthodique : y_t *est le t-ième terme d'une énumération donnée de* A^*.

(7) Informant à la demande : *à l'instant t, l'algorithme d'apprentissage choisit* y_t *en fonction de l'information reçue jusque là.*

Gold (1967) a montré que ces trois dernières méthodes sont équivalentes.

Il reste à voir les différents critères d'apprentissabilité. Dans tous les cas, l'apprentisseur est un algorithme (une MT) M qui produit un nom $N_t = M(S_t)$ d'une classe Γ à tout instant t. A chaque nom N correspond un seul langage, soit L(N).

Définition 3 (Feldman)

L'algorithme M identifie L à la limite dans Γ *s'il produit presque toujours un même nom* $N\in\Gamma$ *de* L : $(\exists\tau)(\forall t)[t\geqslant\tau \Longrightarrow N_t=N_\tau \ \& \ L(N_\tau) = L]$.

M atteint L à la limite dans Γ *s'il produit presque toujours des noms (dans* Γ*) de* L : $(\exists\tau)(\forall t)[t\geqslant\tau \Longrightarrow L(N_t) = L]$.

M approche L à la limite dans Γ *s'il finit par atteindre L et ne peut produire chaque nom d'un autre langage qu'un nombre fini de fois :*

(i) $(\forall y)(\exists\tau)(\forall t)[y\in L \ \& \ t\geqslant\tau \Longrightarrow y\in L(N_t)]$

(ii) $(\forall N)(\exists\tau)(\forall t)[L(N)-L \neq \emptyset \ \& \ t\geqslant\tau \Longrightarrow N_t \neq N]$

M approche fortement L à la limite dans Γ *s'il produit de plus un même nom (dans* Γ*) de L une infinité de fois :*

(iii) $(\exists N)[L(N) = L \ \& \ (\overset{\infty}{\exists} t)[N_t = N]]$.

Remarque 1 :

Feldman (1972) dit en fait que M identifie (atteint, approche) un nom (une grammaire) N de L. Il est plus naturel de dire qu'on apprend un langage, et cette notion se généralise alors facilement à l'identification approximative où à l'approche faible. C'était d'ailleurs la terminologie initiale de Solomonoff (1964) et de Gold (1967).

Remarque 2 :

La condition (ii) est un peut plus faible que la propriété annoncée. Mais si $L(N)\neq L$ et si $L(N) - L=\emptyset$, $\exists y\in L - L(N)$, et (i) assure que M ne peut produire N qu'un nombre fini de fois.

On peut modifier la notion d'apprentissabilité pour tenir compte de ce que

(a) l'information peut être "légèrement" erronée.

(b) la recherche est effectuée dans une classe Γ (de noms) inappropriée ($L\notin L(\Gamma)$).

(c) on cherche le "meilleur" nom possible (le moins complexe, le plus probable,...).

La notion d'approche correspond déjà à (a) et (b). On peut également envisager d'identifier un nom N tel que L(N) soit "presqu'égal" à L, au sens "presque partout" ou au sens d'une métrique introduite sur Λ. Le (c) suggère d'employer des mesures de complexité ou des probabilités.

Définition 4

M identifie faiblement L à la limite dans Γ *s'il produit presque toujours un même nom (dans Γ) d'un langage presque partout égal à L :*

$(\exists \tau)(\forall t)[t \geq \tau \Longrightarrow N_t = N_\tau \ \& \ |L \ominus L(N_\tau)| < \infty]$ §

L'égalité presque partout correspond donc exactement à l'égalité p.p. des fonctions caractéristiques. Plus précisément, supposons donnée une énumération (y_i) de A^* sans répétition (par exemple l'énumération par ordre lexicographique). La fonction caractéristique χ_L de L est donnée par la suite (f_i) de ses valeurs : $f_i = \chi_L(y_i)$.

Définition 5 (Wharton)

Soient $(y_i)_{i \in \mathbb{N}}$ une énumération canonique de A^* et $W=(w_i)_{i \in \mathbb{N}}$ une suite de poids, *c'est-à-dire telle que* $(\forall i)[w_i \in \mathbb{R}_{++}] \ \& \ \sum_{\mathbb{N}} w_i < \infty$ §§. *On supposera W* normalisée, *c'est-à-dire $\sum_{\mathbb{N}} w_i = 1$. Alors*

(1) $\|L\|_W = \sum_{\mathbb{N}} f_i \, w_i$ *définit une* norme *sur* Λ.

(2) $d_W(L_1, L_2) = \|L_1 \ominus L_2\|_W$ *est la* distance *associée. Elle définit une* métrique *sur* Λ.

M identifie approximativement L à la limite dans Γ *s'il peut produire presque toujours un même nom (dans Γ) d'un langage arbitrairement W-proche de L :*

$(\forall \varepsilon)(\exists \tau)(\forall t)[t \geq \tau \Longrightarrow N_t = N_\tau \ \& \ d_W(L, L(N_\tau)) < \varepsilon]$.

Remarque :

Pour la métrique discrète définie par $d(L_1, L_2) = \begin{cases} 0 \text{ si } L_1 = L_2 \\ 1 \text{ sinon} \end{cases}$

L'identification approximative à la limite coïncide avec l'identification à la limite. Mais la métrique discrète n'est pas une métrique "pondérée". D'autre part, l'égalité p.p., "≃", est une relation d'équivalence, et l'identification faible sur Λ coïncide donc avec l'identification à la limite sur Λ/≃.

1.1.2. VALEUR DES GRAMMAIRES

Un procédé d'apprentissage doit chercher à trouver une "meilleure" grammaire. La valeur d'une grammaire peut être calculée intrinsèquement et/ou par rapport à l'échantillon considéré. Deux méthodes parallèles consistent à probabiliser l'ensemble des grammaires et des dérivations possibles ou à définir une "mesure de complexité grammaticale". Les algorithmes d'apprentissage pourront utiliser ces probabilités ou ces mesures de complexité pour limiter le nombre de grammaires à examiner et trouver une "meilleure" grammaire, si Γ est donnée par une "bonne" énumération. Précisons ceci en rappelant quelques définitions introduites au chapitre I sous une forme simplifiée.

Définition 6 (Feldman, Wharton)

Une suite (y_n) est

- ordonnée par une fonction f *si* $(\forall k, t)[t > k \Longrightarrow f(y_t) > f(y_k)]$
- effectivement approximativement ordonnée (EAO) par f *si* $(\exists \tau_f \in R^1)(\forall k, t)[t > \tau_f(k) \Longrightarrow f(y_t) > f(y_k)]$
- effectivement quasi-ordonnée (EQO) par f *si* $(\exists \tau_f \in R^1)(\forall k, t)[t > \tau_f(k) \Longrightarrow f(y_t) > k$ *ou* $(\exists r)[r \leq \tau_f(k) \ \& \ y_r = y_t]]$

On démontre facilement le résultat suivant :

Proposition 1 (*)

Si une suite (y_n) est EAO ou EQO par une fonction f, on peut effectivement produire une énumération (y'_n) de $\{y_n\}$ ordonnée par f.

---

§ - ⊖ désigne la différence symétrique.

§§ - $\mathbb{R}_{++} = \mathbb{R}_+ - \{0\}$. On veut en effet que W définisse une norme sur les langages (ou, de façon équivalente, sur leurs fonctions caractéristiques).

On dit parfois que (y'_n) est une "énumération d'Occam", si f est par exemple une mesure de complexité intri

Définition 7 (Feldman)

Une **mesure de complexité intrinsèque** *d'une classe Γ de grammaires est une mesure de taille (cf. définition I-5) c(G,Γ).*

Une **mesure de complexité dérivationnelle** *est une fonction d(S,G) de* $2^{(A^*)}\times\Gamma$ *dans* $\mathbb{N}$ *qui converge si et seulement si* $S\subset L(G)$, *et telle que la fonction D définie par* $D(S,G,n) = \begin{cases} 1 \text{ si } d(S,G) = n \\ 0 \text{ sinon} \end{cases}$ *soit récursive.*

Une **mesure de complexité grammaticale** *est une fonction* $\gamma_\Gamma(S,G)$ *de* $2^{(A^*)}\times\Gamma$ *dans* $\mathbb{N}$ *exprimable au moyen d'une fonction récursive* γ *croissante sur chaque argument sous la forme* $\gamma_\Gamma(S,G) = \gamma(c(G,\Gamma),d(S,G))$

Ainsi, une mesure de complexité grammaticale définit l'analogue d'un système de programmation mesuré, à cette différence près que la mesure de complexité est définie sur des ensembles finis d'arguments. On peut déduire de cette définition certaines propriétés exigées dans [13], ce qui unifie les définitions de [13] et [40].

Proposition 2

(1) Si Γ est infinie et admet une mesure de complexité intrinsèque c, Γ est r.e. : Γ = (G_i), c n'est pas bornée et toute énumération récursive sans répétition de Γ est EAO par c.

(2) Si Γ est infinie et admet une mesure de complexité grammaticale, γ est non bornée.

(1) Γ est r.e. car c est une mesure de taille, et donc $(\exists f\in R^1)[D_{f(n)} = \{G:c(G,\Gamma)=n\}]$ et donc $\Gamma=\bigcup_{n\in\mathbb{N}} D_{f(n)}$ est r.e.. c ne peut être bornée, sinon Γ serait finie. Si (G_i) est une énumération récursive sans répétition, soit $f\in R^1$ telle que $D_{f(n)}=\{G:c(G,\Gamma)\leqslant n\}$, alors $\tau_c(n)=\mu_j[D_{f(n)}\subset\{G_i\}_{i<j}]$ est récursive, et on peut l'obtenir effectivement à partir de f et de l'énumération de Γ.

(2) résulte du (1) et de la croissance de γ. ⊠

Dans le modèle probabiliste [36,21], on associe une probabilité P(x|G) à toute chaîne W de L(G), par exemple en associant des probabilités aux règles de G, ce qui permet d'attribuer une probabilité à chaque dérivation et à chaque W [25]. Si on suppose que les éléments de S sont produits de façon indépendante, on aura $P(S|G)=\prod_{W\in S}P(W|G)$.
On peut attribuer des probabilités a priori aux G_i (en supposant que Γ est le langage engendré par une métagrammaire elle aussi probabiliste). On peut alors appliquer la formule de Bayes $P(G_i|S) = \frac{P(G_i).P(S|G_i)}{\sum_j P(G_j).P(S|G_j)}$ pour rechercher la grammaire la plus probable. Si (G_i) est "anti-EAO" par P, c'est-à-dire si $(\exists\tau_p\in R^1_{\mathbb{Q}})(\forall_p,t)[t>\tau_p(p) \Longrightarrow P(G_t)<p]$ §, on cherche la première grammaire G_k qui engendre S, et il suffit d'examiner l'ensemble fini des G_i telles que $k\leqslant i\leqslant\tau_p(P(G_k))$ pour trouver la plus probable qui engendre S. Horning (1969) a montré que la probabilité d'identifier L tend vers 1 avec |S|.

Les notations ont été choisies pour souligner le parallélisme entre les deux critères. La "quantité d'information" contenue dans une grammaire $(c(G_i,\Gamma)=-\log P(G_i))$ peut être prise comme complexité intrinsèque. d(W,G) est alors la somme des complexités des règles d'une dérivation de W§§, $d(S,G)=\sum_{W\in S} d(W,G)$ et $\gamma(S,G)=c(G,\Gamma)+d(S,G)$.

---

§ - $R^1_{\mathbb{Q}}$ est l'ensemble des fonctions calculables de $\mathbb{Q}$ dans $\mathbb{N}$, qui s'identifie à un sous-ensemble de R^2.

§§ - Par exemple, la première de complexité minimale (au sens d'une énumération canonique des dérivations).

1.1.3. CLASSES DE GRAMMAIRES ET APPRENTISSABILITE

Nous résumerons l'essentiel des résultats obtenus sous forme graphique, après avoir démontré un résultat sur l'identification faible. On remarquera combien la "pédagogie", ou méthode de présentation de l'information, est importante. En particulier, seules des classes très faibles sont identifiables à la limite en utilisant un "texte" arbitraire. Les résultats s'améliorent si on affaiblit le critère de convergence (approximation, quasi égalité ...), si on utilise également de l'information négative (informant), et si on utilise une certaine connaissance de la structure des grammaires (probabilité, complexité).

Notation

$L^{(m)}$ est l'ensemble des chaînes de L de longueur inférieure à m : $L^{(m)} = L \cap \bigcup_{n=0}^{m} A^n$.

Définition 8

Une classe Γ est continue *si, pour toute énumération (texte) de $L(G)(G\in\Gamma)$, soit (x_i), et pour tout t, elle contient une grammaire G' telle que $L^{(m)}(G') = S_t$, avec $S_t = \{x_i\}_{i\leqslant t}$ et $m = \underset{x\in S_t}{Max}\ |x|$.*

Soient Λ_1 et Λ_2 deux classes de langages, $\Lambda_1 \cup \Lambda_2 \subset \Lambda$. La notion de densité est définie dès qu'on a une topologie sur Λ. On dira donc que Λ_1 est W-($\simeq$-) dense dans Λ_2 si elle l'est pour la topologie induite par la suite de poids W (par l'égalité p.p.).

Remarquons par exemple que la classe Λ_F des langages finis est W-dense dans Λ pour tout W, mais n'est pas $\simeq$-dense dans Λ, ce qui suffit à distinguer ces deux notions.

Théorème 1

Soit Γ une classe récursivement énumérable (G_i) de grammaires, telle que $L(\Gamma)$ soit $\simeq$-dense dans Λ_0. Il existe une machine M qui, pour tout X de Λ_0, identifie faiblement X dans Γ, pour tout "texte" arbitraire de L.

On peut démontrer ce résultat en supposant, de façon plus générale, que la séquence d'information I contient également de l'information "négative" (des éléments de -L) - sans supposer que c'est un informant !

On suppose que, pour tout i, G'_i est un algorithme qui énumère $L(G_i)$: pour fixer les idées, G'_i est une MT à 2 bandes d'alphabet $A\cup\{\flat\}$, sa configuration initiale est blanche (seulement des $\flat$), une des bandes, la bande de sortie, est à sens unique vers la droite et G'_i y inscrit une énumération de $L(G_i)$, en utilisant le $\flat$ comme séparateur. Ceci est toujours possible de manière effective si Γ est une classe r.e. de générateurs ou d'accepteurs. On désignera par $P_i^{(n)}$ l'ensemble des chaînes de $L(G_i)$ énumérées par G'_i pendant ses n premiers pas de calcul.

D'autre part, si $L(G)\simeq L$, $|S^+\ominus L(G)|<\infty$, puisque $S^+=L$. M parcourt les calculs G'_i en "queue d'aronde" (dovetailing) : au pas n, elle simule le (n-i)-ième pas de G'_i et calcule $\delta_i^{(n)} = |S_n^+ \ominus P_i^{(n-i)}|$, pour tous les $i\leqslant n$. On ne peut évidemment se contenter de produire G_t, où $t\leqslant n$ est le plus petit indice rendant $\delta_i^{(n)}$ minima, car il peut y avoir plusieurs grammaires G telles que $L(G)\simeq L$, entre lesquelles M pourrait osciller. On associera alors à chaque G_i un entier $\eta_i^{(n)}$ qui représente, au pas n, le nombre des derniers pas de M pendant lesquels $\delta_i^{(n)}$ est resté constant : M calcule donc, pour tout $i\leqslant n$,

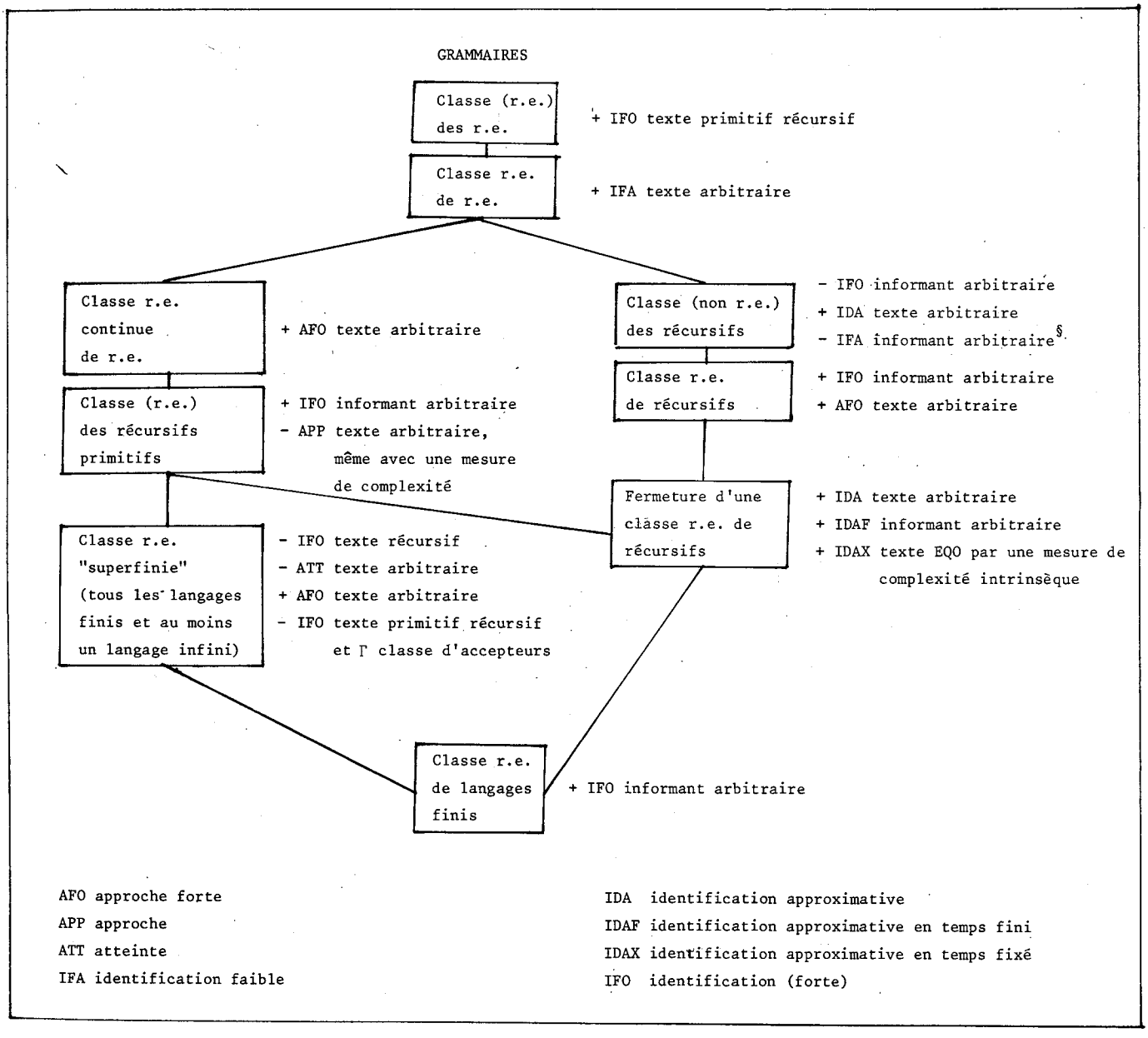
$$\eta_i^{(n)} = \begin{cases} \eta_i^{(n-1)} + 1 & \text{si } \delta_i^{(n)} = \delta_i^{(n-1)} \\ 0 & \text{sinon} \end{cases}$$

et produit la grammaire G_t, ou t est le plus petit indice $i\leqslant n$ rendant $\eta_i^{(n)}$ maximal. Comme $L(\Gamma)$ est dense dans Λ, M produira presque toujours un même indice t_0, celui de la grammaire G_{t_0} telle que $L(G_{t_0})\simeq L$ et que t_0 "stabilise" plus vite δ_{t_0}. ⊠

Remarques

- Dans ce cas, on n'obtient pas mieux en prenant un informant, ce qui contraste avec les résultats habituels.
- Cependant, on peut accélérer sensiblement la convergence si I est un informant et Γ une classe r.e. de récursifs, car on peut définir une machine M' analogue, avec $\delta'^{(n)}_i = |S^+_n \cap \overline{L(G_i)}| + |S^-_n \cap L(G_i)|$: on n'a plus besoin de simuler les G'_i.
- On n'a jamais $\Lambda_0 = \Lambda$ (cf. théorème 2).

Le tableau suivant réunit les résultats précédents et ceux de [13,18,40] sur les grammaires.



§ - Ces deux dernières propriétés découlent du théorème 2 (§ 1.1.2), puisqu'on peut identifier un langage à sa fonction caractéristique.

La méthode énumératoire qu'on emploie pour démontrer tous ces éléments peut paraître bien inefficace. Gold (1967) a cependant montré qu'il n'existe pas de méthode d'apprentissage uniformément plus rapide (pour IDN, et ceci se transpose à IDA). Ainsi, aucune énumération de Γ ne peut être optimale, au sens où l'apprentissage par énumération associé serait uniformément plus rapide.

Dans la plupart des cas précédents, Γ est une classe de générateurs. Si une classe de langages est apprentissable pour l'étiquetage par accepteurs, elle l'est pour l'étiquetage par générateurs, l'inverse étant faux en général. Mais, dans les cas où Γ est une classe de langages récursifs, on a les propriétés d'apprentissabilité indiquées pour les deux étiquetages.

Enfin, remarquons que le langage présenté (par I) peut ne pas être dans $L(\Gamma)$ si on utilise l'identification faible ou approximative.

1.2. APPRENTISSAGE DE FONCTIONS ET DE "BOITES NOIRES"

1.2.1. FONCTIONS

Quand on cherche à "identifier" une fonction (totale)ψ, l'expérience consiste à donner à l'apprentisseur M la valeur $\psi(n)$ à chaque pas n. M produit alors un indice N_n d'une fonction récursive g_n. On peut encore définir l'identification, l'atteinte, l'identification approximative (W est une suite de poids associée aux entiers) et l'approche faible. On ne s'intéressera pas à l'atteinte. Pour l'identification approximative et l'identification faible, on peut supposer que $\psi \notin \Gamma$.

Définition 9

(1)(Gold) Une fonction f à k arguments est **récursive à la limite** *si c'est la limite d'une fonction récursive à k+1 arguments :*

$$(\exists g \in R^{k+1})(\forall x \in \mathbb{N}^k)[f(x) = \lim_{n\to\infty} g(n,x)]$$

(2) Une fonction f à k arguments est **p.p. récursive à la limite** *si c'est presque partout la limite d'une fonction récursive à k+1 arguments :*

$$(\exists g \in R^{k+1})(\overset{\infty}{\forall} x \in \mathbb{N}^k)[f(x) = \lim_{n\to\infty} g(n,x)]$$

L'apprentisseur M est donc caractérisé par g, et $g_n = \lambda x[g(n,x)]$. Si on pose $[f_1 \simeq f_2] \equiv (\overset{\infty}{\forall} x)[f_1(x) = f_2(x)]$, on définit la $\simeq$-densité comme pour les langages. Si $W=(w_i)$ est une suite de poids, on posera $d_W(f_1,f_2) = \sum_{\mathbb{N}} |f_1(n) - f_2(n)|.w_n$. Ceci n'est possible que si cette quantité est finie, ce qui est toujours le cas si f_1 et f_2 sont dans une classe F bornée, $R^1_{[0,1]}$ par exemple. Sinon, il faut se limiter à des suites de poids "fortement convergentes", et nous verrons qu'il en existe dans $\mathbb{Q}_+$.

Définition 10

W=(W_i) est une **suite de poids fortement convergente** *si elle définit une norme sur* R^1 *:*

$$(\forall f \in R^1)[\ \|f\|_W = \sum_n w_n f(n) < \infty].$$

W=(W_i) est une **suite de poids cohyperimmune** *si* $\left(\frac{1}{w_i}\right)$ *énumère (non récursivement !) un ensemble hyperimmune par ordre strictement croissant.*

On a rappelé au chapitre I qu'un ensemble est hyperimmune si et seulement si il n'est majoré par aucune fonction récursive. Il en existe une infinité qui a la puissance du continu [30].

Proposition 3

Une suite de poids W est fortement convergente si et seulement si elle est p.p. majorée par une suite de poids cohyperimmune.

Sinon, il existe une fonction récursive, soit f, qui majore $\left(\frac{1}{w_i}\right)$ p.p., et donc $(\exists n_0)(\forall n)[n\geq n_0 \Longrightarrow f(i)w_i\geq 1]$, et $\|f\|'_W=\infty$, une contradiction.

En effet, soient (z_i) l'ensemble hyperimmune fourni par l'hypothèse, et $f\in R^1$.

Alors $(\overset{\infty}{\forall} i)[\frac{1}{w_i}>z_i>2^i.f(i)]$, puisque $\lambda x[2^x f(x)]\in R^1$. Ainsi, $(\overset{\infty}{\forall} i)[f(i)w_i<\frac{1}{2^i}]$, et donc $\|f\|_W<\infty$. ⊠

Gold (1965) a montré que toute sous-classe r.e. de R^1 est identifiable à la limite, alors que $R^1_{[0,1]}$, classe des fonctions récursives à valeurs dans {0,1}, ne l'est pas. On peut chercher à améliorer ce résultat pour l'identification approximative (ou pour l'identification faible) en utilisant une classe Γ r.e. W- (ou $\simeq$-) dense dans $R^1_{[0,1]}$. Nous avons vu que c'est possible si W est quelconque. Par contre, c'est impossible dans le second cas : aucune classe r.e. n'est $\simeq$-dense dans $R^1_{[0,1]}$! Ceci découle de la "forte" productivité de $R^1_{[0,1]}$.

Notation

Si W_x est un ensemble d'indices de fonctions totales, Γ_x désignera l'ensemble de ces fonctions, et $\emptyset$ sinon : $\Gamma_x=\{f:(\exists y\in W_x)[f=\varphi_y \ \&\ (\forall z\in W_x)[\varphi_x\in R^1]]\}$.

Théorème 2

Si F désigne R^1 ou $R^1_{[0,1]}$:

(1) Il existe des classes Γ r.e. de fonctions récursives W-denses dans F, à condition que W soit fortement convergente si $F=R^1$.

(2) Il n'existe aucune classe Γ r.e. de fonctions récursives $\simeq$-dense dans F. Plus précisément, pour chaque valeur de F, il existe une fonction productive totale $h\in R^1$ telle que, si $\Omega^{\simeq}$ désigne la $\simeq$-fermeture d'un ensemble Ω de fonctions, $(\forall x)[\varphi_{h(x)}\in F - \Gamma_x^{\simeq}]$

(1) Si $F=R^1_{[0,1]}$, ou si F est bornée par une constante C, $(\forall\varepsilon>0)(\exists n_0)[\sum_{n>n_0} C\, w_n<\varepsilon]$

Il existe une infinité de classes r.e. Γ telles que $\Gamma(0,\ldots,n_0)$ prenne les $(C+1)^{n_0}$ valeurs possibles, et qui répondent donc à la question.

Si $F=R^1$, ou si F est non-bornée, on a vu plus haut que :

$$(\forall\varepsilon>0)(\forall f\in F)(\exists n_0)[\sum_{n>n_0} C\, w_n<\varepsilon].$$

On ne peut plus intervertir les deux derniers quantificateurs. Mais on peut construire facilement une énumération $\Gamma=(f_i)$ qui répond à la question : soit $m_n=n^n$ $(m_0=0)$. Quand $m_p\leq n<m_{p+1}$, f_p décrit l'ensemble de (toutes) les fonctions (récursives) f telles que :

$$\begin{cases} f(x)\leq n & \text{si } x\geq n \\ f(x)=0 & \text{sinon} \end{cases}$$

(2) Soit f une fonction récursive (primitive) telle que $(\forall x)[W_x=\mathrm{Im}\varphi_{f(x)} \ \&\ W_x\neq\emptyset \Longrightarrow W_{f(x)}=\mathbb{N}]$ (voir [16], p. 61). Notons pour simplifier $\rho_x=\varphi_{f(x)}$.

- Si $F=R^1$, on définira $\psi(x,n)=\begin{cases} 0 & \text{si } e_x(0) \text{ ne converge pas en moins de n pas} \\ 1+\underset{i\leq n}{\mathrm{Max}}\ e_x(i) & \text{sinon} \end{cases}$

$\psi\in R^2$ par Church, et, via le théorème s-m-n, $(\exists h\in R^1)[\lambda n[\psi(x,n)]=\varphi_{h(x)}]$. h répond à la question, puisque $\varphi_{h(x)}$ est différente de chaque $\varphi_{e_x(i)}$ en tout point.

- Si $F=R^1_{[0,1]}$, il faut produire une fonction à valeurs dans {0,1} et la construction précédente ne convient plus. On peut alors utiliser des "priorités" $\eta_i^{(n)}$, comme dans la démonstration du théorème 1. Intuitivement, $\eta_i^{(n)}$ représente le nombre des derniers arguments consécutifs $y\leq n$ pour lesquels $\psi(x,y)=\varphi_{e_x(i)}(y)$, en définissant ψ par :

$$\psi(x,n)=\begin{cases} 0 & \text{si } e_x(0) \text{ ne converge pas en moins de n pas} \\ 1\dot{-}\varphi^{(n)}_{e_x(t)}(t) & \text{sinon} \end{cases}$$

avec $t=\mu i\leq n[\eta_i^{(n)}=\underset{j\leq n}{\text{Max}}\ \eta_j^{(n)}]$

$$\text{et} \begin{cases} \eta_i^{(0)} = 0 & \text{pour tout i} \\ \eta_i^{(n+1)}= \begin{cases} 0 & \text{si } \psi(x,n)\neq\varphi^{(n)}_{e_x(i)} \\ 1+\eta_i^{(n)} & \text{sinon} \end{cases} \end{cases}$$

h est définie comme plus haut et répond à la question. En effet, si $\varphi_{h(x)}\simeq\varphi_{e_x(i)}$ pour un certain i, soit i_o le plus petit indice possédant cette propriété. η_{i_o} doit donc être de la forme $\eta_{i_o}^{(n)}=n_o+n$, ce qui est impossible par construction, puisqu'il arrive nécessairement un instant n où $t=i_o$, sans quoi i_o ne serait pas le plus petit indice possédant la propriété. ⊠

On peut cependant modifier la formulation du théorème 1 pour obtenir le

Théorème 3

Soit Γ une classe r.e. de fonctions récursives ($\Gamma\subset R^1$). Il existe un algorithme M qui identifie faiblement (approximativement) toute fonction de la $\simeq$-fermeture $\Gamma^{\simeq}$ (de la W-fermeture Γ^W) de Γ dans F, où F désigne R^1 ou $R^1_{[0,1]}$. De plus, M dépend effectivement de Γ.

(1) Cas de $\Gamma^{\simeq}$ et de l'identification faible : Γ s'écrit $\Gamma=\Gamma_x$ pour un certain x. Alors, comme dans la démonstration précédente, $(\varphi_{e_x(i)})_i=(f_i)$ est une énumération de Γ, telle que e_x est partout définie si $\Gamma\neq\emptyset$ et jamais sinon. On ne peut utiliser la méthode de Gold pour l'identification, qui consiste à essayer successivement $f_0,f_1,\ldots$ en passant de f_n à f_{n+1} dès que $f_n(t)\neq\varphi(t)$ pour un certain t : il faut pouvoir essayer chaque f_i une infinité de fois.
Rappelons que λxy[<x,y>] désigne la fonction de Cantor habituelle, d'inverses π_1 et π_2. On posera :

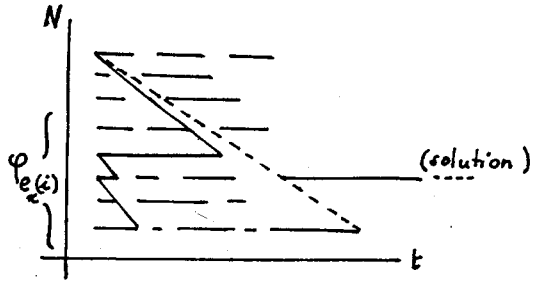
$$\begin{cases} t(x,0) = 0 \\ t(x,n+1)= \begin{cases} t(x,n)+1 & \text{si } \varphi_{N(x,n)}(t(x,n))=\varphi(t(x,n)) \\ \pi_1(<t(x,n),N(x,n)>+1) & \text{sinon} \end{cases} \end{cases}$$

$$\begin{cases} N(x,0) = e_x(0) \\ N(x,n+1)= \begin{cases} N(x,n) & \text{si } \varphi_{N(x,n)}(t(x,n))=\varphi(t(x,n)) \\ \pi_2(<t(x,n),N(x,n)>+1) & \text{sinon} \end{cases} \end{cases}$$

---

§ - Si §=≃ ou W, la §-fermeture de Γ dans F est définie par $\Gamma^{\S}_F=\Gamma^{\S}_{R^1}\cap F$.

On parcourt schématiquement l'ensemble des possibilités de la façon suivante (les traits horizontaux représentent les intervalles où $\varphi_{x(i)}=\varphi$). Il est clair que N et t sont récursives. De plus, $(\forall x,n)[t(x,n)\leq n]$, et donc la suite des arguments est "moins rapide" que l'expérience. Enfin, via le théorème s-m-n, $\lambda n[N(x,n)]=\varphi_{h(x)}$, avec $h\in R^1$, ce qui achève la preuve.



(2) Cas de I^W et de l'identification approximative : la démonstration est analogue à celle de Gold. Notons f[n] la fonction g telle que :

$$g(x) = \begin{cases} f(x) & \text{si } x\leq n \\ 0 & \text{sinon} \end{cases} \quad \text{et fixons } \varepsilon.$$

Alors : $N(x,0) = \bar{e}_x(0)$

$$N(x,n+1) = \begin{cases} N(x,n) & \text{si } d_W(\varphi_{N(x,n)}^{[n]} - \varphi[n]) < \varepsilon \\ N(x,n)+1 & \text{sinon} \end{cases}$$

Comme plus haut, $N\in R^2$ et $(\exists h\in R^1)[\lambda n[N(x,n)] = {}_{h(x)}]$, ce qui achève la preuve.

1.2.2. BOITES NOIRES

On a déjà vu deux classes d'objets différentes : les grammaires et les fonctions. Les "boîtes noires" [17] sont des automates particuliers pour lesquels la situation d'apprentissage est un peu plus complexe.

Définition 11

Une boîte noire *est définie par :*

- *un* alphabet d'entrée *E identifié à* $\{1,\ldots,|E|\}$, *avec* $2\leq|E|<\infty$
- *un* alphabet de sortie *S identifié à* $\{1,\ldots,|S|\}$, *avec* $2\leq|S|<\infty$
- *une* fonction de boîte noire $b:E^+\to S$

Notations

On note e_n et s_n l'entrée et la sortie à l'instant n, et $\bar{e}_n$, $\bar{s}_n$ les chaînes correspondantes : $e_n=e_0e_1\ldots e_n$, $\bar{s}_n=s_0s_1\ldots s_n$. On aura donc $s_n=b(\bar{e}_n)$. E^* opère sur les fonctions de boîtes noires de la façon suivante :

$$(\forall\bar{e}\in E^*)(\forall\bar{e}'\in E^+)[D_{\bar{e}}b(\bar{e}') = b(\bar{e}\bar{e}')]$$

Définition 12 (Gold)

Une fonction de chaîne $b:E^*\to S$ *est* récursive (primitive) *s'il existe une fonction f récursive (primitive) telle que*

$$(\forall\bar{e}\in E^*)[f(\langle\bar{e}\rangle) = b(\bar{e})]^{\S}.$$

Un transducteur d'états finis définit une boîte noire "EF" *si on fixe un état initial et si on néglige le comportement (parcours des états).*

Une situation d'identification à la limite *est constituée par deux fonctions de chaînes récursives*

$$\Theta : S^* \to E$$
$$M : S^+ \to \mathbb{N}$$

telles que $\rho_0 = \Theta(\Lambda)$ *et* $(\forall n)$ $\begin{cases} s_n = b(\bar{e}_n) \\ N_n = M(\bar{s}_n) \\ e_{n+1} = \Theta(\bar{s}_n) \end{cases}$ *Soit* $M(b) = \lim_{n\to\infty}\varphi_{N_n}$

§ – On note encore b(e) l'entier p associé de manière canonique (effective), par exemple, si $e=e_0e_1\ldots e_n$, $p=\sum_{i=0}^{n}e_i$, ce qui correspond au code d'une MT à deux symboles {b,1}.

On dira que (θ,M) identifie fortement b à la limite *si M(b) est définie et si* $(\forall \bar{e}\in E^+)[b(\bar{e})=\varphi_{M(b)}(<\bar{e}>)]$.

(θ,M) identifie faiblement b à la limite *si M(b) est définie et si* $(\exists n)(\forall \bar{e}\in E^*)[D_{\bar{e}_n} b(\bar{e})=D_{\bar{e}_n} M(b)(<\bar{e}>)]$ §.

L'identification faible s'apparente à l'approche faible des cas précédents : à partir d'un certain moment, M(b) simule le comportement ultérieur de b. L'identification forte est en général impossible sauf pour certaines classes de boîtes noires (b peut "piéger" l'apprentisseur, par exemple $b(\bar{e})=\bar{e}_0$ pour tout $\bar{e}$, avec E=S). Gold a démontré les résultats suivants :

Théorème 4 (Gold)

(1) Soit Γ une classe r.e. de boîtes noires, i.e. telle qu'il existe une fonction récursive $\lambda i\bar{e}[b_i(\bar{e})]$ *de* $\mathbb{N}\times E$ *dans* $\mathbb{N}$, *de même E et telle que le* prédicat P de "distingabilité" *soit récursif :* $P(i,j,\bar{e})$ $D_{\bar{e}}b_i\neq D_{\bar{e}}b_j$ $(\bar{e}\in E^*)$. *Alors Γ est faiblement identifiable à la limite.*

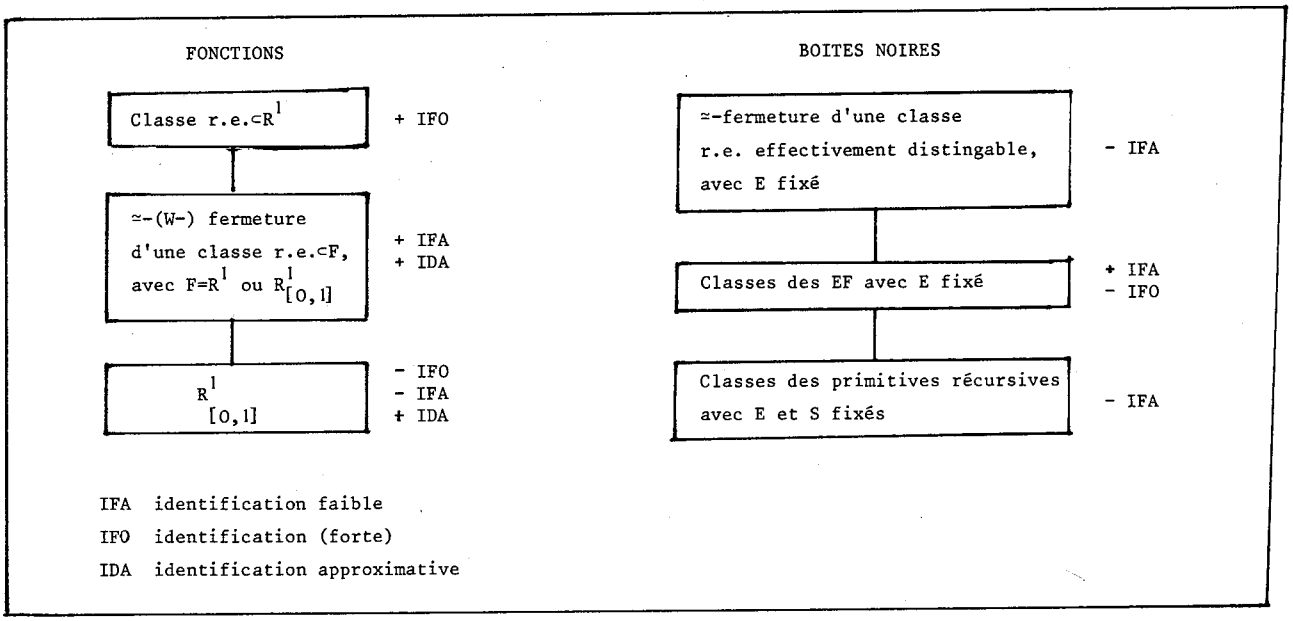
(2) La classe des boîtes noires EF de même alphabet E est faiblement identifiable à la limite.

(3) La classe des boîtes noires primitives récursives de mêmes alphabets E et S n'est pas faiblement identifiable à la limite.

Remarquons que la condition de distingabilité effective est essentielle. On ne peut en effet appliquer une méthode analogue à celle du théorème 3 (parcours par diagonales) car on ne pourrait pas assurer que $D_{\bar{e}_n} b(\bar{e}) = D_{\bar{e}_n} \varphi_{M(b)}(<\bar{e}>)$ pour tous les $\bar{e}\in E^*$. D'autre part, le (1) se généralise trivialement au cas où Γ est contenue dans la ≃-fermeture d'une classe r.e. à distingabilité effective.

1.2.3. RESULTATS

Résumons graphiquement les résultats précédents avant de faire quelques remarques.



On peut se demander pourquoi la classe des langages PR (primitifs récursifs) est IFO, alors que celle des boîtes noires PR n'est pas IFA. Ceci provient de ce que l'ensemble des valeurs d'une fonction est "linéaire", alors que celui d'une boîte noire est "arborescent", et aussi de la méthode de présentation de l'information, complète (informant) dans le premier cas et très contrainte dans l'autre. On notera dans les autres cas la cohérence des résultats obtenus aux paragraphes 1.2.1 et 1.2.2 .

---

§ - $\bar{e}_n$ désigne la chaîne engendrée par (θ,M).

II - METHODES STRUCTURALES

Les méthodes précédentes sont relativement peu puissantes : pour les langages, l'apprentissage "fort" exige un informant, pour les fonctions et les boîtes noires, il faut la plupart du temps se contenter de l'identification faible ou approximative. Ceci est dû au fait qu'on observe seulement le comportement externe des objets.

On peut obtenir des résultats plus forts si on fournit à l'apprentisseur des "exemples de calcul" ou "comportements" (de MT, d'automates d'états finis ...), ou si la structure est donnée par une structure de base et un ensemble de paramètres (modèles stochastiques, perceptrons, systèmes paramétrés ou pondérés). Enfin, on étudiera également dans cette partie le cas du passage effectif entre deux types de structures (passage d'une grammaire sous-contexte (SC) d'un langage hors-contexte (HC) à une de ses grammaires HC,...).

2.1. METHODES CONSTRUCTIVES

Ces méthodes utilisent des renseignements sur la "forme" des grammaires ou des algorithmes, et des séquences d'information souvent plus "précises" : on utilise le plus possible d'information "a priori" pour réduire la longueur des énumérations, en modifiant "localement" N_t au temps t+1 pour obtenir N_{t+1}. On peut donc commencer avec un objet initial (N_0) "vide" ou arbitraire.

2.1.1. GRAMMAIRES

Donnons trois exemples sur les grammaires, tous implémentés.

(1) Crespi-Reghizzi (1970) a introduit un algorithme d'identification à la limite pour les langages de précédence, quand la séquence d'information est formée de descriptions syntaxiques (parenthésages) de chaînes du langage.

(2) Vivier (1974) a introduit la notion de "langages facile à apprendre". Les langages "faciles" et "difficiles" intersectent la hiérarchie de Chomsky. On utilise une représentation en réseau étiqueté assez analogue à celle de l'Ecole de Vienne [27]. L'algorithme est original en ce sens qu'il procède à une réorganisation globale (en faisant des simplifications possibles à cause de la propriété de "remplaçabilité" des langages faciles) quand il ne peut plus faire de modification simple. Là encore, l'algorithme identifie à la limite les langages faciles.

(3) S. Klein (1973) cherche à apprendre des "règles de structure profonde". Une telle règle est le couple d'une règle HC et de sa réalisation sous forme de combinaison de "triplet sémantique"[§]. Par exemple :

S → NP VP // O - R (pour "objet" - "relation")

NPP → adj NPP // O - R (attribut) - O

La séquence d'information consiste en une suite de phrases et de leurs représentations sémantiques. Par comparaison et simplification entre règles, on fabrique petit à petit une telle grammaire. Malheureusement, ce type de description ne peut convenir qu'à des sous-ensembles très restreints de langues naturelles.

2.1.2. BOITES NOIRES

Soit f une fonction : $X^* \to Y$. La relation de Nérode associée R_f est définie par : $(x_1,x_2) \in R_f \equiv (\forall z \in X^*)[f(x_1 z)=f(x_2 z)]$. Biermann et Feldman (1972a) ont modifié la méthode de Nérode pour fabriquer un automate EF qui calcule f, si R_f est finie, à partir d'un ensemble fini S de couples (x,f(x)) assez grand. On peut appliquer cette méthode à une situation d'apprentissage où $(x_t,f(x_t))$ est fourni au temps t, et l'algorithme qui en résulte est "constructif", au sens où il suffit de modifier localement la machine M_t pour trouver M_{t+1}. Si f n'est pas totale, l'identification n'est possible que si on a une borne a priori du nombre des états de la machine M(f).

Biermann (1972) a généralisé ce traitement aux MT en construisant et en implémentant un algorithme qui identifie une MT en utilisant une séquence de calculs (et non de couples argument-valeur) de cette machine. Mais on peut dire que la méthode est trop générale : elle suppose en effet qu'on connaisse la MT à apprendre, a priori. C'est une différence avec les cas précédents, où on peut très bien utiliser une grammaire G pour produire la séquence I et trouver une autre grammaire que G.

---

§ - l'analogue d'un "énoncé élémentaire" en langage pivot (cf. chapitre V), ou d'une "lexis" [32].

2.2. METHODES PARAMETRIQUES

Les modèles "stochastiques" sont utilisés de façon assez générale en psychologie ou en psycholinguistique [1,6,28,31,37]. Les objets considérés sont des arborescences probabilisées correspondant à des situations avec des points à choix multiples : s'il y a k alternatives au point P, on note leurs probabilités $p_1,\ldots,p_k$. L'apprentissage consiste à modifier les $p_i(t)$ en fonction de la réponse au temps t. Ces "modèles" sont autant de formules de réajustement. On peut généraliser ceci au cas de l'apprentissage d'un langage engendré par une grammaire fixée, munie de probabilités, ou (cf. chapitre III) de poids, avec un critère fixé.

Dans le cas des perceptrons [25], ou des "machines à apprendre" [26], on considère qu'on a un ensemble d'opérateurs simples, Φ, et l'opérateur cherché f en est une combinaison linéaire. Les coefficients, ou poids, A, de cette combinaison sont les inconnues. On peut les approcher au moyen d'un algorithme simple d'apprentissage (au sens du 1.2.2), qui s'arrête après un nombre fini de pas, pourvu qu'il y ait au moins une solution. C'est, informellement, le théorème de convergence des perceptrons [25, p. 164]. Cependant, Minsky & Papert montrent bien que ce modèle est (volontairement) trop simple, et qu'on ne peut généraliser cette propriété à des systèmes de perceptrons.

Cependant, la voie de l'ajustement de paramètres semble être la seule qui donne des résultats dans le cas de systèmes complexes [12,38], où, de façon générale, on cherche à maximiser ou minimiser une certaine fonction sur l'ensemble des arguments du système. [38, chapitre IX] en donne toute une série d'exemples, qui incluent d'ailleurs les perceptrons. Il est possible d'appliquer cette idée à certaines classes de systèmes définitionnels.

Définition 13

Soit $D(T,\delta)$ un système définitionnel. Notons $t_1,\ldots,t_N$ les transitions de T (éléments de la relation α de contrôle fini), et supposons que $\delta=(\varepsilon,\tilde{\eta},\gamma,Q)$.
D est un système définitionnel pondéré *si :*

(1) π est une pondération *(sur $\mathbb{N}$) de $\{t_1,\ldots,t_N\}$*

(2) $\iota\in R^1$ est une fonction d'évaluation initiale

(3) $\sigma_1,\ldots,\sigma_N\in R^2$ sont des fonctions de passage *associées aux transitions*

(4) la fonction d'évaluation ε est déterminée par :

$$\varepsilon(x.c) = \begin{cases} \iota\varphi^{-1}(c) & \text{si } x = \Lambda \\ \sigma_k(\varepsilon(x),\pi(k)) & \text{sinon, et si } c \text{ est déduite de } x \text{ via } t_k. \end{cases}$$

La situation d'apprentissage dans ce cas consiste, connaissant T,ι et les σ_i, à chercher π à partir d'une expérience qui fournit les valeurs de ε sur un ensemble (r.e.) de calculs. Il est évident qu'on peut énumérer tous les π possibles de manière effective, et donc l'ensemble des ε possibles est une sous-classe r.e. de R^1.

Notation

Désignons par $\mathcal{D}(T,\tilde{\eta},\iota,\sigma)$ l'ensemble des systèmes définitionnels pondérés associés à T, pour $\tilde{\eta}$, ι, et $\sigma = \{\sigma_i\}$ fixés.

Proposition 4

(1) Toute classe $\mathcal{D}(T,\tilde{\eta},\iota,\sigma)$ est identifiable à la limite.

(2) Sa ≃- (W-) fermeture $\mathcal{D}^{\simeq}(\mathcal{D}^{W})$ est faiblement (approximativement) identifiable à la limite.

Pour ces fermetures, il s'agit des valeurs de ε sur $\Phi^*\theta_0(\mathbb{N})$. Cette proposition est un corollaire immédiat des théorèmes 1 et 3. ∎ Il serait en général impossible de chercher à apprendre les σ_i même s'ils sont tous égaux à σ et si on connaît ι et π, à cause de la "difficulté d'apprentissage" de R^1 (théorèmes 2 et 3).

D'autre part, on utilise implicitement une méthode énumératoire. On pourrait en trouver de plus rapides dans des cas particuliers (e.g. si les σ_i sont inversibles), mais on sait qu'aucune autre méthode ne peut être uniformément plus rapide - au moins pour l'identification forte. Cependant, si le système cherché doit rendre compte d'un phénomène expérimental, il n'est pas interdit d'utiliser la connaissance a priori de ce phénomène pour définir et essayer des algorithmes plus efficaces,

quitte à ce qu'ils fournissent des solutions moins précises. C'est par exemple le cas quand le but de l'apprentissage est de minimiser une fonction de coût connue, même de façon incomplète [38] . Dans son programme de simulation de l'évolution des grammaires (stochastiques) des membres d'une communauté sociale, S. Klein (1966) a même pu pousser la méthode jusqu'au point où l'algorithme d'apprentissage est un système pondéré, lui-même susceptible d'ajustement. Une telle façon de procéder n'a bien sûr pas de valeur universelle, mais seulement expérimentale, c'est-à-dire qu'elle est justifiée par les résultats qu'on en tire.

2.3. SIMPLIFICATION

2.3.1. RAPPELS

Il s'agit ici des méthodes qui permettent de passer d'une description d'un objet dans un formalisme F_1 à une description dans un formalisme F_2 "plus simple", si elle existe. Par exemple, on cherche à passer effectivement (via $\psi \epsilon P^1$) des indices r.e. aux indices caractéristiques pour les ensembles récursifs, de sorte que :

$$(\forall x)[W_x \text{ récursif} \Longrightarrow \psi(x) \downarrow \ \& \ W_x = C_{\psi(x)}]$$

On sait [16] que ceci est impossible, de même que le passage des indices caractéristiques aux indices canoniques pour les ensembles finis.

Shamir (1962) appelle un tel ψ un "algorithme de découverte de F_2 par rapport à F_1". (Il est clair que F_1 permet d'exprimer tous les objets de F_2). On peut aussi parler d'algorithme de simplification de F_1 vers F_2. On a le résultat négatif important :

Théorème 5 (Shamir)

Soient Γ_1 et Γ_2 deux classes r.e. de récursifs, avec $\Gamma_2 \subset \Gamma_1$ et $\Gamma_i = \{W_x | x \epsilon X_i\}$, $i \epsilon \{1,2\}$ et X_i r.e.. Soit P une propriété d'ensemble telle que :

(1) $\{x | x \epsilon X_1 \ \& \neg P(W_x)$ est r.e. non récursif (P est donc indécidable dans Γ_1).

(2) $(\forall x \epsilon X_1)[P(W_x) \Longrightarrow (\exists y \epsilon X_2)[W_x = W_y]]$: tout ensemble de Γ_1 vérifiant P est dans Γ_2.

(3) $\{x | x \epsilon X_2 \ \& \ P(W_x)\}$ est récursif : P est décidable dans Γ_2.

Il n'existe alors pas d'algorithme de simplification de Γ_1 vers Γ_2.

Sinon, soit $x \epsilon X_1$. Il suffit d'énumérer l'ensemble du (1) et d'appliquer en même temps l'algorithme de simplification ψ à x. Nécessairement, ou bien x apparaît dans l'énumération (et alors $\neg P$), ou bien $\psi(x) \downarrow$, et alors P est décidable sur $\psi(x)$, ce qui permet de décider P dans Γ_1, une contradiction. ✉

Corollaire (Shamir)

Il n'y a pas d'algorithme de simplification

(1) des langages hors-contexte vers les langages d'états finis.

(2) des langages sous-contexte vers les langages hors-contexte.

On peut prendre pour P la propriété d'être égal au langage maximal A^* (1) ou au langage vide (2). ✉ Par conséquent, l'algorithme de Peters & Ritchie (1969), qui fournit un HC contenant un SC arbitraire, ne peut pas toujours donner ce SC lui-même, quand il est HC.

2.3.2. "ENVELOPPES"

Ces résultats suggèrent donc de chercher plutôt un ψ, total, tel que $\psi(x)$ nomme dans F_2 un objet assez proche de x (dans F_1). Il est naturel de chercher un récursif (un HC) "assez petit" contenant un r.e. (SC) donné. On ne peut entendre "assez petit" au sens de "le plus petit". En effet :

(1) s'il existait un plus petit $C_{\psi(x)}$ contenant W_x, ψ permettrait de passer des indices r.e. aux indices caractéristiques. D'autre part, si W_x est r.e. non récursif, et si $W_x \subset C$ récursif, $|C-W_x|=\infty$, et on peut toujours diminuer C d'un nombre fini de points pour obtenir C' récursif.

(2) on peut faire les mêmes remarques pour les HC (EF) contenant un SC (HC) infini, puisque les langages finis sont EF et que les classes HC et SC sont stables par intersection avec la classe EF.

Une définition utilisant une suite de poids W ne peut donner que des résultats positifs et triviaux, tout au moins pour ces classes de langages. Elles sont en effet récursives, et on peut approcher tout langage de cette classe, de manière effective et à ε près, par un sous-langage fini. Pour l'approche des r.e. par des récursifs, on peut faire une observation tout aussi décourageante :

Proposition 5

Pour toute suite de poids W, et pour tout $\varepsilon>0$, il existe une classe r.e. infinie d'ensembles r.e. infinis distincts et tous W-approchables par $\emptyset$ à ε près.

En effet, $(\forall\varepsilon)(\exists n)[\sum_{i\geq n} w_i \leq \varepsilon]$. Une telle énumération est par exemple donnée par $W_{\sigma(i)} = \{x | x \geq i\}$ pour tous les $i \geq n$.

Toujours pour l'approche récursive de r.e., Lynch (1973) a proposé une définition fondée sur des critères de densité. Ses résultats montrent que, suivant la gödelisation choisie, le problème de l'arrêt (i.e. l'ensemble $K = \{x | \varphi_x(x) \downarrow\}$) peut être arbitrairement approchable ou inapprochable par un ensemble récursif.§

On peut proposer une définition affaiblie :

Définition 14

*Soient Γ_1 et Γ_2 deux classes d'ensembles, avec $\Gamma_1 \supset \Gamma_2$. L'*enveloppe d'un ensemble A de Γ_1 relativement à Γ_2 *est la classe des ensembles C de Γ_2 tels que :*

$$(\forall B \in \Gamma_2)[A \subset B \subset C \Longrightarrow |C-B| < \infty]$$

Le théorème suivant montre qu'il y a des cas simples où l'enveloppe est vide, ou bien constituée d'ensembles p.p. égaux à l'ensemble total ($\mathbb{N}$ ou Σ^* §§).

Théorème 6

(1) L'enveloppe récursive de tout ensemble simple est constituée d'ensembles p.p. égaux à $\mathbb{N}$.

(2) L'enveloppe récursive de tout ensemble créatif A est vide. Plus précisément,

$$(\exists f \in R^1)(\forall x)[A \subset C_x \Longrightarrow A \subset C_{f(x)} \subset C_x \ \&\ |C_x - C_{f(x)}| = \infty]$$

(3) Tout langage hors-contexte strict qui ne contient aucun langage régulier infini§§§ a une enveloppe EF vide.

---

§ - Plus précisément, soit φ une gödelisation et K_φ l'ensemble associé, on exhibe deux ensembles récursifs A et B, $A \subset K_\varphi$ & $B \subset \bar{K}_\varphi$, tels que $d(A \cup B) > 1-\varepsilon$, ou $< \varepsilon$.

§§ - Σ désigne ici l'alphabet des langages considérés.

§§§ - On pourrait dire : EF-immune.

(1) En effet, si A est simple et C⊃A récursif, $\bar{C}$ est r.e., et ne peut donc être infini, puisque $\bar{A}$ est immune.

(2) Si $C_x \supset A$ est récursif, $\bar{C}=C_{t(x)}$, où $t\in R^1$. Soit h une fonction productive totale de $\bar{A}$. On va l'utiliser pour fabriquer un ensemble D récursif infini, avec $D \subset C_x - A$.

On aura $D=C_{1(x)}$, et $C_x - C_{1(x)} = C_{f(x)}$ vérifiera la propriété.

Soient $u\in R^2$, $g\in R^2$ et $\tilde{h}\in R^3$ définies par :

$$(\forall x)[W_{g(x)}=C_x \ \&\ W_{u(x,y)}=W_x\cup\{y\} \ \&\ W_{\tilde{h}(x)}=W_{f(x,h(x))}=W_x\cup\{h(x)\}]$$

On définit alors $p\in R^2$, $q\in R^1$ et $r\in R^2$ par :

$$\begin{cases} p(x,0) = hgt(x) \\ p(x,n+1)=r(x,n,\mu m[r(n,m)>\{p(0),\dots,p(n)\}]) \end{cases} \qquad \begin{cases} q(x,0) = gt(x) \\ q(x,n+1)=u(q(x,n),p(x,n)) \end{cases} \qquad \text{et } r(x,n,m)=h\tilde{h}^m q(x,n)$$

Il est clair que $\lambda n[p(x,n)]$ énumère un ensemble D infini, $D \subset C_x - A$. Cette fonction est croissante strictement, D est donc récursif et on peut effectivement trouver sa fonction caractéristique à partir de x (via s-m-n), d'où le résultat.

(3) La démonstration repose sur la propriété suivante : si z est une chaîne telle que $|z|\geq p$ d'un EF C, on peut écrire z=uvw, où v est assez petite ($0<|v|\leq q$, p et q sont des constantes pour C), et telle que $C_{uvw}=\{uv^n w \mid n>0\}\subset C$. Le lemme suivant montrera que, si A ne contient pas d'EF infini, $(\forall u,v,w)[C_{uvw}\subset C \implies |A\cap C_{uvw}|<\infty]$.

Choisissons un tel triplet (u,v,w), il existe alors k>0 : $(\forall n)[n\geq k \implies uv^n w\notin A]$, et donc $C-A\supset C_{uv^k w}$. La classe EF étant stable par complémentation et intersection, $B=C-C_{uv^k w}$ est encore EF, et $|C-B|=\infty$. ⊠ Il reste à démontrer le lemme.

Lemme

Soient $A\subset\Sigma^*$ un langage hors-contexte infini et $C_{uvw}=\{uv^n w \mid n>0\}$, avec $u,v,w\in\Sigma^*$ et $|v|>0$. Si $|C_{uvw}\cap A|=\infty$, A contient un langage régulier infini.

Soit G une grammaire HC de A. Elle contient nécessairement un non-terminal X tel que, pour un certain n, une dérivation de $uv^n w$ dans G contienne au moins |uw| fois X. Deux cas se présentent :

-1- On peut trouver un découpage de $uv^n w$ en $\alpha\varphi\theta\psi w$, où $|\alpha|\geq|u|$, $|\omega|\geq|w|$, $X\overset{*}{\underset{G}{\vdash}}\varphi X\psi$ et $X\overset{*}{\underset{G}{\vdash}}\theta$.

Alors $X\overset{*}{\underset{G}{\vdash}}\varphi^{|v|+1}\theta\psi^{|v|+1}$, et cette chaîne "partage v" comme θ, i.e.

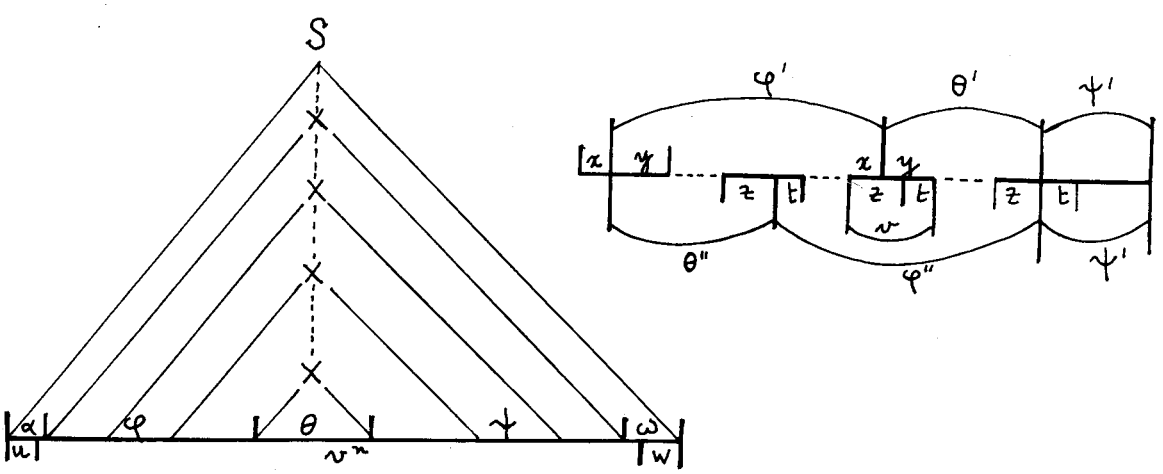
$(\exists x,y,z,t)[v=xy=zt \ \&\ \theta\in yv^*z \ \&\ \varphi^{|v|+1}\in yv^*x \ \&\ \psi^{|v|+1}\in tv^*z]$.

Posons $\varphi'=\varphi^{|v|}$, $\psi'=\psi^{|v|}$ et $\theta'=\varphi\theta\psi$.

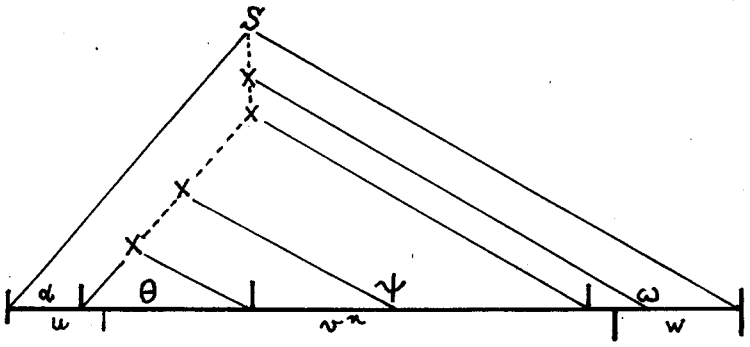
$(\exists m)[uv^m w=\alpha\varphi'\theta'\psi'\omega'\in A]$, et il est clair que :

$(\forall m>0)[\alpha\varphi'^m\theta'\psi'^m\omega\in A]$. Mais on peut réécrire $\varphi'\theta'\psi'=\theta''\varphi''\psi'$, où $|\theta'|=|\theta''|$ et $|\varphi''|=|\varphi'|$, et,

$(\forall m>0)[\varphi'^m\theta'\psi'^m=\theta''(\varphi''\psi')^m]$. Si donc $u'=\alpha'\theta''$, $v'=\varphi''\psi'$ et $w'=\omega'$, on en déduit que $C_{u'v'w'}\subset A$.



-2- Sinon, il existe un couple (φ,ψ) tel que $X \overset{*}{\underset{G}{\vdash}} \varphi X \psi$ et que φ ou ψ soit la chaîne vide (à cause du nombre de X dans l'arbre d'analyse). Soit par exemple $\varphi=\Lambda$ (et donc $\psi\neq\Lambda$). Comme il y a plus de $|u|+|v|$ réécritures de X, et moins de $|u|$ qui développent une chaîne terminale à gauche de X, il y en a plus de $|v|$ qui développent une chaîne terminale à droite de X, et on peut donc trouver un découpage de $uv^n w$ en $\alpha\theta\psi\omega$, où $|\alpha\theta|>|u|$ et $|\omega|>|w|$. Par conséquent, si $u'=\alpha\theta\psi$, $v'=\psi^{|v|}$ et $w'=\omega$, $C_{u'v'w'}\subset A$.



Remarque :

Certains problèmes restent ouverts :

- Peut-on étendre le (2) aux ensembles subcréatifs [30,31] ? A est subcréatif s'il est r.e. et si $(\exists \sigma\in R')(\forall i)(\exists a_i)[W_{\sigma(i)}=A\cup\{a_i\} \;\&\; a_i\in A \iff a_i\in W_i \;\&\; |W_i\cap A|<\infty \implies a_i\in\bar{A}]$. Cette propriété caractérise les ensembles "effectivement accélérables".

- Peut-on obtenir un résultat analogue au (3) pour l'enveloppe HC des SC ?
En particulier, le lemme se généralise-t-il au cas où $C_{uvwxy}=\{uv^n wx^n y \mid n>0\}$, en voyant alors quand A SC contient un HC infini ?

CONCLUSION : APPRENTISSAGE DANS UN SYSTEME DE TA.

Les résultats théoriques sur l'apprentissage ou l'approche des langages formels sont bien modestes, dès qu'on s'intéresse à des classes un peu complexes. Le problème des langues naturelles est qu'on ne sait pas définir une classe formelle dont les langages soient "presque tous" des langues naturelles en puissance, et on en est réduit à chercher (cf. chapitre III) à augmenter des modèles simples (grammaires HC ou réseaux de transitions munis de conditions et d'actions - saturations, validations ou opérations sur des "registres") pour tenter de décrire le plus complètement possible les phénomènes des langues naturelles. Pour ce genre de modèle, les méthodes constructives d'apprentissage sont irréalisables, à cause de la combinatoire énorme des "ajouts". Si par exemple on peut envisager d'"apprendre" un langage d'expressions régulières en construisant des réseaux de transitions récursifs, parce que chaque arc porte au plus un symbole, on ne peut plus le faire dès qu'on autorise des conditions booléennes arbitraires, parce qu'il faudrait toutes les énumérer sur chaque arc possible : la méthode redeviendrait énumératoire.

On peut par contre tenter d'associer une fonction d'apprentissage à chaque module du système, en limitant ses ambitions. Par exemple, on peut rendre l'analyseur morphologique interactif, de façon à permettre un enrichissement des dictionnaires au cours du traitement, si des mots inconnus ou des formes erronées apparaissent, par une interaction "conversationnelle" avec l'utilisateur. Pour les modules plus complexes (algogrammaire, moniteur,...) on peut tenter d'appliquer des méthodes paramétriques, et je parlerai alors plutôt d'adaptation que d'apprentissage.

De toutes façons, dans l'état actuel des connaissances sur l'acquisition du langage et sur les mécanismes de compréhension et de traduction chez l'homme, il ne semble pas possible d'en tirer un modèle implémentable, et à fortiori de les utiliser dans des systèmes de TA exploitables en pratique. Cependant, il est très possible que des progrès dans ces domaines permettent dans l'avenir d'imaginer des systèmes de TA totalement différents, et bien plus adéquats, qui simuleraient beaucoup plus précisément le travail humain. Mais peut-être ceci est-il impossible sans en hériter également les faiblesses ...

TABLE BIBLIOGRAPHIQUE

[1] Atkinson & Estes (1963)
[2] Biermann (1972)
[3] Biermann & Feldman (1972a)
[4] Biermann & Feldman (1972b)
[5] Blum & Marques (1973)
[6] Bush & Mosteller (1955)
[7] Chomsky (1965)
[8] Chomsky & Miller (1963)
[9] Cook & Rozenfeld (1974)
[10] Crespi-Reghizzi (1970)
[11] Crespi-Reghizzi (1974)
[12] Csibi (1972)
[13] Feldman (1972)
[14] Feldman & al (1969)
[15] Feldman & Shields (1972)
[16] Gill & Morris (1974)
[17] Gold (1965)
[18] Gold (1967)
[19] Herault & Lwoff (1976)
[20] Hopcroft & Ullman (1969)
[21] Horning (1969)
[22] S. Klein (1966)
[23] S. Klein (1973)
[24] Lynch (1974)
[25] Minsky & Papert (1969)
[26] Nilsson (1965)
[27] Ollongren (1974)
[28] Peizer & Olmstedt (1969)
[29] Peters & Ritchie (1969)
[30] Rogers (1967)
[31] Rouanet (1967)
[32] Rouault (1971)
[33] Salomaa (1969a)
[34] Schubert (1974)
[35] Shamir (1962)
[36] Solomonoff (1964)
[37] Sternberg (1963)
[38] Tsypkin (1973)
[39] Vivier (1974)
[40] Wharton (1974)

CHAPITRE III

ALGOGRAMMAIRES

I - PRELIMINAIRES

Ce chapitre a un caractère technique malheureusement indispensable pour fournir la preuve des propriétés de l'analyseur baptisé "algogrammaire" et la base de sa programmation effective. Cette méthode d'implantation "à partir de la preuve" permet d'ailleurs de s'assurer plus facilement de la correction des programmes [35] . Le lecteur qui s'intéresse surtout à la définition pratique du système pourra donc passer directement au chapitre IV.

1.1. INTRODUCTION

Le problème de l'analyse syntaxique se pose par exemple quand on veut compiler un langage artificiel ou analyser une langue naturelle dans le cadre d'un système de TA. Mais il se pose de façon fort différente, en ce qui concerne ses données comme en ce qui concerne ses buts.

Dans le premier cas, le langage est donné, ainsi que sa grammaire, qui doit a priori être dans une certaine classe (régulière, hors-contexte, LL(k),...) et parfois satisfaire à certaines contraintes, éventuellement liées à une classe d'algorithmes d'analyse (non-ambiguïté, non-récursivité à gauche). Enfin, la donnée à analyser est une chaîne (de "terminaux"), et le résultat de l'analyse doit être indépendant de l'algorithme utilisé.

Dans le second cas, la donnée à analyser n'est pas directement la chaîne initiale -le texte-, mais le résultat de son analyse morphologique, qui se présente grosso modo soit comme une collection d'"homophrases" [51] , éventuellement déjà pré-structurées (comme dans le système ATEF [10]), soit comme un "graphe de chaînes". Chaque élément contient alors un ensemble déjà complexe de renseignements, comme le genre, le nombre, le cas, le mode. D'autre part, une langue naturelle est une donnée expérimentale, et les grammaires qu'on écrit pour en rendre compte n'en donnent que des approximations (toujours dans une certaine classe, souvent déterminée par l'existence d'un algorithme d'analyse efficace). De plus, il semble que la langue elle-même connaisse des "variations" selon l'auteur, le sujet, le lieu et le temps [28]. Ceci explique, et l'expérience l'a montré, que ces grammaires sont sans cesse remaniées. Il importe de les traiter le plus possible "telles quelles" pour qu'elles conservent leur adéquation linguistique. Les règles de ces grammaires comportent souvent des conditions et des actions liées à chaque élément de la partie droite, et il est alors intéressant de les représenter comme des "réseaux de transitions augmentés" [56] . Enfin, il me semble que le procédé utilisé par l'analyse peut avoir une importance pour le linguiste : un procédé strictement combinatoire (comme l'algorithme de Cocke) n'est sans doute pas linguistiquement très adéquat.

Ainsi, les données d'un analyseur diffèrent dans ces deux cas, leurs buts également. En effet, on s'efforce d'habitude de décrire un langage artificiel par une grammaire non ambiguë, et le procédé d'analyse doit seulement être performant en temps et en place. Par contre, il existe des phrases intrinsèquement ambiguës dans toutes les langues naturelles, et par conséquent les grammaires correspondantes doivent être ambiguës. De plus, le procédé utilisé pour lever les ambiguïtés a une importance : il peut interdire ou permettre une compréhension du processus linguistique. Comme l'écrivaient O.S. Kulagina et I.A. Mel'tchuk (1967, p. 148) :
"Thus we believe that the future of practical AT depends upon the solution of three types of problems : the linguistic problem (creation, with the help of mathematics, of systems of formal description of natural languages) ; the "gnostic-encyclopedic" problem (development of formal descriptions of human knowledge about the external world) ; and the "logical-heuristic", or artificial intelligence, problem (formalization of thinking processes)".

Relions brièvement ce chapitre aux précédents. Toute grammaire formalisée définit un système de transitions, et, si on utilise un formalisme analogue à celui des réseaux de transitions [15] , on obtient une formulation très proche de celle d'un système de machines [33]. Le but du chapitre est de définir une certaine classe d'heuristiques constructibles, et d'appliquer les résultats généraux du chapitre I à la programmation effective de cette classe, au moyen d'un langage formel qui sera présenté au chapitre IV. Enfin, pour tenir compte des "variations dans la langue" [28] , obtenir des évaluations (dans les heuristiques) assez souples et permettre une certaine adaptation, on utilisera une pondération sur les transitions du système.

Il est d'ailleurs généralement admis que les mécanismes humains de compréhension d'un texte ne sont pas purement combinatoires, mais plutôt du type "essai et erreur", "aller et retour", et qu'un de leurs ressorts principaux est l'attente, ou la "prévision" [44]. On a d'autre part souvent noté, en linguistique [22] ou dans des applications comme la reconnaissance automatique de la parole [52] , que les différents niveaux (morphologique, syntaxique, sémantique, par exemple) interréagissent. C'est pourquoi le système d'analyse présenté ici est conçu comme un sous-système d'un système

général de TA dans lequel le linguiste peut définir la stratégie globale et contrôler lui-même l'interaction des modules. Le fonctionnement de l'analyseur est prédictif de façon à donner aux heuristiques programmées une interprétation assez naturelle (et ceci n'exclut pas qu'on puisse en définir pour des procédés d'analyse ascendante [43]) §.

Il est curieux de constater que, pour des raisons différentes, la théorie de la compilation s'intéresse beaucoup aux algorithmes prédictifs ("top-down") [1,30] . En effet, de gros obstacles techniques ont été levés depuis l'introduction des premiers algorithmes prédictifs [32], qui imposaient d'avoir une grammaire sous forme normale de Greibach et refaisaient constamment les mêmes sous-calculs. L'algorithme décrit par Earley [19], qui poursuit le travail de Knuth [29] évite ces écueils au moyen d'une analyse préalable de la grammaire et d'un codage adapté. De plus, dans deux mesures de complexité différentes (temps, espace), il coûte automatiquement d'autant moins cher que la grammaire est simple (quelconque, linéaire, LR(k)).

Nous utiliserons un codage particulier qui possède des qualités analogues et permet aussi de coder l'arborescence $\Phi^*(c)$ des calculs d'une grammaire présentée comme un réseau de transitions. La donnée, c, sera constituée par un "graphe de chaînes". Ce codage utilise le fait qu'on borne la taille des arguments (Cf. chapitre I) et qu'on peut démontrer que $\bar{\Phi}^*(c)$ est bornée en fonction de c. On représentera donc l'arborescence des calculs au moyen d'un stack contenant en particulier des pointeurs vers les autres structures de travail (graphe de chaînes et autres stacks).

Ce chapitre consistera donc à définir ce codage, à construire un certain ensemble F de fonctions de déplacement et à voir quels contrôles utiliser en pratique pour que toute heuristique F-constructible contrôlée soit décidable. Comme le système est hiérarchisé, on peut programmer une telle heuristique sous la forme d'un automate d'états finis. On montrera également que cette présentation permet, en choisissant une heuristique particulière, de retrouver les mêmes bornes de temps et de place qu'Earley.

1.2. RESEAUX

On définit ici un réseau comme un multigraphe sans sommet isolé, c'est-à-dire un ensemble de sommets reliés par des arcs orientés. C'est choisir une définition fonctionnelle analogue à celles qu'on peut trouver en théorie des graphes [5] , des automates [37] ou des catégories [41] . Les notations introduites nous serviront dans la suite.

1.2.1. ELEMENT DE RESEAU

Définition 1

Un élément de réseau *(ER) est une fonction* ρ *de* $\mathbb{N}_+$ *dans* $\mathbb{N}_+^2$.

Un ER est fini *ssi* Dom ρ *l'est.*

Notons π_1 *et* π_2 *les projections canoniques de* $\mathbb{N}_+^2$ *sur* $\mathbb{N}_+$. *On définit alors*

$\sigma = \pi_1 \rho$ *la* fonction source

$\tau = \pi_2 \rho$ *la* fonction but

$A(\rho)$ = Dom ρ, *l'*ensemble des arcs

$S(\rho) = \pi_1(\text{Im } \rho) \cup \pi_2(\text{Im } \rho)$, *l'*ensemble des sommets

Si $n \in A(\rho)$, *on dira que l'*arc n relie $\sigma(n)$ à $\tau(n)$. *Inversement, on peut définir des applications de* $\mathbb{N}_+$ *dans* $\mathcal{P}(\mathbb{N}_+)$:

$\underline{\sigma}(x) = \{n | \sigma(n) = x\} = \sigma^{-1}(x)$, *l'*ensemble des arcs issus de x.

$\underline{\tau} = \tau^{-1}$ $\underline{\tau}(x)$ *est l'*ensemble des arcs de but x.

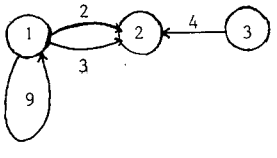
On définit enfin les prédécesseurs *et les* successeurs des sommets :

$\hat{\sigma} = \sigma \underline{\tau}$, $\hat{\tau} = \tau \underline{\sigma}$. $\hat{\tau}(x)$ *est l'*ensemble des sommets reliés à x. *On notera* $\hat{\sigma}^*$ *et* $\hat{\tau}^*$ *les extensions transitives de* $\hat{\sigma}$ *et* $\hat{\tau}$.

Il est évidemment possible de représenter un ER fini par un ensemble de triplets d'entiers de la forme $(n,\rho(n))$, ou par un multigraphe "géométrique" [5].

Exemple

$$\rho = \begin{cases} (2,1,2) \\ (3,1,2) \\ (4,3,2) \\ (9,1,1) \end{cases}$$



§ - Garvin (1971) fait à ce propos une confusion entre la notion d'analyse (ascendante) et la méthode d'analyse (qui peut être ascendante ou descendante). Mais il serait sans doute d'accord avec toute méthode "heuristique" !

1.2.2. NOEUDS PARTICULIERS, CHEMINS

On utilisera indifféremment les termes "sommet" et "noeud". Définissons-en quatre ensembles particuliers.

Définition 2

Les sommets initiaux *(resp.* finals*) sont ceux qui n'ont pas de prédécesseur (resp. de successeur). Les* sommets origines *(resp.* extrémités*) sont ceux qui n'ont pas d'autre prédécesseur (resp. successeur) qu'eux-mêmes.*

Remarquons qu'on appelle souvent ces premiers sommets "source" et "puits". Nous préférons garder le mot "source" pour σ, car c'est son nom en théorie des catégories.

On aura donc les notations :

$I(\rho) = \{x|\hat{\sigma}(x) = \emptyset\}$ $\quad O(\rho) = \{x|\hat{\sigma}(x)\subset\{x\}\}$

$F(\rho) = \{x|\hat{\tau}(x) = \emptyset\}$ $\quad E(\rho) = \{x|\hat{\tau}(x)\subset\{x\}\}$

Définition 3

Un chemin γ *d'un ER* ρ *est une application d'un segment initial de* $\mathbb{N}$ *dans* $A(\rho)$*, c'est-à-dire une suite finie ou infinie d'arcs telle que :* $(\forall i\in \mathrm{Dom}\ \gamma - \{0\})[\tau\gamma(i-1) = \sigma\gamma(i)]$.

On notera $C(\rho)$ *l'*ensemble des chemins de ρ.

Le segment recouvert par γ *est la suite des sommets rencontrés, soit* $\hat{\gamma}(\mathbb{N})$, avec $\begin{cases}\hat{\gamma}(0) = \sigma\gamma(0)\\ \hat{\gamma}(x+1) = \tau\gamma(x)\end{cases}$

*L'*ensemble des chemins reliant deux sommets x et y *sera noté* $\bar{\gamma}(x,y)$

On a donc : $\bar{\gamma}(x,y) = \bigcup_{n\in\mathbb{N}} \{\gamma|\mathrm{Dom}\ \gamma = [0,n] \ \&\ \sigma\gamma(0) = x \ \&\ \tau\gamma(n) = y\}$

Un chemin γ *est* simple *s'il est injectif.*
Un chemin γ *est* élémentaire *si* $\hat{\gamma}$ *est injectif.*
Un circuit *est un chemin* γ *fini d'origine et d'extrémités identiques :* $\sigma\gamma(0) = \tau\gamma(n)$[§].

Une boucle *est un arc de source et de but identiques. C'est encore un circuit à un élément.*
Une chaîne *désignera un chemin simple et élémentaire*[§§].

1.2.3. RESEAUX

En reprenant l'exemple du 1.2.1., on voit bien que la définition "par triplets" doit être définie à l'ordre et à la numérotation près.

On peut parfois définir une fonction f de F^E dans $F'^{E'}$ en utilisant deux applications $f_1 : E \to E'$ et $f_2 : F \to F'$ (mais on ne peut définir tous les f de cette façon). Dans ce cas, on a $(f(\alpha))(f_1(e)) = f_2(\alpha(e))$ pour $e\in E$ et $\alpha\in F^E$.
Ceci nous conduit à la :

Définition 4

Deux ER ρ *et* ρ' *sont* isomorphes *s'il existe un couple* (ξ_1,ξ_2) *de bijections de* $\mathbb{N}$ *sur* $\mathbb{N}$ *tel que :*

(I1) $\mathrm{Dom}\ \rho' = \xi_1(\mathrm{Dom}\ \rho)$

(I2) $\xi_2\ \sigma = \sigma'\ \xi_1$ & $\xi_2\ \tau = \tau'\ \xi_1$

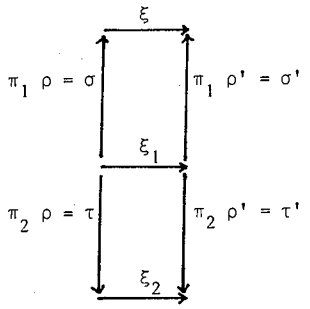
Si de plus ξ_1 *et* ξ_2 *sont récursives, on dira que* ρ *et* ρ' *sont* récursivement isomorphes. *Si* ξ_1 *est* croissante, *l'isomorphie est ordonnée.*

---

§ - Un chemin peut comporter un circuit et être injectif, alors que le segment recouvert par un circuit est nécessairement non injectif.

§§ - Malgré l'usage "non-orienté" qu'on en fait en théorie des graphes, car ce terme a déjà été utilisé dans un contexte analogue (systèmes-Q, [18]).

La propriété (I2) signifie seulement que l'image de la source (du but) est la source (le but) de l'image, ou encore que le diagramme suivant est commutatif :



Il est évident que l'isomorphie et l'isomorphie récursive sont des relations d'équivalence sur l'ensemble des ER.

Définition 5

Un réseau *est une classe d'isomorphie récursive d'éléments de réseau. Un* réseau ordonné *est une classe d'isomorphie récursive ordonnée d'ER.*

Toutes les propriétés et définitions introduites pour les ER se transposent immédiatement aux réseaux. En particulier, les chemins, les circuits, les sommets initiaux, finals... sont conservés par isomorphie. Ceci provient de ce que ξ_1 et ξ_2 sont des bijections. Introduisons enfin la notion d'étiquetage.

Définition 6

Un ER étiqueté sur E et G *(E pour les arcs, G pour les sommets) est un couple* (ρ,ε) *où :*

(1) ρ *est un ER*

(2) $\varepsilon : A(\rho) \oplus S(\rho) \to E \oplus G$ *est une fonction définie par :*

- *l'*étiquetage des arcs $\varepsilon_1 : A(\rho) \to E$
- *l'*étiquetage des sommets $\varepsilon_2 : S(\rho) \to G$

Cette définition permet de considérer une seule fonction d'étiquetage ε pour les arcs et pour les sommets, elle correspond d'ailleurs au codage proposé plus loin.

Définition 7

Deux ER ρ *et* ρ' *étiquetés sont* isomorphes *s'ils sont isomorphes en tant qu'ER et s'il existe un couple d'isomorphie* (ξ_1,ξ_2) *tel que (I3)* $\varepsilon'_1 \xi_1 = \varepsilon_1$ & $\varepsilon'_2 \xi_2 = \varepsilon_2$.

Ceci impose évidemment que les ensembles d'étiquettes soient identiques. D'autre part, tous les couples d'isomorphie vérifient (I3) dès que l'un deux la vérifie. Il est clair que l'isomorphie d'ER étiquetés est une relation d'équivalence.

Définition 8

Un réseau étiqueté sur (E,G) *est une classe d'isomorphie récursive d'ER étiquetés sur (E,G).*
Un réseau étiqueté sur (E,G) ordonné *est une classe d'isomorphie récursive ordonnée d'ER étiquetés sur (E,G).*

1.3. GRAPHES DE CHAINES

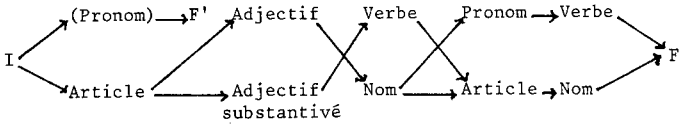
Définition 9

Un graphe de chaînes *est un réseau fini sans circuits comprenant exactement un sommet initial et un ou deux sommet(s) final(s).*

Dans la suite, nous considérerons des graphes de chaînes particuliers, où seuls les sommets sont étiquetés. En fait, on décomposera même l'ensemble d'étiquettes en $G = M \oplus \{I,F,F'\}$, et

- I est l'étiquette du sommet initial
- F et F' sont celles des sommets finals
- M est l'ensemble d'étiquettes réservées aux autres sommets.

En effet, nous utiliserons cette représentation pour décrire le résultat de l'analyse morphologique d'un texte. Donnons d'abord un exemple : "le petit lit le livre".



Ce type de résultat peut être obtenu pas-à-pas à l'aide d'une extension (voir chapitre IV) du système ATEF [10,11], et, bien sûr, de données linguistiques appropriées (variables, modèles, grammaires, dictionnaires).

Les arcs du graphe représentent les liaisons établies en tenant compte des conditions sur l'analyse d'un contexte borné (quatre formes à gauche et une à droite).

Intuitivement, I représente le début du texte, F sa fin et F' une "trappe" qui recueille les chemins non cohérents. Selon les cas, M sera dans la suite appelé ensemble des terminaux ou ensemble des masques. Un masque (de variables) est l'ensemble des valeurs prises par certaines fonctions (les variables) sur une forme donnée (cas = neutre, mode = subjonctif...). C'est donc une entité complexe.

Notons enfin que l'on peut trouver, dans le cas de mots composés ou de tournures, un graphe de chaînes qui n'est pas multipartie (un arc ne correspond pas toujours à la séparation entre deux formes).

1.4. RESEAUX DE TRANSITIONS ET DE RUPTIONS

Les "réseaux de transitions" ont été introduits par Conway (1963) pour présenter plus facilement des grammaires hors-contexte généralisées. Un programme général reçoit comme donnée cette description et fournit un compilateur. Petrick (1965) et Thorne & al (1968) ont développé ce modèle, qui a été généralisé par Woods (1969) sous le nom de "réseaux de transitions augmentés". On trouvera dans Lomet (1974) une formalisation complète des "réseaux de transitions", reliée à une théorie des "systèmes d'automates finis". L'Ecole de Vienne [37] utilise d'ailleurs une présentation analogue.

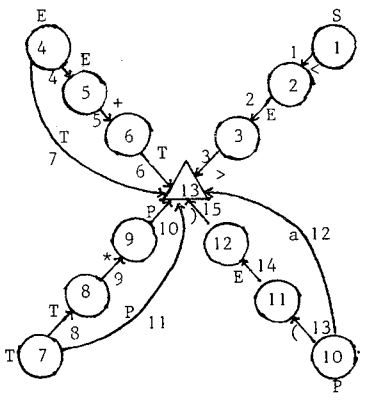
1.4.1. PRESENTATION INTUITIVE

Prenons d'abord un cas simple : il s'agit de représenter une grammaire hors-contexte $\Gamma = (V_N, V_T, S, P)$ par un réseau $R(\Gamma)$. Il suffit de construire un sommet étiqueté par chaque symbole de V_N, puis un sommet Δ correspondant aux "fins de règles", et enfin, pour chaque règle, un ensemble de sommets et d'arcs étiquetés par les symboles de la partie droite.

Exemple :

Γ : $V_N = \{S,E,T,P\}$
$V_T = \{+,(,),<,>,*,a\}$

$S \rightarrow <E>$
$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*P$
$T \rightarrow P$
$P \rightarrow a$
$P \rightarrow (E)$



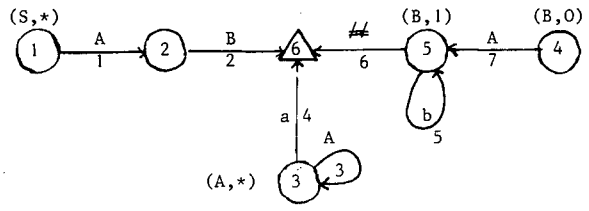
Remarquons que la numérotation des arcs reflète l'ordre des règles. Supposons maintenant qu'on veuille introduire des règles à "contexte gauche". On pourra en rendre compte en utilisant des étiquettes doubles. Enfin, il peut être commode d'accepter des productions "vides" (notation "#") et des expressions régulières (+,*) qui ont une interprétation géométrique immédiate dans la représentation en "réseau".

Exemple :

Γ : $V_N = \{S,A,B\}$
$V_T = \{a,b\}$

$$P \begin{cases} S \to AB \\ A \to Aa + a \\ B \to b^*/A- \end{cases}$$

R(Γ) :



Dans cet exemple, la réécriture d'un "B" se fera à partir de l'état (B,0). Le chemin de (B,0) à (B,1) constitue le contexte : c'est une condition sur la dérivation déjà effectuée ; la réécriture proprement dite s'effectue à partir de (B,1).

Nous remarquons déjà que ce type de représentation permet d'étendre facilement les possibilités des grammaires, en associant aux transitions des tests et des actions (destinées par exemple à construire une structure ou à gérer un certain contrôle), de façon analogue à ce que permet le système de Woods [56] et conforme au voeu de Čulik [17] . Cette représentation définit évidemment un système de transitions (Cf. I), mais, au contraire des modèles précédents, ne suppose pas un critère de définition particulier. C'est dire en particulier qu'une même donnée pourra donner des résultats différents, relativement à un même réseau, si on utilise différentes heuristiques (par exemple pour l'analyse).

Pour revenir brièvement sur des notions introduites au chapitre I, on peut dire qu'un réseau de transitions et de ruptions (RTR) admet en général plusieurs "parcours" ou "analyses" compatibles avec un graphe de chaînes c, qui sont d'ailleurs des feuilles de l'arborescence des calculs $\tilde{\Phi}^*(c)$. Une heuristique est un moyen général de parcours de ces arborescences, et on dira qu'elle engendre une recherche particulière pour tout c (c'est-à-dire une énumération de $\tilde{\rho}^*(c)$)§ . Dans le traitement des langues naturelles, il est (intuitivement) intéressant de pouvoir utiliser des heuristiques, par exemple pour essayer de simuler (ou de définir) un modèle psycholinguistique particulier, ou encore simplement pour éviter une combinatoire exhaustive et souvent explosive : on cherchera à "suivre" le plus vite possible les analyses les plus "vraisemblables".

1.4.2. DEFINITIONS ET PROPRIETES

1.4.2.1. RESEAUX DE TRANSITIONS ET DE RUPTIONS

Donnons la définition formelle d'un RTR avant de l'expliquer plus intuitivement.

Définition 10

Un réseau de transitions et de ruptions *(RTR)* ρ *est un réseau étiqueté sur* (E,G) *possédant les propriétés suivantes :*

(1) $E = ((\mathcal{P}(M) \times N \cup \{\Lambda\}) \cup \{\#\}) \times P$

(2) $G = N \times (\mathbb{N} \cup \{*\}) \cup \{\Delta,\Lambda\}$

(3) $|F(\rho)| = 1 \;\&\; \varepsilon(F(\rho)) = \Delta \;\&\; F(\rho) = E(\rho)$

(4) $\varepsilon|_{S(\rho)} - \varepsilon^{-1}(\Lambda)$ est injective

(5) $x \in O(\rho) \Longrightarrow \varepsilon(x) \in N \times \{*,0\}$

(6) $[x \in S(\rho) \;\&\; \varepsilon(x) \in N \times (\mathbb{N}_+ \cup \{*\})] \Longrightarrow \varepsilon(\hat{\sigma}^*(x) - \{x\}) \subset \{\Lambda,\Delta\}$

(7) $[x \in S(\rho) \;\&\; \varepsilon(x) \in N \times \{0\}] \Longrightarrow [x \in O(\rho) \;\&\; (\exists y \in S(\rho))\, [y \in \hat{\sigma}^*(x) \;\&\; \pi_1\varepsilon(y) = \pi_1\varepsilon(x)]$

(8) $[x \in S(\rho) \;\&\; \varepsilon(x) \in N \times \mathbb{N}_+] \Longrightarrow (\exists y)\; [O(\rho) \cap \hat{\tau}^*(x) = \{y\} \;\&\; \pi_1\varepsilon(y) = \pi_1\varepsilon(x) \;\&\; |\bar{\gamma}(y,x)| = 1]$

Si $\varepsilon(S(\rho)) \subset N \times \{*\} \cup \{\Delta,\Lambda\}$, ρ *est un* réseau de transitions *et les conditions (7) et (8) sont automatiquement vérifiées.*

---

§ - On a tenté au chapitre I de justifier l'emploi de ces termes.

Un RTR a donc des arcs étiquetés soit par "Φ" (transition vide), soit par une condition (récursive) sur M et/ou par un élément de N. P est un ensemble auxiliaire d'étiquettes (1). Les sommets d'un RTR peuvent avoir une étiquette "vide" ("Λ"). Ils comprennent une unique sortie, le sommet final étiqueté par "Δ" (3). Il n'y a pas deux sommets de même étiquette (4), toutes les entrées sont étiquetées (5) et les autres sommets ont des étiquettes du type $(A,n)_{n\geq 1}$, $A\in N$, et sont reliés à une (unique) entrée étiquetée (A,0) par un chemin unique (8). Inversement, toute entrée d'étiquette (A,0) est reliée à au moins un sommet d'étiquette (A,n) (7), et les sommets d'étiquette (A,0) ou (A,*) n'ont pas de successeur strict d'étiquette non vide différente de Δ (6).

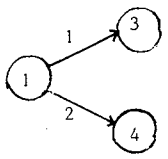
1.4.2.2. RESEAUX ARBORESCENTS

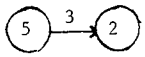
Définition 11

Un réseau arborescent ρ *est un réseau tel que* τ *soit injective.*
Un réseau étiqueté sur (E,G) *sera dit* arborescent *si le réseau non étiqueté sous-jacent est arborescent et si* $E = \emptyset$.

On peut toujours représenter un réseau arborescent fini en utilisant seulement les noeuds, sous forme d'une "forêt" étiquetée par les noeuds :

Exemple :





sera représenté par : 1(3,4),5(2)

Définition 12

Une section *d'un réseau arborescent est une fonction* ψ *de* $\mathbb{N}$ *dans* $S(\rho)$, *telle que :*

(1) $Dom\ \psi$ *est un segment initial de* $\mathbb{N}$
(2) $(\forall x,y\in Dom\ \psi)[x\neq y \Rightarrow \psi(x)\notin \hat{\sigma}^*(y)]$
(3) $\bigcup_{x\in Dom\ \psi} \hat{\sigma}^*(x) \supset F(\rho)$

Comme d'habitude, une section est donc une suite (1) de sommets indépendants (2) et recouvrant l'ensemble des feuilles (les sommets finals d'une arborescence).

1.4.2.3. PARCOURS

Un parcours est l'analogue d'un chemin dans une arborescence de calculs d'un système de transitions.

Définition 13

Un parcours *d'un RTR* ρ *est un réseau arborescent ordonné étiqueté sur* $A(\rho)$ *et défini récursivement de la façon suivante :*

(1) Tout chemin fini $\gamma = \alpha_0,\alpha_1,\ldots,\alpha_n$ *de* ρ *est un parcours de* ρ *de début* $D(\gamma) = \sigma(\alpha_0)$ *et de fin* $G(\gamma) = \tau(\alpha_n)$, *à condition que* $\sigma(\alpha_0)\in I(\rho)$ *et* $\tau(\alpha_n) = F(\rho)$. γ *est un* parcours élémentaire.

(2) Si δ *est un parcours de* ρ, *si* $\alpha\in A(\rho)$ *et si* $\pi_2\pi_1\varepsilon\sigma(\alpha) = \pi_1\varepsilon(D(\delta))$ *(le "nom" de* α *coïncide avec celui du début de* δ), *alors* $\delta' = \alpha(\delta)$ *est un parcours de* ρ, *avec* $D(\delta') = \sigma(\alpha)$ *et* $G(\delta') = \tau(\alpha)$ §.
On dira que δ' *est déduit de* α *et de* δ *par* composition simple.

(3) Si δ_1 *et* δ_2 *sont deux parcours de* ρ *et si* $G(\delta_1) = D(\delta_2)$, *alors* $\delta' = \delta_1,\delta_2$ *est un parcours de* ρ, *avec* $D(\delta') = D(\delta_1)$ *et* $G(\delta') = G(\delta_2)$. *On dira que* δ' *est obtenu par* concaténation *de* δ_1 *et* δ_2.

§ - Ceci correspond à l'analyse d'un sous-groupe "simple".

(4) Soient δ_1 et δ_2 deux parcours de ρ admettant les sections :

$$\psi_1 = \chi_0,\chi_1,\ldots,\chi_p,\ \beta_0,\beta_1,\ldots,\beta_n,\ \alpha,\ \theta_0,\theta_1,\ldots,\theta_q$$

$$\psi_2 = \eta_0,\eta_1,\ldots,\eta_n,\ \mu_0,\mu_1,\ldots,\mu_r$$

avec (1) $(\forall i\in[0,n])[\sigma_{\delta_1}(\beta_i) = \sigma_{\delta_2}(\eta_i)\ \&\ \varepsilon(\beta_i) = \varepsilon(\eta_i)]$ *les "contextes" coïncident*

(2) $\pi_2\pi_1\varepsilon\sigma(\alpha) = \pi_1\varepsilon\sigma(\eta_0) = \pi_1\varepsilon\tau(\eta_n)$

(3) $\pi_2\varepsilon\sigma(\eta_0) = 0\ \&\ \pi_2\varepsilon\tau(\eta_n)\in\mathbb{N}_+$

(4) $\varepsilon\tau(\mu_r) = \Delta$ §

Alors δ', obtenu en remplaçant, dans δ_1, α par $\alpha(\mu_0,\mu_1,\ldots,\mu_r)$ est encore un parcours de ρ, obtenu par composition conditionnelle.

*On notera $P(\rho)$ l'*ensemble des parcours complets *δ de ρ, tels que :*

(1) $\varepsilon(D(\delta))\in\mathbb{N}\times\{*\}$ *le début de δ est "étoilé".*

(2) $\varepsilon(G(\delta)) = \Delta$ *la fin de δ est le sommet final de ρ.*

Exemple

Avec le réseau du 1.4.1., on aura, en notant les étiquettes entre crochets :

δ_1 = 1[A],2[B] ; δ_2 = 4[a] ; δ_3 = 1[A](3[A],4[a]),2[B] ; δ_4 = 7[A],5[b],6[#] ;

δ_5 sera obtenu par composition conditionnelle à partir de δ_3 et δ_4 :

δ_5 = 1[A](3[A],4[a]),2[B](5[b],6[#]).

On voit bien sur cet exemple d'où provient le terme "ruption" : on "saute" dans le parcours du sommet 4[B,0] au sommet 5[B,1], la transition sautée (7[A]) ne servant que comme condition de cohérence. Nous verrons que ceci peut augmenter la commodité, mais pas la puissance d'un RTR.

Définition 14

On appellera analyse *tout préfixe d'un parcours complet au sens de son écriture linéaire. Si ce préfixe est propre, on dira que l'analyse est* partielle.

1.4.2.4. PARCOURS CROISSANTS, REDUITS, MINIMAUX

σ_δ et τ_δ désignent les fonctions source et but de δ, $\leqslant$ l'ordre défini par l'écriture linéaire.

Définition 15

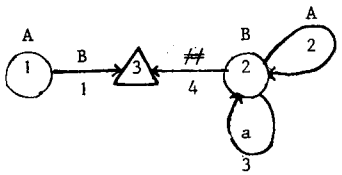
Un parcours δ de ρ est croissant *si :*

(1) $(\forall\alpha\in S(\delta))[\hat\tau_\delta(\alpha)\neq\emptyset \Rightarrow (\exists\beta\in\hat\tau^*_\delta(\alpha))[\pi_2\pi_1\varepsilon_\rho\varepsilon_\delta(\beta) = \Lambda]]$ §§ *: il y a "assez" de transitions terminales.*

(2) $(\forall\alpha_1,\alpha_2\in S(\delta))[[\hat\sigma_\delta(\alpha_1) = \hat\alpha_\delta(\alpha_2)\ \&\ \alpha_1\leqslant\alpha_2\ \&\ \sigma_\rho(\alpha_1) = \tau_\rho(\alpha_2)] \Rightarrow (\exists\beta\in\hat\tau_\delta\hat\sigma_\delta(\alpha_1))[\alpha_1\leqslant\beta\leqslant\alpha_2\ \&\ \pi_1\varepsilon(\beta)\neq\#]]$ *: on ne revient pas sur le même sommet de ρ en n'ayant parcouru que des transitions vides.*

Intuitivement, ceci revient à refuser les "groupes vides" et les "boucles vides".

Exemple :



δ_1 = 1[B](4[#]) n'est pas croissant, mais

δ_2 = 1[B](2[A](1[B](3[a],4[#]))) l'est.

---

§ - On a donc une configuration du type : $0 \xrightarrow[\eta_0]{(A,0)} \cdots \xrightarrow[\eta_n]{(A,k)} \cdot \longrightarrow \cdots \xrightarrow[\mu_r]{} \Delta$

§§ - Rappelons que $\varepsilon S(\delta)\subset A(\rho)$

Définition 16

Un parcours δ *de* ρ *est* **réduit** *si :*

$(\forall \alpha \in S(\delta))(\exists m \in [1, |O(\rho)|])[\tilde{\sigma}_\delta^m(\alpha) = \emptyset$ *ou* $(\exists \beta \in \tau_\delta \sigma_\delta^m(\alpha))[\beta < \tilde{\sigma}_\delta^{m-1}(\alpha)$ & $\pi_2\pi_1\varepsilon(\beta) = \Lambda]]$

Ceci revient à contrôler la récursivité à gauche en exigeant que le parcours ne comporte pas de "chaîne gauche" trop longue de non-terminaux.

Exemple :

Dans le réseau précédent, δ_2 n'est pas réduit, mais :

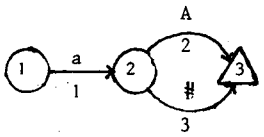
δ_3 = 1[B](3[a],2[A](1[B](3[a],4[#]))) l'est.

Définition 17

Un parcours δ *de* ρ *est* **minimal** *s'il est croissant et réduit. Les parcours complets et croissants, réduits ou minimaux seront notés* $P_C(\rho)$, $P_R(\rho)$, $P_M(\rho)$.

Remarque :

$P(\rho)$, $P_C(\rho)$, $P_R(\rho)$ et $P_M(\rho)$ peuvent être infinis. Il suffit de donner un exemple :



$(\forall n \geq 1)[\text{"2(1,"}^n\text{"3" ")"}^n \in P_M(\rho)]$. Ceci correspond aux "structures" A(a,A(a,...,A(a,#)...))

1.4.2.5. PARCOURS COMPATIBLES AVEC UN GRAPHE DE CHAINES

Définition 18

Soit χ *un graphe de chaînes étiqueté sur* $E = M \cup \{I,F,F'\}$, $G = \emptyset$. *Soit* δ *un RTR étiqueté sur M,N. Si* $\delta \in P(\rho)$, *on notera* $T(\delta) = \delta_0, \delta_1, \ldots, \delta_n$ *la suite des feuilles de* δ. *On dira que* **le parcours** δ **est compatible avec le graphe de chaînes** χ *s'il existe un chemin* γ *de* χ *reliant* $I(\chi)$ *à* $O(\chi)$, *noté* $\gamma = \gamma_0, \ldots, \gamma_{m+1}$ *et une injection* ψ *croissante de* $[0,m+1]$ *dans* $[0,n+1]$ *tels que :*

(1) $(\forall i \in [0,m+1])[i \notin Im\ \psi \Rightarrow \pi_1\varepsilon(\delta_i) = \#]$

(2) $(\forall i \in [0,n+1])[\pi_1^2\varepsilon_\rho\delta_{\psi(i)}(\varepsilon_\chi\tau(\gamma_i)) = I]$

C'est dire que les conditions portées par les transitions non vides de T(δ) sont vérifiées par les étiquettes des sommets non extrêmes d'un chemin de χ.

Notation :

On notera $P(\rho,\chi)$ et $P_M(\rho,\chi)$ l'ensemble des parcours (complets) minimaux de ρ compatibles avec χ.

Théorème 1 :

$P_M(\rho,\chi)$ est fini pour tout couple (ρ,χ)

Divisons les sommets d'un parcours δ en trois classes :

- les sommets α "terminaux", tels que $\pi_2\pi_1\varepsilon(\alpha) = \Lambda$. Ce sont ceux qui sont associés à des sommets de χ.
- les sommets α "non-terminaux", tels que $\pi_1^2\varepsilon(\alpha) \in N$
- les sommets α "vides", c'est-à-dire tels que $\pi_1\varepsilon(\alpha) = \#$.

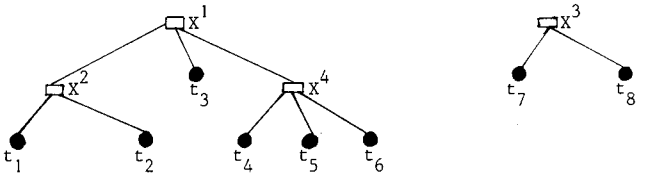
Seuls des sommets terminaux ou vides peuvent être des feuilles de δ. δ induit d'autre part un parenthésage des sommets terminaux qu'on obtient en supprimant les sommets vides et en regroupant les sommets non terminaux à un seul successeur.

Pour démontrer la propriété, il suffit de considérer un graphe χ réduit à une chaîne, puisque tout graphe de chaînes est fini par définition et contient donc un nombre fini de chaînes reliant $I(\chi)$ à $O(\chi)$.

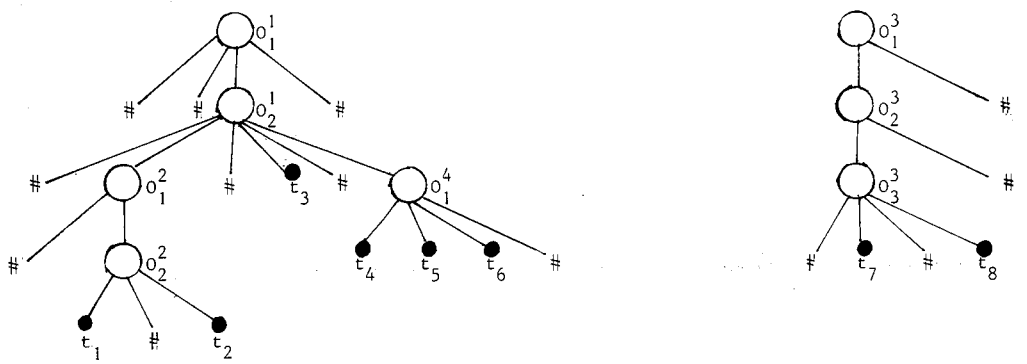
Tout d'abord, le nombre des parenthésages de $\chi' = \gamma_1,\ldots,\gamma_m$ est fini et vaut

$\sum_{\sum n_i=m} C_{n_i}$, avec $C_k = \sum_{0\leq x\leq \frac{k}{2}} (-1)^x \frac{1.3.\ \ldots\ (2k-2x-3)}{x\ !\ (k-2x)\ !}\ 3^{k-2x}\ 2^{-x-2}$ ([13] t.1 p. 70).

Représentons par ▭ les sommets non terminaux d'un tel parenthésage, et par ● les sommets terminaux. On a une structure du type :



Pour trouver tous les parcours δ induisant un parenthésage donné, il suffit d'introduire des sommets vides comme frères des terminaux, et des suites de sommets non terminaux, représentés par ○, à la place des ▭-sommets. D'où par exemple :



A chaque sommet X^i ne peut correspondre qu'une suite finie de ○-sommets, puisque δ est réduit. On note cette suite $O^i_1,\ldots,O^i_k$.

Si X^i avait pour successeur $s^i_1,\ldots,s^i_l$ ($s^i_j = X^{i'}$ ou $s^i_j = t_{i'}$), désignons par $v^i_1,\ldots,v^i_l$ la suite obtenue à partir de $\{s^i_j\}_j$ en remplaçant $X^{i'}$ par $O^{i'}_j$. Alors O^i_k a comme successeurs une suite de sommets dont $\{v^i_j\}_j$ est une sous-suite, et telle que les sommets restants soient vides. La condition (2) de croissance implique que leur nombre est fini dès que $\delta\in P(\rho,\chi)$, sinon il existerait un circuit formé de transitions vides.

D'autre part, $O^i_{k'}$ ($k'<k$) a comme successeurs une suite de sommets dont le seul élément non vide est $O^i_{k'+1}$. En effet, on ne peut ajouter de terminal sans détruire la compatibilité avec χ, ni d'autre non-terminal sans contredire la condition (1) de croissance, puisque cet autre non-terminal ne "recouvrirait" pas de terminal. Le nombre des successeurs vides de $O^i_{k'}$ est encore fini (condition (1) de croissance).⊠

Ainsi, un RTR définit un système de transitions à calculs finis (au sens du chapitre I). Plus précisément, un "calcul" sera isomorphe à un préfixe (de l'écriture linéaire) d'un élément de $P_M(\rho,\chi)$.

Notation :

$Q(\rho,\chi)$ désignera l'ensemble des préfixes de $P(\rho,\chi)$ et

$R(\rho,\chi)$ l'ensemble des préfixes de $P_M(\rho)$ compatibles avec χ (au sens de la compatibilité pour les segments initiaux de χ recouverts).

Corollaire :

$R(\rho,\chi)$ est fini pour tout couple (ρ,χ).

D'après la démonstration précédente, le nombre d'éléments de $R(\rho,\chi)$ qui recouvrent un segment initial donné de χ est fini, sinon on en trouverait un qui ait assez de sommets pour comporter un circuit de transitions vides ou une suite trop longue de O-sommets.

1.4.3. RECHERCHES ET HEURISTIQUES

On vient de voir que $R(\rho,\chi)$ est fini. Intuitivement, $R(\rho,\chi)$ représente tous les débuts d'analyse possibles quand on ne connaît qu'un début de χ ; il possède également une structure arborescente induite par la relation préfixe, pour laquelle on a :

$$F(R(\rho,\chi)) \cap Q(\rho,\chi) = F(Q(\rho,\chi)) = P(\rho,\chi)$$

C'est simplement dire que les feuilles de $R(\rho,\chi)$ sont ou bien des analyses (parcours complets minimaux) de χ selon ρ, ou bien des débuts d'analyses non compatibles.

Nous pouvons appliquer les notions générales étudiées au chapitre I en considérant qu'on a un système de transitions $T=(C,\Phi)$ à trois structures de travail :

- une structure de "graphes de chaînes", avec comme seuls déplacements l'identité et les fonctions "fils" et "frère".
- une structure de "réseau étiqueté" où les déplacements sont encore l'identité et le passage à un sommet connecté (simplement, par appel ou par retour - Cf. I.3.2.).
- une pile de contrôle des appels.

Si T est muni d'une heuristique associée, on sait qu'on peut le réaliser au moyen d'un système de A-machines dont la machine "principale" ("moniteur") possède de plus une structure de travail arborescente. Rappelons qu'une heuristique est un moyen général de décrire une classe d'énumérations (éventuellement partielles) des arborescences des calculs (notées $\Phi^*(c)$ si c est une configuration initiale). Dans le cas présent, $\Phi^*(c)$ correspond exactement à $R(\rho,\chi)$ pour ρ fixé. Ces énumérations constituent les recherches engendrées par l'heuristique considérée et doivent par définition être compatibles avec l'ordre partiel constitué par la relation préfixe de $R(\rho,\chi)$. $R(\rho,\chi)$ est encore ce qu'on appelle l'espace de recherche en Intelligence Artificielle [53].

II - CODAGE ET OPERATIONS ELEMENTAIRES

2.1. CODAGE

Un des fondements de l'algorithmique est que les objets sur lesquels on travaille sont arithmétisables. C'est d'ailleurs pourquoi on peut se ramener à l'étude de fonctions d'entiers. Cependant, le choix d'une bonne "structure de travail" peut permettre de réduire la complexité de description ("taille") et ... d'exposition d'un algorithme, ainsi que sa complexité de calcul, comme on l'a vu à propos de la simulation (Cf. chapitre I). Et il n'est pas étonnant qu'en Intelligence Artificielle, on ait pu dire que : "the question of representation would seem to be the question of heuristics and of strategies at the broadest and more general level" [53, p. 197].

On peut bien sûr coder l'arborescence des parcours (ou des calculs) sur une demi-bande, en utilisant la représentation linéaire canonique des arborescences. Mais ceci conduit à "translater" toute une partie du contenu de la bande pour insérer un nouveau noeud. On préféra imposer une borne à la taille des arguments et coder cette arborescence sur une "bande de travail", ou stack muni de pointeurs. On retrouve alors une structure analogue à celle des automates à stacks avec sauts [31].

D'autre part, on étendra la structure de graphe de chaînes de façon à associer à chaque sommet de χ non pas seulement son étiquette dans M (le masque fourni par l'analyse morphologique), mais aussi une liste des analyses complètes effectuées à partir de ce sommet, ce qui permettra de retrouver "dans les données" les sous-calculs déjà effectués, et donc d'éviter de les refaire.

2.1.1. BANDE DE TRAVAIL

Chaque cellule de la bande de travail correspond à un point de l'arborescence des calculs. Le codage indiquera les valeurs des fonctions de déplacement (ici, "mère", "fille", "soeur") sur cette cellule, et la configuration du système de transitions à laquelle elle correspond (type, position dans ρ et dans χ). On associera de plus à chaque cellule des indicateurs permettant de contrôler les heuristiques qu'on définira. Plus précisément :

Définition 19

Une cellule *de la* bande de travail *est un 11-uplet*
$c = (n,t,r,g,m,f,s,b,x,i,j)$
dont les composantes s'interprètent de la façon suivante :

(1) numéro : $n \in \mathbb{N}$ *correspond à la première apparition de c dans la stratégie particulière considérée.*

(2) type : $t \in \{init, sol, but, rupt, début\}$ *indique si cet élément de calcul est une* initialisation, *une* solution, *une* hypothèse, *une* ruption *ou une* initialisation forcée.

(3) position dans ρ : $r \in A(\rho)$ *si* $t = but$, $r \in S(\rho)$ *sinon. Un "but" correspond donc au début de l'"essai" d'une transition de* ρ.

(4) position dans χ : $g \in S(\chi)$ *est un sommet du graphe de chaînes.*

(5) chaînages propres *(ou "fonctions de déplacement") :*
- $m \in \mathbb{N}$ *est le numéro (n) de la cellule* mère *de c.* $m = 0$ *si* $t = init$ *ou* $t =$ *début (tout nouveau calcul est indépendant).*
- $f \in \mathbb{N}$ *est le numéro (n) de la première cellule* fille *(dans la stratégie considérée).*
- $s \in \mathbb{N}$ *est le numéro (n) de la première cellule* soeur *de c (toujours dans une stratégie particulière).*

(6) chaînage auxiliaire : $b \in \mathbb{N}$ *aura trois interprétations selon le type :*
- *si* $t = but$, *c'est le numéro du* premier "but" *c' égal (c'est-à-dire* $n(c')$, *avec* $r(c') = r(c)$ & $g(c) = g(c')$)
- *si* $t = sol$, *c'est le numéro de l'*élément *de la liste associée à* $gm(c)$ qui a permis la transition.
- *si* $t \in \{init, rupt, début\}$, *c'est le numéro dans la liste associée à* $g(c)$ *du premier élément désignant un calcul de nom* $\varepsilon r(c)$, *c'est-à-dire du* "premier appel".

(7) indicateur de profondeur : $x \in \mathbb{N}$ *n'a de sens que si* $t = sol$, *et désigne alors le nombre de sous-parcours complets (= "sous-calculs") du parcours complet contenant c qui recouvrent le même segment de* χ *que celui-ci.*

(8) indicateur d'activité : $i \in \{+,-,\Theta,\oplus\}$. *C'est un simple sémaphore*
- O *indique que la cellule est* active
- \- *indique qu'elle peut encore avoir des filles*
- \+ *indique le contraire*

(9) indicateur de gel : $j \in \{*,0\}$ *indique si c est "gelée" ou non. Seuls les "buts" peuvent être gelés.*

Enfin, un calcul *est une suite* $c_1,\ldots,c_k$ *de cellules telle que :*

(1) $t(c_1) \in \{init, début, rupt\}$

(2) $m(c_{i+1}) = n(c_i)$ *pour* $1 \leqslant i \leqslant k-1$

(3) $t(c_k) = sol$ & $\varepsilon r(c_k) = \Delta$

Un calcul correspond à un parcours complet. Son nom *est* $\pi_1 \varepsilon r(c_1) \in \mathbb{N}$, *l'étiquette du sommet d'entrée* $r(c_1)$ *de* ρ.
Ses bornes *sont les numéros des cellules où il a commencé et fini, soit* $(n(c_1), n(c_k))$.

2.1.2. GRAPHE DE CHAINES

Notations

On associe à chaque sommet s non extrême du graphe de chaînes χ une liste finie $\bar{l}(s)$

- *de premier élément l'étiquette $\epsilon(s) \epsilon M$ du sommet*
- *dont les éléments suivants sont les calculs (parcours complets) effectués à partir de s, notés sous la forme $l(k,s) = (u',a)$, où :*

(1) $l(k,s)$ désigne le k-ième élément de $\bar{l}(s)$

(2) u' est le nom du calcul : $u' \epsilon N x \{*,0\}$. On posera $u = \pi_1 u'$.

(3) si $l(k,s)$ est le premier élément de $\bar{l}(s)$ de nom u, a est le numéro de la première cellule "but" qui a "appelé" u en ce point s :

$$a = \mu n[t(n) = but \ \& \ \epsilon r(n) = u \ \& \ g(n) = s]$$

(4) Sinon, a est le numéro de la dernière cellule du calcul considéré.

Ainsi, on a une "sur-structure" du graphe de chaîne, la structure élémentaire étant "partagée" par le module d'analyse morphologique. On aura d'ailleurs la possibilité d'analyser χ au fur et à mesure de sa construction.

2.1.3. INTERPRETATION ET EXEMPLE

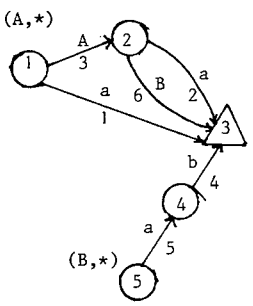
Si $t(c) \epsilon \{$init, rupt,début$\}$, $\pi_1 \epsilon r(c) = A \epsilon N$, g(c) = s et b(c) = k, la cellule c correspond à la "prédiction" du non-terminal A à partir de s. On se trouve sur l'entrée de ρ étiquetée par A. Les filles de c sont des "buts". k est le numéro du premier élement de $\bar{l}(s)$ de nom A.

Si t(c) = but, $r(c) \in A(\rho)$ et g(c) = s, la cellule c correspond à l'essai de la transition (de ρ) r(c) à partir du sommet s de χ. Les filles de c sont des "sol".

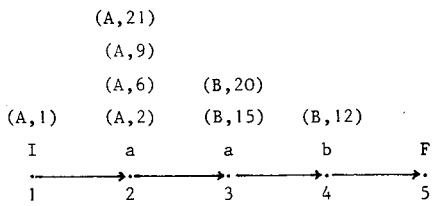
Si t(c) = sol, $r(c) \epsilon S(\rho)$, g(c) = s et b(c) = k, la cellule c correspond à la réalisation de la transition r(c') de la mère c' de c au moyen du k-ième élément de la liste $\bar{l}g(c')$. Si c'est un calcul qui recouvre tout un segment, s en est l'extrémité. Les filles de c sont des "buts". Si $\epsilon r(c) = \Delta$, on est à la fin d'un calcul, et Δ n'a pas de fille.

Donnons maintenant un exemple simple d'analyse, en supposant donnée une heuristique particulière qui s'arrête dès qu'on a une solution. On prendra :

ρ :



χ :



n	t	r	εr	g	εg	m	f	s	b	x	i	j	Commentaires
1	début	1	(A,*)	1	I	0	2						Début de l'analyse : on "appelle" un calcul A, et on introduit 1(2,1)=((A,*),1).
2	but	3	A	2	a	1	22		4			*	On passe sur le (seul) point suivant de X, et on introduit 1(2,2)=((A,*),2)
3	init	1	(A,*)	2	a	0	4		2				On initialise ce calcul et on "gèle" 2 pour éviter de cycler.
4	but	3	A	2	a	3	7	5				*	Il n'y a pas encore de solution A sur le point 2 de X : on gèle 4, et on chaîne à 2 par b (but égal).
5	but	1	a	2	a	3	6						
6	sol	3	Δ	2	a	5			1				On a trouvé un groupe A. On dégèle 2 et 4, et on introduit 1(3,2)=((A,*),6).
7	sol	2	Λ	2	a	4	8	10	3				On réalise la transition demandée par 4, qu'on regèle (pas d'autre solution), ainsi que 2.
8	but	2	a	3	a	7	9	15					
9	sol	3	Δ	3	a	8			1				Une autre solution : on ajoute 1(4,2)=((A,*),9) et on dégèle 2 et 4.
10	sol	2	Λ	3	a	4	11		4				
11	but	2	a	4	b	10		12			+		Ce but n'est pas réalisable, puisque εg=b.
12	but	6	B	4	b	10						*	
13	init	5	(B,*)	4	b	0	14						On initialise un calcul de B à partir du point 4, et on gèle 12.
14	but	5	a	4	b	13					+		But non réalisable. On revient à 7, qui admet une fille.
15	but	6	B	3	a	7	21						
16	init	5	(B,*)	3	a	0	17						On gèle 15, et on introduit 1(2,3)=((B,*),15).
17	but	5	a	3	a	16	18						
18	sol	4	Λ	3	a	17	19		1				
19	but	4	b	4	b	18	20						
20	sol	3	Δ	4	b	19			1				On ajoute 1(3,3)=((B,*),20) et on dégèle 15.
21	sol	3	Δ	4	b	15			3				On utilise le 3ième élément de Ī(3). On ajoute 1(5,2)=((A,*),21). Le successeur de 4⌊b⌋ étant F, on a une solution qu'on met sur 1 : 1(3,1)=((A,*),22).
22													

2.2. OPERATIONS ELEMENTAIRES

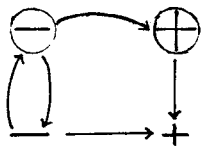
On considère que l'automate est muni d'une tête de lecture-écriture particulière qui pointe à tout moment sur la cellule "active", i.e. celle qui est justement énumérée par la stratégie choisie. Pour la commodité, on pourra introduire deux têtes "auxiliaires" (cf. IV) destinées à garder l'information de calculs précédents. On notera c la cellule active. Les opérations élémentaires sur la bande de travail correspondent aux fonctions de déplacement dans l'arborescence des parcours (minimaux) possibles, et à de nouvelles fonctions de déplacement utiles et simplement exprimables à cause du codage choisi (par exemple, l'ordre sur la bande reflète l'ordre déterminé par la première occurrence dans l'énumération).

Elles se traduiront par trois types d'actions :

(1) Pointer sur une autre cellule (déjà énumérée)

(2) Construire une nouvelle cellule (une fille de c), d'où une modification éventuelle de f ou s sur une autre cellule (c ou une fille de c)

(3) Modifier le chaînage auxiliaire b ou un indicateur (i ou j) ; i peut évoluer suivant le diagramme :



Les paragraphes suivants donnent l'expression formelle de ces fonctions. Elle tient compte de certains contrôles destinés à obtenir la décidabilité de toutes les heuristiques (elles aussi contrôlées) possibles, ce qui est une propriété essentielle du système, et permet la démonstration de cette propriété et de la correction de l'implantation présentée au chapitre IV, qui est directement réalisée à partir de cette formulation.

Ces opérations élémentaires seront notées TROUV(Z,c,C),MOUV(Z,c,C),TROUVC(Z,c,C),MOUVC(Z,c,C),TROUVB(Z,c,C), MOUVB(Z,c,C),SOEUR(c,C),DEBUT(v,s,c),MORT(c) et CONST(c,C).

2.2.1. FONCTIONS SIMPLES

Définissons quelques fonctions utiles pour la construction des opérations élémentaires, avec les conventions suivantes :

a) $\mu k[P(k)]$ signifie "le plus petit k, s'il existe, tel que la proposition P(k) soit vraie, 0 sinon". Cette fonction sera ici toujours récursive, car k aura toujours un domaine borné (nombre de successeurs, d'arcs issus, d'éléments dans une liste associée à un sommet).

b) pour toute fonction Θ, $\Theta(0) = 0$. Ainsi, 0 est un élément singulier, et les fonctions seront définies même sur la "cellule vide".

c) on utilisera le symbole c pour noter l'emplacement de c sur la bande ; n(c) n'est affecté que si c est "construite" (correspond à un point de l'énumération), et vaut alors "c".

(1) $\mathrm{INIT}(c) = m(c)^{\mu k[tm^k n(c) \in \{init, rupt, début\}]}$ — C'est la cellule initiale du calcul contenant c.

(2) $\mathrm{BUTIN}(v,s) = al(\mu k[ul(k,s) = v], s)$ — C'est le numéro du premier but demandant un calcul de nom v en s.

(3) $\mathrm{BUTFI}(v,s) = b^{\mu k[b^k.\mathrm{BUTIN}(v,s) \neq 0 \;\&\; b^{k+1}.\mathrm{BUTIN}(v,s) = 0]}$ — C'est le numéro du dernier but demandant un calcul de nom v en s.

(4) $\mathrm{FILCAD}(c) = s^{\mu k[s^k f(c) \neq 0 \;\&\; s^{k+1} f(c) = 0]}.f(c)$ — C'est le numéro de la "fille cadette" de c.

(5)
$$\mathrm{SEGID}(c) = \begin{cases} 1 \text{ si } t(c) = sol \;\&\; (\exists k)[m^k(c) \neq 0 \;\&\; tm^k(c) = sol \;\&\; SEG(c) \overset{def}{=} \{gm^q(c) \mid q \leq \mu q'[tm^{q'}(c) = init]\} \\ \quad = \{gm^q al(bm^k(c), gm^k(c)) \mid q \leq \mu q'[tm^{q'} al(bm^k(c), gm^k(c)) \in \{init, rupt\}]\} \\ 0 \text{ sinon} \end{cases}$$

SEGID(c) vaut 1 si le calcul contenant c a utilisé un sous-calcul recouvrant exactement le même segment de χ. SEG(c) est le segment recouvert par ce calcul jusqu'à c.

2.2.2. MOUVEMENT SUR LA BANDE

Donnons trois types de déplacement, qui utilisent plus ou moins l'ordre ("temporel") induit par la stratégie et la structure arborescente des calculs.

a) On part d'une origine c et, en allant vers la droite (Z=D) ou vers la gauche (Z=G), on s'arrête à la première cellule qui vérifie une condition C (récursive) donnée. On aura :

$$\text{TROUV}(Z,c,C) = \begin{cases} \mu c'[c'>c \ \&\ (\forall c'')[c<c''<c' \Rightarrow n(c'') \neq 0] \ \&\ C(c')] & \text{si } Z=D \\ \max c'[k \leqslant c' \leqslant c \ \&\ C(c')] & \text{si } Z=G \end{cases}$$

b) Dans le calcul contenant c, on trouve, vers la gauche ou vers la droite, la première cellule vérifiant une condition C. La fonction CHERCH(c) trouve la première cellule suivant c dans l'énumération canonique de l'arborescence (ordonnée par la stratégie choisie) de racine INIT(c), et la fonction CONDIT(c,Z,c',C) vérifie la condition et le sens de parcours :

$$\text{CHERCH}(c) = \mu c'[n(c')>0 \ \&\ f(c) \in \{0,c'\} \ \&\ c' = sm(c)] \qquad \mu k[m^k(c)\neq 0 \ \&\ sm^k(c)\neq 0 \ \&\ tm^k(c)\notin\{\text{init,rupt,début}\}]$$

$$\text{CONDIT}(c,Z,c',C) = \begin{cases} 1 \text{ si } i(c')\neq + \ \&\ j(c')\neq * \ \&\ [(Z=G \ \&\ c'<c) \text{ ou } (Z=D \ \&\ c'>c)] \ \&\ C(c') \\ 0 \text{ sinon} \end{cases}$$

Alors :

$$\text{TROUVC}(Z,c,C) = \text{CHERCH}(c) \qquad \mu k[\text{CONDIT}(c,Z,\text{CHERCH}^k(c),C) = 1]$$

c) On revient au premier but qui a appelé le calcul contenant c :

$$\text{TROUVB}(Z,c,C) = al(b.\text{INIT}(c),g(c))$$

En fait, ces fonctions ne font que trouver une cellule, sans la rendre "active". Nous compléterons ceci au 2.2.4.1.

Enfin, on se donnera la possibilité d'utiliser un certain nombre fixé, de variables à valeurs dans l'ensemble des cellules construites, notées $c1,c2,\ldots,c_n$, et des fonctions d'accès très simples pour généraliser la fonction (TROUV) (et donc MOUV, Cf. 2.2.4.2.). De plus, on peut introduire deux constantes triviales "top" et "down" pour 0 et $\max\{n(c) \mid c\neq\emptyset\}$. On affectera aux ci des expressions représentant des cellules. Pour les fonctions d'accès, on prendra :

$$M(c) = m(c) \qquad F(c) = f(c) \qquad S(c) = s(c) \qquad \text{et} \qquad B(c) = \begin{cases} b(c) & \text{si } t(c) = \text{but} \\ \text{BUTIN}(N(c),g.\text{INIT}(c)) & \text{si } t(c) = \text{sol} \ \&\ \varepsilon r(c) = \Delta \\ a(b(c)) & \text{si } t(c)\in\{\text{init,début,rupt}\} \\ m(c) & \text{si } t(c) = \text{sol} \ \&\ \varepsilon r(c) \neq \Delta \end{cases}$$

d) On définira alors TROUV(Z,c,C) pour $Z\in\{M,F,S,B\}$, ces fonctions portant sur c :

$$\text{TROUV}(Z,c,C) = \begin{cases} Z(c) & \text{si } C(Z(c)) \\ 0 & \text{sinon} \end{cases}$$

2.2.3. CONSTRUCTION D'UNE NOUVELLE CELLULE

Il s'agit soit de construire une fille d'une cellule c (fertile), en utilisant une certaine condition de choix, C, soit de "lancer" un calcul de nom v en un sommet s de χ. D'où les fonctions CONST(c,C) et DEBUT(v,s,c). On définira aussi SOEUR(c,C) = CONST(m(c),C)).

2.2.3.1. CONST : FILLES DES "BUT"

Ce sont des cellules "sol", "init" ou "rupt". Intuitivement, réaliser un but revient à trouver une solution dans la liste $\tilde{I}(s)$ du sommet associé à c, ou à commencer (sur s) le calcul du nom demandé, soit $N(c) = \pi_2\pi_1 \varepsilon r(c)$.

On introduit deux critères de choix :

C_1 : trouver (sur s) le premier calcul de nom N(c) qui n'a pas été utilisé pour produire une fille de c.

C_2 : trouver le premier calcul de nom N(c) non encore utilisé qui recouvre un segment de longueur maximale.

On posera :

$$N(c) = \begin{cases} \pi_2\pi_1 \varepsilon r(c) & \text{si } t(c) = \text{but} \\ \pi_1 \varepsilon r(c) & \text{sinon} \end{cases}$$

$$K1(k,c) = [ul(k,g(c)) = N(c) \;\&\; (\exists k'<k)[ul(k',g(c)) = N(c)] \;\&\; (\forall q)[s^q f(c)\neq 0 \Longrightarrow l(bs^q f(c), gs^q f(c))\neq l(k,g(c))]]$$

(k' correspond à l'élément de liste qui renvoie à l'initialisation du calcul de nom N(c)).

$$K2(k,c) = (\forall q)[ul(k.g(c)) = ul(q,g(c)) \Longrightarrow |SEG\ a\ l(k,g(c))| < |SEG\ a\ l(q,g(c))|] .$$

Alors :

$$CALCUL(c,C1) = \begin{cases} 0 & \text{si } \varepsilon r(c) = \Lambda \;\&\; (\exists q)\,[s^q f(c)\neq 0 \;\&\; l(bs^q f(c), gs^q f(c)) = l(l,g(c))] \\ l(l,g(c)) & \text{si } \varepsilon r(c) = \Lambda \;\&\; \neg\ \text{----------} \\ l(\mu k[K1(k,g(c))], g(c)) & \text{sinon} \end{cases}$$

$$CALCUL(c,C2) = \begin{cases} CALCUL(c,C1) & \text{si } \varepsilon r(c) = \Lambda \\ l(\mu k[K1(k,g(c)) \;\&\; K2(k,g(c))], g(c)) & \text{sinon} \end{cases}$$

DEPART(c,C) = INIT.a.CALCUL(c,C)

Pour la nouvelle cellule CONST(c,C), on aura les composantes :

n = $\mu k[n(k) = 0]$. C'est la première cellule libre.

$$t = \begin{cases} sol & \text{si } [CALCUL(c,C) \neq 0 \;\&\; [t.DEPART(c,C) = rupt \Longrightarrow m.DEPART(c,C) = c \text{ ou } \varepsilon r(c) = \#] \\ init & \text{si } (\forall k)[ul(k,g(c)) \neq N(c)] \;\&\; (\exists e\in S(\rho))[\varepsilon(e) = (N(c),*)] \text{, c'est-à-dire si aucun calcul de ce} \\ & \quad \text{nom n'a été lancé et si le non-terminal considéré est sans ruption.} \\ rupt & \text{si } (\forall k)[ul(k,g(c)) \neq N(c)] \;\&\; (\exists e\in S(\rho))[\varepsilon(e) = (N(c),0)] \end{cases}$$

r = $\sigma_\rho r(c)$: on arrive dans ρ au but de l'arc (transition) associé à c.

$$g = \begin{cases} g(c) & \text{si } CALCUL(c,C) = l(l,g(c)), \text{ car le premier élément ne recouvre que lui-même.} \\ g(c) & \text{si } t\in\{init, rupt\} \text{ ou si } [t = sol \;\&\; \varepsilon r(c) = \#\#] \quad \text{(transition vide).} \\ ga\ CALCUL(c,C) & \text{si } CALCUL(c,C) \neq l(l,g(c)) : \text{on progresse dans } \chi \text{ jusqu'au dernier sommet recouvert par le calcul} \\ & \quad \text{utilisé.} \end{cases}$$

$$m = \begin{cases} n(c) & \text{si } t\in\{sol,rupt\} \\ 0 & \text{si } t = init \end{cases}$$

f = s = 0

$$b = \begin{cases} \mu k[l(k,g(c)) = CALCUL(c,C)] & \text{si } t = sol \\ \mu k[ul(k,g(c)) = N(c)] & \text{si } t\in\{init,rupt\} \end{cases}$$

$$x = \begin{cases} 0 & \text{si } t \{init,rupt\} \\ xm(c) + SEGID.CONST(c,C) & \text{si } t = sol \end{cases}$$

$$i = \begin{cases} \Theta & \text{si } x > |N| \\ \theta & \text{sinon} \end{cases}$$

$j = 0$

Si $i(c) = \Theta$, $CONST(c,C) = 0$: on ne peut poursuivre l'analyse à partir d'une cellule "stérile".

2.2.3.2. CONST : FILLES DES "SOL" ET "INIT"

Ce ne peuvent être que des "but". Intuitivement, on se trouve sur un sommet r du RTR ρ et sur un sommet s du graphe de chaînes χ, avec plusieurs possibilités (le produit des successeurs de s et des arcs issus de r). On définira deux critères de choix :

C1 : choisir la première transition et le premier successeur non essayés.
C2 : choisir le premier successeur et la première transition non essayés.

Notation :

$\bar{\sigma}(k,s)$ désignera le k-ième arc issu du sommet s dans un réseau (donc $\sigma\bar{\sigma}(k,s) = s$), et $\sigma(s)$ l'ensemble des arcs issus de s (cf. 1.2.1.).

Pour la nouvelle cellule CONST(c,C), on aura les composantes :

$n = \mu k[n(k) = 0]$

$t = but$

$$r = \begin{cases} \bar{\sigma}(\mu k[|\tau\bar{\sigma}g(c) - \{gs^q f(c) | s^q f(c) \neq 0 \ \& \ rs^q f(c) = \bar{\sigma}(k,r(c))\}| \neq 0], r(c)) & \text{si } C = C1 \\ \bar{\sigma}(\mu k[(\forall q)[[s^q f(c) \neq 0 \ \& \ g.CONST(c,C) = gs^q f(c)] \Longrightarrow \bar{\sigma}(k,r(c)) \neq rs^q f(c) \\ \quad \& \ gm^q(c) = g.CONST(c,C) \Longrightarrow \bar{\sigma}(k,r(c)) \neq rm^q(c)]], r(c)) & \text{si } C = C2 \end{cases}$$

$$g = \begin{cases} \tau\bar{\sigma}(\mu k[(\forall q)[[s^q f(c) \neq 0 \ \& \ r.CONST(c,C) = rs^q f(c) \Longrightarrow \tau\bar{\sigma}(k,g(c)) \neq gs^q f(c) \\ \quad \& \ [rm^q(c) = r.CONST(c,C) \ \& \ tm^q(c) = but] \Longrightarrow \tau\bar{\sigma}(k,g(c)) \neq gm^q(c)]], g(c)) & \text{si } C = C1 \\ \tau\bar{\sigma}(\mu k[|\tau g r(c) - \{rs^q f(c) | s^q f(c) \neq 0 \ \& \ gs^q f(c) = \tau\bar{\sigma}(k,g(0))\}| \neq 0], g(c)) & \text{si } C = C2 \end{cases}$$

$m = n(c)$
$f = s = b = j = 0$
$x = x(c)$
$i = \Theta$

De plus, il faut "chaîner" la nouvelle cellule au dernier but égal, s'il existe. Sinon, on ajoute un nouvel élément à $\bar{l}g(c)$ (voir plus loin). Remarquons que la définition de g et de r interdit de construire un but égal à l'un de ses ascendants. C'est un des "contrôles" mentionnés plus haut, et destinés à ce que les parcours construits soient toujours minimaux.

2.2.3.3. CONST : FILLES DES "RUPT"

Ce peuvent être des cellules toutes du type "rupt" ou toutes du type "but". Plus précisément, une cellule "rupt" fille d'un "but", a pour filles des "rupt" (correspondant aux sommets de ρ étiquetés par $A \times \mathbb{N}_+$, si $A \in N$ est l'étiquette de ce but), et une cellule "rupt" fille d'une "rupt" a pour fille des "buts". Séparons ces deux cas :

1) Si $t(c) = tm(c) = rupt$, CONST(c,C) est défini comme si $t(c) = init$ (2.2.3.2.)
2) Si $t(c) = rupt$ & $tm(c) = but$, CONST(c,C) aura les composantes :

$n = \mu k[n(k) = 0]$
$t = rupt$

Notation :

r est le premier sommet de ρ d'étiquette dans $Nm(c)\times N_+$ relié à r(c) par un chemin (unique par définition de ρ) $\gamma = \alpha_0,\ldots,\alpha_p$ dont les étiquettes coïncident avec celles des p+1 dernières transitions non vides qui précèdent "horizontalement" c dans le calcul contenant c. On notera $\beta(c) = \beta_0,\ldots,\beta_p$ cette suite de transitions. On a alors :

$r = \mu h[\varepsilon(h)\in Nm(c)\times N_+ \ \&\ (\forall_q \in [0,p])[Nm^{2q+3}(c) = \pi_1\varepsilon(\alpha_1)]$

$\&\ (\forall q)[s^q f(c) \neq 0 \Longrightarrow rs^q f(c) \neq r.CONST(c,C)]]$

g = g(c) : on reste sur le même sommet de χ.

m = n(c)

f = s = b = j = 0

x = x(c)

i = θ

2.2.3.4. DEBUT ET CONST : INITIALISATION ARBITRAIRE ET FILLES DES "DEBUT"

Les cellules "début" correspondent aux "axiomes" (non-terminaux) dont on espère trouver une réalisation dans χ. On s'en servira pour initialiser l'analyse ou pour lancer un nouveau calcul, si par exemple toutes les analyses ont échoué.

DEBUT(v,s,c) aura les composantes :

$n = \mu k[n(k) = 0]$ si $\mu k[u1(k,s) = v] = 0$ (ce calcul n'a pas déjà été commencé).

t = début

$r = \mu\beta[\varepsilon(\beta)\in\{v\}\times\{0,*\}]$ c'est l'entrée de ρ étiquetée par v∈N.

g = s

m = f = s = x = j = 0

i = θ

$b = \mu k[1(k,s) = 0] + 1$ (et on complètera $\tilde{1}(s)$).

Les filles des "début" sont des "but" placés sur les successeurs de s si ε(s) = I, sur s sinon. Pour la nouvelle cellule CONST(c,C), on aura les composantes :

$n = \mu k[n(k) = 0]$

t = but

r et g sont définis
- comme au 2.2.3.2. si ε(s) = I (C étant libre)
- par g=s et r comme au 2.2.3.2. (avec C=C2) sinon.

m = n(c)

f = s = b = x = j = 0

i = θ

2.2.4. MODIFICATIONS DE CELLULES CONSTRUITES ET DE LISTES DU GRAPHE DE CHAINES

2.2.4.1. ASSOCIEES A CONST

Quand on construit une nouvelle cellule c' = CONST(c,C), il faut modifier un chaînage propre de c ou de FILCAD(c) et éventuellement le chaînage auxiliaire de "dernier but égal".

1) Chaînages propres

F(c) := n(c') si f(c) = 0 & t(c') ∉ {init,début,rupt}

S.FILCAD(c) := n(c') si f(c) ≠ 0 & t (c') ∉ {init,début,rupt}

2) Chaînage auxiliaire, gel et modification de χ

Après avoir construit une cellule

- "init" ou "rupt", on gèle les "but" correspondants
- "but", on la chaîne au dernier but égal, s'il existe
- "sol", si on a achevé un calcul (sommet Δ de ρ), on ajoute ce calcul à la liste du sommet "appelant" de χ et on "dégèle" les buts de même nom associés à ce sommet.

Après avoir construit une cellule "début" ou la première cellule "but" de nom v sur un sommet s de χ, on ajoute un nouvel élément à $\tilde{1}(s)$.

Définissons d'abord :

$$GEL(v,s) = \bigwedge_{k\leq\mu q[b^q BUTIN(v,s) = 0]} jb^k BUTIN(v,s) := *$$

et DEGEL(v,s) en remplaçant "*" par "0" dans cette définition.

On notera c' = CONST(c,C) ou c' = DEBUT(v,s,c). Alors :

- Si t(c') = init	On fait GEL(N(c'),g(c')) (N a été défini au 2.2.3.1.)
- Si t(c') = but & BUTIN(N(c'),g(c')) $\neq$ 0	On fait b.BUTFI(N(c'),g(c')) := n(c')
- Si t(c')$\in${début,but} & BUTIN(N(c'),g(c')) = 0	On ajoute à $\tilde{1}$g(c') l'élément
	$1(\mu k[1(k,g(c')) = 0]+1, g(c')) = (u,n(c))$, avec

$$u = \begin{cases} (N(c'),w) \text{ si } t(c') = but \ \&\ (\ s\in O(\rho))[\varepsilon(s) = (N(c'),w)] \\ (v,w) \quad \text{si } c' = DEBUT(v,s,c) \ \&\ (\exists s\in O(\rho))[\varepsilon(s) = (v,w)\ ] \end{cases} \qquad (w\in\{0,*\})$$

- Si c' = 0 & t(c) = but	On "gèle" c : j(c) := *
- Si t(c') = sol & εr(c') = Δ	On fait DEGEL(N(INIT(c')),g(c')) et on ajoute à $\tilde{1}$(g.INIT(c')) l'élément (ε.r.INIT(c'),n(c')). On pourrait introduire ici une condition supplémentaire de "look-ahead" et exiger qu'il existe dans χ un chemin successeur de g(c') de longueur k dont la suite des étiquettes (dans M^k) appartienne à l'ensemble H_k(N.INIT(c')) des segments de longueur k successeurs possibles de N.INIT(c')[18,19,29]. k est alors un paramètre du système.

2.2.4.2. MODIFICATIONS DE L'INDICATEUR D'ACTIVITE

1) Fonctions de déplacement

Les fonctions TROUV, TROUVB, TROUVC (2.2.2.) ne font que calculer le numéro de la cellule cherchée. Pour changer la position de la tête de l'automate, on utilisera les opérations MOUV, MOUVB, MOUVC définies de manière uniforme. Par exemple, pour MOUV(Z,c,C), on aura :

$$MOUV(Z,c,C) = \begin{cases} 0 & \text{si i TROUV(Z,c,C) = +} \\ TROUV(Z,c,C), \text{ avec la différence que i=}\Theta \text{ sinon} \end{cases}$$

$$\text{et, si i.TROUV(Z,c,C)} \neq +,\ i(c) := \begin{cases} - \text{ si } i(c) = \Theta \\ - \text{ si } i(c) = \Theta \end{cases}$$

2) Construction d'une cellule

Si c' = CONST(c,C), SOEUR(c,C) ou DEBUT(v,s,c), on modifie i(c) par :

$$\begin{cases} \Theta \text{ si } c' = 0 \ \&\ t(c) = but \\ - \text{ si } c' \neq 0 \ \&\ 1(c) = \Theta \\ + \text{ si } c' \neq 0 \ \&\ i(c) = \Theta \end{cases} \quad \text{(cas où t(c') = début, ou encore si t(c') = sol \& b(c') = 1).}$$

3) Arrêt d'une analyse

Le codage permet enfin de "tuer" définitivement une analyse partielle en définissant MORT(c) par :

$$i(c) := \begin{cases} \oplus \text{ si } i(c) = \theta \\ + \text{ si } i(c) = - \end{cases}$$

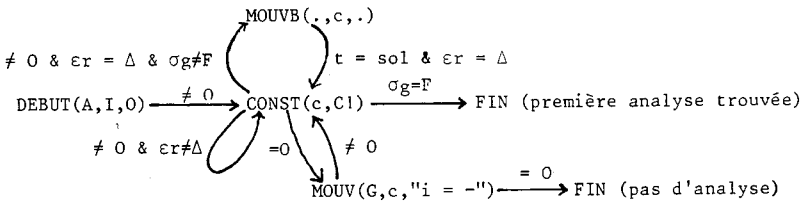
Une heuristique pourra l'utiliser pour bloquer des analyses dont l'estimation (cf. plus loin les "poids") sera trop faible.

2.3. HEURISTIQUES SUR UN RTR

2.3.1. REPRESENTATION

Les opérations élémentaires constitueront l'ensemble F des fonctions à partir desquelles on pourra construire des heuristiques sur un RTR. Elles ont été définies de façon que tout calcul soit un parcours minimal (et que tout sous-calcul soit dans $R(\rho,\chi)$. Comme le système est hiérarchisé, le "moniteur" définissant l'heuristique est un automate d'états finis. On n'utilisera pas la représentation standard, où toutes les actions et tests figureraient sur les arcs, mais plutôt une représentation en réseau étiqueté : les sommets porteront les "actions élémentaires" et les arcs des "tests" (correspondant à un test de caractère sur une structure de travail). C'est d'ailleurs une représentation naturelle pour le contrôle fini de toute A-machine. On n'admettra qu'un sommet initial, portant en général une opération DEBUT §.

Exemple : (cf. 2.1.3)



2.3.2. DECIDABILITE

On a vu que $R(\rho,\chi)$ était fini. La définition des opérations élémentaires assure de n'associer qu'un nombre fini de cellules à chaque élément de $R(\rho,\chi)$ et contient des contrôles destinés à limiter le nombre de passages sur certaines cellules. Il faudra encore ajouter un contrôle ("de production", cf. I) pour s'assurer que toute heuristique (F-constructible) est décidable.

2.3.2.1. NOMBRE DE CELLULES

Pour un couple (ρ,χ) donné, le nombre de cellules "init", "début" et "rupt" qu'on peut construire est évidemment borné par $3|S(\chi)|.|N|.|S(\rho)|$. Le nombre de "but" filles d'une cellule c "init", "début", "rupt" ou "sol" est également fini et vaut $|\sigma r(c)|.|\tau\sigma g(c)|$.

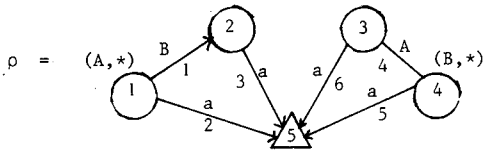
Il reste à voir que le nombre des "sol" est fini. Ceci est obtenu dans la construction en exprimant la minimalité (1.4.2.4.) au moyen des fonctions x, g et r. Intuitivement, on oblige les calculs à "avancer" suffisamment en interdisant de poursuivre une construction qui a déjà utilisé $|N|$ sous-constructions pour recouvrir le même segment (2.2.3.1.). D'autre part, on limite la "profondeur" des calculs en interdisant de construire un but égal à l'un de ses ascendants (2.2.3.2.).

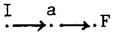
---

§ - Pour des raisons pratiques, on généralisera cette représentation au chapitre IV, de façon à admettre des actions ou listes d'actions conditionnelles sur les arcs et les sommets.

Exemples :

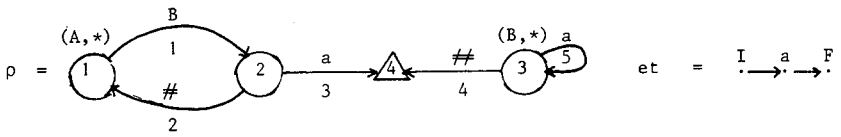
(1) La première restriction interdit de construire les structures du type $A[BA]^n(a)$ et $A[BA]^nB(a)$ dans le cas où :



et χ = 

On pourra seulement construire A(a) et A(B(a)).

(2) La seconde restriction interdit les "boucles de transitions vides", comme on pourrait en avoir avec :



en utilisant les transitions 1[B], 4[#] et 2[#], soit par exemple :

A : 1[B](4[#]),2[#],1[B](4[#]),...,2[#],1[B](4[#]),3[a]

La première restriction revient à exiger que les calculs partiels (parcours) soient réduits, la seconde qu'ils soient croissants. Et l'ensemble des débuts de calculs minimaux est justement $R(\rho,\chi)$, qui est fini. D'où :

Proposition 1

Pour tout couple (ρ,χ), le nombre des calculs qu'on peut engendrer en utilisant les opérations élémentaires est fini.

2.3.2.2. NOMBRE DE PASSAGES

1 - Gel

La définition des opérations élémentaires fait intervenir l'"indicateur de gel" j de façon à éviter de "tourner en rond" en oscillant entre deux ou plusieurs cellules "but" si le réseau est récursif à gauche. Avec l'exemple (1) précédent, on pourrait lancer le calcul de A, puis celui de B, et, si on choisit toujours la première transition (critère C1), chercher à chaque passage si $\hat{I}(1)$ contient une solution pour A.

Pour cette raison, on "gèle" les "but" dès qu'on commence à les réaliser (en créant un "init" ou un "rupt"). On ne les réactive (dégel) que si on augmente la liste du sommet de χ considéré. Ainsi, le nombre de passages sur un "but" est au plus égal à la longueur de la liste du sommet associé, qui est finie.

D'autre part, on rend "stériles" les cellules d'un autre type dès qu'on a construit tous leurs descendants possibles, qui sont en nombre fini.

2 - Contrôle sur les heuristiques

Le contrôle obtenu dans la définition des opérations élémentaires n'est pas encore suffisant : il serait possible de construire des heuristiques qui cyclent sur les points de l'arborescence des calculs déjà énumérés, en utilisant par exemple MOUV, MOUVB ou MOUVC.
On sait (I, Prop. 16) qu'il suffit d'utiliser un contrôle de production $\mathcal{C}$ pour que les recherches $\mathcal{C}$-contrôlées engendrées par une heuristique $\mathcal{H}$ soient décidables, au sens où on sait décider quand elles ne peuvent plus énumérer de nouveau point.

On peut définir ce contrôle en notant dans une liste de contrôle les cellules parcourues depuis la dernière construction de cellule, et d'arrêter la recherche dès que la nouvelle cellule active figure dans la liste et qu'on est revenu au même état de l'heuristique.

2.3.2.3. DECIDABILITE

Définition 19

Une heuristique R-contrôlée *est une heuristique de RTR à contrôle de production.*

Proposition 2

Toute opération élémentaire est décidable.

En effet, le domaine de l'opérateur μ et des quantificateurs qui apparaissent dans les définitions du 2.2. est toujours fini (contenu dans l'ensemble des cellules construites ou des éléments d'une liste $\bar{1}(s)$). ⊠

D'où finalement le résultat annoncé :

Théorème 2

Toute heuristique R-contrôlée est décidable.

On peut remarquer que la composante x des cellules n'est pas utile si ρ n'est pas récursif à gauche et ne contient pas de chemin constitué de transitions vides et reliant une origine au sommet final Δ (les boucles vides à l'intérieur du réseau sont évitées grâce à la définition de g et r). Il y a des techniques bien connues [33,55] pour ramener un réseau de transitions à une forme standard qui évite ces cas. Cependant, il est intéressant dans la pratique de conserver la liberté de représentation (pour un linguiste par exemple), et, de plus, ces techniques deviennent très difficiles à mettre en oeuvre dès qu'on augmente la complexité du modèle en associant aux transitions des conditions et des actions sur des "registres" auxiliaires (cf. infra.).

Enfin, on peut caractériser simplement les heuristiques exhaustives, telle que toute stratégie engendrée énumère tout R(ρ,χ).

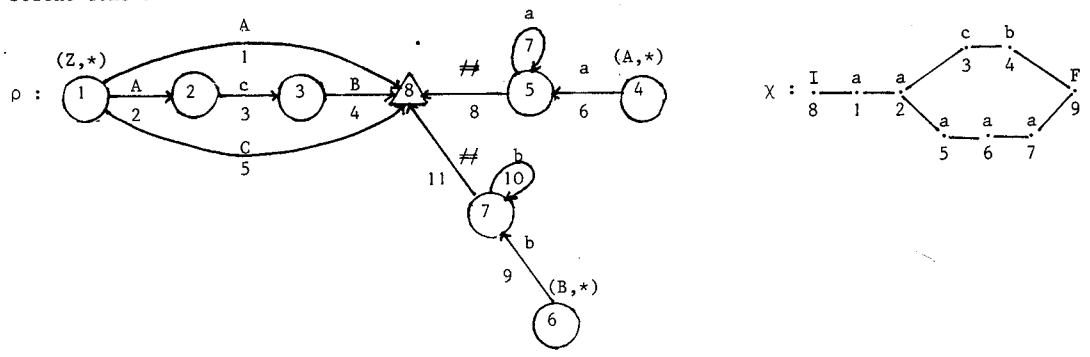
Proposition 3

Une heuristique est exhaustive dès qu'elle n'utilise pas la fonction MORT et ne s'arrête pas tant qu'il reste une cellule fertile non gelée.

2.3.3. UN EXEMPLE

Dans l'exemple qui suit, on va utiliser la stratégie du 2.3.1. avec les deux critères de choix C1 et C2 (2.2.3.2.), et pour tout état initial DEBUT(Z,O,I). Le choix entre C1 et C2 se reflète dans le diagramme qui suit (cases "doubles" coupées par une diagonale).

Soient donc :



n	t	r	N	εg	g	n	f	s	b	x	i	j	Commentaires
1	début	1	(Z,*)	8	I	0	2				-		On ajoute 1(2,8) = ((Z,*),1) à $\bar{1}$(8). Cas C = C1
2	but	1	A	1	a	1	18/11	0/12	0/12		-		On ajoute 1(2,1) = ((A,*),2) à $\bar{1}$(1).
3	init	4	(A,*)	1	a	0	4		2		-		Gel de 2.
4	but	6	a	1	a	3	5				+		
5	sol	5	Λ	1	a	4	6		1		-		i(4):= + car on a construit la seule fille possible.
6	but	7	a	2	a	5	7				+		
7	sol	5	Λ	2	a	6	8				-		
8	but	7	a	3	c	7		9			+		Pas de solution : on revient en 7 (première où i=-).
9	but	7	a	5	a	7	10				+		On poursuit en essayant encore cette transition avec un autre successeur dans χ (critère C1).
10	sol	5	Λ	5	a	9	11		1				
11	but	7	a	6	a	10	12				+		
12	sol	5	Λ	6	a	11	13		1				
13	but	7	a	7	a	12	14				+		
14	sol	5	Λ	7	a	13	15		1				
15	but	7	a	9	F	14		16					Arrivée à la fin de χ.
16	but	8	#	7	a	14	17						
17	sol	8	Δ	7	a	16			0				On ajoute 1(3,1) = ((A,*),17) à $\bar{1}$(1), on dégèle 2, et on revient à 2 en utilisant MOUVB. On ajoute 1(3,8) = ((Z,*),18) à 1(8) et on s'arrête. La première solution trouvée est donc Z(A(a,a,a,a,a)).
18	sol	8	Δ	7	a	2			3				
9	but	8	#	2	a	7							Cas C = C2
10	sol	8	Δ	2	a				0				On ajoute 1(3,1) = ((A,*),10) à $\bar{1}$(1) et on dégèle 2.
11	sol	8	Δ	2	a	2			3				On ne construit pas de Z, car on n'est pas à la fin de χ.
12	but	2	A	1	a	3	13				-		Chaîné à 2 par b (buts de même nom au même point).
13	sol	2	Λ	2	a	12	14		3				On utilise (3) la construction de A déjà trouvée.
14	but	3	c	3	c	13	15				+		
15	sol	3	Λ	3	c	14	16		1				
16	but	4	B	4	b	15	23						On ajoute 1(4,2) = ((B,*),16) à $\bar{1}$(4).
17	init	6	(B,*)	4	b		18						Gel de 16.
18	but	9	b	4	b	17	19				+		
19	sol	7	Λ	4	b	18	20						(22) On ajoute 1(3,4) = ((B,*),22) à $\bar{1}$(4) et on dégèle 16.
20	but	10	b	9	F	19		21			+		(23) On ajoute 1(4,1) = ((A,*),23) à $\bar{1}$(1), on dégèle 2, et on revient à 2 via MOUVB. On ajoute 1(3,8) = ((Z,*),24) à $\bar{1}$(8) et on s'arrête. Avec le critère C2, la solution trouvée est donc Z(A(a,a),c,B(b)).
21	but	11	#	4	b	19	22						
22	sol	8	Δ	4	b	21							
23	sol	8	Δ	4	b	16			3				
24	sol	8	Δ	4	b	2			4				

Comme on peut s'y attendre, une même heuristique peut donner des résultats différents avec différents critères de choix. Bien sûr, toutes les heuristiques exhaustives donnent les mêmes résultats, à savoir l'ensemble des analyses totale

D'autre part, on voit comment le codage utilisé permet de ne pas refaire les calculs déjà faits, tout en gardant une grande liberté de parcours de l'espace de recherche. On verra plus loin comment relier ce codage avec celui d'Earley, qui utilise une heuristique exhaustive particulière.

qui utilise une heuristique exhaustive particulière.

III - EXTENSION : ALGOGRAMMAIRES

La définition des RTR contient déjà une généralisation par rapport à la notion de grammaire hors-contexte, qui revient à admettre des expressions régulières[§]. Si l'on veut associer à chaque analyse un résultat qui ne soit pas nécessairement la structure du calcul effectué, on peut ajouter des "actions" aux arcs, ce qui correspond au passage général d'un algorithme d'analyse à un algorithme de transduction. Pour augmenter la puissance et la commodité du modèle, on peut également munir les arcs de conditions sur le contenu de certains "registres", comme dans les réseaux de transitions augmentés [56]. Cette idée n'est d'ailleurs pas nouvelle, puisque Kuno et Oettinger (1963) l'utilisaient dans leur analyseur prédictif, et que les grammaires employées effectivement [9,10,54] pour le traitement des langues naturelles contiennent souvent des conditions sur des "variables" associées aux non-terminaux.

D'autres extensions sont désirables. Joshi, Kosaraju et Yamada (1972) ont par exemple introduit les "grammaires d'adjonction" qui permettent en particulier de regrouper dans une même classe des langages "naturellement" voisins comme $\{a^n\}$, $\{a^n b^n\}$ et $\{a^n b^n c^n\}$ qui sont dans trois classes de Chomsky distinctes. Čulik (1973) préférerait munir les grammaires hors-contexte de "conditions de comptage" et de "conditions de contexte", de façon à garder une description intuitivement claire[§§] tout en augmentant la classe des langages engendrés. Enfin, on peut introduire au niveau de la grammaire un contrôle sur les algorithmes d'analyse en utilisant des probabilités ou des poids (Salomaa 1969 a), ou en demandant que le langage des productions soit dans une certaine classe [21,25].

Les extensions qu'on propose pour les RTR sont obtenues en introduisant des registres "numériques" ou "linguistiques" et en associant un poids à chaque arc du RTR. Les registres seront de nouvelles composantes des cellules de la bande de travail, et une structure de travail supplémentaire (stack) servira à garder les résultats (arborescences) partiels sous forme résolue ou symbolique. Les arcs et certains sommets du RTR pourront alors être étiquetés par des conditions et des actions sur ces registres. Un RTR ainsi "augmenté" sera appelé "algogrammaire".

3.1. MODIFICATIONS DU CODAGE

3.1.1. STACK

Définition 20

Chaque cellule du stack est un quadruplet $c = (n, \mu, f, s)$, *où :*

- n, f, s *ont la même signification que dans la bande de travail*
- $\mu \in M \in (\mathcal{O} \times \mathbb{N}_+)$, *le* contenu, *est un masque ou le couple d'un* opérateur d'arborescences *et d'un numéro de cellule du stack.*

Si $\mu \in M$, *c est dite* résolue, *sinon* symbolique.

Toute cellule c définit une arborescence $\mathcal{A}(c)$ de racine c, via f et s. Les cellules non résolues de $\mathcal{A}(c)$ font elles-mêmes référence à d'autres cellules, et leur opérateur indiquera comment placer les arborescences associées dans l'arborescence principale. Dans tous les cas, ce processus terminera, et toute cellule c définit finalement une arborescence $\mathfrak{M}(c)$ étiquetée sur M, dite arborescence associée à c.

Ceci correspond à une présentation "dépendantielle", où les sommets portent des étiquettes homogènes. Mais il suffit que M soit un ensemble complexe pour qu'on puisse naturellement associer plusieurs interprétations à $\mathfrak{M}(c)$ (constituants, dépendances ou autres)[§§§].

3.1.2. REGISTRES

On enrichit la structure de la bande de travail et on la relie à celle du stack de la façon suivante :

Définition 21

Chaque cellule de la bande de travail s'écrira :

$c = (n, t, r, g, m, s, f, b, x, i, j, \nu_1, \ldots, \nu_k, \lambda_1, \ldots, \lambda_l)$ *où k et l sont fixés et*

$\lambda_1, \ldots, \lambda_k \in \mathbb{N}$ *sont les* registres numériques

$\lambda_1, \ldots, \lambda_l \in \mathbb{N}$ *sont les* registres structuraux *("linguistiques"), qui s'interprètent comme des pointeurs vers le stack.*

§ - Woods (1969) les appelle "grammaires d'expressions régulières".

§§ - Par exemple, les grammaires CS posent certains problèmes inattendus quand on les utilise "en analyse seulement" (Peters & Ritchie 1969) : elles définissent des langages strictement CF !

§§§ - C'est par exemple le cas quand M est un ensemble de masques de variables, ces variables étant associées à différents niveaux d'analyse linguistique (renseignements morphologiques, classe syntaxique, fonction syntaxique, fonction logique).

Ces registres ne sont pas modifiables lors d'un retour sur la cellule. Le fonctionnement de l'automate impose d'avoir une même initialisation de registres pour tous les appels d'un sous-calcul. C'est seulement lors d'un retour (cellule "sol") qu'on combinera les registres d'appel à ceux du résultat. C'est pourquoi chaque entrée de ρ étiquetée sur $Nx\{*,0\}$ pourra comporter des valeurs initiales pour les ν_i et ν_j.

On pourra aussi associer des initialisations aux sommets de ρ étiquetés sur $N \times N_+$, éventuellement fonctions des registres d'appel, puisque les sous-calcul "à contexte" sont séparés pour chaque appel. Par contre, on ne peut en général permettre d'initialiser en fonction de l'appel ou de transmettre des valeurs "vers le bas" (fonction SENDR de Woods [56]), si on ne veut pas répéter les sous-calculs"§.

3.2. CONDITIONS ET ACTIONS

3.2.1. CONDITIONS

Supposons qu'on progresse dans une analyse en passant d'un sommet S1 de ρ à un sommet S2 au moyen de l'arc T1.

Notation

On notera P l'analyse "précédente" (sur S1), T l'analyse "transitoire", (T1) et C l'analyse "courante" obtenue en S2.

T est "ce qui cause la transition", c'est-à-dire un élément de l(s), et s'interprète donc soit comme une cellule de la bande, soit comme un sommet de χ muni d'un masque. On pourra dans ce cas convenir que tous les registres linguistiques de T contiennent cet unique masque, et que les registres numériques contiennent une constante donnée, ou bien, dans un développement futur, un poids calculé par l'analyse morphologique.

Définition 22

Le "masque" d'un registre structural *est le masque de la racine de l'arborescence qu'il contient.*

Les conditions structurales *sont des prédicats formés au moyen des fonctions booléennes usuelles et portant sur les masques des registres structuraux de P et de T.*

Les conditions numériques *sont prises dans un ensemble $\mathcal{P}$ de prédicats et portent sur les registres numériques et les valeurs booléennes de conditions structurales de P et T.*

Ainsi, les conditions numériques peuvent servir de "conditions de comptage" ou de "contrôle", pour limiter par exemple la profondeur de certaines constructions (principe d'Yngve), et elles peuvent aussi "moduler" des conditions "grammaticales".

Remarque

Dans le cas où χ est le résultat d'une analyse morphologique, chaque sommet est associé à un élément du texte d'entrée ("forme"), ce qui définit une relation d'équivalence sur $S(\chi)$, l'homographie. On peut alors envisager d'introduire des conditions linguistiques portant sur la classe d'homographie de s, si $T \in l(s)$.

Ces conditions permettent ou interdisent la construction d'une nouvelle cellule, correspondant à C, à partir du "but" P et de T.

---

§ - A ce propos, on peut noter que Woods a montré [55] comment obtenir un algorithme efficace pour les réseaux de transitions en généralisant l'algorithme d'Earley, mais qu'il semble [56] être revenu au back-tracking classique pour les réseaux de transitions augmentés, pour pouvoir autoriser des fonctions très générales. Nous cherchons donc à "localiser" ces fonctions, tout en obtenant un gain de puissance et en conservant le bénéfice du codage adopté.

3.2.2. ACTIONS

Si les conditions sont vérifiées, on construit la nouvelle cellule et on affecte ses registres en exécutant les actions portées par la transition de ρ utilisée.

Définition 23

Une action structurale *consiste :*

- *soit à modifier le masque d'un registre structural de C en fonction des masques des registres structuraux de P, T et C lui-même.*
- *soit à affecter à un registre structural de C une arborescence symbolique dont les sommets sont des registres structuraux de P, T ou C.*

Une action numérique *consiste à affecter à un registre numérique de C la valeur d'une fonction prise dans un ensemble* $\mathcal{F}$ *et portant sur les registres numériques et les valeurs booléennes de conditions structurales de P, T et C.*

L'arborescence symbolique construite sera décrite au moyen de O. La puissance supplémentaire obtenue par ces ajouts dépend évidemment de O, P et F.

3.2.3. OPERATEURS D'ARBORESCENCES, PREDICATS ET FONCTIONS NUMERIQUES

1°) $\mathcal{O}$

Une action structurale consistera par exemple à effectuer $\lambda_1(C) := \lambda_2(P)[.\lambda_3(T).,.\lambda_4(C) : [\lambda_1(C)]]$.
Il s'agit donc de construire une arborescence à partir de "morceaux" que les opérateurs indiquent ou placer, et non d'effectuer des transformations générales. On peut proposer les opérateurs suivants :

- "X" pour le placement en dernier fils de la racine.
- ".X" " " en premier fils de la racine.
- ".X." " " sous la feuille la plus à gauche.
- "X." " " " " " droite.
- ":X" " " en frère gauche de la feuille la plus à gauche.
- "X:" " " en frère droit de la feuille la plus à droite.

(avec X = '[' ou ',').

Dans cet exemple, si $\lambda_1(C) = a(b,c)$, $\lambda_2(P) = c(d(e),f)$, $\lambda_3(T) = g(h)$ et $\lambda_4(C) = i(j,k(l))$, on aurait comme résultat $\lambda_1(C) = c(d(e(i(a(b,c),j,k(l)))),f(g(h)))$. Par contre, $\lambda_1(C) := \lambda_2(P)[\lambda_3(T) :,\lambda_4(C).[\lambda_1(C)]]$ donnerait

$\lambda_1(C) = c(d(i(a(b,c),j,k(l)),e),f,g(h))$.

Pour cette dernière expression, on calcule de la façon suivante :

$\lambda_2[\lambda_3:,\lambda_4.[\lambda_1]] = c(d(e(\lambda_4.[\lambda_1])),f(\lambda_3))$, puis $\lambda_4:[\lambda_1] = i(\lambda_1,j,k(1))$.

2°) $\mathcal{P}$ et $\mathcal{F}$

Il est naturel de construire ces prédicats et fonctions à partir de prédicats et de fonctions de base au moyen d'opérations de clôture. Deux résultats de R. Robinson montrent qu'on peut obtenir très simplement deux sous-classes très intéressantes en n'utilisant que des fonctions à un nombre fixé d'arguments.

Théorème 3 (Robinson)

(1) La classe des fonctions récursives primitives RP^1 peut s'obtenir :

- *à partir des fonctions de base $\lambda x[x+1]$ et $\lambda x[x-\lfloor\sqrt{x}\rfloor^2]$*
- *au moyen des opérations d'addition, de composition et d'itération*

(2) La classe des fonctions récursives R^1 peut s'obtenir :

- *à partir des fonctions de base $\lambda x[x+1]$ et $\lambda x[x-\lfloor\sqrt{x}\rfloor^2]$*
- *au moyen des opérations d'addition, de composition et d'inversion.*

Rappelons que - l'itération J est définie par : $f = Jg \iff \begin{cases} f(0) = 0 \\ f(n+1) = g(f(n)) \end{cases}$

- l'inversion I est définie par : $f = Ig \iff f(x) = \mu y[g(y) = x]$, et n'est utilisée (2) que sur des fonctions surjectives[§].

On peut aussi limiter $\mathcal{P}$ et $\mathcal{F}$ aux prédicats et fonctions élémentaires au sens de Kalmár[§§], car toutes les classes de langages usuelles semblent l'être. On en verra plus loin une illustration à propos des langages d'adjonction distributive.

---

§ - Pour (1), voir [34] p. 77, pour (2), [45], théorème III ou [34] p. 117. On a des théorèmes analogues pour RP^n et R^n, où I et J sont alors définies sur les dernières composantes. D'autre part, on peut prendre d'autres fonctions de base, par exemple les deux projections de Cantor.

§§ - Obtenues à partir de $\{\pi_1, +, -, \mathcal{Q}_m^n$ (choix d'arguments)$\}$ par substitution, somme et produit. En particulier, $\lambda x[2^x]$, $\lambda x[2^{2^x}]$... sont élémentaires. C'est aussi la classe $\mathcal{E}^3$ de Grzegorczyk. Plus généralement, on peut l'obtenir à partir de $\{S, +, -, ., \exp\}$ par substitution et par l'une des opérations élémentaires (somme, produit, maximum, minimum et récursion limités, valeur maximale et minimale bornée)[24].

IV- PUISSANCE, COMPLEXITE, ADEQUATION ET ADAPTATION

C'est l'utilisation de registres numériques et de classes $\mathcal{P}, \mathcal{F}$ suffisamment grandes qui permettent de sortir du hors-contexte généralisé.

4.1. PUISSANCE SANS REGISTRES

Il est immédiat d'associer un RTR à une grammaire hors-contexte ou à une grammaire d'expressions régulières. Ces deux classes de grammaires sont d'ailleurs (faiblement) équivalentes, puisqu'on peut toujours trouver un automate à pile (non déterministe) de reconnaissance. Cependant, une expression régulière permet de caractériser des structures non bornées "horizontalement". Remarquons que les RTR associés sont "sans contexte" (pas de sommet étiqueté sur $N \times N_+$).

D'autre part, on a vu qu'on peut utiliser des "contextes gauches" et on pourrait penser pouvoir atteindre la classe CS des langages sous-contexte, puisque Pentonnen (1974) a montré que tout langage sous-contexte était sous-contexte gauche, concluant ainsi un problème longtemps ouvert.

Rappelons un exemple simple de langage sous-contexte gauche non hors-contexte [47, p. 102] :

$G = (V_N, V_T, X_0, P) \qquad V_T = \{a,b,c\} \qquad V_N = \{X_0, B, B_1, C, C_1\}$

$$P = \begin{cases} X_0 \to aaBBC & C_1 \to C/B_1 - \\ B \to aBB_1/a - & B_1 \to B/B - \\ C \to C_1C/B_1B - & B \to b \\ B \to B_1/B_1 - & C \to c \end{cases}$$

$L(G) = \{a^n b^n c^m \mid n \geq m \geq 1\} \in CF$

On peut lui associer simplement un RTR "à contexte". Mais son utilisation "en analyse seulement", et non pas en génération, augmente le langage reconnu. Ici, on reconnaîtrait $\{a^n b^n cc^*\}$. Comme l'ont montré Peters et Ritchie [39, th. 3.8.], "en analyse", ces conditions portent sur la structure et non sur l'histoire de la dérivation, et sont strictement plus faibles puisqu'en aucun cas on ne dépasse la classe CF.

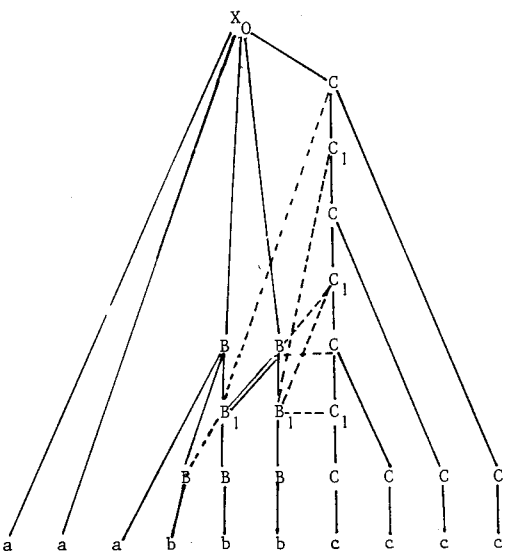
Proposition 4

Un RTR non augmenté ne permet pas de reconnaître plus que les langages hors-contexte.

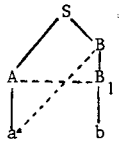
Il suffit de réduire χ à une chaîne et d'utiliser les remarques précédentes ⊠.

Exemples

Avec la grammaire précédente, la structure suivante convient "en analyse" à $a^3b^3c^4$:



- Si G est donnée par : $P \begin{cases} S \to AB \\ A \to a \\ B \to B_1/a - \\ B_1 \to b/A - \end{cases}$



Il est clair que L(G) = ∅ et qu'on reconnaît ab par :

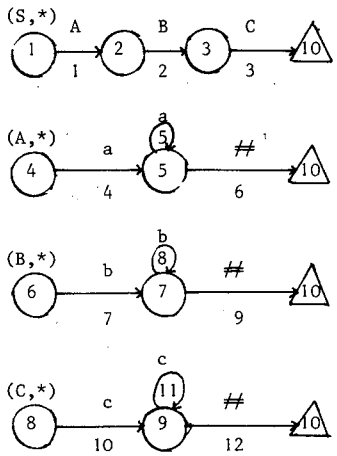
En somme, l'introduction des "ruptions" permet seulement d'augmenter la compacité de l'écriture de certaines grammaires, mais diminue celle des calculs, puisqu'on doit répéter le calcul d'un groupe "à contexte" pour chaque appel. Cependant, les sous-groupes sans contexte d'un groupe à contexte ne sont, eux, calculés qu'une fois, et l'ensemble des calculs est finalement comparable à ceux d'un RTR sans contexte où il aurait fallu multiplier les sommets et les arcs pour chaque contexte.

4.2. PUISSANCE AVEC REGISTRES (ALGOGRAMMAIRES)

L'utilisation de registres numériques permet d'augmenter strictement la puissance du modèle. Donnons un exemple avant de préciser la correspondance entre $\mathcal{P}, \mathcal{F}$ et les classes atteintes.

4.2.1. EXEMPLE

ρ :



On utilise un seul registre ν , et on pose :

sur 1[A]	ν(C) := ν(T)
sur 2[B] et 3[C]	ν(C) := ν(T) si ν(T) = ν(P)
sur 4[a] ,7[b] et 10[c]	ν(C) := 1
sur 5[a] ,8[b] , et 11[c]	: ν(C) := ν(P)+1

Cette algogrammaire reconnaît alors $L = \{a^n b^n c^n \mid n \geq 1\}$ pour toute heuristique exhaustive, et L est strictement sous-contexte. On voit sur cet exemple comment reconnaître de manière uniforme des langages "voisins" comme $\{a^n\}, \{a^n b^n\}$ et $\{a^n b^n c^n\}$, sans utiliser de mécanisme d'"adjonction distributive".

4.2.2. CODAGE ET CLASSES ATTEINTES

Soit V_T un ensemble fini, $V_T = \{a_1, \ldots, a_p\}$. On définit un codage bijectif g de V_T^* sur $\mathbb{N}$ en identifiant V_T à $\{0, \ldots, p-1\}$ et en associant à tout mot $x = y_k, \ldots, y_0$ de V_T^* l'entier $g(x) = \sum_{i=0}^{k} y_i p^i$ dont x est la représentation p-adique.

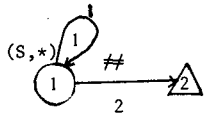
Définition 24

On dira qu'un langage $L \subset V_T^*$ possède une propriété P d'ensembles si son code $g(L) \subset \mathbb{N}$ la possède. Ainsi, L est récursif primitif (récursif, r.e.) si g(L) est récursif primitif (récursif, r.e.).

Soit | un symbole particulier, on définira :

BARRE(L) = $\{|^{g(x)} \mid x \in L\} \subset \{|\}^*$. Si $L' \subset \{|\}^*$, L' est un langage-barre *(tally language).*

g étant une bijection algorithmique, il est évident qu'on sait reconnaître L si on sait reconnaître BARRE(L). D'autre part, le fait pour L de posséder la propriété P est récursivement invariant. On peut associer à P l'existence d'une fonction $f \in \mathcal{G}_p$ possédant certaines propriétés (caractéristique et dans RP^1 ou R^1, semi-caractéristique et dans P^1). Il est alors immédiat d'associer à L l'algogrammaire associée, qui reconnaît BARRE(L) :

ρ : 

avec { sur 1[a] l'action ν(C) := ν(P)+1 ; sur 2[#] la condition f(ν(P)) = 1

Proposition 5

Soit P une propriété d'ensembles associée à une classe $\mathcal{G}_p$ de fonctions. On peut reconnaître les langages possédant cette propriété au moyen des algogrammaires associées et d'une heuristique exhaustive très simple dès que $\mathcal{G}_p \subset \mathcal{F}$.

Ainsi, on peut reconnaître les langages (primitifs) récursifs dès que $\mathcal{F}$ contient R^1 (RP^1), et on a vu plus haut comment obtenir simplement ces classes. A ce propos, on peut rappeler des résultats intéressants :

1) Les fonctions récursives primitives sont calculables par la classe des transducteurs réguliers (à sens unique sur la bande d'entrée) déterministes composés linéairement et récursivement, ainsi que par la classe des transducteurs à pile composés linéairement et récursivement [8, th. II.9 et II.10].

2) Ces mêmes fonctions sont également celles qu'on peut programmer dans des langages "simples", les langages $Loop$ et $Loop_{min}$ dûs à Ritchie et à Minsky. Constable et Borodin (1972) ont pu montrer que ces formalismes permettent de caractériser les fonctions élémentaires au sens de Grzegorczyk [24] : ce sont exactement celles que calculent les programmes Loop où la profondeur d'imbrication des boucles est inférieure à 2 ($\mathcal{L}_2$).

4.2.3. RECONNAISSANCE ET ELEMENTARITE DES DAL

Définition 25 (Joshi)

Une grammaire d'adjonction distributive, *ou DAG[27] est un couple $G = (\Sigma_C, J)$, où Σ_C est un ensemble de* chaînes axiomes *et $J = \{u_t\}$ un ensemble de* règles d'adjonction *de la forme $u_t = (\sigma_i, (\sigma_j), \xi_k)$. σ_i est une* chaîne hôte. *Dans toute chaîne $\sigma_i = a_{i1} \ldots a_{ip_i}$, on considère qu'il y a deux* points d'adjonction, *soient l_j et r_j, à gauche et à droite de chaque symbole a_{1j}. (σ_j) désigne un tronçonnement de σ_j en m parties, soit $(\sigma_j) = (\sigma_{j_1}) \ldots (\sigma_{j_m})$, et ξ_k est un* vecteur d'adjonction *de dimension m et de composantes prises dans $\{l_j, r_j\}_{j \leq p_i}$ en respectant l'ordre défini par : $s_v < t$ si $v<w$ ou si $v=w$, $s=l$ et $t=r$.*

L'effet d'une règle d'adjonction est d'adjoindre les morceaux de σ_j aux points d'adjonction de σ_i spécifiés par ξ_k. Une chaîne adjointe peut déjà avoir reçu des adjointes. Le langage d'adjonction distributive, *ou DAL,* engendré par G *est l'ensemble des chaînes qu'on peut obtenir à partir de Σ_C par adjonction selon J.*

Exemple

$G = (\Sigma_C, J)$, $\Sigma_C = \{abc\}$, $J = \{u_1, u_2, u_3, u_4\}$, avec

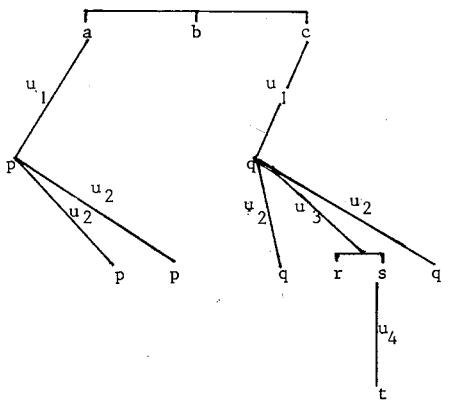
$u_1 = (abc, (p)(q), l_1 l_3)$

$u_2 = (pq, (p)(q), r_1 r_2)$

$u_3 = (pq, rs, r_2)$

$u_4 = (rs, t, l_2)$

p p p a b q q r t s q c ∈ L(G)



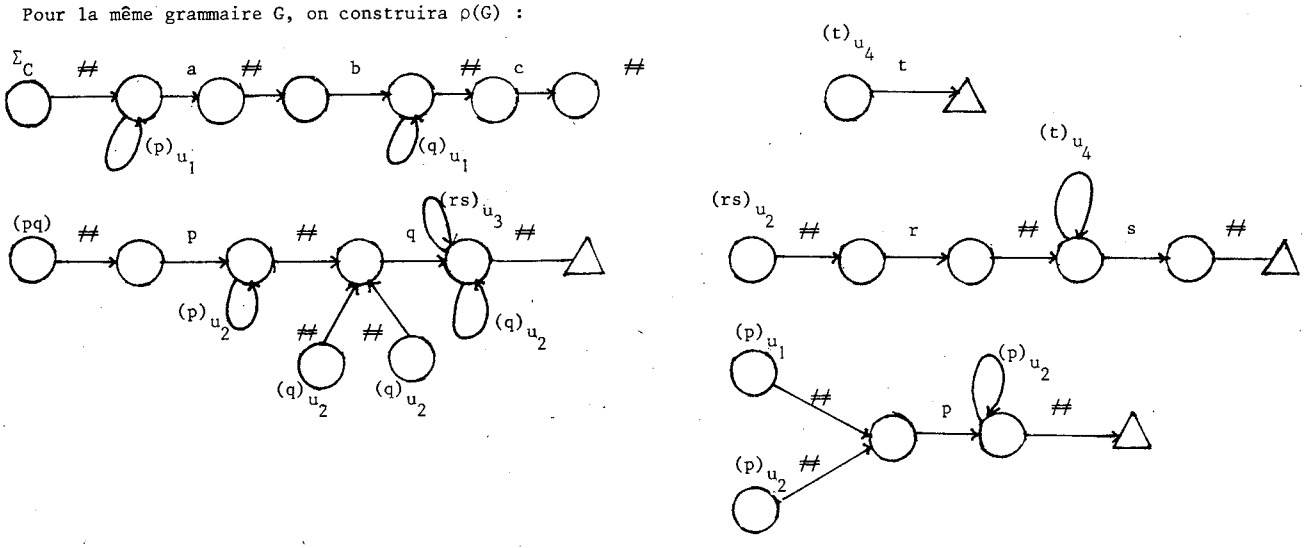
On va maintenant construire une algogrammaire associée à une DAG G de façon standard, au moyen d'un RTR simplement déduit de G et d'une gödelisation convenable des parcours permettant la reconnaissance au moyen d'un registre et de fonctions élémentaires.

1) RTR associé

On décrira chaque chaîne par une suite d'arcs alternativement "vides" ou étiquetés par un symbole de la chaîne. On obtient ainsi que représentation des points d'adjonction. Chaque règle u_t se traduit alors par m arcs de source égale au but, étiquetés par les "buts de chaînes" et le nom de la règle qui constituent N.

Exemple

Pour la même grammaire G, on construira ρ(G) :



2) Gödelisation

Il est clair qu'il faut utiliser au moins un registre pour contrôler la correction du parcours ; en particulier, il faut que tous les morceaux d'une chaîne adjointe soient présents. Pour n'utiliser qu'un registre, on propose la gödelisation suivante pour les parcours, ou, ce qui revient ici au même, pour les dérivations représentées (comme plus haut) en étiquetant les arcs par les noms de règle.

Notation

Pr(k) désigne le k-ième nombre premier. Pr(1) = 2, Pr(0) = 1.

On attribue un code <x> à tout symbole x de G. Plus précisément, $\langle u_i \rangle = 4i+3$ et $\langle a_i \rangle = 4i+5$.

Définition 26

Une dérivation dans G *est une expression*

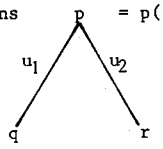
$x = a_1(y_1), a_2(y_2), \ldots, a_n(y_n)$ *où* $a_1 a_2 a_n \in V^*$ *est la* chaîne *et les* y_i *les* structures adjointes. *Chaque structure adjointe s'écrit* $y = u_1(x_1), u_2(x_2), \ldots, u_m(x_m)$, *où les* x_j *sont des dérivations et les* u_j *des règles de G.*

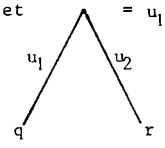
Le code d'une expression $t = s_1(t_1), s_2(t_2), \ldots, s_p(t_p)$ *est défini récursivement par :*

$$\langle t \rangle = \prod_{k=1}^{p} Pr(2k)^{\langle s_k \rangle} Pr(2k+1)^{\langle t_k \rangle} \quad \text{et} \quad \langle \Lambda \rangle = 0.$$

Exemples :

1) Dans $p = p(u_1(q),u_2(r))$, p est la chaîne, et $= u_1(q),u_2(r)$ la structure adjointe.





2) La dérivation de l'exemple initial s'écrit :

$x = a(u_1(p(u_2(p),u_2(p)))),b,c(u_1(q(u_2(q),u_3(r,s(u_4(t))),u_2(q))))$, et, si on a les codes :

$<u_1> = 7$	$<a> = 9$	$<q> = 25$
$<u_2> = 11$	$<b> = 13$	$<r> = 29$
$<u_3> = 15$	$<c> = 17$	$<s> = 33$
$<u_4> = 19$	$<p> = 21$	$<t> = 37$

on aura $<x> = 3^9.5^{3^7}.5^{3^{21}}.5^{3^{11}}.5^{21}.7^{21}.11^{3^{11}}.5^{21}.7^{13}.11^{0}.13^{17}.17^{3^7}.5^{3^{25}}.5^{3^{11}}.5^{25}.7^{15}.11^{3^{29}}.5^{0}.7^{33}.11^{3^{19}}.5^{37}.13^{11}.17^{25}$

Bien sûr, on n'envisage ce codage qu'à une fin toute théorique ![§] On définit classiquement les fonctions suivantes, récursives primitives et même élémentaires :

$Id(m,k)$ = "l'exposant de $Pr(k)$ dans la décomposition de m"
$= \mu i \leqslant m[Pr(k)^{i} | m \;\&\; \neg Pr(k)^{i+1} | m]$

$|m|$ = "le nombre de facteurs premiers de m"
$= \sum_{k \leqslant m} (1 \dot{-} Id(m,k))$

$R(i,t,m)$ = "le rang k du i-ème facteur d'exposant t dans m"

$$\begin{cases} R(0,t,m) = 0 \\ R(1,t,m) = \mu k \leqslant |m|[Id(m,2k) = t] \\ R(i+1,t,m) = \begin{cases} \mu k \leqslant |m|[Id(m,2k) = t \;\&\; k > R(i,t,m)] & \text{si } R(i,t,m) \neq 0 \\ 0 & \text{sinon} \end{cases} \end{cases}$$

Rappelons que $\mu k \leqslant m[P(k)]$ vaut 0 si $\bigwedge_{k=0}^{m} \neg P(k)$

$x*y$ = "le code du mot obtenu en concaténant les mots de codes x et y "
$= x.\prod_{k=1}^{|y|} Pr(|x|+1)^{Id(y,k)}$ on vérifie aisément l'associativité.

§ - Comme d'ailleurs toutes les gödelisations de systèmes formels.

Définissons enfin, si $u_t = (\sigma_i,(\sigma_j),\xi_k),\delta_t = <\sigma_j>$

On peut alors définir un prédicat à deux arguments $A_G(m,\delta)$ qui exprime que m est le code d'une dérivation dont la chaîne principale a pour code δ et qui est compatible avec G.

$$A(m,\delta) \equiv [\delta = \prod_{k=1}^{\left\lfloor \frac{|m|}{2} \right\rfloor} Pr(2k)^{Id(m,2k)} \;\&$$

$$\max_{1\leq j\leq \left\lfloor \frac{|m|}{2} \right\rfloor (|Id(m,2j+1)|)} \bigwedge_{i=1}^{|J|} \bigwedge_{t=1} A\left(\left\lfloor \frac{|m|}{2} \right\rfloor \underset{k=1}{*} Id(Id(m,2k+1),R(i,<u_t>,Id(m,2k+1))+1),\delta_t \right)]$$

Ceci exprime simplement qu'on considère les adjonctions règle par règle et de gauche à droite, qu'on "recolle" les morceaux trouvés et qu'on vérifie leur correction.

A est élémentaire. Il reste à voir qu'on peut construire m dans l'algogrammaire de façon élémentaire, c'est-à-dire en prenant $\mathcal{F}_0\mathcal{E}$. Le fait qu'on a un codage qui commence à Pr(2)=3 permettra d'utiliser Pr(1)=2 pour "stocker les adjoints" au cours d'un calcul.

Ainsi, en notant $m=\nu(P)$ et $n=\nu(T)$

- sur les arcs d'adjonction, on prendra :

$$f_a(m,n) = \frac{m}{2^{Id(m,1)}} \cdot 2^{Id(m,1)*(2^{<\mu_t>}.3^n)}$$

- sur les arcs de transition étiquetés, on prendra :

$$f_t(m,n) = m.Pr(|m|+1)^n$$

- sur les arcs vides issus d'un sommet où arrive un arc étiqueté, on met à sa place le code des adjoints du dernier symbole :

$$f_p(m) = \frac{m}{2^{Id(m,1)}} \cdot Pr(|m|+1)^{Id(m,1)}$$

- enfin, toutes les initialisations de ν sont à 1.

Proposition 6

Les langages d'adjonction distributive sont élémentaires.

Remarquons que cette propriété concerne une fonction caractéristique, et non une fonction d'énumération. Pour ces dernières, Grzegorczyk [24, th. 5.3] a en effet montré que tout r.e. est énumérable par une fonction de $\mathcal{E}^0$.

4.3. COMPLEXITE

L'algorithme d'Earley (1968) pour les grammaires hors-contexte (et même pour les grammaires d'expressions régulières) a été adapté par Woods (1969) aux réseaux de transitions. Si n est la longueur d'une chaîne d'entrée, cet algorithme possède des complexités en place L et en temps T telles que :

- en général, $L(n)\backsim n^2$ et $T(n)\backsim n^3$
- si la grammaire est non ambiguë, $T(n)\backsim n^2$
- si la grammaire est "à ambiguïté directe bornée" [49] (LR(k) en particulier), $L(n)\backsim n$ et $T(n)\backsim n$.

Nous allons rappeler la méthode de Woods et lui associer une heuristique particulière qui permet de conserver les mêmes classes de complexité. Notons toutefois que ces classes sont toujours définies à un facteur multiplicatif près, et on verra que notre codage est légèrement plus complexe (pour permettre toute une classe d'heuristiques).
D'autre part, les bornes calculées ne sont valables que sur un modèle de machine à accès aléatoire (RAM)§.

§ - "Random access machines". Voir Cook (1973) pour le lien entre la complexité des RAM et des MT multibandes.

4.3.1. L'ALGORITHME DE WOODS-EARLEY

Définition 27

Soient $q\in S(\rho)$ et $x\in M\cup N$. L'ensemble des sommets de ρ accessibles depuis *q* via x *est défini par :*

$\eta(q,x) = \{q' \mid (\exists \gamma\in C(\rho))(\exists p)[\sigma\gamma(0) = q \ \&\ \tau\gamma(p) = q' \ \&\ \varepsilon\gamma\{0,\ldots,p\} = x]\}$

Si $A\in N$, l'ensemble des (symboles) descendants gauches directs de A dans ρ *est* $G(A) = \{x\in M\cup N \mid \eta(\varepsilon^{-1}(A,*),x)\neq\emptyset\}$

Si G^* est l'extension transitive de G, l'ensemble des états initiaux prédictibles à partir de A *est :*

$L^*(A) = \varepsilon^{-1}(G^*(A),*)$.

Enfin, le nom d'un sommet *q de ρ est l'étiquette (dans N) du sommet initial associé à q :*

$h(q) = \pi_1\varepsilon(\iota q'[q'\in 0(\rho) \ \&\ q\in\hat{\tau}^*(q')])$.

L'algorithme consiste, pour toute chaîne d'entrée $x = x_1x_2\ldots x_n$, à construire des ensembles d'états $(Q_0,Q'_0),(Q_1,Q'_1),\ldots,(Q_n,Q'_n)$. Chaque état est un couple $[q,i]$ ou q est un sommet du réseau et $i\leqslant n$ un entier qui indique que la réduction correspondante a été "lancée" sur x_i. Plus précisément, si S désigne l'axiome de la grammaire d'expressions régulières associée à ρ, on a les étapes suivantes :

(1) Initialisation : on construit $Q_0 = \{[\varepsilon^{-1}(S,*),0]\}$ et sa clôture

$$Q'_0 = Q_0\cup\{[q,0] \mid q\in L^*(S)\}$$

(2) Progression : pour i croissant de 1 à n, on construit Q_i et Q'_i de la manière suivante :

(2.1) Transitions : $Q_i = \{[q',j] \ \ q'\in\eta(q,x_i) \ \&\ [q,j]\in Q'_{i-1}\}$ ("scanner" d'Earley)

(2.2) Clôture : on initialise Q'_i à Q_i et on le considère comme un ensemble ordonné, qu'on parcourt depuis le début en appliquant à chaque état $[q,j]$ les deux opérations suivantes, qui rajoutent éventuellement (et au total un nombre fini de fois) de nouveaux éléments à la fin, de façon qu'ils subissent eux aussi ces deux opérations.

(2.2.a) Appel : pour tout $q'\in L^*(\varepsilon(q))$, on ajoute $[q',i]$ à la fin de Q'_i, sauf s'il s'y trouve déjà.

(2.2.b) Retour : si $\varepsilon(q) = \Delta$, on parcourt les états $[q',j']$ de Q'_j qui ont pu "appeler" q (c'est-à-dire tels que $\eta(q',h(q))\neq\emptyset$), et, pour chacun d'eux et pour chaque q" de $\eta(q',h(q))$, on ajoute $[q",j']$ à la fin de Q'_i, sauf s'il s'y trouve déjà.

Q_i est achevé dès qu'on a parcouru le dernier état sans en ajouter de nouveau.

(3) Décision : $x_1x_2\ldots x_n$ est accepté si Q'_n contient $[\varepsilon^{-1}(\Delta),0]$, et refusé sinon ou si l'un des Q_i est vide.

Bornes

L'obtention des bornes annoncées dépend essentiellement du fait qu'on a une mémoire à accès aléatoire et de la façon dont elle est organisée. Par exemple, si, à (2.2.a) ou (2.2.b), on vérifiait la condition "sauf s'il s'y trouve déjà" en parcourant séquentiellement Q'_i, il faudrait une puissance de n supplémentaire pour estimer T(n). En effet, $|Q'_i|$ peut croître proportionnellement à i. Si par contre Q'_i est indexé par les pointeurs de retour j', il suffit pour vérifier cette condition de parcourir un ensemble de taille bornée par $|S(\rho)|$. On peut alors considérer comme opérations élémentaires la vérification de cette condition ou la production d'un pointeur vers la liste $[q',j']$ des états de Q'_j qui ont pu "appeler" un q donné.

Soient $U = |S(\rho)|$ le nombre de sommets de ρ

$V = \underset{S(\rho)}{\text{Max}} |\underline{\sigma}(q)|$ le nombre maximum d'arcs issus d'un sommet de ρ

$W = \underset{S(\rho)}{\text{Max}} |L^*(\varepsilon(q))|$ le nombre maximum de sommets prédictibles

Q'_i contient au plus U(i+1) états [q,j]. D'où la borne de place $L(n)\sim n^2$. Pour le nombre d'opérations, reprenons l'algorithme :

(1) Q'_0 est construit en au plus W+1 opérations.

(2) Pour tout i entre 1 et n,

(2.1) Il faut au plus V opérations pour chaque état de Q_{i-1}, donc au plus UVi.

(2.2) (2.2.a) Il faut moins de W opérations, puisque $|L^*(\varepsilon(q))|\leqslant W$.

(2.2.b) Il faut au plus UV(j+1) opérations pour chaque [q,j] final.

Pour (2.2), on a donc au plus $U(i+1)(W+UV(j+1))\leqslant UW(i+1)+U^2V(i+1)^2=X(i)$

Pour (2), $\sum_{i=1}^{n} X(i) = U^2Vn^3+O(n^2)$, d'où le résultat final.

Pour obtenir une borne de temps en n^2 si la grammaire est non ambiguë, il faut pouvoir borner le nombre d'opérations pour chaque [q,j] de Q'_i au (2.2.b). Pour ceci, on ne peut plus parcourir séquentiellement Q'_j, et il faut donc indexer Q'_j par les symboles non-terminaux appelés, qui sont en nombre borné par U. Pour tout couple (j,h(q)), on obtient alors en une seule opération un pointeur vers la liste des [q',j'] de Q'_j qui ont pu "appeler" q, soit $Q''_j(h(q))$. Alors le nombre d'opérations aux (2.2) est au plus proportionnel à U(i+1) sauf si, au (2.2.b), on a trouvé au moins un état à ajouter, soit [q",j'], de plus d'une façon. Avec la structure des Q''_j, il ne faut en effet qu'une opération par état [q,j] tel que $Q''_j(h(q))=\emptyset$, et par état à ajouter sinon. D'autre part, si on trouve un état à ajouter plus d'une fois, la grammaire associée acceptera au moins une chaîne ambiguë $y_1,\ldots y_m$, sauf si cet état correspond à un sommet "trappe" de ρ, q_0, tel qu'aucun parcours de début q_0 n'ait son mot des feuilles sur M^*, auquel cas la grammaire n'a pas été donnée sous forme réduite, c'est-à-dire sans règles inutiles et/ou symboles superflus.

La borne en n pour les "grammaires à ambiguïtés directe bornée" provient du fait que $|Q'_i|$ est borné pour ces grammaires, et on peut donc remplacer les produits Ui par une nouvelle U'.

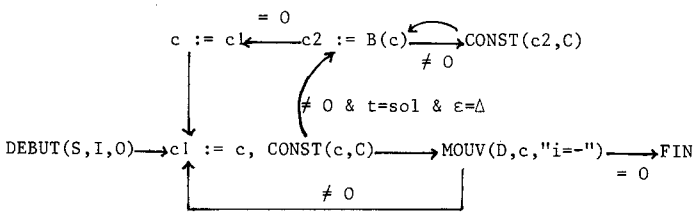
4.3.2. CORRESPONDANCE ENTRE LES CODAGES

Si χ est réduit à la chaîne $x_1x_2,\ldots,x_n$, l'ensemble de toutes les cellules c telles que $g(c)=x_i$ correspondra à Q'_i. Sans modification du codage, les Q''_j correspondent aux cellules liées par b (buts égaux). Il faut cependant modifier les listes $\bar{1}(s)$ en les indexant par les non-terminaux, d'où des listes $\bar{1}(x_i,u)$ $u\in N$ pour décider si CONST produit un "but" déjà présent sur x_i en un nombre d'opérations borné.

Plus précisément,

- la création de cellules "but" ou "init" correspond à la "prédiction" d'Earley (2.2.a)
- l'insertion du résultat d'un calcul dans $\bar{1}(\chi)$ et l'utilisation de calculs $1(k,x_i)$ $(k\geqslant 2)$ pour construire des "sol" correspond à la "complétion" (2.2.b).
- l'utilisation d'un $1(1,x_i)$ (terminal) pour créer une cellule "sol" correspond au "scanner" (2.1).

L'heuristique dans l'algorithme de Woods-Earley consiste à construire, pour tous les x_i successifs, toutes les cellules possibles associées. Ceci veut dire qu'on va progresser sur la bande de travail en ne revenant en arrière que pour les "retours" (2.2.b), c'est-à-dire pour réaliser les "but" appelants. On peut la décrire par :



Proposition 7

Il existe une heuristique qui permet, pour tout RT associé à une grammaire d'expressions régulières, de conserver les bornes en temps et en place obtenues par l'algorithme de Woods-Earley.

4.4. ADEQUATION ET ADAPTATION

4.4.1. ADEQUATION

Dans un modèle en réseau de transitions, il est très facile d'associer des poids aux arcs. Ceci revient à associer un poids à chaque symbole de chaque règle d'une grammaire d'expressions régulières, ce qui généralise la notion de grammaire pondérée, en tant que formalisme de description. Pour ce qui est de l'interprétation, elle est différente, puisque le critère de définition n'est pas nécessairement du type "somme" ou "max" [46] : on peut utiliser ces poids comme des contrôles sur les heuristiques associées aux algogrammaires. Par rapport à une heuristique donnée, un tel système de poids peut donc caractériser une "variante" d'une grammaire[§], correspondant à un corpus, à un auteur, ou plus modestement à une étape de la mise au point !.

D'autre part, le modèle qu'on a proposé, comme les réseaux de transitions augmentés, possède la clarté liée au mode de représentation choisi, ainsi qu'une puissance accrue par rapport aux modèles hors-contexte. Cependant, alors que les réseaux de transitions augmentés possèdent la puissance de calcul maximale, celle d'une MT [56] , on a imposé certaines restrictions pour assurer la décidabilité, qui est une propriété essentielle d'un module destiné à s'intégrer à un système informatique.

Enfin et surtout, ce modèle permet d'utiliser toute une classe d'algorithmes, les heuristiques[§§], sur les données linguistiques (algogrammaires), et donc de chercher, selon le mot de Mel'tchuk, à "simuler le processus humain de compréhension". D'un point de vue pratique, on peut remarquer que le codage utilisé n'est pas beaucoup plus complexe que celui de Woods-Earley, seule l'organisation diffère, et que l'introduction de procédés heuristiques peut se révéler d'autant plus féconde que la donnée à analyser est plus ambiguë, ce qui èst par exemple le cas de la parole [52] . Dans ce cas d'ailleurs, on voit que le modèle peut facilement permettre l'interaction entre différents modules, puisqu'il suffit d'intégrer l'heuristique au "moniteur" du système.

4.4.2. ADAPTATION

La nature statistique des phénomènes linguistiques, et surtout grammaticaux, est encore mal connue, et il n'est pas sûr que, comme pour les bases dans un texte scientifique slave, les unités grammaticales en général se stabilisent après 2 000 à 4 000 mots de texte.[§§§] Certains résultats laissent d'ailleurs penser le contraire [57].

Il n'est donc pas question d'un véritable apprentissage de grammaires ni d'un apprentissage d'heuristiques : on se bornera à proposer une "adaptation" très simple, en espérant trouver par la suite des résultats expérimentaux.

Pour ceci, on peut associer un ensemble ψ de compteurs à $A(\rho)\oplus S(\rho)$, et les incrémenter à chaque utilisation effective de CONST.

- pour les compteurs associés aux sommets $s\subset S(\rho)$, à chaque création de cellule c de type "init" ou "début", telle que $r(c)=s$.

- pour les compteurs associés aux arcs $a\subset A(\rho)$, à chaque création de cellule "sol" telle que $rm(c)=a$.

L'adaptation consiste alors à réajuster les poids $\pi(s)$ et $\pi(a)$ après un certain volume de traitement et avec un certain algorithme, tous deux à déterminer expérimentalement.

Par analogie avec le modèle linéaire de Bush et Mosteller (1955), on peut proposer d'utiliser une formule comme :

$$\pi'(a) = \lambda\ \pi(a) + (1-\lambda)\frac{\psi(a)}{\psi\sigma(a)}\ ,\ \lambda\in[0,1]\ .$$

---

§ - "Variante" au sens W. Klein (1974), par exemple.

§§ - Et on peut penser à essayer des procédures bien connues de recherche dans les arborescences des choix qui apparaissent dans les jeux [36,48,49] .

§§§ - D. Hérault, séminaire IRIA, Avril 1974.

Comme on l'a vu au chapitre précédent, on ne peut guère espérer faire de l'apprentissage grammatical "structural", puisque les méthodes employées supposent, dans les cas intéressants, qu'on connaisse a priori la réponse. Et alors cet "apprentissage" est un exercice de style fort coûteux qui n'a pas sa place dans un système de T.A. utilisable.
Par contre, l'idée d'introduire des pondérations a une motivation linguistique [28] et logique [17], et on peut espérer que l'ajustement paramétrique converge et permette d'automatiser une partie de la mise au point, si toutefois l'algogrammaire (sans les poids) et l'heuristique choisies sont "suffisamment proche" d'une description adéquate de la langue et du processus d'analyse à simuler.

On pourrait encore penser à des méthodes énumératoires d'apprentissage, utilisant éventuellement des critères de complexité ou de probabilité (cf. Chapitre II). Mais les classes de grammaires pour lesquelles on a des résultats sont toujours des sous-classes strictes des grammaires hors-contexte (HC). Et Chomsky (1957) a montré voici longtemps que le modèle HC n'est pas assez puissant pour décrire les langues naturelles[§].

§ - Cette question a d'ailleurs été reprise par Postal (1964), et Corstius (1975).

TABLE BIBLIOGRAPHIQUE

[1] Aho & Ullman (1972)
[2] Boitet (1973)
[3] Booth, ed. (1967)
[4] Brandt Corstius (1975)
[5] Busacker & Soaty (1965)
[6] Bush & Mosteller (1955)
[7] Chauché (1973)
[8] Chauché (1974)
[9] Chauché (1975a)
[10] Chauché, Guillaume, Quézel (1973a)
[11] Chauché, Guillaume, Quézel (1973b)
[12] Chomsky (1957)
[13] Comter (1970)
[14] Constable & Borodin (1972)
[15] Conway (1963)
[16] Cook & Rechow (1973)
[17] Culik (1973)
[18] Earley (1968)
[19] Earley (1970)
[20] Feldman & Yakimovsky (1974)
[21] Friant (1971)
[22] Fry (1970)
[23] Garvin (1971)
[24] Grzegorczyk (1963)
[25] Guinsburg & Spanier (1968)
[26] Hopcroft & Ullman (1969)
[27] Joshi, Kosaraju & Yamada (1972)
[28] W. Klein (1974)
[29] Knuth (1965)
[30] Knuth (1971)
[31] Kosaraju (1974)
[32] Kuno & Oettinger (1962)
[33] Lomet (1973)
[34] Mal'cev (1970)
[35] Manna & Waldinger (1971)
[36] Nilsson (1971)
[37] Ollongren (1974)
[38] Pentonnen (1974)
[39] Peters & Ritchie (1969)
[40] Petrick (1965)
[41] Poitou (1964)
[42] Postal (1964)
[43] Quinton (1975)
[44] Riesbeck (1973)
[45] Robinson (1950)
[46] Salomaa (1969a)
[47] Salomaa (1973)
[48] Slagle (1970)
[49] Slagle & Dixon (1969)
[50] Thorne, Bratley & Dewar (1968)
[51] Tseitin (1971)
[52] Tubach (1970)
[53] Uhr (1973)
[54] Veillon (1970)
[55] Woods (1969)
[56] Woods (1970)
[57] Zampolli (1973)
[58] * rapport TAUM 71 (1971)

CHAPITRE IV

PRINCIPES DE REALISATION D'UN PROTOTYPE

INTRODUCTION

Dans le cadre de ce travail, j'ai été amené à proposer et à commencer à réaliser un système de TA expérimental, qui incorpore les idées exposées dans le chapitre d'introduction, à l'aide des méthodes étudiées par la suite. Le but recherché est de construire un système, baptisé TAUPHA[§] , en attendant mieux,

1) où l'organisation du traitement, sa "stratégie", soit définissable dans un langage externe au même titre que les données linguistiques des composants du système

2) où certains composants soient munis d'un dispositif d'apprentissage, ou, plus modestement, d'adaptation pendant l'exécution.

Pour ceci, j'ai bien sûr cherché à utiliser les composants déjà mis au point au GETA et écrits pour la plupart en assembleur 360. Pour des raisons de compatibilité et de facilité de programmation, j'ai programmé moi-même en PL360[§§] . Tous les systèmes du GETA sont actuellement réalisés sur l'ordinateur IBM 360/67 de l'USMG, sous le système CP/CMS.[§§§]

D'autre part, des circonstances sur lesquelles il serait vain d'épiloguer ont fait que j'ai dû entreprendre seul l'implantation des composants nouveaux et la modification de certains composants anciens pour l'intégration dans TAUPHA. Ceci explique que la réalisation de ce prototype ne soit achevé qu'en partie et doive certainement être améliorée.

Je me limiterai donc ici à présenter l'organisation générale de TAUPHA et à décrire plus précisément les langages externes des composants ALGOG (algogrammaires) et MONIT (moniteur), en donnant quelques exemples.

---

§ Traduction Automatique Utilisant un Prototype Heuristique et Adaptatif.

§§ Cf. D. SUTY, "La langage PL360", Grenoble, 1971.

§§§ Cf. "Manuel d'utilisation du système CP 67/CMS", Grenoble, 1969 & svt.

I - ORGANISATION GENERALE

1.1. LE SYSTEME DU G.E.T.A.

1.1.1. GENERALITES

Le système du CETA (Grenoble) est bien connu [6,10,11]et leurs références . Outre ce qui a été dit plus haut, on peut lui reprocher de ne pas observer la seconde règle d'or pour la morphologie, d'utiliser une "cote mal taillée" (règles binaires de Chomsky + variables véhiculaires + validations et saturations) pour l'analyse syntaxique et de remettre en cause dans certaines phases le travail d'une phase antérieure. On a cherché à y remédier dans le système du GETA.

Il faut d'abord traduire en clair quelques expressions de notre jargon.

1) Une variable est définie par un nom et une liste de valeurs particulières. L'ensemble de ses valeurs possibles est :

- l'ensemble des éléments de la liste et une valeur "vide", si elle est "exclusive"
- l'ensemble des parties de l'ensemble précédent, si elle est "non-exclusive"
- l'ensemble des entiers inférieurs ou égaux à l'unique élément de la liste, si la variable est "arithmétique.

On écrira par exemple GENRE := (MASC,FEM,NEUTRE). D'autre part on considère des "variables générales", ou sur-variables, qui regroupent d'autres variables.

2) Un masque de variable[§] est une combinaison de valeurs des variables utilisables ("déclarées" au système). Les étiquettes portées par les structures (chaînes, arbres,...) seront toujours des masques de variables, le jeu de variables pouvant d'ailleurs changer d'une phase à l'autre du traitement. Au contraire d'une habitude trop répandue, les étiquettes sont donc complexes. Ceci permet d'éviter de faux problèmes (discontinuités artificielles) et de séparer les propriétés géométriques des structures (noeud "ancêtre" d'un autre...) de leur interprétation.

3) Un format est un masque de variables particulier et constant auquel on a donné un nom, qui peut servir de référence dans des dictionnaires ou des grammaires.

4) Une forme est une succession de caractères non blancs encadrée par deux blancs.

5) Une unité lexicale est une valeur de la variable UL, prédéfinie et exclusive. Les unités lexicales sont introduites dans les dictionnaires et non pas à la déclaration des variables.

1.1.2. ORGANISATION

Les composants du système sont :

ATEF — analyseur morphologique qui transforme la chaîne d'entrée en une arborescence étiquetée [5,2] . Il réalise un automate d'états finis non-déterministe. Ses données externes comportent des déclarations de variables et de formats, des dictionnaires et une grammaire.

CETA — transducteur d'arborescences qui permet de simuler toute grammaire d'arbres [2,4] . Il est décidable, peut calculer les fonctions primitives récursives et est de complexité linéaire pour toute transformation[§§]. Ses données externes sont constituées par des déclarations de variables et de formats et par un fichier contenant les transformations élémentaires et leur organisation en sous-grammaires [3] . Il n'y a pas de dictionnaire à ce niveau.

---

§ Dans le système du CETA, on parlait plutôt de "syntagmes élémentaires".

§§ Remarquons que le découpage en unités (paragraphes, phrases) ne provient pas d'une préédition, mais est déterminée dans la grammaire.

TRANS pour le passage d'une arborescence associée à un fragment de texte en langue source à sa correspondante en langue cible. Pour l'instant, les manipulations de structure autorisées sont réduites au développement de noeuds en sous-arborescences. Les données externes contiennent les déclarations des variables et des formats source et cible, des procédures de condition et d'affectation, et la description du passage sous la forme d'un dictionnaire dont les articles sont des règles.

SYGMOR générateur morphologique qui transforme une arborescence étiquetée en une chaîne de masques, puis en une chaîne de caractères [8]. Outre les déclarations des variables et des formats, on trouve dans les données externes des dictionnaires (adressés par les valeurs de variables particulières, dont l'UL) et une grammaire.

L'organisation du système est séquentielle : un fragment de texte, dont la taille est déterminée par la place disponible en mémoire, est traité successivement par ces composants, chacun utilisant, pour une phase considérée, les tables provenant de la compilation des données externes correspondantes. Puis le système passe au fragment suivant, où les parties hâchurées représentent les programmes et les autres les données.

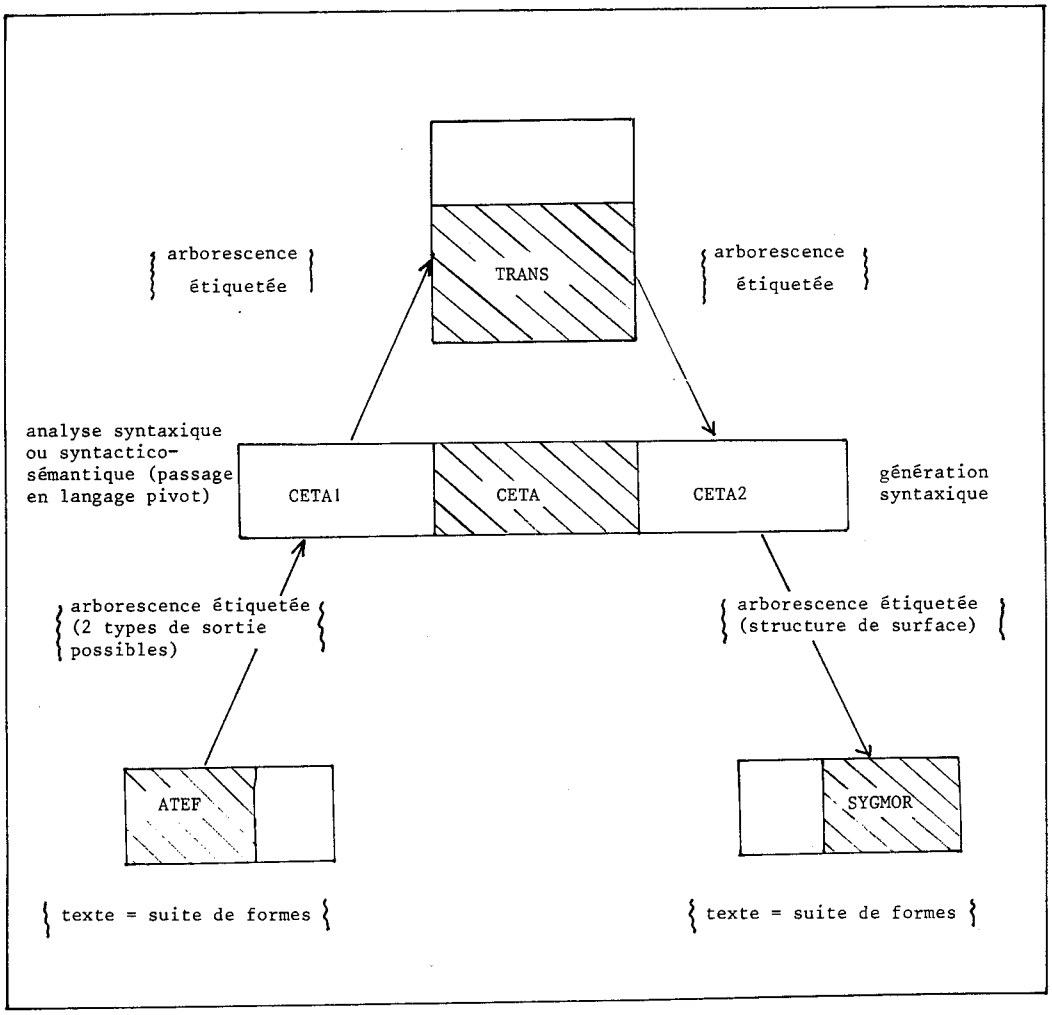


figure 1 - Le système du GETA

1.2. TAUPHA

1.2.1. GENERALITES

Le système précédent représente évidemment beaucoup de travail, d'expérience et de réflexion. Il est donc normal d'en partir pour chercher à avancer dans les directions indiquées dans l'Introduction. On peut déjà définir des stratégies de traitement au niveau de chaque composant, et on désire généraliser ceci à l'ensemble du traitement. Deux idées se dégagent naturellement : essayer de faire interagir les composants de manière plus souple et donner la possibilité de définir la stratégie globale de traitement, i.e. leur enchaînement, de manière externe. De plus, on aimerait donner au système, ou tout au moins à certaines de ses parties, des possibilités d'auto-adaptation et d'interaction avec l'utilisateur. Enfin, il importe de modifier aussi peu que possible les composants existants.

Compte tenu du caractère global d'ATEF et de CETA, on est alors conduit à modifier ATEF et à définir deux composants nouveaux, MONIT pour la stratégie globale et ALGOG pour une analyse syntaxique "heuristique".

1.2.2. ORGANISATION

Dans TAUPHA, ATEF est modifié pour permettre :

- la construction d'un graphe de chaînes parallèlement à l'analyse, c'est-à-dire complété après l'analyse de chaque forme ; il contient sous forme condensée les "homographes" (chaînes "terminales" possibles) [9]
- la correction des formes non reconnues et l'enrichissement des dictionnaires en cours d'exécution
- l'appel sur un certain nombre de formes, ou jusqu'à un marquant (forme particulière ou forme reconnue comme la dernière d'une unité de texte - cf. 1.1.2.).

Il va de soi que ces possibilités ne sont en aucun cas impératives. En particulier, l'interaction avec l'utilisateur se conçoit plutôt dans une phase de mise au point que dans une phase de traitement de masse.

Les deux nouveaux composants sont :

ALGOG qui définit un certain nombre d'opérations élémentaires à partir desquelles on peut construire, au niveau supérieur, des heuristiques d'analyse syntaxique. Sa donnée externe est une grammaire sous une forme proche des "réseaux de transitions augmentés" de [12], dite "algogrammaire".

MONIT qui construit le superviseur à partir d'une donnée externe présentée comme un automate d'états finis étiqueté par des actions conditionnelles et complété par un jeu de variables propres. Une action (une condition) est un appel (un test) d'un composant, ou une manipulation (un test) des variables propres.

On peut résumer l'organisation dans le schéma suivant.

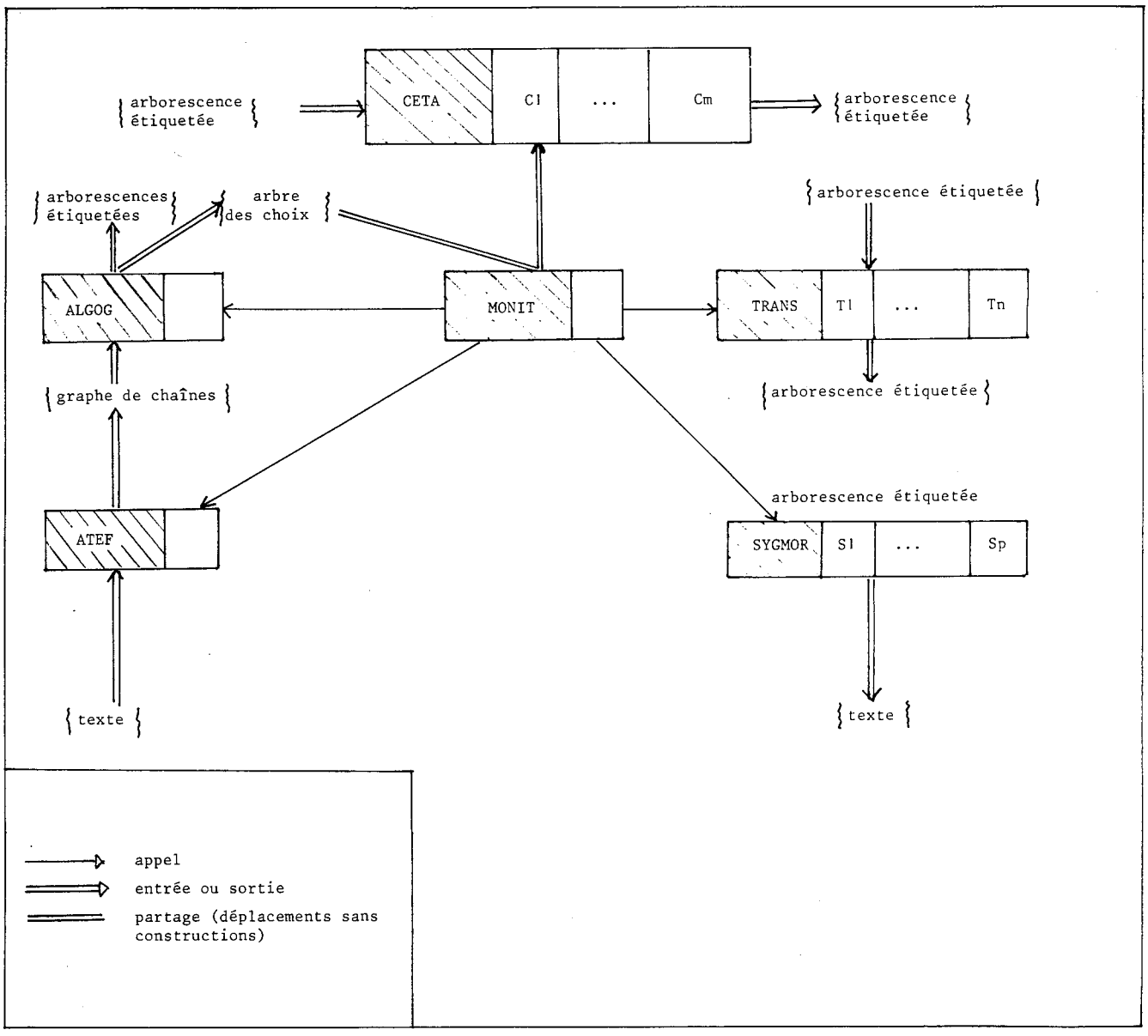


figure 2. Organisation de TAUPHA

Les langages externes se rapprochent le plus possible de ceux qu'on utilise déjà pour ATEF, CETA, TRANS et SYGMOR. En particulier, les conventions lexicales sont les suivantes,avec quelques généralisations pour MONIT (Cf III) :

- le blanc n'est pas significatif
- les commentaires apparaissent entre "**" et "." et ne contiennent pas de "." .

Les identificateurs sont alphanumériques et comportent 8 caractères au plus.

< chiffre > := 0|1|2|3|4|5|6|7|8|9

< lettre > := A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z

< caractère alphanumérique > ::= < lettre >|< chiffre>

< identificateur> ::= <lettre> <caractère alphanumérique>*

< chaîne > ::= < caractère >*

< caractère > ::= { ensemble des caractères disponibles }

Dans une chaîne, " ' " est représenté en le doublant (" '' "), et le blanc est un caractère significatif.

< littéral> ::= ' <chaîne>'

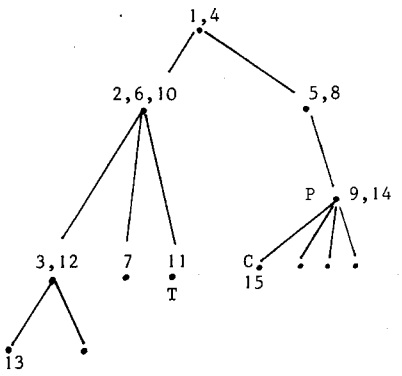
II - ALGOG : DEFINITION EXTERNE

2.1. GENERALITES

On a défini formellement les algogrammaires au chapitre précédent. ALGOG contient un langage pour les écrire et une partie algorithmique constituée par les opérations élémentaires permettant de construire des heuristiques d'analyse d'un graphe de chaînes (fourni par ATEF) selon une algogrammaire.

Plus intuitivement, disons qu'une algogrammaire, comme une grammaire formelle habituelle, donne une description des structures possibles, sans imposer un procédé d'analyse particulier. Le procédé choisi, ou heuristique, permettra de se déplacer dans chaque "arbre des analyses" associé à une algogrammaire ρ est à un graphe de chaînes χ , soit $R(\rho,\chi)$. Dans cet arbre, la racine représente l'état initial de toute analyse, et tout chemin de la racine à une feuille une analyse complète d'un chemin de χ. Une heuristique est un moyen de parcourir $R(\rho,\chi)$, en totalité ou en partie, sans jamais passer par un point sans être auparavant passé par son "père", ce qui reflète l'ordre naturel des calculs (dans le système de transitions défini par le couple (ρ,χ)).

Exemple



Voici un exemple très simple. On peut imaginer qu'à tout point correspond un certain calcul, qui n'est pas refait quand on revient à ce point, et dont le résultat est nécessaire pour passer aux calculs "fils". Au 15-ième pas de l'énumération, l'analyse courante, C, est complète, et son prédécesseur est P. Il se peut que le passage de P à C ait utilisé une autre analyse complète, soit T, précédemment calculée (au pas 11).

Une algogrammaire est un réseau (RTR) qui comporte :

- un vocabulaire propre N, ensemble de "noms" analogues aux non-terminaux d'une grammaire de constituants. Ils étiquettent les sommets d'entrée et certains arcs.

- un vocabulaire externe M constitué par les étiquettes (masques de variables) possibles des sommets des graphes de chaînes à analyser.

- des noms de registres. A chaque analyse partielle (i.e. sommet de l'arbre des analyses) est associé un ensemble de tels registres. Les registres linguistiques contiennent des arborescences étiquetées sur M et les registres numériques des entiers.

- sur les arcs, des actions conditionnelles permettant de modifier les registres de l'analyse courante, C, (associée au but de l'arc) en fonction de ceux de l'analyse précédente (associée à l'origine de l'arc), de ceux de l'analyse (associée à l'arc lui-même) qui a permis la transition, et de constantes.

- sur les sommets d'entrées, des actions conditionnelles destinées à initialiser C à partir de constantes

- des groupes de noms

- des variables entières

- trois variables particulières servant à la communication avec MONIT et prenant pour valeur un nom de registre numérique, un nom de registre linguistique et un nom de groupe, soient &NUM, &LIN, &GRU.

- trois variables ("origines") pour C, P et T, soient C, P, T.

- trois origines particulières correspondant à des sommets de χ, soient X, Y, Z.

Pour faciliter la compilation, on a pris un langage LL(1) [!] . La grammaire qui suit n'est pas sous forme normale : pour abréger, on utilise des productions vides, l'opérateur * de Kleene, et les crochets "[","]" pour entourer une chaîne éventuellement absente. Il est cependant trivial d'en déduire une grammaire BNF LL(1) §.

<algogrammaire> ::= <affectations initiales>< arcs><sommets> -FIN-

2.2. AFFECTATIONS INITIALES

<affectations initiales> ::= -DEF- <affectation initiale>*
<affectation initiale> ::= |&LIN := <registre 1> .
|&NUM := <registre n> .
|&GRU := <groupe > .
|<groupe> := (<nom a.g.>[, <nom a.g.>]*).
<registre 1> ::= <identificateur>
<registre n> ::= <identificateur>
<nom a.g.> ::= <identificateur>
<groupe> ::= &GR <chiffre>

Les "groupes de noms" &GRO, ..., &GR9 sont vides par défaut, et le "nom de groupe", &GRU, vaut &GRO par défaut.

2.3. ARCS

<arcs> ::= -ARC- <arc> *
<arc> ::= <nom d'arc> : <source> - <but><partie nom a.g.><partie poids> == <partie droite>
<nom d'arc> ::= <identificateur>
<source> ::= <identificateur>
<but> ::= <identificateur>
<partie nom a.g.> ::= |- <nom a.g.>
<partie poids> ::= |/ <nombre>
<nombre> ::= <chiffre><chiffre> *

Le poids est inférieur à $2^{15}-1 = 32767$.

<partie droite> ::= [<actions 1> [/<actions n> [/<actions g> [/<conditions 1>
[/<conditions n> [/<conditions g>]]]]]] .
<actions 1> ::= |<action 1>[, <action 1>] *
<actions n> ::= |<action n>[, <action n>] *
<actions g> ::= |<action g>[, <action g>] *

2.3.1. ACTIONS LINGUISTIQUES

<action 1> ::= <affectation de variables>|< construction d'arborescence>

Il s'agit d'affecter le masque du sommet de l'arborescence contenue dans un registre (linguistique), ou de construire une telle arborescence à partir d'autres registres linguistiques.

<affectation de variables> ::= <affectation exclusive>| <affectation non exclusive>
<affectation exclusive> ::= <variable exclusive> <but a.l> := <valeur e.a>
<variable exclusive> ::= <identificateur>
<but a.l> ::= (<registre 1> (C))
<valeur e.a> ::= <valeur exclusive>| <valeur e.r>
<valeur e.r> ::= <variable e> (<origine a.l>)
<origine a.l> ::= <nom de registre 1> (<origine a>)
<origine a> ::= C|P|T

---

§ utilisée pour la compilation

<affectation non exclusive> ::= <variable non exclusive><but a.1> := <valeur n.e.a>
<valeur n.e.a> ::= <valeur n.e.s> |<valeur n.e.s><relateur n.e> <valeur n.e.a.1>
<valeur n.e.s> ::= <valeur non exclusive> |<valeur n.e.r>
<valeur n.e.r> ::= <variable non exclusive> (<origine a.1>)
<valeur exclusive> ::= <identificateur>
<valeur non exclusive> ::= <identificateur>
<variable non exclusive> ::= <identificateur>
<relateur n.e.> ::= -U-|-I-
<valeur n.e.a.1> ::= <valeur n.e.a> |(<valeur n.e.a>)
<construction d'arborescence> ::= $ <registre 1> (C) :: <arborescence r>
<arborescence r> ::= <registre 1>(<origine a.>) <dépendants>
<dépendants> ::= |(<fonction 1><arborescence r> [, <fonction 1><arborescence r>]*)
<fonction 1> ::= |-R1-|-G1-|-D1-|-GF-|-DF-

2.3.2. ACTIONS NUMERIQUES

Les actions numériques consistent à affecter un registre numérique de C.

<action n> ::= <affectation n>
<affectation n> ::= <registre n> (C) := <valeur n>
<valeur n> ::= <valeur n1>|<valeur n1> + <valeur n>|<valeur n1> - <valeur n>
<valeur n1> ::= <valeur n2>|<valeur n2> * <valeur n1>|<valeur n2> : <valeur n1>
<valeur n2> ::= <nombre>|<valeur de condition 1>|<opérateur 1> (<valeur n>)
|<opérateur 2> (<valeur n> [,<valeur n>]*)| <valeur n r>| NBHOM (<origine a>)
<valeur de condition 1> ::= VAB (<exp. bool. de variables>)
<opérateur 1> ::= FACT| LG2
<opérateur 2> ::= MOY|MIN|MAX|PLUS
<valeur n.r> ::=<registre n> (<origine a>)

Ici, nombre est pris inférieur à $2^{32}-1$. D'autre part, on se donne la possibilité d'évaluer une condition linguistique et d'utiliser le résultat dans une expression numérique. NBHOM (X) est le nombre d'homographes de X distincts de X.

2.3.3. ACTIONS ET CONDITIONS GLOBALES

<action g> ::= &LIN := <registre 1> | &NUM := <registre n>| &GRU := <groupe>
<condition g> ::= &LIN <relateur e><registre 1>| &NUM <relateur e>< registre n>
|&GRU <relateur e> <groupe>

2.3.4. CONDITIONS LINGUISTIQUES

<condition 1> ::= <exp. bool. de variables>
<exp. bool. de variables> ::= <exp. bool. v1> |<exp. bool. v1> -OU- <exp. bool. de variables>
<exp. bool. v1> ::= <exp. bool. v2>|<exp. bool. v2> -ET- <exp. bool. v1>
<exp. bool. v2> ::= -N- <exp. bool. v2>|<relation propre>| (<exp. bool. de variables>)
|EXHOM (<origine a> ,<condition 1h>) TTHOM (<origine a>,<condition 1h>)
<relation propre> ::= <rel. propre e>|<rel. propre n.e>
<rel. propre e> ::= <valeur e.r> <relateur e> <valeur e.a>
<relateur e> ::= -E- | -NE-
<rel.propre n.e> ::= <valeur n.e.r> <relateur n.e> <valeur n.e.a>
<relateur n.e> ::= -E-|-NE-|-INC-|-DANS-|-NDANS- |-NINC-

Ces symboles sont mis pour =, ≠, ⊃, ⊂, ⊄, et -N- pour la négation. -N-, -ET- et -OU- ont des priorités décroissantes. <condition 1h> désigne une condition où on autorise de plus l'origine H (homographe), à tous les niveaux. EXHOM (o,c) s'interprète comme "il existe un homographe H (de l'origine o) tel que c", et TTHOM (o,c) comme "tous les homographes vérifient C".

2.3.5. CONDITIONS NUMERIQUES

<condition n> ::= <cond. n1>|<cond. n1> -OU- <condition n>
<cond. n1> ::= <cond. n2>|<cond. n2> -ET- <cond. n1>
<cond. n2> ::= -N- <cond. n2> | <prédicat n>| (<condition n>)
<prédicat n> ::= <valeur n.r> <relateur n> <valeur n>
<relateur n> ::= =|¬=|>|>=|<|<=

2.4. SOMMETS

Si un sommet est un sommet d'entrée, il a une étiquette composée d'un nom et d'un type, soit (N,t). Si t = *, c'est un sommet normal. Sinon, t=0, et on trouve d'autres sommets (N,t), t∈{1,2,...} , qui ne sont pas des entrées, mais caractérisent la fin d'une chaîne (partant de (N,o)) utilisée pour d'éventuelles "ruptions" (Cf. Ch. III). L'unique sommet final est étiqueté par &FIN. Tous les sommets n'apparaissent donc pas nécessairement.

<sommets> ::= -SOM- <sommet>*
<sommet> ::= <identificateur> : <étiquette> [/<actions 1c>[/<actions nc> [/<actions g>]]] .
<étiquette s> ::= &FIN |<nom a.g> , <type>
<type> ::= * |<nombre>

Le nombre est pris inférieur à 255. Les actions linguistiques, numériques et globales sont des initialisations : les origines P et T sont interdites.

III - MONIT

3.1. GENERALITES

On a vu au chapitre I que le moniteur d'un système hiérarchisé de "G-machines" était simulable par un automate d'états finis. Le système MONIT permettra donc d'écrire une classe de tels automates. Chacun d'eux pourra être interprété comme la description d'une certaine organisation d'un système de TA utilisant des composants prédéfinis ATEF, ALGOG, CETA, TRANS, SYGMOR.

La deuxième partie de la description est alors constituée par un réseau dont les arcs sont étiquetés par des tests et les sommets par des appels paramétrés et conditionnels des composants. Pour avoir plus de souplesse, on multiplie d'autre part les états de l'automate (jusqu'ici identiques aux sommets du réseau) en utilisant un certain nombre de masques de variables, pour un jeu de variables propre au moniteur. Les arcs et les sommets du réseau pourront alors comporter également des affectations et des conditions portant sur ces masques. L'heuristique d'analyse syntaxique, ou organisation du traitement ALGOG, apparaît donc comme une partie intégrante de la stratégie de traduction.

De façon externe, le moniteur sera défini par trois fichiers :

- déclaration des variables
- formats (états complexes)
- moniteur proprement dit

Cette séparation a seulement pour but de faciliter l'emploi des compilateurs de variables et de formats qui servent déjà pour ATEF et CETA. On ne donnera donc ici que la syntaxe du moniteur proprement dit. Notons que les conventions lexicales généralisent celles des autres composants :

- identificateurs plus longs (maximum 256 caractères)
- identificateurs pouvant commencer par &, !, @, ? ou %
- commentaires entre '**' et '.' ou entre deux '≠'.

<moniteur> ::= <bloc redéfinition> <bloc déclaration>[<bloc procédure>*<bloc arc>*<bloc sommet>*]* -FIN-

3.2. REDEFINITIONS

Cette partie sert à redéfinir les identificateurs prédéclarés.

<bloc redéfinition> ::= |-DEF- <redéfinition>*
<redéfinition> ::= <ancien identificateur> = <nouvel identificateur> .

A l'initialisation, on a :

<ancien identificateur> ::= &MODULE|&ORIGINE|&CHAINE|&NOMBRE
|MORPHO|ALGOG|ARBOR|TRANS|SYNTH|LIRE|ECRIRE|EXTRACT
|OK|FINTXT|X|C|P|T|D|G|MOTINC|INIT|&NUM|&LIN|&NOM|&GRU
|MOY|MAX|MIN|PLUS|VAB|FACT|LG2|NBFORM|NBSOM
|MORTE|STERILE|FGRAPHE|TROUV|MOUV|TROUVB|MOUVB|TROUVC|MOUVC
|MOR|DEBUT|CONST|SOEUR|C1|C2|FORME

On notera S(<ancien identificateur>) tout synonyme déclaré à ce niveau.
MØRPHØ, ARBØR et SYNTH sont les noms des fonctions associées à ATEF, CETA et SYGMØR. EXTRACT est la fonction d'extraction d'une arborescence (et non de construction de graphe) à partir du résultat de l'analyse par ATEF.

&NOMBRE désigne l'ensemble des variables numériques du moniteur.

&CHAINE désigne l'ensemble des variables chaînes.

&ORIGINE désigne l'ensemble des points de communication avec les modules (arborescences en général, analyses partielles pour ALGOG.

&MODULE indique quels modules sont utilisés pour quels codes langue.

On aura par exemple : &NUM = REGNUM.

3.3. DECLARATIONS

C'est dans ce bloc qu'on définit le contenu des 5 descripteurs précédents.

<bloc déclaration> ::= -DEC- <modules><origines>[<chaînes>][<nombres>]
<modules> ::= S(&MODULE) := (<liste d.m>).
<liste d.m> ::= <d.m>|<d.m>,<liste d.m>
<d.m> ::= <n.module>(<liste de codes>) |S(TRANS) (<liste p.c>)
<liste de codes> ::= <code>|<code>,<liste de codes>
<code> ::= <identificateur>
<n.module> ::= S(MORPHO)|S(ALGOG)|S(ARBOR)|S(SYNTH)
<liste p.c> ::= <p.c>|<p.c>,<liste p.c>
<p.c> ::= (<code>,<code>)

On aura par exemple :

&MODULE := (MORPHO(L1), ALGOG(L2), ARBOR(L3,L4,L5), TRANS((L3,L4),(L3,L5)), SYNTH(L4,L5)).

<origines> ::= S(&ORIGINE) := (<liste d.o>).
<liste d.o> ::= <d.o>|<d.o>, <liste d.o>
<d.o> ::= <code>(<liste d'origines>)
<liste d'origines> ::= <origine>|<origine>,<liste d'origines>

On aura par exemple :
&ORIGINE ::= (L1(&ATEF),L2(C,P,T,C1,C2),L3(A,B,C),L4(T),L5(U)).

C,P,T (ou leurs synonymes) sont les points distingués dans l'arbre des analyses (ALGOG).

<chaînes> ::= S(&CHAINE) := (<liste d.c>).
<liste d.c> ::= <d.c>|<d.c>, <liste d.c>)
<d.c> ::= <org. chaîne> [= <littéral>]
<nombres> ::= S(&NOMBRE) := (<liste d.n>).
<liste d.n> ::= <d.n>|<d.n> , <liste d.n>
<d.n> ::= <org. nombre> [= <nombre>]
<org. chaîne> ::= <identificateur>
<org. nombre> ::= <identificateur>

<nombre> est encore pris inférieur à 2^{32}-1. On aura par exemple :
&CHAINE := (&CH1 = '.' , &CH2,CH3 = '/CHAINE/').
&NOMBRE := (N1=3,N2,N3,N4=625).

3.4. PROCEDURES

Elles constituent les étiquettes des arcs et des sommets. Un bloc procédure permet de définir des actions conditionnelles indépendamment d'un arc ou d'un sommet. On peut également appeler une procédure précédemment définie dans une autre procédure. Les arcs et les sommets définissent implicitement des procédures portant leurs noms.

<bloc procédure> ::= -PROC- <définition de procédure>*
<définition de procédure> ::= <nom de procédure> : <procédure> .
<nom de procédure> ::= <identificateur>
<procédure> ::= <condition>|<proc. s.a>|<proc. a.a>

3.4.1. CONDITIONS

<conditions> ::= <cond. 1>|<cond. 1> -OU- <condition>
<cond. 1> ::= <cond. 2>|<cond. 2> -ET- <cond. 1>
<cond. 2> ::= -N- <cond. 2>|(<condition>)| <c. élém>
|$<nom de procédure>|/<nom d'arc>|!<nom de sommet>
<c. élém> ::= <c. variables>|<c. ATEF>|<c. ALGOG>|<c. chaîne>|<c. nombre>
<c. variables> ::= <rel. propre e>|<rel. propre n.e>
<relation propre> ::= <rel. propre e>|<rel. propre n.e>
<rel. propre e> ::= <valeur e><relateur e><valeur e>
<valeur e> ::= <valeur de variable e>|<nom de variable e> (<nom de format>)
<relateur e> ::= -E-|-NE-
<rel. propre n.e> ::= <valeur n.e><relateur n.e><valeur n.e>
<valeur n.e> ::=<valeur n.e.1>|<valeur n.e.1> -U- <valeur n.e>
<valeur n.e.1> ::= <valeur n.e.2>|<valeur n.e.2> -I- <valeur n.e.1>
<valeur n.e.2> ::= -N- <valeur n.e.2>|(<valeur n.e>)
|<valeur de variable n.e>|<nom de variable n.e> (<nom de format>)
<relateur n.e> ::= <relateur e>| -DANS-|-INC-|-NDANS-

Les noms et valeurs de variables, ainsi que les noms de format, sont fournis par les deux premiers fichiers externes. Les procédures "appelées" (par $, / ou !) doivent être des conditions (forme -S(SI)- <condition>).

<c. ATEF> ::= S(MORPHO) = <r. ATEF>
<r. ATEF> ::= S(OK)|S(FINTXT)|S(MOTING)|S(INIT)
<c. ALGOG> ::= S(ALGOG) = <r. ALGOG>|S(&NOM)<relateur n.e> S(&GRU)
<r. ALGOG> ::= S(OK)| S(MORTE)| S(STERILE)| S(FGRAPHE)
<c. nombre> ::= <valeur n.a><relateur n><valeur n.a>
<valeur n.a> ::= <val. n.a. 1>|<val. n.a 1>+<valeur n.a>
<valeur n.a. 1> ::= <val. n.a. 2>|<val. n.a. 2> - <val. n.a. 1>
<val. n.a. 2> ::= <nombre>|<org. nombre>| S(&NUM) (<org. ALGOG>) |S(NBFORM)|S(NBSOM)
|(<valeur n.a>)| S(VAB) (<condition>)|<opérateur 1>(<valeur n.a>)
|<opérateur 2>(<valeur n.a>[,<valeur n.a>]*)
<opérateur 1> ::= S(FACT) |S(LG2)
<opérateur 2> ::= S(MOY)|S(MIN)|S(MAX)|S(PLUS)
<relateur n> ::= =|¬=|>|>=|<|<=
<c. chaîne> ::= <not. chaîne><relateur e><not. chaîne>
<not. chaîne> ::= <org. chaîne>|<littéral>|S(FORME) (<org. ALGOG>)
<not. nombre> ::= <org. nombre>|<nombre>

3.4.2. PROCEDURES SANS APPEL

Rappelons que les procédures avec appel de modules ne peuvent être portées que par les sommets.

<proc. s.a> ::= -SI- <condition> -ALORS- <liste affcom> -SINON- <liste affcom>
|-FAIRE- <liste affcom>
<liste affcom> ::= <affcom>[,<affcom>]*
<affcom> ::= <affectation m>|<comm>|<appel p>
<comm> ::= S(LIRE)|S(ECRIRE) (<not. chaîne>)
<appel p> ::= $<nom de procédure>|/<nom d'arc>|!<nom de sommet>
<affectation m> ::= <aff. n.m>|<aff. c.m>|<aff. v.e.m>|<aff. v.n.m>
<aff n.m> ::= <org. nombre> := <valeur n.a>
<aff. c.m> ::= <org. chaîne> := <not. chaîne>
<aff. v.e.m> ::= <nom de variable e> (<nom de format>) := <valeur e >
<aff. v.n.m> ::= <nom de variable n.e>(<nom de format>):=<valeur n.e>

Dans le cas où <affcom> est un appel d'une procédure portée par un sommet, il faut bien sûr que celle-ci soit sans appel.

3.4.3. PROCEDURES AVEC APPEL

<proc. a.a> ::= -SI- <condition> -ALØRS- <liste appaffcom> [-SINØN- <liste appaffcom>]
|-FAIRE- <liste appaffcom>
<liste appaffcom> ::= <appaffcom> [, <appaffcom>]*
<appaffcom> ::= <appel m>|<affcom>
<appel m> ::= <appel ATEF>|<appel ALGOG>|<appel CETA>|<appel TRANS>|<appel SYGMOR>
<appel ATEF> ::= <a. analyse>|<a. extraction>
<a. analyse> ::= S(MORPHO) (<born form>,<mode a>,<nb détail>,<sort graph>)
<born form> ::= * |<not. nombre>|<not. chaîne>|- S(INIT) -
<mode a> ::= 1|2|3|<org. chaîne>
<nb détail> ::= * <not nombre>
<sort graph> ::= <mode de sortie> [N]|S
<mode de sortie> ::= I|T|D
<a. extraction> ::= <origine ATEF> := S(MORPHO) (S(EXTRACT))

<born form> donne le nombre de formes à analyser (* si c'est tout le texte), ou jusqu'à quelle occurrence analyser, ou encore indique que l'analyse doit s'arrêter au premier "INIT" (fonction d'initialisation de l'automate, qui entraîne un découpage, interprété comme la séparation entre deux phrases).

<mode a> indique l'interaction possible avec l'utilisateur si la forme n'est pas reconnue : correction de la forme et/ou indexation dans les dictionnaires (1), correction seule (2), ou rien (3) par défaut.

<nb détail> est le nombre des (dernières) formes analysées pour lesquelles il faudra imprimer le graphe construit, après l'analyse.

<mode détail> indique l'unité de sortie (terminal, imprimante, les deux) et l'absence éventuelle (N) du développement des masques de variables.

<appel ALGOG> ::= S(ALGOG) (<fonc par ALGOG>)
<fonc par ALGOG> ::= <depl>|<constr>|<arrêt>|<début>
<depl> ::= <f depl> , <sens>[,<condition>]|<org. ALGOG> := <org. ALGOG>
<f depl> := S(TROUV)| S(MOUV)|S(TROUVC)|S(MOUVC)|S(TROUVB)|S(MOUVB)
<sens> ::= S(G)|S(D)|<not. chaîne>
<constr> ::= <f constr>, <opt const>
<f constr> ::= S(CONST)|S(SOEUR)
<opt constr> ::= S(C1)|S(C2)|<org. chaîne>

Dans <sens> et <opt const> , la première valeur est prise par défaut (si <not. chaîne> ne vaut aucune des deux premières).

<arrêt> ::= S(MORT) , <org. ALGOG>
<début> ::= S(DEBUT) , <not. sommet> , S(&GRU)
<not. sommet> ::= <org. ALGOG>|<not. nombre>

Le <nombre> désignera le numéro du sommet choisi. <début> initialise des calculs pour tous les noms (ALGOG) de &GRU.

<appel CETA> ::= S(ARBOR) (<org. CETA> [, <num gram>])

<num gram> est un nombre inférieur à 32 et désigne le numéro de la sous-grammaire CETA (du code langue correspondant à l'arborescence <org. CETA>) à appliquer, et vaut 0 par défaut (toute la grammaire).

<appel TRANS> ::= <org. arbre> := S(TRANS) (<origine>)
<org. arbre> ::= <org. CETA>|<org. SYGMOR>
<appel SYGMOR> ::= S(SYNTH) (<org. SYGMOR>)

3.5. ARCS ET SOMMETS

<bloc arc> ::= -ARC- <définition d'arc>*
<définition d'arc> ::= <nom d'arc>:< nom de source> - <nom de but> == <proc. s.a> .
<bloc sommet> ::= -SOM- <définition de sommet>*
<définition de sommet> ::= <nom de sommet> : <proc. a.a> .

TABLE BIBLIOGRAPHIQUE

[1] Aho & Ullman (1972)
[2] Chauché (1974b)
[3] Chauché (1975a)
[4] Chauché (1975b)
[5] Chauché & al (1973)
[6] Kulagina & Mel'tchuk (1971)
[7] Quézel-Ambrunaz & Guillaume (1973)
[8] Thouin (1975)
[9] Tseitin & al (1966)
[10] Vauquois (1975)
[11] Veillon (1970)
[12] Woods (1970)

ANNEXE

Les sorties au terminal se trouvent sur les pages de gauche et les sorties correspondantes à l'imprimante sur les pages de droite, en regard si possible. Il est évident que l'exemple présenté n'est que démonstratif et ne prétend à aucune valeur linguistique.

```
geta
** A (194) READ-ONLY **

A.T.E.F. ET C.E.T.A. A VOTRE SERVICE ....

R; T=0.10/0.24 09:32:07

traigen
REPONDEZ PAR 'OUI' OU PAR 'NON' SI RIEN N EST SPECIFIE
POUR ARRETER LE TRAITEM. NOTEZ 'ARRET' EN REPONSE A UNE QUESTION

NOTEZ LA LANGUE SOURCE D ESSAI ( 3 CARAC TERES MAX. )
eg

INDIQUEZ LE TRAITEM. PAR :
DICMOR , DICET1 , DICTRAN , DICET2
MORPHO , CETA1 , TRANSF , CETA2
COMPGEN , LISGEN , DESTRUC , DUPLG , DET (DETAIL DES C.DES)
morpho

INDIQUEZ LE TRAITEM. PAR :
EXECUT , MODIF , CREAT , LISTE , ELIM , LG , TRAIT ,
COMPIL , INIT OU DET ( DETAIL DE CES COMM.DES )
liste
*VOULEZ VOUS LA LISTE DES NOMS DE TOUS VOS FICHIERS ?
non
DONNEZ LE NUMERO DU TEXTE A LISTER : X T/I
T OU I SI LISTE SUR TERMINAL OU IMPRIMAN TE
OU : FIN SI VOUS AVEZ TERMINE
2 t

LE PETIT LIT LE LIVRE

DONNEZ LE NUMERO DU TEXTE A LISTER : X T/I
T OU I SI LISTE SUR TERMINAL OU IMPRIMAN TE
OU : FIN SI VOUS AVEZ TERMINE
3 t

LE MECHANT PETIT MECHANT PETIT LIT LE LIVRE

DONNEZ LE NUMERO DU TEXTE A LISTER : X T/I
T OU I SI LISTE SUR TERMINAL OU IMPRIMAN TE
OU : FIN SI VOUS AVEZ TERMINE
fin

INDIQUEZ LE TRAITEM. PAR :
EXECUT , MODIF , CREAT , LISTE , ELIM , LG , TRAIT ,
COMPIL , INIT OU DET ( DETAIL DE CES COMM.DES )
execut
DONNEZ LE NUMERO DU TEXTE ?
3

PARAME TRES POUR CHAQUE PHASE DE TRAITEM. --- NOTEZ ---
1 1; 1 2 XY; 2 1 X; 2 2 XY --(Y POSSIBLE SI 'I')--
X = T; I; TI; TR; IR; TIR; TQ; IQ; TIQ
Y = A OU 'BLANC' -- NOTEZ 'DET' (DETAIL DE CES CM.DES)

PARAME TRES POUR LA PHASE 'MORPHO'
2 2 tl
EXECUTION BEGINS...
EXECUTION BEGINS...
EXECUTION BEGINS
```

Liste des fichiers de cette langue (p. 143)

Texte 2 (pas de mot inconnu)

Texte 3 ("méchant" = mot inconnu)

```
                    *** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE***
                                                                    PAGE   1
DETAIL DE L'EXECUTION , TEXTE : 3          SYSTEME A.T.E.F.
                                                                    CODE LANGUE : EG


  *** OCCURRENCE : LE
MORPHE : LE                          REGLE :  RARTI    FORMATS :ARTI     SG
   --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE :  RPRON    FORMATS :PRON     SG
   --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

  *** OCCURRENCE : MECHANT
  MOT INCONNU
MORPHE :  MECHANT                    REGLE :  MOTINC   FORMATS :MODINC   MODINC
   --> DECOUPAGE : MECHAN MECHANT: UL(MECHANT).

  *** OCCURRENCE : PETIT
MORPHE : PETIT                       REGLE :  RADJS    FORMATS :ADJS     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE :  RADJE    FORMATS :ADJE     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

  *** OCCURRENCE : MECHANT
  MOT INCONNU
MORPHE :  MECHANT                    REGLE :  MOTINC   FORMATS :MODINC   MODINC
   --> DECOUPAGE : MECHAN MECHANT: UL(MECHANT).

  *** OCCURRENCE : PETIT
MORPHE : PETIT                       REGLE :  RADJS    FORMATS :ADJS     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE :  RADJE    FORMATS :ADJE     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

  *** OCCURRENCE : LIT
MORPHE : LIT                         REGLE :  RVERB    FORMATS :VERB     V3
   --> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT                         REGLE :  RSUBS    FORMATS :SUBS     SG
   --> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

  *** OCCURRENCE : LE
MORPHE : LE                          REGLE :  RARTI    FORMATS :ARTI     SG
   --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE :  RPRON    FORMATS :PRON     SG
   --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

  *** OCCURRENCE : LIVRE
MORPHE : LIVRE                       REGLE :  RVERB    FORMATS :VERB     V3
   --> DECOUPAGE : LIVRE LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE                       REGLE :  RSUBS    FORMATS :SUBS     SG
   --> DECOUPAGE : LIVRE LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

```
                    *** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE***
                                                                    TV-18   PAGE   1
                                    SYSTEME A.T.E.F.
DETAIL DE L'EXECUTION , TEXTE : 3                                   CODE LANGUE : EG


*** OCCURRENCE : LE
MORPHE : LE                         REGLE :  RARTI    FORMATS :ARTI     SG
  --> DECOUPAGE : LE LE: LL(LE),CAT(ART),NBR(SNG),K(FORME)
MORPHE : LE                         REGLE :  RPRON    FORMATS :PRON     SG
  --> DECOUPAGE : LE LE: LL(IL),CAT(PRO),NBR(SNG),K(FORME)

*** OCCURRENCE : MECHANT
MOT INCONNU
MORPHE :  MECHANT                   REGLE :  MOTINC   FORMATS :MOTINC   MOTINC
  --> DECOUPAGE : MECHAN MECHANT: LL(MECHANT).

*** OCCURRENCE : PETIT
MORPHE : PETIT                      REGLE :  RADJS    FORMATS :ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)
MORPHE : PETIT                      REGLE :  RADJE    FORMATS :ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)

*** OCCURRENCE : MECHANT
MOT INCONNU
MORPHE :  MECHANT                   REGLE :  MOTINC   FORMATS :MOTINC   MOTINC
  --> DECOUPAGE : MECHAN MECHANT: LL(MECHANT).

*** OCCURRENCE : PETIT
MORPHE : PETIT                      REGLE :  RADJS    FORMATS :ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)
MORPHE : PETIT                      REGLE :  RADJE    FORMATS :ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE : LIT
MORPHE : LIT                        REGLE :  RVERB    FORMATS :VERB     VB
  --> DECOUPAGE : LIT LIT: LL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT                        REGLE :  RSUBS    FORMATS :SUBS     SG
  --> DECOUPAGE : LIT LIT: LL(LIT),CAT(NOM),NBR(SNG),K(FORME)

*** OCCURRENCE : LE
MORPHE : LE                         REGLE :  RARTI    FORMATS :ARTI     SG
  --> DECOUPAGE : LE LE: LL(LE),CAT(ART),NBR(SNG),K(FORME)
MORPHE : LE                         REGLE :  RPRON    FORMATS :PRON     SG
  --> DECOUPAGE : LE LE: LL(IL),CAT(PRO),NBR(SNG),K(FORME)

*** OCCURRENCE : LIVRE
MORPHE : LIVRE                      REGLE :  RVERB    FORMATS :VERB     VB
  --> DECOUPAGE : LIVRE LIVRE: LL(LIVRER),CAT(VRB),TPS(FRE),PRS(3),K(FORME),
MORPHE : LIVRE                      REGLE :  RSUBS    FORMATS :SUBS     SG
  --> DECOUPAGE : LIVRE LIVRE: LL(LIVRE),CAT(NOM),NBR(SNG),K(FORME)
```

```
                    *** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
                                                                    PAGE    1
                                    SYSTEME A.T.E.F
RESULTAT DE L'EXECUTION , TEXTE :3                                  CODE LANGUE : EG

     .1 ULTXT (.2.ULERA (.3.ULCCC (.4.LE ,  5 IL) , .6 ULCCC (.7 MECHANT) ,  8 ULCCC ( 9 PETIT ,  10 PETIT) ,  11
     ULCCC (.12.      ME) , .13.ULCCC (.14.PETIT , .15.PETIT) , .16.ULCCC (.17.LIRE , .18.LIT) ,  19 ULCCC ( 20 LE ,
     .21.IL) ,  22.ULCCC (.23.LIVRER , .24.LIVRE)))

          SOMMET 1    : LL(ULTXT),

          SOMMET 2    : LL(ULERA),

          SOMMET 3    : LL(ULCCC),

          SOMMET 4   LE: LL(LE),CAT(ART),NBR(SNG),K(FORME)

          SOMMET 5   LE: LL(IL),CAT(PRO),NBR(SNG),K(FORME)

          SOMMET 6    : LL(ULCCC),

          SOMMET 7   MECHANT: LL(MECHANT),

          SOMMET 8    : LL(ULCCC),

          SOMMET 9   PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)

          SOMMET 10   PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)

          SOMMET 11    : LL(ULCCC),

          SOMMET 12   MECHANT: LL(MECHANT)

          SOMMET 13    : LL(ULCCC),

          SOMMET 14   PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

          SOMMET 15   PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME)

          SOMMET 16    : LL(ULCCC),

          SOMMET 17   LIT: LL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME)

          SOMMET 18   LIT: LL(LIT),CAT(NOM),NBR(SNG),K(FORME)

          SOMMET 19    : LL(ULCCC),

          SOMMET 20   LE: LL(LE),CAT(ART),NBR(SNG),K(FORME)

          SOMMET 21   LE: LL(IL),CAT(PRO),NBR(SNG),K(FORME)

          SOMMET 22    : LL(ULCCC),

          SOMMET 23   LIVRE: LL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME)

          SOMMET 24   LIVRE: LL(LIVRE),CAT(NOM),NBR(SNG),K(FORME)
```

RESULTAT DE L'EXECUTION , TEXTE :3 CODE LANGUE EG

```
 .1.ULTXT (.2.ULFRA (.3.ULOCC (.4.LE , .5.IL) , .6.ULOCC (.7.MECHANT) , .8.ULOCC (.9.PETIT , .10.PETIT) , .11.
ULOCC (.12.     ME) , .13.ULOCC (.14.PETIT , .15.PETIT) , .16.ULOCC (.17.LIRE , .18.LIT) , .19.ULOCC (.20.LE ,
 .21.IL) , .22.ULOCC (.23.LIVRER , .24.LIVRE)))

     SOMMET 1   : UL(ULTXT).

     SOMMET 2   : UL(ULFRA).

     SOMMET 3   : UL(ULOCC).

     SOMMET 4  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     SOMMET 5  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     SOMMET 6   : UL(ULOCC).

     SOMMET 7  MECHANT: UL(MECHANT).

     SOMMET 8   : UL(ULOCC).

     SOMMET 9  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 10  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 11   : UL(ULOCC).

     SOMMET 12  MECHANT: UL(MECHANT).

     SOMMET 13   : UL(ULOCC).

     SOMMET 14  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 15  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 16   : UL(ULOCC).

     SOMMET 17  LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

     SOMMET 18  LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

     SOMMET 19   : UL(ULOCC).

     SOMMET 20  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     SOMMET 21  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     SOMMET 22   : UL(ULOCC).

     SOMMET 23  LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

     SOMMET 24  LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).


TRAITEM. ' MORPHO. ' TERMINE POUR CE TEXTE

VOULEZ VOUS MODIFIER LE TEXTE D ENTREE ?
arret
TRAITE MENT TERMINE
R; T=5.40/9.28 09:44:06
```

Version interactive d'ATEF ("vprime")

```
LOGGED
ENTER PASSWORD:
********
READY AT 10:10:04 ON WEDNESDAY 02/25/76
CP
i cms
CMS..VERSION 3.0 octobre 74

vprime

** A (194) READ-ONLY **
** B (195) READ-ONLY **

VERSION PRIME A VOTRE SERVICE ...

R; T=0.31/0.67 10:10:24

tralgen
REPONDEZ PAR 'OUI' OU PAR 'NON' SI RIEN N'EST SPECIFIE ,
PAR '?' POUR OBTENIR DES PRECISIONS
ET PAR 'ARRET' POUR ARRETER LE TRAITEMENT

NOTEZ LA LANGUE SOURCE D'ESSAI ( 3 CARACTERES MAXIMUM )
eg

TRAITEMENT GENERAL?
?
INDIQUEZ LE TRAITEMENT PAR :
DICMOR , DICET1 , DICTRAN , DICET2
MORPHO , CETA1 , TRANSF , CETA2
COMPGEN , LISGEN , DESTRUC , DUPLG , DET (DETAIL DES C.DES)
lisgen
LISTE DES FICHIERS DEPUIS LA MORPHOLOGIE (IMPRIMANTE)
DERNIERE PHASE DES FICHIERS A LISTER ?
?
REPONDEZ PAR : DICMOR , DICET1 , DICTRAN , DICET2
dicmor
LISTE SEULEMENT A L'IMPRIMANTE
VOULEZ-VOUS AUSSI LA LISTE DU (DES) DICTIONNAIRES ?
oui
VOULEZ-VOUS OBTENIR LES LISTES TRIEES ?
POUR LE(S) DICTIONNAIRE(S) TRI 'ALFA' SEULEMENT
non
EXECUTION BEGINS...
EXECUTION BEGINS...
EXECUTION BEGINS...
EXECUTION BEGINS...
EXECUTION BEGINS...
*EXECUTION BEGINS...
*LISTES TERMINEES
AUTRE TRAITEMENT ? OUI OU NON
oui

NOTEZ LA LANGUE SOURCE D'ESSAI ( 3 CARACTERES MAXIMUM )
eg

TRAITEMENT GENERAL?
morpho
*
TRAITEMENT EXECUTION?
?

EXECUT , MODIF , CREAT , LISTE , ELIM , LG , TRAIT ,
COMPIL , INIT , AUXI OU DET ( DETAIL DE CES COMMANDES )
det

EXECUT.. EXECUTION IMMEDIATE
MODIF... MODIFS SUR UN TEXTE PUIS EXECUTION
CREAT... CREATION D'UN TEXTE PUIS EXECUTION
LISTE... LISTE DE TEXTES
ELIM.... EFFACEMENT DE TEXTES
LG..... CHANGEMENT DE LANGUE
TRAIT... CHANGEMENT DE PHASE DE TRAITEMENT
COMPIL.. DEMANDE DE COMPILATION
INIT.... REPRISE EN DEBUT DE TRAITEMENT
AUXI.... SAUVEGARDE DES DICTIONNAIRES AUXILIAIRES

liste
VOULEZ-VOUS LA LISTE DES NOMS DE TOUS VOS TEXTES ?
non
X T/I OU FIN ?
?
DONNEZ LE NUMERO DU TEXTE A LISTER : X ET
T OU I SI)LISTE SUR TERMINAL OU IMPRIMANTE
OU : FIN SI VOUS AVEZ TERMINE
2 t

LE PETIT LIT LE LIVRE

X T/I OU FIN ?
3 t

LE MECHANT PETIT MECHANT PETIT LIT LE LIVRE

X T/I OU FIN ?
fin

TRAITEMENT EXECUTION?
execut
DONNEZ LE NUMERO DU TEXTE
3

MODE INTERACTIF? (NON, X Y OU DET)
?
AVEC X = 1, 2, OU 3, Y = G(I), GS(I), OU N; DET POUR LE DETAIL
det
X : TRAITEMENT DES MOTS INCONNUS
X = 1 : DICTIONNAIRE AUXILIAIRE ET CHGT DE FORME
X = 2 : CHANGEMENT DE FORME
X = 3 : PAS DE TRAITEMENT SPECIAL
Y : SORTIE EN GRAPHE DE CHAINES
Y = G(I) : AVEC GRAPHE DE CHAINES ET ARBORESCENCE
Y = GS(I) : AVEC GRAPHE DE CHAINES SEUL
Y = N : AVEC ARBORESCENCE SEULE
"I" PERMET D'INVERSER LES CHAINAGES INTERNES
1 g
PARAMETRES : 1 1; 1 2 XY; 2 1 X; 2 2 XY

PARAMETRES POUR LA PHASE 'MORPHO' ?
?
PARAMETRES POUR CHAQUE PHASE DE TRAITEMENT: NOTER
1 1; 1 2 XY; 2 1 X; 2 2 XY --(Y POSSIBLE SI 'I')--
X = T; I; TI; TR; IR; TIR; TQ; IQ; TIQ
Y = A OU 'BLANC' -- NOTEZ 'DET' (DETAIL DE CES CM.DES)

PARAMETRES POUR LA PHASE 'MORPHO' ?
? ? ri
```

① Liste des fichiers de la langue eg

Texte 2 (sans mot inconnu)

Texte 3 ("méchant" = mot inconnu)

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

① Liste des fichiers

FICHIER : FILEMEG VARBM — 25 FEVRIER 1976 — 10H 12MN 18S — PAGE 1

Variables morphologiques

```
   -EXC-
CAT := (ART.ADJ.ADS.NUM.VRB.PRO.PRP.INV).
   -NEX-
CICT := (1).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

FICHIER : FILEMEG FORMATM — 25 FEVRIER 1976 — 10H 12MN 20S — PAGE 1

Formats morphologiques

(servent à adresser les règles)

```
ADJE    01==. ** CAT -E- ADJ.
ADJS    01==. ** CAT -E- ADS.
ARTI    01==. ** CAT -E- ART.
INVA    01==. ** CAT -E- INV.
MODINC  01==. ** .
PREP    01==. ** CAT -E- PRP.
PRON    01==. ** CAT -E- PRO.
SUBS    01==. ** CAT -E- NOM.
VERE    01==. ** CAT -E- VRB.
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

FICHIER : FILEMEG VARBS — 25 FEVRIER 1976 — 10H 12MN 21S — PAGE 1

Variables syntaxiques

```
   -EXC-
NBR := (SNG.PLU).
TPS := (PAS.PRE.FUT).
PRS := (1.2.3.4.5.6).
   -NEX-
K := (FORME).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

FICHIER : FILEMEG FORMATS — 25 FEVRIER 1976 — 10H 12MN 23S — PAGE 1

Formats syntaxiques

```
MODINC  01==. ** .
SG      01==. ** NBR -E- SNG. K -E- FORME.
V3      01==. ** PRS -E- 3. TPS -E- PRE. K -E- FORME.
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

FICHIER : FILEMEG GRAMR — 25 FEVRIER 1976 — 10H 12MN 29S — PAGE 1

Grammaire

```
GPRON   :    PRON == VAR(C) := VAR(A)
        ///CAT(S) -E- VRB.

GARTI   :    ARTI == VAR(C) := VAR(A)
        ///(CAT(S) -E- ADJ -OU- CAT(S) -E- ADS -OU- CAT(S) -E- NOM)
        -ET- CAT(PL) -NE- NOM.

GADJE   :    ADJE == VAR(C) := VAR(A)
        ///CAT(S) -E- NOM.

RSUBS   :    SUBS-PREP-INVA == VAR(C) := VAR(A).-SOL-.

RVERB   :    VERB == VAR(C) := VAR(A)
        ///CAT(S) -NE- PRO.

RADJS   :    ADJS == VAR(C) := VAR(A)
        ///CAT(S) -NE- NOM.

POTINC  :    MODINC == .
RDICT   :    (1/NU).

        -FIN-
```

On n'a de pronom qu'avant un verbe

On n'a d'article qu'avant un adjectif ou un nom, et pas après un nom.

Pas de verbe avant un pronom.

Pas d'adjectif substantivé avant un nom.

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

FICHIER : FILEMEG DICB1 — 25 FEVRIER 1976 — 10H 12MN 26S — PAGE 1

Dictionnaire

```
CHAISE          ==SUBS  (SG     .CHAISE     ).
GRAND           ==ADJE  (SG     .GRAND      ).
GRAND           ==ADJS  (SG     .GRAND      ).
LA              ==ARTI  (SG     .ELLE       ).
LA              ==PRON  (SG     .ELLE       ).
LE              ==ARTI  (SG     .LE         ).
LE              ==PRON  (SG     .IL         ).
LIT             ==SUBS  (SG     .LIT        ).
LIT             ==VERB  (V3     .LIRE       ).
LIVRE           ==SUBS  (SG     .LIVRE      ).
LIVRE           ==VERB  (V3     .LIVRER     ).
MAUVAIS         ==ADJE  (SG     .MAL        ).
MAUVAIS         ==ADJS  (SG     .MAL        ).
PETIT           ==ADJE  (SG     .PETIT      ).
PETIT           ==ADJS  (SG     .PETIT      ).
TABLE           ==SUBS  (SG     .TABLE      ).
TABLE           ==VERB  (V3     .TABLER     ).
```

```
EXECUTION BEGINS...
EXECUTION BEGINS...

*** PARAMETRES INITIAUX ***
PHI?
>?
PHI : nombre de formes a analyser ( * si tout le reste), ou 'X' si X = forme limite, ou I si arret sur -INIT-
>'petit'
MU?
>?
MU : traitement du mot inconnu. MU = 1 pour changer les formes et completer les dictionnaires,
MU = 2 pour seulement changer les formes, MU = 3 sinon.
>1
GAMMA?
>?
GAMMA : sortie du graphe - I(N), T(N) (n sans masque), ou S sans sortie
>d
SIGMA?
>?
SIGMA : nombre de formes a lire (* si tout)
>*
PLACEZ VOTRE PAPIER ET FAITES UN RETOUR CHARIOT
>
```

d = i+t (Les E/S sont centralisées dans ATEF', et leurs paramètres légèrement différents)

```
                                *** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***
                                        SYSTEME A.T.E.F.                25/02/76 @ 10:18:34   PAGE 1
DETAIL DE L'EXECUTION , TEXTE : 3                                                    CODE LANGUE : EG

*** OCCURRENCE 1 : LE
MORPHE : LE                             REGLE : RARTI     FORMATS : ARTI     SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                             REGLE : RPRON     FORMATS : PRON     SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 2 : MECHANT
MOT INCONNU
MOT INCONNU : MECHANT
G, C ,I, F ou ARRET?
>?
Tapez C pour continuer, I pour indexer, F pour lire ou changer la forme analysee, ARRET pour arreter
et G pour examiner le graphe
>f
FORME INCONNUE:     MECHANT
LA CHANGER?
>oui
NOUVELLE FORME?
>mauvais
MAUVAIS
FORME CORRECTE?
>oui
NOUVELLE INDEXATION?
>
SUITE DU TRAITEMENT?
>
G, C ,I, F ou ARRET?
>

*** OCCURRENCE 2 : MAUVAIS
MORPHE : MAUVAIS                        REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MAUVAIS MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).
MORPHE : MAUVAIS                        REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : MAUVAIS MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** BORNE ATTEINTE *** : PETIT

*** GRAPHE ***

     -0- (1,2)    <ORIGINE>

     -1- (3,4)    LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -2- ()    LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     -3- ()    MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

     -4- ()    MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

GRAPHE?
>min@@on
```

```
INDEXAGE?
>oui
NOUVELLE INDEXATION?
>oui
NUMERO DU DICTIONNAIRE?
>12
12?
NUMERO DU DICTIONNAIRE?
>2
DICTIONNAIRE INEXISTANT
AUTRE DICTIONNAIRE?
>oui
NUMERO DU DICTIONNAIRE?
>1
ARTICLE?
>mechant==adjs(mal
FORMAT SYNTAXIQUE INCORRECT
ARTICLE?
>mechant== adjs(sg,  mal)
MECHANT                ==ADJS    (SG       ,MAL                  ).
ENREGISTRER?
>oui
ARTICLE?
>mechant ==   adjs  (sg,mechant
MECHANT                ==ADJS    (SG       ,MECHANT              ).
ENREGISTRER?
>oui
ARTICLE?
>
AUTRE DICTIONNAIRE?
>
SUITE DU TRAITEMENT?
>
PHI?
>2
MU?
>
GAMMA?
>tn
SIGMA?
>3

 *** OCCURRENCE 3 : PETIT                                                    (4)
 MORPHE : PETIT                        REGLE : RADJS     FORMATS : ADJS     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
 MORPHE : PETIT                        REGLE : RADJE     FORMATS : ADJE     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

 *** OCCURRENCE 4 : MECHANT
 MORPHE : MECHANT                      REGLE : RADJS     FORMATS : ADJS     SG
   --> DECOUPAGE : MECHANT MECHANT: UL(MECHANT),CAT(ADS),NBR(SNG),K(FORME).
 MORPHE : MECHANT                      REGLE : RADJS     FORMATS : ADJS     SG
   --> DECOUPAGE : MECHANT MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

*** BORNE ATTEINTE *** : PETIT

*** GRAPHE ***
```



```
    -3- (5,6)    MAUVAIS

    -4- ()    MAUVAIS

    -5- (7,8)    PETIT

    -6- ()    PETIT

    -7- ()    MECHANT

    -8- ()    MECHANT

GRAPHE?
>
INDEXAGE?
>oui
NOUVELLE INDEXATION?
>oui
NUMERO DU DICTIONNAIRE?
>1
ARTICLE?
>petit = = adjs ( sg, petiot
?
ARTICLE?
>petit == adjs(sg,petiot
PETIT                  ==ADJS    (SG       ,PETIOT               ).
ENREGISTRER?
>oui
ARTICLE?
>petit == adje(sg,petiot
PETIT                  ==ADJE    (SG       ,PETIOT               ).
ENREGISTRER?
>oui
ARTICLE?
>
AUTRE DICTIONNAIRE?
>
SUITE DU TRAITEMENT?
>
PHI?
>3
MU?
>2
GAMMA?
>1
SIGMA?
>3

 *** OCCURRENCE 5 : PETIT                                                    (5)
 MORPHE : PETIT                        REGLE : RADJS     FORMATS : ADJS     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
 MORPHE : PETIT                        REGLE : RADJE     FORMATS : ADJE     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
 MORPHE : PETIT                        REGLE : RADJE     FORMATS : ADJE     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).
 MORPHE : PETIT                        REGLE : RADJS     FORMATS : ADJS     SG
   --> DECOUPAGE : PETIT PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).
```

```
*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

                                SYSTEME A.T.E.F.                    25/02/76 à 10:16:34   PAGE 1

DETAIL DE L'EXECUTION , TEXTE : 3                                   CODE LANGUE : EG

*** OCCURRENCE 1 : LE
MORPHE : LE                          REGLE : RART1     FORMATS : ART1     SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE : RPRON     FORMATS : PRON     SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 2 : MECHANT
MOT INCONNU

*** OCCURRENCE 2 : MAUVAIS
MORPHE : MAUVAIS                     REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MAUVAIS MALVAIS: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
MORPHE : MAUVAIS                     REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : MAUVAIS MALVAIS: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** GRAPHE ***

     -0- (1,2)    <ORIGINE>

     -1- (3,4)    LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -2- ()    LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     -3- ()    MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

     -4- ()    MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 3 : PETIT
MORPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 4 : MECHANT
MORPHE : MECHANT                     REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MECHANT MECHANT: LL(MECHANT),CAT(ADJ),NBR(SNG),K(FORME).
MORPHE : MECHANT                     REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MECHANT MECHANT: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 5 : PETIT
MORPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
```

② φ = 'petit'

Changement de forme

③ $(\gamma = d, \sigma = *)$

Indexage de "méchant"

④ $\varphi = 2$

"méchant" normalement analysé

Indexage de "petit"

⑤ $\varphi = 3$

```
*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

                                SYSTEME A.T.E.F.                    25/02/76 à 10:30:12   PAGE 2

DETAIL DE L'EXECUTION , TEXTE : 3                                   CODE LANGUE : EG

MORPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 6 : LIT
MORPHE : LIT                         REGLE : RVERB     FORMATS : VERB     V3
  --> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),IPS(PRE),PRS(3),K(FORME).
MORPHE : LIT                         REGLE : RSUBS     FORMATS : SUBS     SG
  --> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

*** OCCURRENCE 7 : LE
MORPHE : LE                          REGLE : RART1     FORMATS : ART1     SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE : RPRON     FORMATS : PRON     SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** GRAPHE ***

     -9- (13)    PETIT
          SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     -10- (14)    PETIT
          SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     -11- (14)    PETIT
          SOLUTION 1    PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

     -12- (13)    PETIT
          SOLUTION 1    PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

     -13- (15)    LIT
          SOLUTION 1    LIT: UL(LIRE),CAT(VRB),IPS(PRE),PRS(3),K(FORME).

     -14- (15,16)    LIT
          SOLUTION 1    LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

     -15- ()    LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -16- ()    LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 8 : LIVRE
MORPHE : LIVRE                       REGLE : RVERB     FORMATS : VERB     V3
  --> DECOUPAGE : LIVRE LIVRE: LL(LIVRER),CAT(VRB),IPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE                       REGLE : RSUBS     FORMATS : SUBS     SG
  --> DECOUPAGE : LIVRE LIVRE: LL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

*** GRAPHE ***
```

$(\gamma = i, \sigma = 3)$

⑥ $\varphi = *$

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

1+2

*** OCCURRENCE 6 : LIT
MORPHE : LIT REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

*** OCCURRENCE 7 : LE
MORPHE : LE REGLE : RARTI FORMATS : ARTI SG
--> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE REGLE : RPRON FORMATS : PRON SG
--> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

⑥

*** BORNE ATTEINTE *** : LIVRE
GRAPHE?
>
PHI?
>
MU?
>
GAMMA?
>d
SIGMA?
>

(non par défaut)
(* " ")
(μ initial " ")
(* " ")

*** OCCURRENCE 8 : LIVRE
MORPHE : LIVRE REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

*** GRAPHE ***

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6) MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-4- () MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MECHANT),CAT(ADS),NBR(SNG),K(FORME).

-8- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

-9- (13) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-10- (14) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- (14) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-12- (13) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).

-13- (15) LIT
SOLUTION 1 LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-14- (15,16) LIT
SOLUTION 1 LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-15- (18) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-16- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-17- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-18- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

-19- () <EXTREMITE>

FIN DU TEXTE. Pour examiner le graphe, tapez G; pour le conserver, R; pour continuer, C; pour arreter ARRET.
>r
GRAPHE CONSERVE
G, R, C ou ARRET?
>r
COMMANDE DEJA EXECUTEE
G, R, C ou ARRET?
>c

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:34:30 PAGE 3

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6) MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-4- () MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MECHANT),CAT(ADS),NBR(SNG),K(FORME).

-8- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-9- (13) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-10- (14) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- (14) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-12- (13) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).

-13- (15) LIT
SOLUTION 1 LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-14- (15,16) LIT
SOLUTION 1 LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-15- (18) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-16- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:37:06 PAGE 4

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

-17- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-18- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

-19- () <EXTREMITE>

RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

```
.1.ULTXT (.2.ULFRA (.3.ULOCC (.4.LE , .5.IL) , .6.ULOCC (.7.MAL , .8.MAL) , .9.ULOCC (.10.PETIT , .11.PETIT)
, .12.ULOCC (.13.MECHANT , .14.MAL) , .15.ULOCC (.16.PETIT , .17.PETIT , .18.PETIOT , .19.PETIOT) , .20.ULOCC
(.21.LIRE , .22.LIT) , .23.ULOCC (.24.LE , .25.IL) , .26.ULOCC (.27.LIVRER , .28.LIVRE)))

     SOMMET 1   : UL(ULTXT).

     SOMMET 2   : UL(ULFRA).

     SOMMET 3   : UL(ULOCC).

     SOMMET 4  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     SOMMET 5  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     SOMMET 6   : UL(ULOCC).

     SOMMET 7  MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 8  MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 9   : UL(ULOCC).

     SOMMET 10  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 11  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 12   : UL(ULOCC).

     SOMMET 13  MECHANT: UL(MECHANT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 14  MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 15   : UL(ULOCC).

     SOMMET 16  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 17  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 18  PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

     SOMMET 19  PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).

     SOMMET 20   : UL(ULOCC).

     SOMMET 21  LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

     SOMMET 22  LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

     SOMMET 23   : UL(ULOCC).

     SOMMET 24  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     SOMMET 25  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     SOMMET 26   : UL(ULOCC).
```


RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

```
     SOMMET 27  LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

     SOMMET 28  LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

```
TRAITEMENT TERMINE
TRAITEMENT ' MORPHO ' TERMINE POUR CE TEXTE

VOULEZ-VOUS MODIFIER LE TEXTE D'ENTREE ?
non
ESSAI SUR UN AUTRE TEXTE ?
oui
*DONNEZ LE NUMERO DU TEXTE
2

MODE INTERACTIF? (NON, X Y OU DET)
3 c?
AVEC X = 1, 2, OU 3, Y = G(I), GS(I), OU N; DET POUR LE DETAIL
3 g

PARAMETRES POUR LA PHASE 'MORPHO' ?
2 2 ti

EXECUTION BEGINS...
*EXECUTION BEGINS...
EXECUTION BEGINS...

*** PARAMETRES INITIAUX ***
GAMMA?
>d
SIGMA?
>
PLACEZ VOTRE PAPIER ET FAITES UN RETOUR CHARIOT
>
```

Pas d'interaction, mais graphe

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:38:09 PAGE 5

RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

.1.ULTAT (.2.ULFRA (.3.ULOCC (.4.LE , .5.IL) , .6.ULOCC (.7.MAL , .8.MAL) , .9.ULOCC (.10.PETIT , .11.PETIT) , .12.ULOCC (.13.MECHANT , .14.MAL) , .15.ULOCC (.16.PETIT , .17.PETIT , .18.PETIOT , .19.PETIOT) , .20.ULOCC (.21.LIRE , .22.LIT) , .23.ULOCC (.24.LE , .25.IL) , .26.ULCCC (.27.LIVRER , .28.LIVRE)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULOCC).

SOMMET 4 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

SOMMET 5 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

SOMMET 6 : UL(ULOCC).

SOMMET 7 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 8 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 9 : UL(ULOCC).

SOMMET 10 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 11 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 12 : UL(ULOCC).

SOMMET 13 MECHANT: UL(MECHANT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 14 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 15 : UL(ULOCC).

SOMMET 16 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 17 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 18 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 19 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

SOMMET 20 : UL(ULOCC).

SOMMET 21 LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

SOMMET 22 LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

SOMMET 23 : LL(ULOCC).

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:41:06 PAGE 6

RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

SOMMET 24 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

SOMMET 25 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

SOMMET 26 : LL(ULOCC).

SOMMET 27 LIVRE: LL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

SOMMET 28 LIVRE: LL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : EG

(8)

*** OCCURRENCE 1 : LE
MORPHE : LE REGLE : RARTI FORMATS : ARTI SG
--> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE REGLE : RPRON FORMATS : PRON SG
--> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 2 : PETIT
MORPHE : PETIT REGLE : RADJS FORMATS : ADJS SG
--> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
MORPHE : PETIT REGLE : RADJE FORMATS : ADJE SG
--> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 3 : LIT
MORPHE : LIT REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

*** OCCURRENCE 4 : LE
MORPHE : LE REGLE : RARTI FORMATS : ARTI SG
--> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE REGLE : RPRON FORMATS : PRON SG
--> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 5 : LIVRE
MORPHE : LIVRE REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

*** GRAPHE ***

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-4- (6) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7) LIT
SOLUTION 1 LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-6- (7,8) LIT
SOLUTION 1 LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-7- (10) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : EG

-8- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-9- (11) LIVRE
SOLUTION 1 LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-10- (11) LIVRE
SOLUTION 1 LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

-11- () <EXTREMITE>

FIN DU TEXTE. Pour examiner le graphe, tapez G; pour le conserver, R; pour continuer, C; pour arreter ARRET.
>r
GRAPHE CONSERVE
G, R, C ou ARRET?
>c

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:44:49 PAGE 1

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : EG

(8)

```
*** OCCURRENCE 1 : LE
MORPHE : LE                          REGLE : RARTI      FORMATS : ARTI      SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE : RPRON      FORMATS : PRON      SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 2 : PETIT
MORPHE : PETIT                       REGLE : RADJS      FORMATS : ADJS      SG
  --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
MORPHE : PETIT                       REGLE : RADJE      FORMATS : ADJE      SG
  --> DECOUPAGE : PETIT PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 3 : LIT
MORPHE : LIT                         REGLE : RVERB      FORMATS : VERB      V3
  --> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT                         REGLE : RSUBS      FORMATS : SUBS      SG
  --> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

*** OCCURRENCE 4 : LE
MORPHE : LE                          REGLE : RARTI      FORMATS : ARTI      SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE                          REGLE : RPRON      FORMATS : PRON      SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 5 : LIVRE
MORPHE : LIVRE                       REGLE : RVERB      FORMATS : VERB      V3
  --> DECOUPAGE : LIVRE LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE                       REGLE : RSUBS      FORMATS : SUBS      SG
  --> DECOUPAGE : LIVRE LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

$\gamma = d$, $\sigma = *$

```
*** GRAPHE ***

    -0- (1,2)    <ORIGINE>

    -1- (3,4)    LE
         SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

    -2- ()    LE
         SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

    -3- (5)    PETIT
         SOLUTION 1    PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

    -4- (6)    PETIT
         SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

    -5- (7)    LIT
         SOLUTION 1    LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
```

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:46:08 PAGE 2

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : EG

```
    -6- (7,8)    LIT
         SOLUTION 1    LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

    -7- (10)    LE
         SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

    -8- ()    LE
         SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

    -9- (11)    LIVRE
         SOLUTION 1    LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

    -10- (11)    LIVRE
         SOLUTION 1    LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 10:50:46 PAGE 3

RESULTAT DE L'EXECUTION , TEXTE : 2 CODE LANGUE EG

(9)

```
 .1.ULTAT (.2.ULFRA (.3.ULOCC (.4.LE , .5.IL) , .6.ULOCC (.7.PETIT , .8.PETIT) , .9.ULOCC (.10.LIRE , .11.LIT
) , .12.ULOCC (.13.LE , .14.IL) , .15.ULOCC (.16.LIVRER , .17.LIVRE)))

     SOMMET 1   : LL(ULTAT).
     SOMMET 2   : LL(ULFRA).
     SOMMET 3   : LL(ULOCC).
     SOMMET 4  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 5  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 6   : LL(ULOCC).
     SOMMET 7  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
     SOMMET 8  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 9   : LL(ULOCC).
     SOMMET 10  LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 11  LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).
     SOMMET 12   : LL(ULOCC).
     SOMMET 13  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 14  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 15   : LL(ULOCC).
     SOMMET 16  LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 17  LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

RESULTAT DE L'EXECUTION , TEXTE : 2 CODE LANGUE EG

```
.1.ULTXT (.2.ULFRA (.3.ULOCC (.4.LE , .5.IL) , .6.ULOCC (.7.PETIT , .8.PETIT) , .9.ULOCC (.10.LIRE , .11.LIT (9)
) , .12.ULOCC (.13.LE , .14.IL) , .15.ULOCC (.16.LIVRER , .17.LIVRE)))
     SOMMET 1   : UL(ULTXT).
     SOMMET 2   : UL(ULFRA).
     SOMMET 3   : UL(ULOCC).
     SOMMET 4  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 5  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 6   : UL(ULOCC).
     SOMMET 7  PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).
     SOMMET 8  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 9   : UL(ULOCC).
     SOMMET 10  LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 11  LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).
     SOMMET 12   : UL(ULOCC).
     SOMMET 13  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 14  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 15   : UL(ULOCC).
     SOMMET 16  LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 17  LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

```
TRAITEMENT TERMINE
TRAITEMENT ' MORPHO ' TERMINE POUR CE TEXTE

VOULEZ-VOUS MODIFIER LE TEXTE D'ENTREE ?
arret
TRAITEMENT TERMINE
R; T=17.87/31.92 10:54:55

p filemeg dicx1

MECHANT              ==ADJS    (SG      ,MAL              ).
MECHANT              ==ADJS    (SG      ,MECHANT          ).
PETIT                ==ADJS    (SG      ,PETIOT           ).
PETIT )              ==ADJE    (SG      ,PETIOT           ).

R; T=0.05/0.08 10:55:14

o pr grapheg 3
R; T=0.17/0.35 10:55:52

o pr grapheg 2
R; T=0.12/0.25 10:56:03
```

Dictionnaire auxiliaire n°1 (le seul)

Graphes conservés.

(10)

(11)

(10)

```
-0- (1,2)    <ORIGINE>

-1- (3,4)    LE
     SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- ()    LE
     SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6)    MAUVAIS
     SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-4- ()    MAUVAIS
     SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8)    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-6- ()    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- (9,10,11,12)    MECHANT
     SOLUTION 1    MECHANT: UL(MECHANT),CAT(ADS),NBR(SNG),K(FORME).

-8- (9,10,11,12)    MECHANT
     SOLUTION 1    MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-9- (13)    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-10- (14)    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- (14)    PETIT
     SOLUTION 1    PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-12- (15)    PETIT
     SOLUTION 1    PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).

-13- (15)    LIT
     SOLUTION 1    LIT: LL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-14- (15,16)    LIT
     SOLUTION 1    LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-15- (16)    LE
     SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-16- ()    LE
     SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-17- (19)    LIVRE
     SOLUTION 1    LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
```



```
-18- (19)    LIVRE
     SOLUTION 1    LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

-19- ()    <EXTREMITE>
```


(11)

```
-0- (1,2)    <ORIGINE>

-1- (3,4)    LE
     SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- ()    LE
     SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5)    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-4- (6)    PETIT
     SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7)    LIT
     SOLUTION 1    LIT: LL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-6- (7,8)    LIT
     SOLUTION 1    LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-7- (10)    LE
     SOLUTION 1    LE: LL(LE),CAT(ART),NBR(SNG),K(FORME).

-8- ()    LE
     SOLUTION 1    LE: LL(IL),CAT(PRO),NBR(SNG),K(FORME).

-9- (11)    LIVRE
     SOLUTION 1    LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-10- (11)    LIVRE
     SOLUTION 1    LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

-11- ()    <EXTREMITE>
```

IV-35 Autre essai (vprime)

```
NOTEZ LA LANGUE SOURCE D'ESSAI ( 3 CARACTERES MAXIMUM )
eg

TRAITEMENT GENERAL?
morpho

TRAITEMENT EXECUTION?
liste
*VOULEZ-VOUS LA LISTE DES NOMS DE TOUS VOS TEXTES ?
non
X T/I OU FIN ?
3 t

LE MECHANT PETIT MECHANT PETIT LIT LE LIVRE

X T/I OU FIN ?
fin

TRAITEMENT EXECUTION?
execut
DONNEZ LE NUMERO DU TEXTE
3

MODE INTERACTIF? (NON, X Y OU DET)
1 g
PARAMETRES : 1 1; 1 2 XY; 2 1 X; 2 2 XY

PARAMETRES POUR LA PHASE 'MORPHO' ?
2 2 ia
*
EXECUTION BEGINS...
EXECUTION BEGINS...
EXECUTION BEGINS...

*** PARAMETRES INITIAUX ***
PHI?
>?
PHI : nombre de formes a analyser ( * si tout le reste), ou 'X' si X = forme limite, ou I si arret sur -INIT-
>'petit'
MU?
>?
MU : traitement du mot inconnu. MU = 1 pour changer les formes et completer les dictionnaires,
MU = 2 pour seulement changer les formes, MU = 3 sinon.
>2
GAMMA?
>I
SIGMA?
>
```

'a' pour sortir l'arborescence en fin de morphologie

(12) Détail à l'imprimante

```
                                        SYSTEME A.T.E.F.          25/02/76 : 11:08:04   PAGE 1
   DETAIL DE L'EXECUTION , TEXTE : 3                                        CODE LANGUE : EG

   MOT INCONNU : MECHANT
   G, C, F ou ARRET?
   >?
   Tapez C pour continuer, I pour indexer, F pour lire ou changer la forme analysee, ARRET pour arreter
   et G pour examiner le graphe
   >I
   PAS D'INDEXAGE SI MU ¬= 1
   G, C, F ou ARRET?
   >f
   *FORME INCONNUE:     MECHANT
   LA CHANGER?
   >oui
   NOUVELLE FORME?
   >mauvais
   MAUVAIS
   FORME CORRECTE?
   >oui
   SUITE DU TRAITEMENT?
   >
   G, C, F ou ARRET?
   >

   *** BORNE ATTEINTE *** : PETIT
   GRAPHE?
   >
   PHI?
   >2
   MU?
   >1
   GAMMA?
   >d
   SIGMA?
   >
   MOT INCONNU : MECHANT
   G, C ,I, F ou ARRET?
   >I
   NOUVELLE INDEXATION?
   >oui
   NUMERO DU DICTIONNAIRE?
   >1
   ARTICLE?
   >mechant == adjs(sg,mal
   MECHANT                ==ADJS    (SG      ,MAL                  ).
   ENREGISTRER?
   >oui
   ARTICLE?
   >mechant==adje(sg,mal
   MECHANT                ==ADJE    (SG      ,MAL                  ).
   ENREGISTRER?
   >oui
   ARTICLE?
   >
   AUTRE DICTIONNAIRE?
   >
   SUITE DU TRAITEMENT?
   >
   G, C ,I, F ou ARRET?
   >
```

(13)

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:08:03 PAGE 1

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

```
*** OCCURRENCE 1 : LE
MCPPHE : LE                          REGLE : RARTI     FORMATS : ARTI     SG
  --> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MCPPHE : LE                          REGLE : RPRON     FORMATS : PRCA     SG
  --> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 2 : MECHANT
MOT INCONNU

*** OCCURRENCE 2 : MAUVAIS
MCPPHE : MAUVAIS                     REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MAUVAIS MALVAIS: LL(MAL),CAT(ADJS),NBR(SNG),K(FORME).
MCPPHE : MAUVAIS                     REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : MAUVAIS MALVAIS: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** GRAPHE ***

     -0- (1,2)   <ORIGINE>

     -1- (3,4)   LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -2- ()   LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     -3- ()   MAUVAIS
          SOLUTION 1    MAUVAIS: LL(MAL),CAT(ADJS),NBR(SNG),K(FORME).

     -4- ()   MAUVAIS
          SOLUTION 1    MAUVAIS: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 3 : PETIT
MCPPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJS),NBR(SNG),K(FORME).
MCPPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

*** OCCURRENCE 4 : MECHANT
MOT INCONNU

*** OCCURRENCE 4 : MECHANT
MCPPHE : MECHANT                     REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : MECHANT MECHANT: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
MCPPHE : MECHANT                     REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : MECHANT MECHANT: LL(MAL),CAT(ADJS),NBR(SNG),K(FORME).

*** GRAPHE ***
```

(12) φ = 'petit'

$\gamma = i$, $\sigma = *$

Indexage de "méchant"

(13) $\varphi = 2$

$\gamma = d$, $\sigma = *$

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:11:52 PAGE 2

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

```
     -0- (1,2)   <ORIGINE>

     -1- (3,4)   LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -2- ()   LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     -3- (5,6)   MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJS),NBR(SNG),K(FORME).

     -4- ()   MAUVAIS
          SOLUTION 1    MALVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

     -5- (7,8)   PETIT
          SOLUTION 1    PETIT: UL(PETIT),CAT(ADJS),NBR(SNG),K(FORME).

     -6- ()   PETIT
          SOLUTION 1    PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

     -7- ()   MECHANT
          SOLUTION 1    MECHANT: LL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

     -8- ()   MECHANT
          SOLUTION 1    MECHANT: UL(MAL),CAT(ADJS),NBR(SNG),K(FORME).

*** OCCURRENCE 5 : PETIT
MCPPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJS),NBR(SNG),K(FORME).
MCPPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
MCPPHE : PETIT                       REGLE : RADJE     FORMATS : ADJE     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).
MCPPHE : PETIT                       REGLE : RADJS     FORMATS : ADJS     SG
  --> DECOUPAGE : PETIT PETIT: LL(PETIOT),CAT(ADJS),NBR(SNG),K(FORME).

*** GRAPHE ***

     -0- (1,2)   <ORIGINE>

     -1- (3,4)   LE
          SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

     -2- ()   LE
          SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

     -3- (5,6)   MAUVAIS
          SOLUTION 1    MAUVAIS: UL(MAL),CAT(ADJS),NBR(SNG),K(FORME).

     -4- ()   MAUVAIS
```

(14) Indexage de "petit"

(15) $\varphi = 1$

(16) $\gamma = d$, $\sigma = *$

*** BORNE ATTEINTE *** : PETIT (14)

*** GRAPHE ***

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6) MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-4- () MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- () MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-8- () MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

GRAPHE?
>
INDEXAGE?
>oui
NOUVELLE INDEXATION?
>oui
NUMERO DU DICTIONNAIRE?
>1
ARTICLE?
>petit**adjs(sg,petiot).
PETIT **ADJS (SG ,PETIOT).
ENREGISTRER?
>oui
ARTICLE?
>petit¢petit ** adje(sg, petiot
PETIT **ADJE (SG ,PETIOT).
ENREGISTRER?
>oui
ARTICLE?
>
AUTRE DICTIONNAIRE?
>
SUITE DU TRAITEMENT?
>
PHI?
>1
MU?
>

GAMMA? (15)
>

*** BORNE ATTEINTE *** : LIT (16)
GRAPHE?
>oui
GAMMA?
>1
SIGMA?
>
GRAPHE?
>
INDEXAGE?
>
PHI?
>
MU?
>
GAMMA?
>d
SIGMA? (17)
>

*** GRAPHE ***

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6) MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-4- () MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- () MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-8- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADS),NBR(SNG),K(FORME).

-9- (13) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADS),NBR(SNG),K(FORME).

-10- (14) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- (14) PETIT

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:15:09 PAGE 3

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- () MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-8- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-9- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-10- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- () PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-12- () PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

(17) $\varphi = *$

*** OCCURRENCE 6 : LIT
MORPHE : LIT REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIT LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIT REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIT LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

*** OCCURRENCE 7 : LE
MORPHE : LE REGLE : RART1 FORMATS : ART1 SG
--> DECOUPAGE : LE LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
MORPHE : LE REGLE : RPRON FORMATS : PRLN SG
--> DECOUPAGE : LE LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

*** OCCURRENCE 8 : LIVRE
MORPHE : LIVRE REGLE : RVERB FORMATS : VERB V3
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
MORPHE : LIVRE REGLE : RSUBS FORMATS : SUBS SG
--> DECOUPAGE : LIVRE LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

$\gamma = d$, $\sigma = *$

*** GRAPHE ***

-0- (1,2) <ORIGINE>

-1- (3,4) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:15:44 PAGE 4

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

(18)

-2- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-3- (5,6) MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-4- () MAUVAIS
SOLUTION 1 MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-5- (7,8) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-6- () PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-7- () MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-8- (9,10,11,12) MECHANT
SOLUTION 1 MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).

-9- (13) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-10- (14) PETIT
SOLUTION 1 PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).

-11- (14) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-12- (13) PETIT
SOLUTION 1 PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

-13- (15) LIT
SOLUTION 1 LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-14- (15,16) LIT
SOLUTION 1 LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

-15- (16) LE
SOLUTION 1 LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

-16- () LE
SOLUTION 1 LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

-17- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

-18- (19) LIVRE
SOLUTION 1 LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:18:24 PAGE 5

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : EG

-19- () <EXTREMITE>

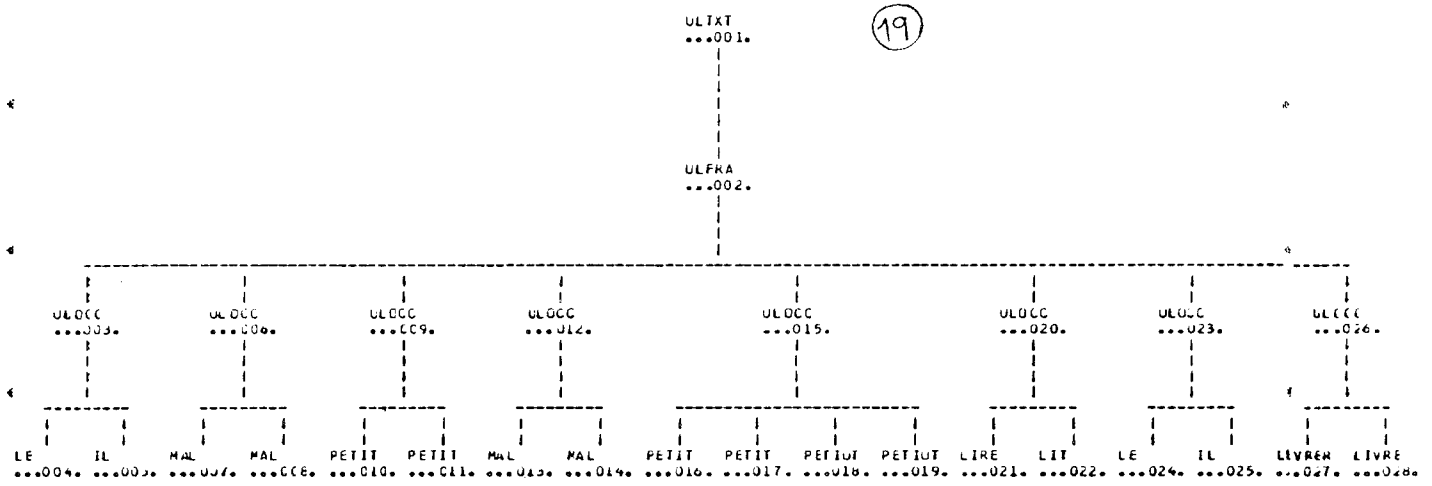
```
        SOLUTION 1    PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).

   -12- (13)    PETIT
        SOLUTION 1    PETIT: UL(PETIOT),CAT(ADS),NBR(SNG),K(FORME).

   -13- (15)    LIT
        SOLUTION 1    LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

   -14- (15,16)    LIT
        SOLUTION 1    LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).

   -15- (18)    LE
        SOLUTION 1    LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).

   -16- ()    LE
        SOLUTION 1    LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).

   -17- (19)    LIVRE
        SOLUTION 1    LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).

   -18- (19)    LIVRE
        SOLUTION 1    LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).

   -19- ()    <EXTREMITE>

FIN DU TEXTE. Pour examiner le graphe, tapez G; pour le conserver, R; pour continuer, C; pour arreter ARRET.   (18)
>c
```

(Résultat final) (19)

```
TRAITEMENT TERMINE
TRAITEMENT ' MORPHO ' TERMINE POUR CE TEXTE

VOULEZ-VOUS MODIFIER LE TEXTE D'ENTREE ?
arret

FROM OPERATOR:  RECUPERATION MACHINE ASP

*TRAITEMENT TERMINE
R; T=7.39/13.61 11:19:29
```

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:19:01 PAGE 6



*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:19:02 PAGE 8

RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

```
 .1.ULTXT (.2.ULFRA (.3.ULLOCC (.4.LE , .5.IL) , .6.ULOCC (.7.MAL , .8.MAL) , .9.ULOCC (.10.PETIT , .11.PETIT)
, .12.ULOCC (.13.MAL , .14.MAL) , .15.ULOCC (.16.PETIT , .17.PETIT , .18.PETIOT , .19.PETIOT) , .20.ULOCC (.21
.LIRE , .22.LIT) , .23.ULOCC (.24.LE , .25.IL) , .26.ULOCC (.27.LIVRER , .28.LIVRE)))

     SOMMET 1   : LL(ULTXT).
     SOMMET 2   : LL(ULFRA).
     SOMMET 3   : LL(ULOCC).
     SOMMET 4  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 5  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 6   : LL(ULOCC).
     SOMMET 7  MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 8  MAUVAIS: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 9   : LL(ULOCC).
     SOMMET 10  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 11  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 12   : LL(ULOCC).
     SOMMET 13  MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 14  MECHANT: UL(MAL),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 15   : LL(ULOCC).
     SOMMET 16  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 17  PETIT: UL(PETIT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 18  PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 19  PETIT: UL(PETIOT),CAT(ADJ),NBR(SNG),K(FORME).
     SOMMET 20   : LL(ULOCC).
     SOMMET 21  LIT: UL(LIRE),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 22  LIT: UL(LIT),CAT(NOM),NBR(SNG),K(FORME).
     SOMMET 23   : LL(ULOCC).
```

*** GROUPE D'ETUDES POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F. 25/02/76 à 11:19:04 PAGE 9

RESULTAT DE L'EXECUTION , TEXTE : 3 CODE LANGUE EG

```
     SOMMET 24  LE: UL(LE),CAT(ART),NBR(SNG),K(FORME).
     SOMMET 25  LE: UL(IL),CAT(PRO),NBR(SNG),K(FORME).
     SOMMET 26   : LL(ULOCC).
     SOMMET 27  LIVRE: UL(LIVRER),CAT(VRB),TPS(PRE),PRS(3),K(FORME).
     SOMMET 28  LIVRE: UL(LIVRE),CAT(NOM),NBR(SNG),K(FORME).
```

CHAPITRE V

SEMANTIQUE

" ... nor did Pnin, as a teacher, ever presume to approach the lofty halls of modern scientific linguistics, that ascetic fraternity of phonemes, that temple wherein earnest young people are taught not the language itself, but the method of teaching others to teach that method ; which method, like a waterfall splashing from rock to rock, ceases to be a medium of rational navigation but perhaps in some fabulous future may become instrumental in evolving esoteric dialects - Basic Basque and so forth - spoken only by certain elaborate machines".

Nabokov, "Pnin"

INTRODUCTION

Les systèmes de traduction automatique (TA) qui sont actuellement utilisés, ou qui ont déjà été testés sur de grandes quantités de textes, n'utilisent qu'une "sémantique" assez rudimentaire, limitée à l'utilisation de "traits" (animation, abstraction, champ sémantique, etc.). On pense en particulier aux systèmes issus des travaux de l'Université de Georgetown, utilisés depuis près de vingt ans, et au système du CETA (Grenoble), testé sur un corpus de plus de 400000 mots. L'utilisation de tels "traits" permet de se limiter à des traitements purement combinatoires § . Et il faut bien reconnaître que la qualité des traductions obtenues est assez élevée pour qu'on puisse se demander s'il est utile (et rentable) d'intégrer plus de "sémantique" à ces systèmes, puisqu'aussi bien il faudrait avoir totalement résolu le problème de l'intelligence artificielle (IA) pour pouvoir obtenir des traductions de "haute qualité", au sens de Bar-Hillel, c'est-à-dire aussi bonnes que des traductions soignées effectuées par des spécialistes bilingues. Cependant, bien des modèles de TA, implémentés ou non, utilisent ou prévoient une composante sémantique élaborée §§ .

Le mot "modèle" a deux sens bien différents en logique et en linguistique, et Schreider (1973) montre qu'une théorie linguistique correspond en fait à un modèle logique. Un troisième sens existe en informatique, le "modèle implémentable", qui est plutôt un schéma explicatif et algorithmique.

Nous distinguons donc soigneusement entre "systèmes" et "modèles". Un système de TA doit satisfaire des exigences pratiques d'ampleur (dictionnaires, grammaires, textes) et d'efficacité, alors qu'un modèle, implémenté ou non, vise d'abord une certaine adéquation linguistique. Citons par exemple les implémentations réalisées par Y. Wilks [60] , T. Winograd [63] ou R. Shank [45] . Ces auteurs reconnaissent d'ailleurs eux-mêmes que leurs modèles traitent des aspects très intéressants du langage, mais dans un contexte restreint, et il est clair qu'on ne pourrait étendre tels quels ces modèles à des systèmes (de TA, ou d'IA) généraux sans buter sur le gros problème de place et de complexité de traitement. Dans un autre ordre d'idées, on peut citer les modèles de TA de linguistes soviétiques (Mel'tchuk, Zholkovskij, équipe du Pr. Rosenzweig), qui prévoient une partie sémantique dans les dictionnaires utilisés, mais dont l'implémentation n'est pas réalisée pour l'instant. Nous reviendrons bien sûr sur ces différentes approches.

Jusqu'ici, nous avons parlé de "sémantique" sans dire ce que c'était. Il n'est pas inutile de le préciser, car on utilise souvent ce terme dans plusieurs domaines avec des sens différents, ce qui conduit à de fâcheux malentendus. Dans la première partie, je chercherai donc à préciser la terminologie et à situer le problème de la "sémantique en TA". Dans la seconde, je présenterai divers "langages pivots" utilisés en TA. Dans la troisième, j'examinerai brièvement divers types de représentation du "sens" (ou de la "connaissance") utilisés par des modèles, presque tous implémentés, destinés plus particulièrement à la TA ou à l'IA. Enfin, nous verrons en quatrième lieu comment on peut envisager d'utiliser une composante "gnosto-encyclopédique" §§§ dans des systèmes de TA.

---

§ - Au sens linguistique de "combinatoire de classes", ou "étude des conditions et des conséquences des cooccurrences d'éléments" (Pottier). Outre l'expression consacrée de "système combinatoire", on parle aussi de "méthode combinatoire" pour désigner un algorithme dans lequel on construit toutes les possibilités de combiner les éléments de l'entrée avant de déterminer la sortie. L'algorithme de Cocke ou les systèmes-Q en sont des exemples.

§§ - Voir en particulier l'introduction très intéressante de Mel'tchuk (1974).

§§§ - Ce terme est emprunté à Mel'tchuk et Kulagina (1967).

I - SEMANTIQUE, NIVEAUX DE SENS ET TRADUCTION AUTOMATIQUE

1.1. SEMANTIQUE

En mathématique, une sémantique est la donnée d'un système formel et d'une interprétation. Citons Mendelson [34] "Concept and propositions which involve the notion of interpretation, and related ideas such as truth, model, etc., are often called semantical to distinguish them from syntactical concepts, which refer to simple relations amoung symbols and expressions of precise formal languages".

En linguistique, la situation est loin d'être aussi claire, car ni le système (le langage), ni le domaine (le monde, réel ou imaginaire), ni l'interprétation (le sens des énoncés) ne sont fournis de manière formelle. On pourrait dire qu'à la limite, pour l'homme, dans le langage tout est sens, et rien ne l'est pour la plante. Les difficultés commencent au moment où l'on tente de formaliser le langage et son domaine, en cherchant à en extraire un système combinatoire (un langage formel) S, une "connaissance formelle" C (base de données), et des règles d'interprétation. En effet, le linguiste continuera d'appeler "sémantiques" certains phénomènes qu'il aura formalisés dans S (traits sémantiques, classes morphologiques liées à l'animation, comportement des verbes de mouvement, etc.). C'est en ce sens qu'on parle de "sémantique par la syntaxe" [1,2,39] . Il faudrait plutôt dire "sémantique retrouvée dans la syntaxe", puisqu'il s'agit au fond de parties de l'interprétation que la langue a codé avec assez de régularité, dans les mots eux-mêmes ou dans leur arrangement.

Pour être clair, nous allons introduire quelques définitions : nous dirons que le niveau grammatical, N1, est celui qu'on atteint quand on dégage un ensemble de propriétés purement formelles (formation et combinatoire des unités graphiques) d'une langue. Quant à la formalisation de domaine, elle trouve un obstacle que ne connaît habituellement pas la logique mathématique: il évolue, et parfois même si l'on s'est limité à un "micro-univers" donné a priori [63] . On conçoit naturellement deux étapes dans la formalisation : décrire d'abord les constantes du domaine (N2, c'est le royaume des dictionnaires et "règles du sens commun"), et ensuite ses variations (N3, la "mise en situation"), toujours en les liant formellement à S. L'énorme ouvrage de Mel'tchuk [36] est conçu comme un "essai" de formalisation au niveau N2.

1.2. NIVEAU DE SENS

Ces trois niveaux semblent perçus de manière assez générale, et il est remarquable qu'ils correspondent à trois types d'implémentation, de plus en plus complexes (au sens des structures et des algorithmes utilisés). Droste [16] définit par exemple une hiérarchie de "niveaux de langage" tout à fait parallèle.

L1 est le langage profond, universel. C'est le domaine des concepts. Un concept est plus ou moins consistant, et un énoncé plus ou moins grammatical.

L2 est le langage de référence. C'est le domaine des idées, c'est-à-dire des concepts appliqués à un domaine extérieur. Une idée est plus ou moins vraie, et un énoncé plus ou moins "textuel".

L3 enfin est le langage de communication. C'est le domaine des faits. Un fait est plus ou moins approprié (à la réalité extérieure, i.e. à une situation donnée), et un énoncé de L3 (tout énoncé émis par un locuteur) est plus ou moins acceptable.

L1 correspond à peu près au niveau de la linguistique "pure" [35] , à la grammaire "en soi", L2 à la sémantique (par rapport à des référents figés), et L3 à la pragmatique. Les limites effectives entre ces niveaux dépendent bien sûr de la façon dont le linguiste étudie chacun d'eux. A la limite, on pourrait dire que c'est plus une hiérarchie de types de formalisation que de niveaux "objectifs" du langage. Par exemple, il serait possible d'inclure dans L2 un modèle des mathématiques connues à ce jour et de déterminer un référent anaphorique ambigu en utilisant la cohérence mathématique des énoncés possibles, mais il est plus vraisemblable qu'on laisserait ce type de question à L3. Par contre, le robot de Winograd [63] peut tester la cohérence logique des commandes qu'on lui donne, puisque sa "composante dans L2" contient un modèle de son univers (une table sur laquelle sont disposés des cubes, des cônes et des pyramides) et de sa logique (impossibilité de placer un cube sur une pyramide, un cône, ou dans un cube plus petit).

On peut bien sûr, et c'est d'ailleurs une nécessité en TA, raffiner ces niveaux. Mel'tchuk [38] donne comme exemple les neuf phrases suivantes :

(1) Tous savent que le peuple allemand combat le fascisme.
(2) Tous savent que le peuple allemand combat contre le fascisme.
(3) Tous connaissent le combat du peuple allemand contre le fascisme.
(4) Il est connu de tout le monde que le peuple allemand mène le combat contre le fascisme.
(5) Tous savent que se mène la lutte antifasciste du peuple allemand.
(6) Qui ne sait qu'en Allemagne le peuple ne s'est pas soumis à la peste brune ?
(7) Nous savons qu'en Allemagne les fascistes ne connaissent pas de repos, l'organisation clandestine du peuple agit.
(8) Nous savons que les descendants de Thomas Munzer se lèvent sur la route des hitlériens.
(9) Tous savent que, des Alpes à la Mer du Nord, du Rhin à l'Oder, le peuple allemand poursuit l'oeuvre d'Edgar André.

D'après Mel'tchuk , les phrases
(1) à (2) ont même structure de surface
(1) à (3) ont même structure lexico-syntaxique (LSS)
(1) à (5) ont même structure "basic"
(1) à (6) ont même "inscription de sens"
(1) à (9) représente la même situation

(1) à (5) peuvent être équivalentes dans L1 dès que L1 est assez bien défini. L"inscription de sens" est plutôt une représentation dans L2. En effet, pour que les phrases (1) à (6) soient synonymes, il faut avoir des équivalences du type "le peuple allemand $\Longleftrightarrow$ le peuple en Allemagne" et "La peste brune" $\Longleftrightarrow$ "le fascisme". Et toutes ces phrases peuvent être équivalentes dans L2 si on dispose d'équivalences encore plus fortes que les précédentes, purement "lexicales". Par exemple, la définition géographique de l'Allemagne, ou l'oeuvre d'E. André.

De tels exemples conduisent à penser qu'on ne peut donner "a priori" une définition du sens. Il faudrait plutôt dire que le sens d'un énoncé, dans un certain niveau de langage, est la classe des énoncés équivalents à ce niveau, c'est-à-dire qui y ont la même représentation. Les représentations utilisées varient beaucoup, et il est intéressant de les comparer pour voir si les équivalences d'un niveau se conservent quand on change de représentation. Dans l'affirmative, on peut penser que ceci exprime bien une réalité linguistique ou psychologique, et nous montrerons qu'il en est ainsi pour les "représentations du sens" dans L1 les plus connues ("pivot", ou "basic" pour Mel'tchuk).

1.3. UTILISATION EN TA

En TA, on cherche finalement à fournir une traduction pour un "énoncé" (phrase, paragraphe, texte). La première étape de la traduction consiste à analyser cet énoncé à un certain niveau, au sens précédent (LSS ou basic le plus souvent, car on doit faire un compromis entre la qualité de la traduction et l'efficacité du traitement). Si on trouve plus d'une représentation, on dit que l'énoncé est ambigu, et il s'agit non pas de trouver "la bonne" représentation, mais plutôt la plus vraisemblable, étant donnés les renseignements dont on dispose (à un niveau supérieur). On dira encore qu'un énoncé est ambigu pour un couple de langues donné s'il peut avoir plusieurs traductions, même si, au niveau d'analyse considéré, il n'a qu'une représentation.

Reprenons un exemple de Y. Wilks [60] : "The crook drank a glass of water". Le mot "crook" peut signifier "escroc" ou "houlette". L'ambiguïté peut être levée si on dispose d'une certaine connaissance du monde extérieur, ici que l'agent de boire est (le plus souvent) animé. On n'aura pas besoin de ce renseignement s'il n'y a pas ambiguïté, comme dans "my car drinks gasoline" ; mais il se peut fort bien que, dans la situation considérée, on ait choisi la mauvaise interprétation. C'est d'ailleurs une observation fondamentale que le sens d'un énoncé réel (dans L3) peut être contraire au sens apparent (dans L2), ou encore être un "non-sens", une incohérence. L'expérience montre qu'on peut toujours imaginer une situation où il aurait fallu accepter l'interprétation rejetée. La houlette de Moïse peut bien boire l'eau du rocher...

C'est pourquoi les critères qu'on emploie ne peuvent être que préférentiels et approximatifs. C'est seulement dans un univers restreint que la "connaissance" de cet univers peut lever définitivement une ambiguïté. Dans les systèmes usuels, cette connaissance, ou "base de données", se présente souvent comme un ensemble d'énoncés descriptifs (définitions, réseaux sémantiques) et de règles de type lexico-syntaxique ("inférences") permettant de relier les atomes du système formel (mots, "unités lexicales", "éléments sémantiques") de manière plus souple. Pour traduire correctement "The monkeys ate bananas. They are ripe." par "Les singes ont mangé des bananes. Elles sont mûres." Il faut savoir qu'une banane est un fruit et que l'adjectif mûr est surtout associé aux fruits. C'est ce qu'utilise Wilks, en ne se servant que des définitions de ces termes. Par contre, pour trouver l'antécédent de "they" dans "The soldiers fired at the women and they fell", il faut savoir que tirer sur quelqu'un implique (souvent) le blesser, et qu'un blessé a tendance à tomber : c'est une "inférence du sens commun" au sens de Wilks. Encore une fois, ceci ne permet pas d'affirmer avoir trouvé la bonne interprétation, c'est simplement une meilleure approximation, au sens où, dans la plupart des cas, on aura un meilleur résultat qu'avec un choix arbitraire ou aléatoire.

Quand on dit qu'un système ou un modèle a une "composante sémantique", il s'agit souvent ... d'une syntaxe poussée. C'est à dire qu'on définit un langage abstrait de "représentations profondes" ou des "langages pivot". Leur utilité est évidente dès qu'on cherche à construire des systèmes multilingues. Pour ce qui est de l'utilisation effective d'une "vraie" sémantique (L2, voire L3) dans les systèmes de TA, l'objection majeure tient au problème de place et de complexité de traitement. En effet, il semble qu'on doive payer très cher une légère amélioration de qualité, si on n'arrive pas à limiter l'utilisation de la sémantique (pire, de la "pragmatique") aux seuls cas vraiment litigieux : intégrer une telle composante à un système suppose a priori que celui-ci possède une organisation souple. C'est alors seulement que peut apparaître le bénéfice lié à la présence d'informations supplémentaires et à l'interaction entre plusieurs niveaux de traitement. C'est bien ce qu'ont montré Tubach [50] en reconnaissance de la parole, ou Winograd [63] en IA, dans des contextes volontairement restreints.

1.4. DEUX OBJECTIFS ET DEUX APPROCHES

On l'a dit, les systèmes de TA n'utilisent pas encore de "vraie" sémantique. Il serait évidemment tentant d'essayer directement de leur incorporer la pragmatique. Mais ce serait sans doute mettre la charrue avant les boeufs, car seuls des modèles très partiels ont pu être implémentés jusqu'à présent [17,45] . Et ces modèles utilisent des structures et des opérations très complexes dont il faudrait avoir une connaissance mathématique assez poussée (en particulier en ce qui concerne les problèmes de complexité et de décidabilité) avant de pouvoir se lancer dans l'écriture d'un système.

Par contre, j'essaierai de montrer comment adapter les solutions indiquées par Wilks (au niveau L2) : on peut coder la "connaissance du monde" comme un dictionnaire constitué d'énoncés déclaratifs, ou plus exactement de leurs structures "pivot", c'est-à-dire comme un ensemble d'arborescences. Et le traitement des arborescences est fondamentalement moins complexe que celui des réseaux [11,12] , ou que celui de structures récursives (représentations conceptuelles de Shank [46]).

Dans cette optique, utiliser une "vraie" sémantique consiste, étant donnée une certaine connaissance initiale, à la combiner avec les représentations possibles du sens d'un énoncé pour en déduire la résolution d'ambiguïtés (TA) et/ou une nouvelle connaissance, ou une commande (IA). On peut distinguer deux approches :

- la déduction "forte", qui consiste à formaliser L2 (ou 13) comme une logique mathématique classique et à lui appliquer les techniques de la démonstration automatique. Cette méthode est bien adaptée aux "micromondes" [17,45] .

- La déduction "faible" ("preference semantics"), dans laquelle on ne cherche que certaines conséquences utilisables, et qui emploie la notion de vraisemblance par opposition à celle de vérité ("densité sémantique" [60]). Elle semble plus adaptée aux problèmes liés au traitement de langues naturelles dans leur totalité (ou quasi-totalité).

II - LANGAGES "PIVOT" UTILISES EN TA

2.1. LE LANGAGE PIVOT DU SYSTEME "TITUS II".

2.1.1. BUTS DU SYSTEME

Le système Titus II, développé à l'Institut textile de France [17] , est un système de documentation multilingue C'est dire que ses impératifs sont d'abord pratiques : constitution de banques de données, édition de bulletins analytiques, de thésaurus, de dictionnaires, recherche de documentation, etc. En conséquence, l'utilisation d'un langage pivot découle plus d'une nécessité d'économie de place pour le stockage et de temps pour la traduction que d'impératifs théoriques. En 1973, Titus II travaillait sur le français, l'anglais, l'allemand et l'espagnol.

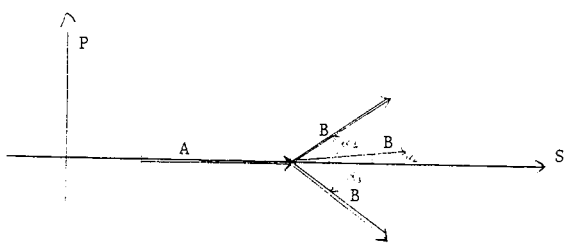
On peut enfin noter le caractère particulier des informations à traiter. Il ne s'agit pas d'articles scientifiques quelconques, mais de résumés d'articles sur les techniques du textile, munis d'une structure précise (titre et zones thématique , argumentaire et numérique). On peut de plus imposer aux rédacteurs des résumés , l'utilisation d'une syntaxe et d'un codage destinés à éviter le plus possible les ambiguités.

2.1.2. SOLUTIONS APPORTEES

Tout d'abord, l'analyse morphologique est réduite par l'utilisation de signes spéciaux dans le texte d'entrée, ce qui ne constitue d'ailleurs pas une limitation du système.

Ensuite, on utilise une syntaxe définie à l'avance, et fondée sur les notions suivantes : les auteurs distinguent dans la langue un axe "paradigmatique" et un axe "syntagmatique", et représentent la structure d'une phrase comme une suite de segments articulés représentant les syntagmes. Les angles des segments avec l'axe syntagmatique caractérisent les fonctions des syntagmes.

Exemple :

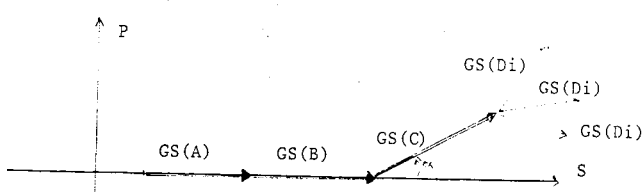


Ici, on pourra convenir que :
α_1 correspond à "avant"
α_2 correspond à "pendant
α_3 correspond à "après"

d'où les énoncés : A avant (pendant, après) B.

Sur ce modèle de base, les auteurs de Titus II ont construit, à l'aide de 18 syntagmes de base, 530 stuctures de phrase possibles. On a réduit cette syntaxe pour qu'elle soit compatible entre les 4 langues, simple, similaire à la syntaxe classique, et facilement traduisible sans ambiguïté.

Donnons enfin, pour préciser les idées, le schéma d'un "syntagme standard à deux actants" :



GS(X) désigne le groupe syntagmatique X. Ici, A,B,C et sont fixes. C est l"actant fixe" et Di le second actant, variable. A correspondra par exemple à un groupe au nominatif, B à un complément de nom, C à un complément régi par A et D à un circonstanciel.

Notons enfin qu'on indique dans le texte d'entrée la structure de la phrase, pour éviter les ambiguïtés.

2.1.3. CARACTERISTIQUES DU LANGAGE PIVOT

Remarquons d'abord que, dans ce système, le langage pivot est à la fois le langage graphique utilisé dans les deux exemples précédents, et aussi son codage sous une forme binaire très compacte, destinée à la mémorisation.

Ce qui nous intéresse est bien sûr le type de représentation utilisée. On remarque d'abord la linéarité du langage : on n'a pas affaire ici à une représentation arborescente, mais à une représentation en chaîne. Remarquons à ce propos une certaine analogie avec le type de structures utilisées par certains programmes de TA "pure" (de toute implication documentaire !), comme celui de Georgetown [68] .

Notons aussi l'utilisation de notions de linguistique structurale comme celle d'actants. Dans un syntagme, un actant a une certaine fonction logique, qui peut correspondre à différentes réalisations syntaxiques dans une même langue. Ceci permet donc de coder de façon unique deux phrases de structures syntaxiques différentes mais équivalentes. Remarquons que, pour des raisons pratiques, on a choisi dans Titus II d'utiliser le plus possible la forme nominale, quand elle existe (l'étude d'une méthode ... plutôt que "on étudie une méthode"...).

Enfin, outre les vertus de simplicité et d'adéquation que nous venons de signaler, notons l'utilité d'une telle représentation "pivot". Elle est double :

- au niveau de la traduction, on n'a pas besoin d'écrire autant de programmes de traduction qu'il y a de couples de langues, mais seulement un programme d'analyse et un programme de synthèse pour chacune d'elles
- au niveau du stockage, puisqu'on ne cherche pas à obtenir une traduction immédiatement, mais à l'occasion d'une demande ultérieure, et qu'il est moins onéreux de garder une représentation condensée que quatre traductions.

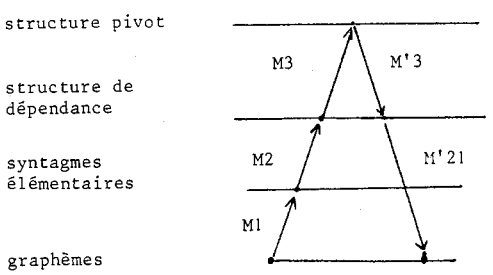
2.2. LE LANGAGE PIVOT DU SYSTEME DU CETA

2.2.1. BREF RAPPEL SUR LE SYSTEME

Le système développé au CETA de 1963 à 1969 sur IBM 7044 est un système de traduction automatique multilingue de deuxième génération -et même presque de troisième, si l'on en croit O.S. Kulagina et I.A. Mel'tchuk [27] .
Il est destiné à traiter des textes scientifiques ou techniques quelconques, présentés avec une préédition très réduite (marquage des fin de phrases et codes correspondant aux indications typographiques).

On trouvera toutes les précisions omises ici dans les publications des membres du CETA [9 et 51 à 57] notamment.
Le système utilise la notion de niveau linguistique. A chaque niveau est associée une représentation spéciale du texte d'entrée. Les "modèles" du système sont les outils (théoriques et pratiques) qui permettent de passer d'une représentation du texte à un niveau à sa représentation au niveau suivant.

Schéma :



Le modèle M1 effectue l'analyse morphologique en deux passes. Le modèle M2 trouve d'abord une (ou plusieurs) structure (s) de constituants à partir d'une grammaire sous forme normale de Chomsky, puis en déduit une structure de dépendance. Enfin, le modèle M3, "d'étiquetage", transforme cette structure en une structure pivot. M'3 et M'21 sont "grosso modo" les inverses de M3 et (M1,M2).

2.2.2. CARACTERISTIQUES DU LANGAGE PIVOT

Définition

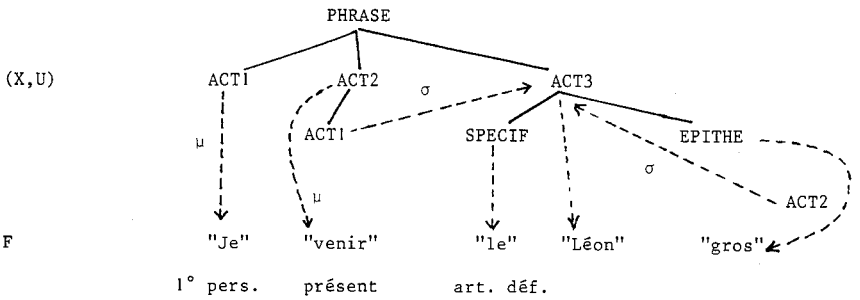
Une expression *en langage pivot est définie par :*

- *un* graphe arborescent *(X,U), où X est l'ensemble des sommets et U l'ensemble des arcs*
- *une application* ε *de X dans un ensemble E d'étiquettes (tout sommet est* étiqueté*).*
- *un ensemble F* d'occurrences *(combinaisons d'une "unité lexicale" et de valeurs de "variables persistantes")*
- *une fonction* σ *(substitution ou élision) de X dans X et une fonction* μ *(occurrence) de X dans F telles que :* $\sigma(x) = \emptyset \iff \mu(x) \neq \emptyset$

Une telle structure est une quasi-arborescence étiquetée.

Les étiquettes sont des noms de fonctions syntactico-logiques. L'étiquette d'un sommet indique donc sa fonction par rapport à son père.

Exemple : "Je dis au gros Léon de venir"



On trouvera dans [9] toutes les étiquettes et leurs significations. Ici, ACT1(2,3) notent le premier (second, troisième) actant §, SPECIF un spécificateur et EPITHE un modificateur faible (épithète).

Premier trait :

Ce langage pivot n'est pas purement arborescent, à cause de la présence des fonctions μ et σ. S'il est concevable d'éliminer la fonction μ en associant à un sommet, outre une étiquette dans E, une étiquette dans F, on ne peut utiliser un tel artifice pour σ. C'est dire que σ ne représente pas une propriété invariante par transformation, c'est un pointeur à recalculer après chaque transformation. Cependant, une quasi-arborescence est une hiérarchie, et il semble possible de généraliser les modèles de transduction d'arborescences [11] à ces structures, moyennant une augmentation de la complexité (quadratique au lieu de linéaire), qui reste malgré tout bien en deça de ce qu'il faut pour traiter les multigraphes.

Remarquons enfin que le système du CETA peut traiter ce type de structure parce qu'il utilise une représentation par pointeurs, qui permet de représenter des graphes plus généraux que les arborescences.

Deuxième trait :

La plupart des langages "pivot" proposés par les linguistes, outre des langages "naturels" comme le volapük ou l'espéranto, sont des langages formels munis d'une syntaxe et d'un lexique. Dans le pivot du CETA, il n'y a pas de lexique propre. C'est donc un langage hybride au sens de Shaumjan [47]. On utilise les éléments lexicaux ("unités lexicales" et variables persistantes) de la langue source ou de la langue cible, et la phase de transfert a en partie pour objet d'effectuer des substitutions d'unités lexicales en utilisant un dictionnaire de transfert.

Pour conclure, disons qu'il est remarquable de pouvoir utiliser une formulation aussi simple dans le traitement effectif d'un volume de textes déjà considérable (400.000 mots environ). Il semble même qu'on puisse se limiter à une structure purement arborescente ; c'est du moins le principe de nouveau langage pivot développé au GETA.

---

§ Extension de la notion d'actant chez Tesnière, au sens linguistique de "combinatoire de classes", ou "étude des conditions et des conséquences des cooccurrences d'éléments" (Pottier). Outre l'expression consacrée de "système combinatoire", on parle aussi de "méthode combinatoire" pour désigner un algorithme dans lequel on construit toutes les possiblités de combiner les éléments de l'entrée avant de déterminer la sortie. L'algorithme de Cocke ou les systèmes-Q en sont des exemples.

2.3. LANGAGE PIVOT AU GETA

2.3.1. POURQUOI DES LANGAGES ARBORESCENTS ?

La plupart du temps, comme on le verra plus loin, on utilise des graphes, voire des multigraphes pour représenter le sens d'un ou plusieurs énoncés. Cependant, le traitement de telles structures est long et compliqué. Par exemple, on ne connait pas d'algorithme de reconnaissance d'un sous-graphe dans un autre qui ne soit pas de complexité exponentielle. D'autre part, si on décompose ces graphes en arborescences, la recherche est compliquée du fait qu'on dissymétrise la structure [15] , et on n'y gagne pas en temps de calcul.

Comme on l'a vu plus haut, on peut aussi utiliser une structure quasi-arborescente, c'est-à-dire une structure arborescente munie d'une ou plusieurs relations binaires entre les sommets. On peut donc les traiter comme des arborescences, en utilisant un mécanisme particulier pour traiter les relations supplémentaires lors des transformations (stockage et modification de pointeurs). C'est ce qui est réalisé dans le système du CETA [57] .

Or il se trouve que les structures arborescentes peuvent s'écrire de façon linéaire simple (en utilisant la notation parenthésée habituelle), ce qui n'est pas le cas des graphes ou multi-graphes, et qu'on peut ramener les transformations d'arborescences à des transformations de sous-chaînes. C'est l'idée fondamentale des "transducteurs d'arborescences" [11] . La complexité d'un transducteur d'arborescences est une fonction linéaire du nombre de sommets de l'arborescence d'entrée, ce qui est un avantage décisif. C'est pourquoi, à condition de garder l'adéquation linguistique, on cherche à utiliser le plus possible des représentations arborescentes.

2.3.2. PIVOT "ABSTRAIT"

Plusieurs langages pivot ont été envisagés ; ils correspondent à différents degrés d'abstration (de l'expression dans une langue donnée). Je me bornerai ici à décrire le plus abstrait, et on trouvera dans [36] une présentation très complète d'un pivot de niveau moins élevé, destiné à une réalisation multilingue à moyen terme, dans le cadre des travaux du groupe LEIBNIZ.

Toute expression "pivot" est une arborescence étiquetée sur l'ensemble des "masques de variables". Une variable est définie par un nom et une liste de valeurs particulières. L'ensemble des valeurs possibles d'une variable est :

- l'ensemble des éléments de la liste, plus une valeur "nulle", si la variable est "exclusive".
- l'ensemble des parties de l'ensemble précédent, si la variable est "non-exclusive".
- l'ensemble des entiers inférieurs ou égaux à l'unique élément de la liste, si la variable est "arithmétique".

On distingue d'autre part dans l'ensemble des variables l'ensemble des variables spécifiques du pivot, toutes exclusives et toujours présentes.

2.3.2.1. VARIABLES PIVOT

Voici leurs déclarations, avec quelques commentaires.

NOM	DECLARATION	COMMENTAIRES
Etiquette	ETQ := (PHRASE, TITRE, CIRINT, CIRAPR, DEGRE, COMPAR, COOR, IDENT, SPECIF, QUAL1, QUAL2, APPOS, POSS, RELAT, PARTP, A1, A2, A3, A12).	- sur la racine d'une phrase avec ou sans verbe. - circonstants placés avant ou après les prédicats - pour les deux parties d'une comparaison. - coordination IDENT indique une étiquette égale à celle du "grand-père" (cf 2.3.2.2.). - spécificateur (démonstratif, cardinal, ordinal). - qualificatifs forts et faibles. - apposition, complément déterminatif. - relative, participiale - Actants. A12 désigne A1 et A2 (réfléchi).
Unité lexicale	UL := valeurs définies dans les dictionnaires	
Désignation	DES := (DEC,INT, EXC, IMP).	- déclaratif, interrogatif, exclamatif, impératif,
Détermination	DET := (DEF,IDF,VID).	- défini, indéfini, vide.
Mode	MOD := (IND,CDL,NEU).	- indicatif, conditionnel, neutre - pour l'infinitif ou la nominalisation-.
Aspect	ASP := (ACH,INC).	- Achevé, inachevé.
Temps	TIM := (ANT, PST, ATL).	- antérieur, postérieur, actuel. La combinaison des valeurs de ASP et TIM permet de déterminer le temps syntaxique d'une langue donnée.
Référence	RFR := (ELD, REP,ATR,SBT,RPL).	- élidé, répété, attribut, substitué, remplacé. REP indique que l'UL est référée par un pronom. ATR sert pour l'attribut de l'objet, SBT pour le substitut habituel (relatives et participiales), et RPL pour le remplacement de l'actant 1 dans les infinitives.
Emphase	EMP := (THM,THN,DNG).	- thème, thème négatif, dénégation.
Assertion	ASS := (AFF,NEG).	- affirmatif, négatif.
Nombre	NBR := (SIN,PLU,CRD,PLT).	- singulier, pluriel, cardinal, plurale tantum.
Sexe	SEX := (MAS,FEM)	- au sens sémantique, pas morphologique !

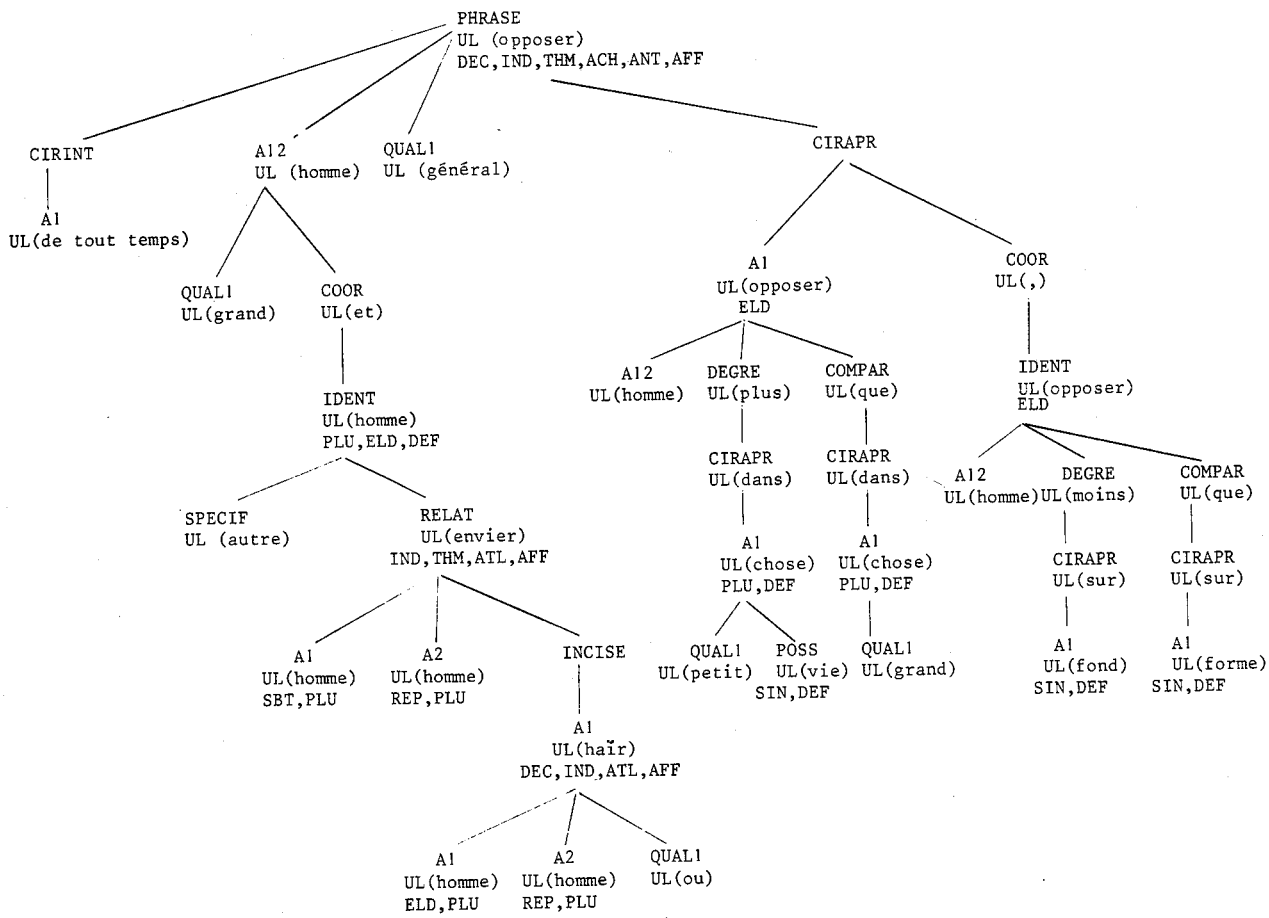
2.3.2.2. SYNTAXE DU LANGAGE

Donnons-la sous une forme de Backus généralisée (dérivations vers le vide), pour simplifier la notation. On sait qu'il existe une grammaire hors-contexte équivalente. L'axiome est <sentence> .

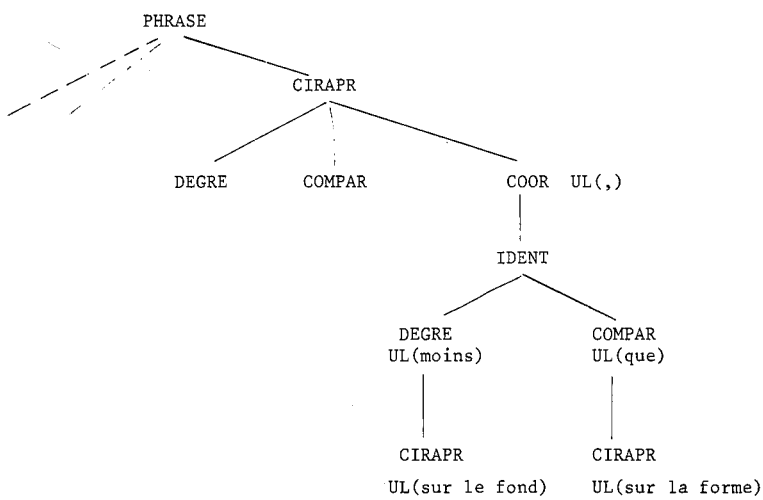
```
<sentence>        ::= PHRASE ( <début> , <noyau> , <fin>)| TITRE (A1 ( <contenu> )).
<debut>           ::= | CIRINT ( <debut> , A1 ( <contenu> ), <fin> ).
<noyau>           ::= <modification> , <arguments> , <qualification> .
<fin>             ::= <circonstance> , <corrélation> , <coordination> .
<contenu>         ::= <début> , <noyau> , <fin>|<qualification> , <fin> .
<modification>    ::= |MODAL (<modification> , <fin> ).
<arguments>       ::= A1 (<contenu> )| A12 ( <contenu> )| A1 ( <contenu> ), A2( <contenu> )|
                      A1 (<contenu> ), A2 ( <contenu> ), A3 ( <contenu> ) | A12 ( <contenu>), A3(<contenu>).
<qualification>   ::= <spécification> , <appréciation> , <apposition> , <possession> , <relation> .
<circonstance>    ::= |CIRAPR (<début> , A1 (<contenu>), <fin> ).
<corrélation>     ::= |DEGRE (<fin>, COMPAR (A1 (<contenu>)))| DEGRE (<fin>).
<coordination>    ::= |COOR (<début>, IDENT (<contenu>)).
<spécification>   ::= |SPECIF (<spécification>, <fin>).
<appréciation>    ::= |QUAL1 (<épithète>)| QUAL2 (<épithète>)| QUAL1 (<épithète>), QUAL2 (<épithète>).
<apposition>      ::= |APPOS (<contenu>) |INCISE (A1 (<contenu>)).
<possession>      ::= |POSS (<contenu>).
<épithète>        ::= <fin>|QUAL1 (<fin>), <fin>| QUAL1 ( QUAL1 (<fin>), <fin>).
<relation>        ::= |<participiale> , RELAT (<début> , <noyau> , <fin>)| <participiale> .
<participiale>    ::= PARTP (A1 ( <contenu>)) | PARTP ( A2 ( <contenu>)).
```

2.3.2.3. UN EXEMPLE

"De tout temps, les grands hommes et les autres, qui les envient (ou les haïssent), se sont généralement opposés plus dans les petites choses de la vie que dans les grandes, moins sur le fond que sur la forme".



Cet exemple est seulement indicatif, et suppose implicitement certains choix linguistiques. On pourrait par exemple préférer introduire dans le dictionnaire "sur le fond" et "sur la forme" comme des expressions figées (tournures) et obtenir deux unités lexicales correspondantes. De plus, on pourrait simplifier la structure en choisissant de ne pas répéter le verbe élidé (opposer), ce qui donnerait la structure :



Enfin, rappelons qu'on n'a pas mentionné les valeurs de toutes les variables pivot, et à plus forte raison des variables "source". On pourrait y noter que l'incise est entre parenthèses, ce qui peut être pertinent pour un couple de langues déterminé, au moment du transfert. Et bien d'autres remarques seraient également justifiées !

2.3.3. AVANTAGES ET INCONVENIENTS

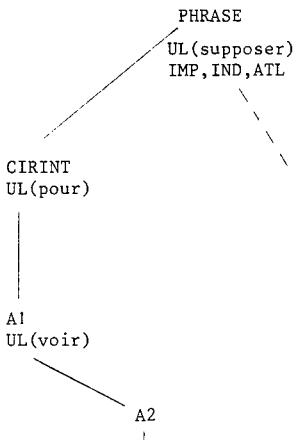
On l'a dit, le principal avantage de ce langage pivot est d'être strictement arborescent, d'où sa clarté et la possibilité d'employer des algorithmes de transductions d'arborescences pour le traiter.

On peut bien sûr l'attaquer sur deux points au moins. Tout d'abord, il n'est pas évident qu'il soit assez complet pour contenir tous les énoncés d'une langue naturelle exprimable de façon arborescente : il faudrait alors modifier sa syntaxe ou ses variables, ce qui ne serait pas trop lourd de conséquences.

On peut aussi se demander si tous les énoncés d'une langue naturelle peuvent se représenter de façon arborescente. On a indiqué dans l'exemple précédent comment représenter des structures syntaxiquement discontinues (coordination, cas de "respectivement"), en introduisant dans la structure des sommets élidés. Par contre, on peut se demander comment traiter des cas d'anaphore plus complexe que ceux prévus (par les valeurs de RFR), par exemple : "la sphère est fermée dans l'espace ; pour le voir, supposons que ...".

On peut remarquer que ce problème est mal posé : en effet, on peut toujours, semble-t-il, trouver un énoncé équivalent dont la représentation (la même, par équivalence) est aisée à construire. Ici : "pour voir que la sphère est fermée dans l'espace, supposons que..". Le problème ne semble pas être l'existence d'une représentation pivot, mais sa construction, i.e. l'analyse. Et il n'est pas interdit au linguiste d'utiliser tout un jeu d'autres variables propres à la langue utilisée, dont certaines destinées à traiter ces cas, au cours de l'analyse. Là encore, il semble qu'on doive faire un compromis entre la qualité de la représentation trouvée et le coût de l'analyse.

Enfin, notons que ceci n'est pas un artifice pour éviter de rendre compte des "opérateurs anaphoriques" [42] Bien au contraire, une analyse qui produirait le résultat suivant l'aurait nécessairement mis en évidence. Elle aurait directement formé ce que Lecomte appelle la "condensation" des deux énoncés [31] . Notons que la représentation linéaire qu'il utilise est strictement équivalente à la représentation quasi-arborescente utilisée dans le système de CETA. D'autre part, elle ne semble pas prévoir d'opérateurs d'anaphore assez complets pour traiter les cas où le renvoi est au niveau d'un énoncé et non d'un argument.



III - D'AUTRES SYSTEMES DE REPRESENTATION DU "SENS"

3.1. INTRODUCTION

Dans cette partie, je passerai en revue quelques modèles de représentation du sens, implémentés ou non, afin de voir quels types de structures ont été retenus en fonction des applications envisagées. Ces applications conduisent d'ailleurs à donner au mot "sens" différentes interprétations. Peut-être un peu arbitrairement, on peut distinguer entre les modèles qui donnent une représentation au niveau de L2 et ceux qui vont jusqu'à L3. Les premiers, pourrait-on dire, sont purement "linguistiques" et semblent être adéquats pour une TA de haute qualité. Ils utilisent des structures arborescentes. Les seconds, plus "psychologiques", cherchent à permettre une représentation du processus même de compréhension. Ils sont plus orientés vers des buts d'IA, et les structures qu'ils utilisent sont plus complexes (réseaux ou "sur-réseaux").

Dans ces deux cas, la structure d'un énoncé particulier reste arborescente. Mais la structure de la base de données est différente : pour les premiers, c'est une collection de définitions analogues à celles d'un dictionnaire classique, mises sous une forme profonde (pivot) arborescente. La base de données n'est pas modifiée par l'analyse d'un texte. Pour les seconds, la base de données a une structure de réseau et est modifiée après l'analyse d'un texte. Elle représente à la fois la connaissance générale du monde et la compréhension de texte traité, dont les énoncés se trouvent reliés entre eux et avec la "connaissance de base".

Dans chaque groupe, j'ai choisi trois modèles qui ont fait l'objet de publications et sont implémentés ou destinés à l'être. Tout d'abord, j'essaierai de présenter le "dictionnaire sémantique" développé à Moscou, le système §§ de "génération sémantique" de R.F. Simmons et celui de "sémantique préférentielle" dû à Y. Wilks ; et ensuite les systèmes de "compréhension de langues naturelles" de M.R. Quillian, T. Winograd et R. Shank.

3.2. DES SYSTEMES ORIENTES VERS LA TA

3.2.1. REPRESENTATION DU CONTEXTE SEMANTIQUE DANS UN DICTIONNAIRE

Il existe à Moscou un groupe dont les membres appartiennent à divers instituts et qui s'occupe de linguistique formalisée, en vue de l'application à la TA. Au sein de ce groupe, A.K. Zholkovskij et I.A. Mel'tchuk ont présenté leur fameux "modèle sens-texte" [66,67], et quelques articles d'un dictionnaire "sémantico-combinatoire" [3]§.

Z.M. Shaljapina et N.G. Arsenteva ont développé les principes de constitution d'un dictionnaire analogue, destiné à la TA. Le système de TA prévu comporte deux dictionnaires de même forme, dont certaines zones sont propres, pour chaque langue, au couple de langues traitées (ici, anglais-russe), et une grammaire (de dépendance).

---

§ Voir aussi Mel'tchuk (1974), p. 113-133.

§§ On revient à ce terme pour désigner un ensemble de programmes.

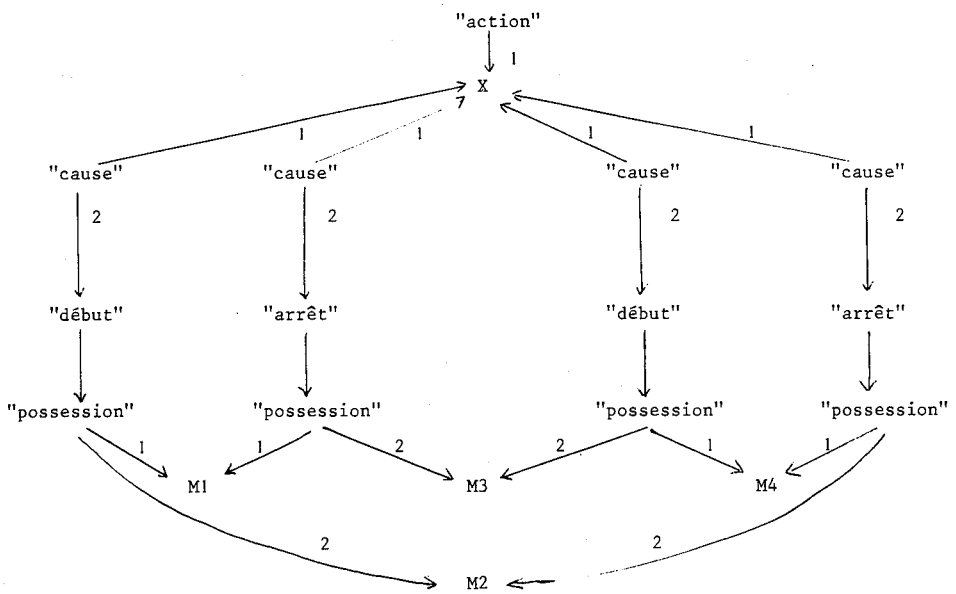
En gros, chaque dictionnaire contient quatre zones, pour la morphologie, les dérivations et le contexte sémantiques, le fonctionnement syntaxique, et la dernière pour les transformations ("fonctions lexicales").

C'est la représentation du contexte sémantique qui nous intéresse. Nous donnerons l'exemple d'un tel contexte et de la représentation d'un énoncé.

EXEMPLE : Voici un article de dictionnaire pour le verbe "acheter" (en transposant à un système français-russe)

ARTICLE	COMMENTAIRES
01 ACHETER 1(2)	Unité lexicale "acheter", premier sens sur les deux retenus (acheter, suborner).
011 ACHET	Base
11 Vrb	Catégorie morphologique
121 1 ПОКУПАТЬ	Sous-catégorie
212 213 "action et $\mathcal{D}_1$ ("cause") $\xrightarrow{1}$ "début" $\longrightarrow$ "possession" et $\mathcal{D}_1$ ("cause") $\xrightarrow{1}$ "arrêt" $\longrightarrow$ "possession" et $\mathcal{D}_1$ ("cause") $\longrightarrow$ "début" $\longrightarrow$ "possession" et $\mathcal{D}_1$ ("cause") $\longrightarrow$ "arrêt" $\longrightarrow$ "possession"	Description du "contexte sémantique". Les index sur les flèches donnent la place d'argument de la partie droite. Les index sur les renvois ont de plus une fonction d'identification. Notons l'utilisation d'"atomes", sémantiques aux propriétés figées ("action", "début", ...).
22 [Jean achète à Pierre un ballon pour deux francs] .	
31 VERBE	Catégorie syntaxique.
321 <table><tr><th>M1 [acheteur]</th><th>M2 [achat]</th><th>M3 [prix]</th><th>M4 [vendeur]</th></tr><tr><td>SBS</td><td>1.SBS 2.PRON(A)</td><td>1.<pour→SBS 2.<au prix de →> SBS 3.SBS</td><td>1.<à → SBS> 2.PRON(D)</td></tr></table>	Fonctionnement syntaxique. C'est un verbe à 4 actants. On donne leurs fonctions dans l'arborescence constituant la structure de dépendance d'un énoncé où ce verbe apparaît.
322 M1 = "personne"; M2 ≠ "prédicat", "caractéristique", "grandeur" ; M3 ≠ "personne"; M3 = SNUM ; M4 = "personne", M31 = SNUM, "grandeur" ; M32 = SNUM	Propriétés sémantiques des actants
323 Il y a M2	Actant obligatoire
41 Conv $\overset{=}{42.31}$ vendre S1 = acheteur ; S2 = achat ; S3 = prix ; S4 = vendeur	Fonctions lexicales, permettant de relier systématiquement les mots de la langue. Voir encore [66] .

Remarquons l'écriture particulière du "contexte sémantique". Il en existe une écriture quasi-arborescente équivalente. Nous remplaçons ici la notation " $\overset{\sigma}{\dashrightarrow}$ " de substitution par l'identité des sommets. On a alors :



Enfin, M1,M2,M3 et M4 sont en fait des paramètres de la structure. X est son "definiendum", et l'écriture choisie a pour but de préciser sa place dans la structure. Si cette structure était étiquetée par un masque de variables et pas seulement par des étiquettes d'atomes sémantiques ou de paramètres, il est clair qu'on pourrait utiliser une variable dont les valeurs seraient ces paramètres et travailler sur la structure arborescente sous-jacente.

D'ailleurs, le modèle d'analyse (en dépendance) extrait une ou plusieurs structures arborescentes à partir du graphe dont les noeuds sont les mots de la phrase et les arcs les relations de dépendance possibles entre ces mots.

3.2.2. LES "RESEAUX SEMANTIQUES"

Examinons maintenant le système de "réseaux sémantiques" présenté par R.F. Simmons et J. Slocum [48,49] . Le bu poursuivi est de fabriquer des phrases "sémantiquement cohérentes" à partir d'une représentation sémantique profonde. L'analyse et la synthèse utilisent des grammaires en "réseaux de transitions augmentés" [65] .

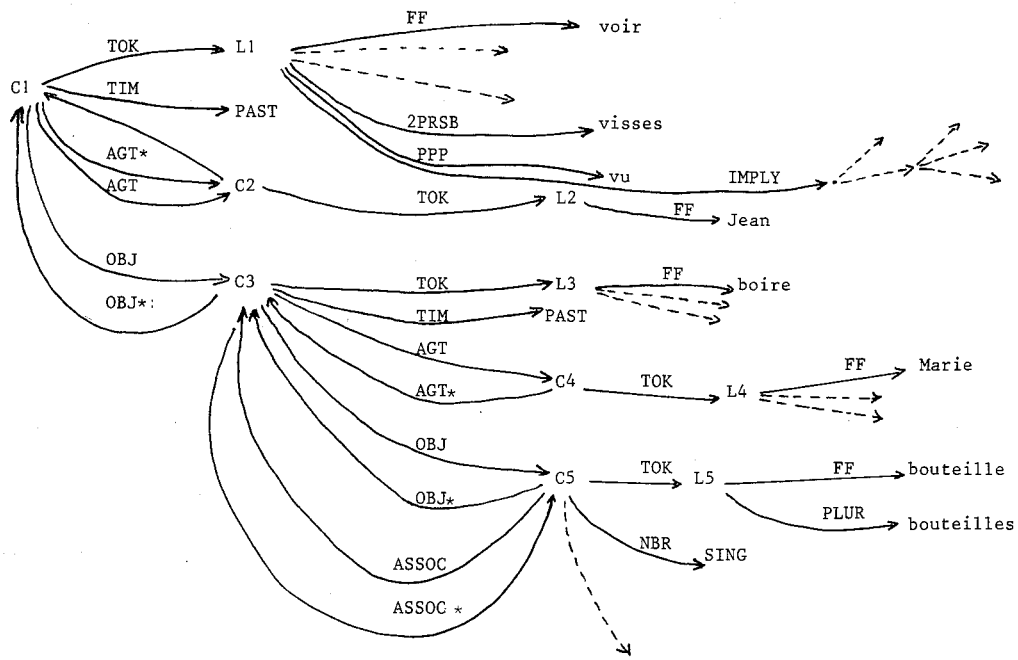
Le terme de "réseau sémantique" semble contredire ce qu'on avait annoncé dans l'introduction. En fait, si l'implémentation de la base de données est celle d'un réseau (S-graphes de LISP), sa structure réelle est arborescente comme on va le voir.

Les ambitions du système sont d'être un "modèle des idées", de permettre une grande aisance de calcul et de donner une bonne adéquation linguistique. La théorie linguistique invoquée a été exposée par Fillmore et Celce-Murcia. On utilise 6 "cas profonds" (cause, actant, thème, lieu, source et but). Il est seulement important de savoir qu'on a un nombre fini et déterminé d'arguments possibles.

Dans toutes les phases d'un traitement (analyse vers la structure profonde et génération en "thématisant" certains points de cette structure), on utilise un dictionnaire profond qui est une collection de représentations sémantiques.

Une représentation sémantique est un réseau dont les sommets sont des concepts et les arcs des relations sémantiques (cas profonds, relations ou attributs comme l'unité lexicale (TOK), le temps (TIM) ou la détermination (DET). L'unité lexicale elle-même renvoie à ses différentes réalisations.

Exemple : "Jean a vu Marie boire la bouteille"



Remarquons que certaines relations sémantiques ont des inverses, de façon à pouvoir ensuite générer un énoncé en partant d'un sommet Ci arbitraire. C'est le principe du paraphrasage, qui consiste ici à "thématiser" différemment la structure sémantique. On pourra par exemple appeler en génération GR(C4;C1) et obtenir "Marie buvait la bouteille". Jean l'a vu".

D'autre part, on voit très clairement que, par un souci d'uniformité, on représente de façon identique les relations de syntaxe profonde, les relations de type logique (IMPLY) et les attributs. Il est clair qu'on pourrait représenter les attributs de façon bien plus compacte en utilisant un masque de variables. Enfin, la structure sans inverses ni relations logiques est arborescente. La représentation complexe utilisée a pour effet d'éviter toute transformation d'arborescence compliquée : en suivant la grammaire de génération, on parcourt ce graphe et on marque les sommets parcourus et/ou explicitement réalisés, de façon à générer un discours cohérent et correct (utilisation de pronoms, d'ellipses ...).

Enfin, le système comporte des fonctions lexico-syntaxiques (au sens de Mel'tchuk) permettant de paraphraser en changeant les mots utilisés ("gagner" est le converse de "perdre", et (sic) Bonaparte = Napoléon, par exemple).

On voit bien que ce système est un "modèle des idées" au sens de L2, c'est-à-dire des équivalences formelles entre différentes structures, vraies indépendamment d'un texte particulier. Ce n'est certainement pas un modèle de la formation des idées. En fait, le modèle reste strictement combinatoire, comme les modèles "syntaxiques" : il s'agit de combiner des fragments de "connaissance" à des fragments de texte et de procéder à des identifications pour l'analyse et à un développement pour la synthèse. Pour l'instant, le système ne permet pas de relier les énoncés successifs pour trouver leur signification relativement à une situation particulière et en tirer les conséquences (i.e. exécuter certaines commandes, répondre à des questions ...). Son grand avantage se situe au niveau du traitement, puisqu'on utilise le même dictionnaire pour l'analyse et la synthèse, et le même type de grammaires.

3.2.3. LA "SEMANTIQUE PREFERENTIELLE

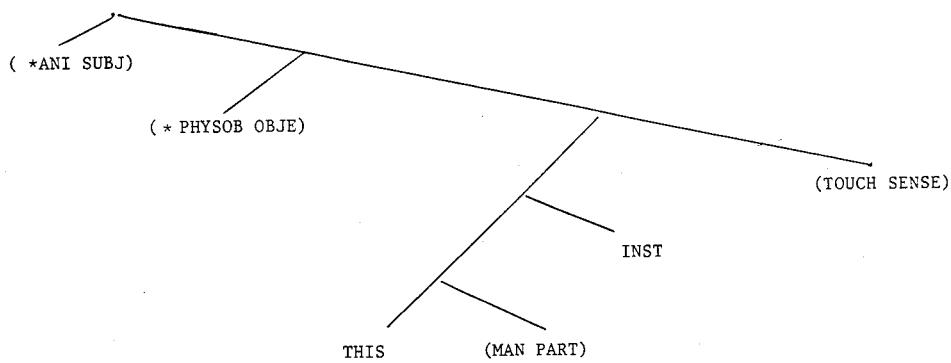
Le but de ce modèle, implémenté par Y. Wilks et son équipe [58 à 62], est analogue à celui de Simmons. Il s'agit de "comprendre" des textes écrits en langue naturelle. L'accent est mis sur la résolution des ambiguïtés, plus dans une optique de T A que d'I A. En particulier, il n'y a pas de représentation "conceptuelle" du sens global du texte traité.

Par contre, on fournit effectivement des traductions exactes (en français) dans des cas assez difficiles d'anaphores. Enfin, la notion de "préférence" s'oppose fortement à celle de "déduction".

3.2.3.1. REPRESENTATION

Le système contient un "dictionnaire sémantique" dans lequel on associe à chaque mot un arbre binaire qui représente son "sens", son comportement syntaxique et éventuellement ses caractéristiques morphologiques. Il y a environ 70 "éléments sémantiques", parmi lesquels plus de 10 "cas profonds", des actions (STRIK, CHANGE, ...) ou des classes (THING, MAN, EVNT, ...). Chaque noeud de l'arbre dépend normalement de son "frère droit".

Exemple : "grasp" (1° sens) : ((* ANI SUBJ)((* PHYSOBJ OBJE)(((THIS (MAN PART)) INST)(TOUCH SENSE)))))



Remarquons que, comme dans le dictionnaire développé à Moscou, on trouve dans chaque article une description selon les trois niveaux traditionnels. Mais il n'y a pas de structure préétablie pour chaque article. Dans la présentation de son système, Wilks insiste sur le fait qu'il n'y a pas de séparation entre ces trois niveaux. Cela ne veut pas dire qu'on ne fait aucune distinction théorique , mais seulement que tous les renseignements sont placés sur le même plan.

Le traitement s'effectue en fait selon des phases traditionnelles. L'analyse morphologique consiste à chercher la forme examinée dans un dictionnaire de formes, et à la remplacer par les renseignements obtenus, sous forme d'une liste d'arborescences binaires. Il est certain qu'il faudrait un outil plus sophistiqué pour l'analyse morphologique de langues flexionnelles comme le russe ou l'allemand.

Ensuite a lieu l'analyse syntaxique. Elle consiste à fragmenter l'énoncé en fonction des classes des mots, de façon à retrouver des "gabarits" ("templates") sous-jacents , et à les assembler.

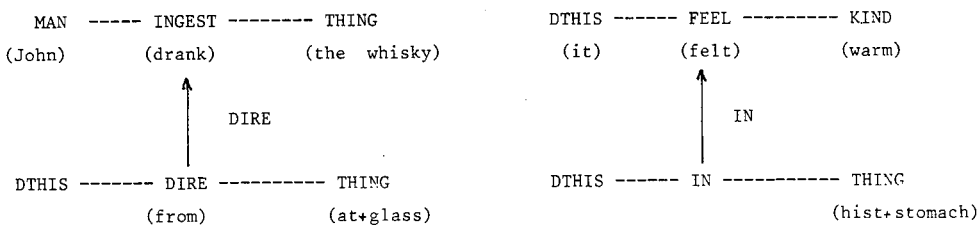
Exemple : "Small men sometimes father big sons"

On trouvera deux séquences ("homophrases" pour Tseïtin") :

KIND MAN HOW MAN KIND MAN
KIND MAN HOW CAUSE KIND MAN

Un "gabarit" est un triplet agent-action-objet précisé par les classes de chaque élément. MAN-CAUSE-MAN en est un, mais pas MAN-MAN-MAN, ce qui lève l'ambiguïté. On pourrait dire qu'un tel triplet est l'analogue d'un schéma de lexis [42] . Le but de l'analyse est d'expliciter tous ces énoncés élémentaires présents dans la phrase donnée et leurs relations. Une telle relation est un "cas profond" et relie un gabarit à un sommet d'un autre gabarit.

Exemple : "John drank the whisky from a glass and it felt warm in his stomach"



Les spécificateurs (articles, adjectifs, ...) sont directement attachés aux spécifiés. On voit clairement que la structure obtenue est arborescente. La technique utilisée pour "préférer" une solution à une autre consiste à renforcer le poids de certains liens si une condition se trouve vérifiée (si par exemple l'agent de "boire" est bien animé, comme le demande la description de ce verbe). On fait le total et on choisit la solution la plus lourde, la plus "sémantiquement dense". C'est ainsi qu'on préfère le sens "d'escroc" à celui de "houlette" pour le mot "crook" dans la phrase "The big policeman interrogated the crook".

Dans l'ensemble précédent, on aura aussi résolu la première "place vide" (DTHIS) en y mettant le seul référent possible (whisky). De plus, le référé de la seconde place vide est (par construction) le même que celui de "it".

3.2.3.2. LES INFERENCES

Le problème posé consiste à résoudre les ambiguïtés anaphoriques. On commence d'abord à appliquer la technique précédente en substituant au référent les divers référés possibles et en calculant la densité du résultat. Il se peut qu'on échoue ("The soldiers fired at the women and they fled"). Dans ce cas, on fait toutes les "extractions" possibles à partir des gabarits contenant le référent et/ou ses référés possibles.

Une "extraction" consiste à expliciter les relations "casuelles" présentes dans les définitions. Dans l'exemple précédent, on aurait :

```
      THING  --- IN  --- THING
    whisky         in      John+part          , car la définition de boire dit que l'objet va dans une partie de l'agent.
et  ? DTHIS -- IN   ----  THING
        it         in        his+stomach      , directement à cause du cas "IN".
```

On cherche à identifier un gabarit contenant la question avec une autre contenant une réponse. Dans cet exemple, on a la solution. On a donc trouvé le référé par une technique de type combinatoire qui consiste finalement, en termes de traitements d'arborescences, à extraire des sous-arborescences réalisant certains schémas simples (à trois sommets) et à les comparer.

En cas d'échec, on a épuisé les méthodes de type analytique et on a recours à des "règles du sens commun", qui sont des déductions faibles. Une telle règles est un couple de gabarits et la correspondance entre leurs sommets. Pour exprimer par exemple que, si l'on frappe quelqu'un, il tombe, on écrira :

```
(1(THIS STRIK) (*ANI2))   ↔   ((*ANI 2) (NOTUP BE) DTHIS)
    (frapper)                          (tomber)
```

La correspondance est indiquée par les numéros portés par les sommets concernés. On applique toutes les règles possibles à l'ensemble des gabarits ou "extractions" contenant la question et/ou ses réponses possibles, et on cherche encore une fois à identifier. En termes de traitements d'arborescences, on a donc effectué quelques transformations simples sur un ensemble d'arborescences (qui sont conservées : on ajoute de nouveaux éléments à cet ensemble).

En dernier recours, on fait l'hypothèse que le référé est ce dont on parlait précédemment.

Remarquons que les "inférences sémantiques" ne sont sémantiques que par leur contenu. Leur forme est exactement celle d'une transformation d'arborescence. L'identification est évidemment un cas particulier de la recherche d'une sous-arborescence dans une autre [10,11].

3.2.4. COMPARAISON

On peut comparer ces trois modèles en se fondant sur au moins trois critères : le type de représentation profonde utilisé, son implémentation et les algorithmes de traitement, qui déterminent la puissance finale du modèle.

Dans ces trois modèles, la structure des énoncés et celle de la base de données est fondamentalement arborescente. Si les deux derniers mettent tous les renseignements sur le même plan, le modèle de Moscou est plus structuré, puisque le dictionnaire est hiérarchisé et que les renseignements morphologiques, syntaxiques et sémantiques sont séparés. Ceci devrait permettre une implémentation plus efficace, puisqu'aussi bien le traitement a toujours lieu en phases distinctes, quoi qu'on en dise. D'autre part, le modèle de Wilks est le seul à être indépendant du langage, puisque la représentation profonde possède un lexique propre. C'est à mon avis une grande qualité, puisqu'on peut définir des règles valables pour des classes de mots (STRIK recouvrant strike, batter, fire, hit, ...). Il resterait toutefois à s'assurer du bien-fondé des hypothèses linguistiques.

Le système moscovite, pour de multiples raisons, n'est pas implémenté§ Cependant, le formalisme adopté permet du supposer que le "contexte sémantique" aura une représentation arborescente ou quasi-arborescente. Les deux autres, écrits en LISP, n'utilisent pas toute la puissance de ce langage et contiennent des structures strictement arborescentes.

Enfin, le traitement. Les algorithmes définis à Moscou ne sont pas implémentés, mais ont été définis très précisément et testés à la main. Ils consistent, après l'analyse morphologique, à établir toutes les relations de dépendance possibles entre les mots, à "nettoyer" le graphe obtenu en applicant certaines règles générales, puis à extraire toutes les structures de dépendance vérifiant une grammaire globale (projectivité, conditions de cohérence des traits, ...) et les règles syntaxiques particulières aux mots de la phrase, et présentes dans le dictionnaire, et enfin, pour terminer l'analyse, à résoudre les ambiguïtés restantes en utilisant le contexte sémantique d'une manière sans doute analogue à celle de Wilks -ce point reste obscur pour l'instant- Ces deux modèles vont sans doute, et celui de Wilks certainement, au-delà de la pure combinatoire : on introduit la notion d'estimation, c'est-à-dire déjà celle d'heuristique. La conséquence fondamentale est qu'on n'élimine une solution que s'il y en a une meilleure, alors que les méthodes purement combinatoires peuvent conduire à ne trouver aucune solution.

L'apport de Y. Wilks est donc d'avoir prouvé que l'on peut résoudre un bon nombre de problèmes épineux d'ambiguïtés sans aller jusqu'au niveau de la compréhension "totale", en utilisant des structures simples, mais en introduisant à tous les niveaux des traitements heuristiques.

3.3. DES SYSTEMES ORIENTES VERS L'I.A.

Les trois systèmes suivants, dus à M.R. Quillian, T. Winograd et R. Shank ont pour but de simuler la compréhension la plus générale, c'est-à-dire au niveau L3. Tous trois utilisent un ensemble de données initiales que la compréhension des messages reçus (et leur exécution, si ce sont des ordres), modifie continuellement. Avant de discuter plus avant des fondements et postulats de ces travaux, présentons-les brièvement.

3.3.1. UN SYSTEME "ENSEIGNABLE" POUR COMPRENDRE LES LANGUES NATURELLES : TLC

3.3.1.1. REPRESENTATION ET ALGORITHMES

La représentation utilisée par M.R. Quillian [40] est fondée sur la distinction classique entre unités et propriétés.

Une unité est définie comme "un objet, un évènement, une idée, une affirmation..." ou encore tout ce qui peut s'exprimer en anglais au moyen "d'un mot, d'un groupe nominal, d'une phrase ou d'un fragment plus long de texte", c'est-à-dire à peu près n'importe quoi. On suppose de plus, et on impose dans la structure, que l'univers des unités soit une hiérarchie : toute unité est un "raffinement" d'une autre unité, sa "super-unité". Pour différencier une unité de sa super-unité, on lui associe un ensemble de propriétés qui servent justement à la raffiner.

Une propriété est une propriété simple éventuellement raffinée par d'autres propriétés. Une propriété simple est le couple d'un attribut et d'une valeur. L'attribut est une unité de "type prédicatif", c'est-à-dire qui correspond en anglais aux verbes et aux prépositions. Mais il se peut qu'il n'existe pas de mot de la langue qui l'exprime. La valeur est une autre unité qui correspond au second argument de la relation induite par l'attribut (et dont le premier est une unité raffinée par cette propriété). Cette valeur peut correspondre aux objets des verbes,

§ L'implémentation d'un système voisin, destiné à la communication homme-machine, est en cours à Novosibirsk : voir Ershov, Narinyjani, Mel'tchuk (1975).

aux compléments introduits par des prépositions ou aux adjectifs. On aurait par exemple la propriété (goût salé).

Le lien avec la langue est fait en utilisant un dictionnaire qui associe à chaque mot un ensemble d'unités. réciproquement, une unité peut renvoyer à un mot du dictionnaire.

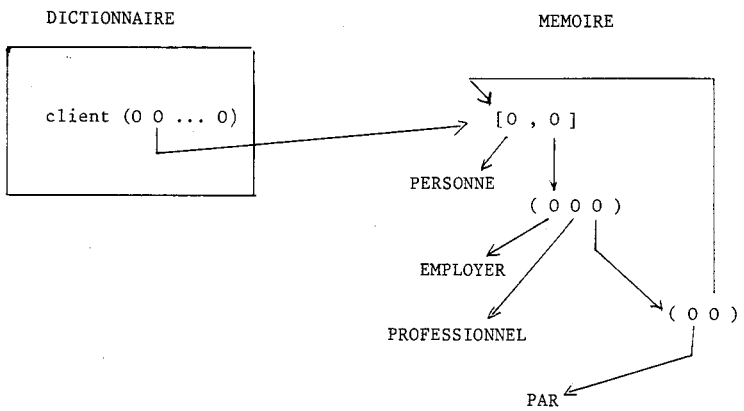
Exemples :

"Jean" sera une entité précisant
- que c'est une personne (super-unité)
- qu'il est français, etc.. (propriétés)

"Jean à neuf ans" sera une autre entité
- de super-unité "Jean"
- de propriétés : "avoir neuf ans", "être anglais", etc...

Sur cet exemple, on voit qu'une même propriété peut avoir diverses valeurs pour une entité et sa super-unité, ce qui n'a pas d'inconvénient, puisque le raffinement n'est pas l'implication, mais plutôt la précision.

Si un des sens du mot "client" est "celui qui emploie un professionnel", on aura une représentation en mémoire comme :



Les mots en majuscules ou les crochets droits indiquent des unités, les parenthèses des propriétés.

Ainsi, la structure de la mémoire est celle d'un multigraphe étiqueté (par : "super-unité", "propriété", "attribut" ou "valeur"). L'interconnexion des unités et propriétés entre elles est arbitrairement complexe. Cependant, on pourrait dire qu'on a là un dictionnaire de définitions : un client est "une personne qui..." et employer veut dire "..." et un professionnel est "une personne qui ...".

Pour "comprendre" une phrase, TLC crée une nouvelle unité, initialement vide, de ses super-unités possibles, parmi lesquelles les unités indiquées par le dictionnaire et les référents possibles dans le texte. Puis il cherche, pour chaque candidat, à relier ses propriétés (parcourues sur une certaine profondeur, en recherche "horizontale" ("breadth first") à d'autres mots de la phrase, sous certaines conditions grammaticales, associées aux attributs des propriétés. Si par exemple on parle de "client de l'avocat", on examinera la propriété de client, on marquera les unités "employer" et "professionnel", puis en examinant la super-unité d'avocat, soit "professionnel", on trouvera une intersection. Alors, dans la nouvelle unité créée pour le mot "client", on fera pointer la valeur de la propriété correspondant à "par" sur la nouvelle unité créée pour le mot "avocat", si la condition portée par l'attribut de cette propriété ("par") est vérifiée. L'algorithme utilise le back-track et choisit apparemment la première solution complète qu'il trouve. Tout un système de sémaphores permet de savoir si on est déjà passé sur un noeud du graphe, et de quelle façon.

D'autre part, l'utilisateur peut définir de nouvelles conditions grammaticales de façon interactive, et les associer à des unités. Notons encore la possibilité de définir une propriété comme une fonction de plusieurs propriétés, au moyen des opérateurs d'intersection, d'union et d'union exclusive.

3.3.1.2. DISCUSSION

On peut considérer ce système soit comme un programme de communication homme-machine, soit comme un programme de simulation du procédé de compréhension humain. Ce n'est pas équivalent. Dans le second cas, on considère, comme Quillian, que le programme est un "modèle du cerveau" ou plutôt d'une fonction du cerveau. Les deux postulats principaux de TLC sont que :

- le procédé de base de la compréhension consiste à relier le texte à des fragments appropriés de la "connaissance du monde", ce qui est généralement admis.
- toute la connaissance est stockée dans un format homogène.

Il est permis d'avoir des doutes sur ce dernier point. En effet, toutes les recherches linguistiques ont amené à distinguer différents "niveaux" de la langue, dont le fonctionnement est largement indépendant. Il n'est pas de même ordre de dire que "cheval" a un pluriel irrégulier ou que c'est un mammifère domestique ! Et d'ailleurs, Quillian est amené à abandonner en pratique cette notion d'homogénéité : ni le dictionnaire ni les conditions grammaticales n'entrent dans le format standard, non plus que les règles de l'analyseur syntaxique à coupler avec TLC. S'il s'éloigne ainsi du modèle initial où "tout est fait par la sémantique", c'est pour se rapprocher, guidé par des considérations pratiques, d'un modèle plus adéquat. Il est curieux de retrouver la même tendance et la même évolution dans les travaux de Shank et de Wilks : on affirme se placer au seul niveau sémantique et ignorer superbement les problèmes morphosyntaxiques, or ils existent, et on est donc conduit à les traiter de manière ad hoc , puis, pour des raisons d'efficacité, de manière générale, donc séparée, ce qui revient à distinguer un nouveau niveau !

D'autre part, puisqu'il s'agit d'un "modèle psychologique", Quillian cite quelques expériences (temps de réponse) à l'appui de la structure choisie (unités-propriétés). Même s'il disposait d'une "connaissance du monde" écrite dans son format et équivalente à celle d'un homme, et si on obtenait des différences de temps de réponse assez proches sur un nombre de questions assez grand, cela ne prouverait en aucune façon l'exactitude du modèle. Tout ce qu'on peut dire de ces expériences, c'est qu'elles permettent de penser que certains types de concepts sont organisés en hiérarchie (ex : canari < oiseau < animal, etc.). Ceci nous amène à examiner de plus près ce que sont les entités et les propriétés. Leurs définitions, fort vagues d'ailleurs sont purement expérimentales et leur fonctionnement strictement ensembliste. Or, si on fait un parallèle, semble-t-il justifié, avec les "notions" et les "relations" qui sont étudiées par la sémantique formelle, on voit que, là aussi, le modèle est inadéquat. Il fonctionne en effet comme si toutes les notions étaient "distributives" [31,42] . Or il est bien clair que la relation entre "Jean" et "personne" est celle de l'appartenance à une classe, alors que la relation entre "Jean à neuf ans" et "Jean" est d'un autre type : "Jean" est une classe "collective" dont "Jean à neuf ans" fait partie, mais pas au sens d'un élément. De plus, un modèle de la compréhension ne saurait exclure la notion de "mise en situation" d'un énoncé, c'est-à-dire l'aspect communicatoire de la langue (L3).

Enfin, on pourrait dire que l'algorithme de compréhension lui-même, même réalisé par un dispositif de recherche "en parallèle", n'est sans doute pas adéquat. Il est en effet purement combinatoire, et "aveugle" en ce sens qu'on suppose a priori qu'il n'existe qu'une façon de relier le texte à la connaissance, et qu'on choisit donc la première solution trouvée. On s'interdit donc le double sens, ou le "faux sens" corrigé par la suite du texte. De même on ne rend pas compte du mécanisme connu (et bien étudié dans d'autres domaines comme les échecs) qui consiste à ne poursuivre que quelques possibilités, les plus vraisemblables. Remarquons que l'algorithme que Wilks utilise dans un modèle qui ne se veut pas psychologique est sans doute plus adéquat.

Si par contre on considère ce système uniquement comme un système de communication homme-machine, presque toutes ces objections tombent, puisqu'elles concernent l'adéquation. Il reste un modèle très séduisant de l'organisation de la mémoire dans un tel système, lié à un ensemble complexe de procédures de recherche et de vérification dans un graphe, et un système qui est le premier à "apprendre" par interaction avec un "moniteur" humain, dans ce domaine tout au moins.

3.3.2. UN SYSTEME DESTINE A COMPRENDRE ET EXECUTER DES ORDRES

C'est le fameux système implémenté par T. Winograd [63] . Le système est muni d'un "univers" ou "micro-monde" propre : des cubes et des pyramides de différentes tailles et teintes posés sur une table. On peut demander au robot de "mettre la pyramide verte sur le cube rouge", il demandera laquelle s'il y a ambiguïté, exécutera l'ordre et pourra ensuite expliquer le pourquoi de ses actions. Il peut aussi comprendre de nouvelles définitions et les utiliser par la suite.

Pour comprendre un énoncé, on utilise un dictionnaire "procédural", et une analyse syntaxique simple interagissant avec la "sémantique". Dans le dictionnaire, on ne trouve que des formes non-ambiguës, associées à des traits grammaticaux et à des fonctions liées au micromonde (tests, actions, ...). La grammaire "systémique" est en fait d'un type classique et utilise un système de variables hiérarchisé. Par exemple, seule la catégorie du substantif admettra un "trait" (une variable) d'animation. La "sémantique" consiste à relier un énoncé avec la représentation interne du micromonde. Pour cela, on utilise un mécanisme de démonstration automatique de théorèmes. En effet, le caractère construit, logique et fini de l'univers considéré permet de résoudre de cette façon les problèmes d'ambiguïtés. Par exemple, on trouvera une contradiction si on cherche à mettre la grande pyramide verte dans un cube trop petit, il s'agit donc de l'autre. En fait, la représentation d'un énoncé est celle d'un théorème. La preuve de ce théorème permet d'en établir la cohérence et de l'exécuter, si c'est un ordre.

Si ce système applique de façon très spectaculaire les méthodes de démonstration automatique à un problème d'IA, il ne semble pas qu'on puisse y trouver de méthode pour le traitement automatique de textes en langues naturelles. Le monde réel n'est en aucune façon exprimable au moyen de la logique habituelle, et on a plus d'une fois souligné, par exemple, qu'il faut pouvoir y admettre la contradiction [3,58,59] . Ainsi, la composante sémantique d'un traitement assez général ne peut être purement logico-déductive. D'ailleurs, les tentatives de formalisation de la sémantique [31,42] ont buté dès le départ sur l'inadéquation de la théorie des ensembles, classique et tentante, pour représenter les relations entre objets et les niveaux d'abstraction (notion de classe). Même des théories plus adaptées, comme celles de Lesniewski, ne peuvent que servir de cadre à ce genre de travail : ici comme ailleurs, il faut trouver un système de description formelle adapté à son objet. Tout ceci pour souligner que les problèmes linguistiques sont tels qu'on ne peut espérer les résoudre en utilisant directement une théorie mathématique proposée de façon indépendante. Winograd lui-même insiste d'ailleurs sur ce point en concluant [64] .

3.3.3. LA "MEMOIRE CONCEPTUELLE"

Le système développé est implémenté par R. Shank et ses collaborateurs à l'Université de Stanford [45] est destiné à "répondre intelligemment, sur la base de son propre modèle du monde, en réaction à une phrase donnée" (en langue naturelle, ici en anglais). Ce système contient un analyseur qui transforme l'entrée en une "structure conceptuelle profonde", un ensemble de données, la "mémoire conceptuelle", et un ensemble de mécanismes d'inférence destinés à relier différentes informations pour déterminer des référents, ou mieux la possibilité, le but ou la cause de la phrase donnée. Des recherches récentes [24,26] indiquent que la"mémoire conceptuelle" est sans doute un modèle assez adéquat de l'organisation de la mémoire humaine.

3.3.3.1. LA STRUCTURE CONCEPTUELLE PROFONDE

Cette représentation se veut indépendante de la langue ; elle a donc son vocabulaire propre. Les éléments de plus bas niveau (atomes) sont répartis en six catégories conceptuelles : PP (objets réels), ACT (actions réelles), PA (attributs d'objets), AA (attributs d'actions), T (temps) et L (lieux). Il y a par exemple 14 "actions réelles", parmi lesquelles ATRANS (transfert de possession), PTRANS (déplacement d'un objet), SPEAK, LOOK-AT, LISTEN-TO, SMELL et MBUILD (construction par un animé d'une nouvelle information à partir d'une ancienne). Les attributs peuvent être JOY, HEALTH, COLOR, etc... On affirme que ces éléments suffisent pour construire toutes les conceptualisations possibles, ce qui est bien sûr un point délicat.

Avant d'examiner comment on construit une conceptualisation, rappelons la base linguistique de ce modèle, tirée des travaux de Fillmore [21] : chaque action a une rection précise : elle implique un certain nombre d'arguments, chacun à un "cas conceptuel" précis (objectif, réceptif, directif, instrumental). Un argument peut être complexe.

Une conceptualisation est alors la donnée d'une action et de ses arguments, ou encore d'un prédicat (CAUSE, CANCAUSE, ...) et de ses arguments, qui sont des conceptualisations. Il y a 16 règles de formation.

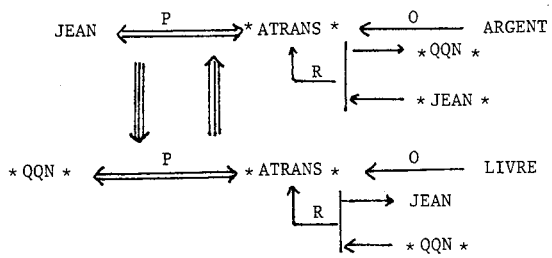
Exemples :

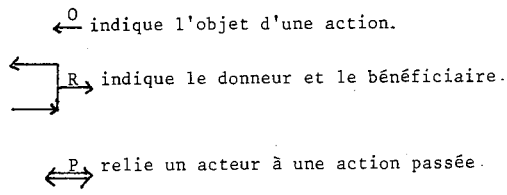
1) PP ⇐ (PA2 ← PA1) indique que l'objet (PP) a changé d'état (PA1 → PA2)

2) X ⇐ Y indique que X a causé Y (CAUSE X Y).

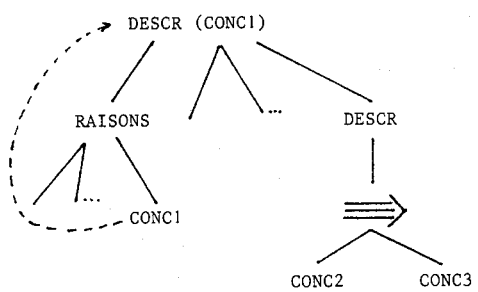
Il est clair que chaque conceptualisation peut s'écrire comme une arborescence de dépendance ("dépendance conceptuelle"). R. Shank a cependant introduit une notation graphique plus compacte.

Exemple : "Jean a acheté un livre"





L'implémentation du système est faite en LISP. Chaque conceptualisation est représentée par un "superatome" muni d'une liste de propriétés (raisons, inférences, vraisemblance, temps, ... et description). La description est un pointeur vers la représentation de la structure de la conceptualisation. Au total, la mémoire est organisée de façon à conserver à la fois les structures arborescentes et la multiplicité des pointeurs. Par exemple, si une conceptualisation fait référence à elle-même, ce ne peut être que par l'intermédiaire de son "nom" (son superatome associé), et sa description reste finie et arborescente. On aura par exemple :



La flèche en pointillé indique que l'on peut trouver la description du CONC1 dans sa liste de propriétés, et c'est bien un atome. Par contre, DESCR (CONC1) est une structure complexe.

3.3.3.2. LES INFERENCES CONCEPTUELLES

Le but du système est de comprendre une histoire, en utilisant des données initiales. L'analyse d'un nouvel énoncé fournit une structure profonde "primaire". Comprendre, c'est alors expliciter toute l'information contenue dans l'énoncé, et ses liens (de causalité, de cohérence, ...) avec l'information existante. D'où la nécessité de procédés d'inférence.

R. Shank les divise en plusieurs catégories. Donnons-en seulement un bref aperçu, pour indiquer la puissance du système.

Tout d'abord, les inférences linguistiques. Elles consistent à prédire la présence d'un objet non mentionné à un certain cas, à partir d'un mot ou d'une construction syntaxique. Par exemple, le mot "acheter" implique la présence d'argent, à moins que le contraire ne soit spécifié. On remarque dès maintenant que les inférences ne sont pas et ne doivent pas être des déductions au sens de la logique formelle. Une histoire peut fort bien être contradictoire, et il faut pourtant pouvoir la comprendre.

Les autres inférences sont indépendantes de la langue. Par exemple, "Jean veut un livre" permet d'inférer qu'il le veut (normalement) pour le lire, et "Jean est parti à Marseille" qu'il y est arrivé. De même, si "Jean frappe Marie", on peut supposer qu'elle est blessée, et ensuite qu'elle est fâchée (sa "joie" a diminué). La structure de la mémoire permet de conserver les chaînes d'inférences, et de modifier ultérieurement la vraisemblance de certaines conceptualisations sur la base d'une nouvelle information qui les renforce ou les contredit.

En conclusion, on peut dire que ce système semble être le premier à définir et à implémenter une structure profonde véritablement indépendante de la langue. Le système est même plutôt un modèle psychologique. La structure des énoncés est une structure arborescente en "dépendance conceptuelle", mais l'organisation de la mémoire permet de les relier arbitrairement comme constituants, inférés, inférents, ... les uns des autres. Finalement, ce système semble être le plus prometteur pour l'I.A. en général, c'est-à-dire pour la simulation des processus généraux de compréhension de la langue. De tous les systèmes examinés ici, c'est le seul à permettre d'espérer une compréhension au niveau de L3, i.e. au niveau de la communication.

IV - POUR UNE COMPOSANTE GNOSTO-ENCYCLOPEDIQUE EN T.A.

J'ai brièvement présenté les six modèles précédents pour montrer quels types de représentation du sens ou de la connaissance ils utilisent et quelle puissance ils atteignent. Par puissance, on entend la profondeur linguistique de l'analyse effectuée : celle de la "mémoire conceptuelle" est certainement supérieure à celle de TITUS. Mon propos est de voir de quelle manière intégrer une sémantique "assez" élaborée dans un système de TA L'objectif est évidemment plus restreint que dans un système d' IA , puisqu'on cherche à résoudre des ambiguïtés au niveau de L1 ou L2, mais pas de L3. D'autre part, il faut autant que possible conserver l'efficacité du système. C'est pourquoi je propose d'utiliser une "référence", ou "connaissance du monde" présentée d'une façon analogue à celle qu'on trouve dans les dictionnaires monolingues, en utilisant des définitions exprimées dans le langage "pivot" du GETA (2.3) . La puissance atteinte doit être sensiblement la même que celle de systèmes sans "inférences". On verra ensuite comment représenter et utiliser des "règles du sens commun", ou règles de "déduction faible", au sens de R. Shank ou Y. Wilks, au moyen de grammaires transformationnelles traitables par le système de "transductions d'arborescences" mis au point au GETA. La puissance atteinte devrait alors être comparable à celle du système de Wilks, tout en gardant une efficacité et une compacité acceptables dans un système de TA.

4.1. LES DIVERSES REPRESENTATIONS DU SENS ET LE "PIVOT".

Dans la troisième partie, on a pu remarquer à plusieurs reprises que la structure des énoncés était arborescente, bien que la structure globale de la base de données ne le soit pas. C'est dire qu'on peut extraire de cette base des énoncés exprimables en langue naturelle, donc en pivot et inversement. Ceci est particulièrement net pour les représentations sémantiques de dictionnaire de Shaljapina ou des phrases dans les systèmes de Simmons et de Wilks. Mais il faut savoir si la collection de ces énoncés suffit à décrire toute l'information contenue dans la base de données. Cela est sans doute vrai pour ces trois systèmes et faux pour ceux de R. Shank et de M.R. Quillian. On mettra à part celui de T. Winograd qui, on l'a vu, est très particulier à un certain type de données, analytique et logique.

Dans le système de Quillian, on a pu observer que la description d'une entité pouvait être faite exactement comme dans un dictionnaire classique : la "définition" comprend la superunité et les propriétés (sous forme attribut-valeur), puis ces termes sont eux-mêmes susceptibles de définitions. Utiliser la stratégie du "breadth-first", c'est parcourir un dictionnaire en se renvoyant successivement aux mots de la définition, puis à ceux de la leur, etc. . Les définitions des entités et les propriétés sont très vagues, mais les exemples donnés montrent qu'en fait on doit pouvoir aisément passer à une représentation "pivot". On en donnera plus loin un exemple.

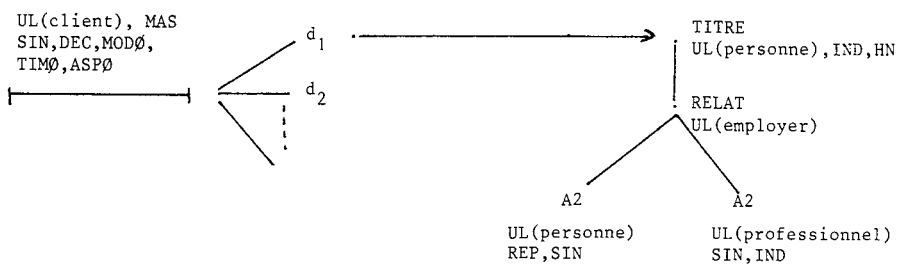
Dans le système de R. Shank, il est clair qu'une conceptualisation est plus qu'une définition, puisqu'elle est décrite dans ses rapports avec d'autres conceptualisations, et que ces rapports sont créés par le processus de compréhension. On peut facilement trouver des cycles dans le graphe formé par les conceptualisations et leurs rapports, mais il est toujours possible de développer la description d'une conceptualisation sous forme arborescente en utilisant l'opérateur "et" et les foncteurs formateurs de conceptualisations comme $\Longrightarrow$ ou $\Rrightarrow$. On revient ainsi nécessairement au niveau des atomes et on obtient une expression transformable en "pivot", mais évidemment au prix d'une grande perte de place (puisqu'on répète le développement d'une conceptualisation dans celui de toutes les conceptualisations qui l'utilisent directement ou non dans leur description). Il ne semble donc pas qu'on puisse, au niveau de la représentation, atteindre efficacement la puissance de cette sorte de réseau "récursif".

4.2. UN DICTIONNAIRE DE "DEFINITIONS PIVOT"

On peut proposer l'organisation suivante, qui consisterait à se rapprocher d'un dictionnaire monolingue, et permettrait peut-être même de construire automatiquement la base de données à partir de son expression en langue source, en utilisant l'analyseur écrit pour cette langue dans le système de T.A. considéré.

Les entrées du dictionnaire seraient constituées par les masques de variables des mots à définir. Le but n'est donc pas de définir des "entités", ou des "notions", mais seulement des mots par rapport à d'autre, de façon très classique et, il faut bien l'avouer, sans plus préciser ce qu'est ou doit être une "définition". On suppose seulement qu'on peut la représenter par une arborescence "pivot", dite "définition pivot". A chaque masque, on associerait donc une liste de définitions pivots.

Exemple : pour définir "client" par la phrase "une personne qui emploie un professionnel", on trouverait :



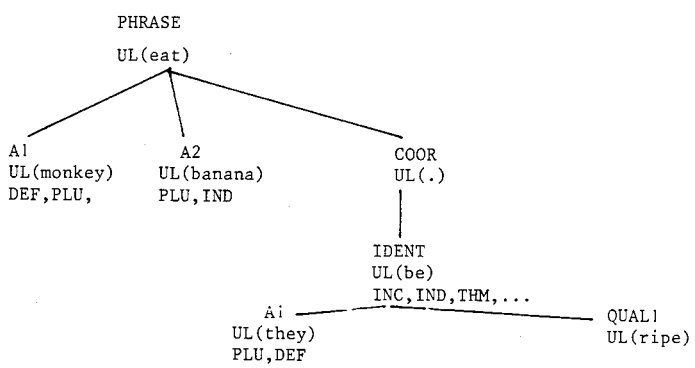
A lire les définitions des dictionnaires classiques, il semble bien, comme le remarque aussi Quillian, qu'un mot soit presque toujours défini par remplacement et précision. Ici, le "remplaçant" est "personne" et la "précision" la relative. C'est pourquoi on peut pour l'instant faire cette hypothèse. Le "remplaçant" se trouve donc toujours en sommet de la "définition pivot".

On n'a fait apparaître dans l'exemple précédent que des variables pivot, mais rien n'interdirait d'utiliser les variables propres à la langue source dans les conditions d'identification, puisqu'aussi bien on cherche toujours à lever les ambiguïtés au niveau de la langue source -quitte à associer ensuite un ou plusieurs équivalents en langue cible à chaque définition.

Pour préciser les idées, on peut indiquer comment utiliser un tel dictionnaire pour résoudre des ambiguïtés anaphoriques. Prenons l'exemple de Wilks :

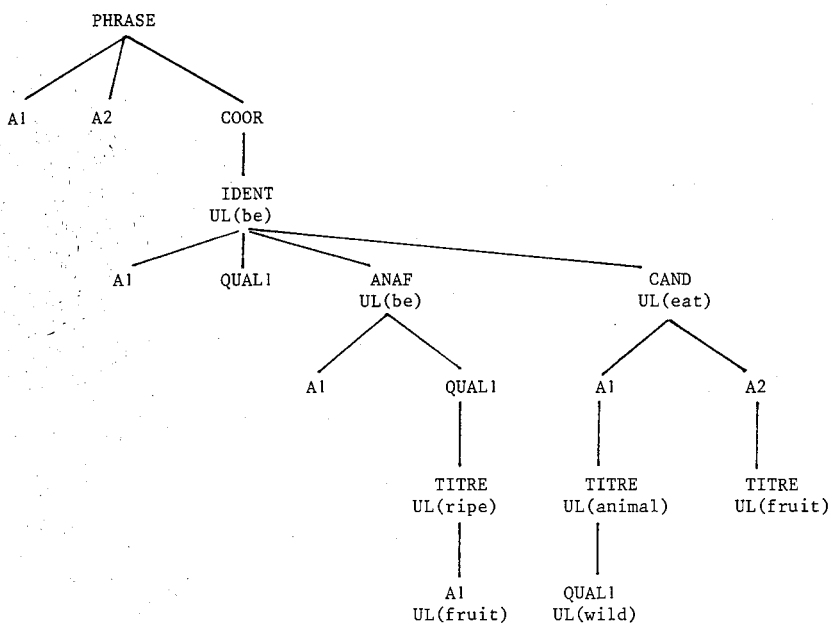
"The monkeys eat the bananas. They are ripe".

Supposons qu'on dispose d'un analyseur morphologique et d'un outil de transformation d'arborescences comme le système CETA, qui permettre de trouver l'analyse :



L'analyse n'est pas terminée, la valeur des variables "source" indiquant que "they" est un pronom : il faut donc trouver son antécédent. Il est alors facile de trouver les "candidats" possibles et, en utilisant une variable "source" particulière ou une nouvelle variable pivot ANTE := (CAND,ANAF,DEFN) de recopier les bouts de construction contenant la ou les anaphores et les candidats. La taille de ces "bouts" de construction

dépend seulement du linguiste qui écrit les règles transformationnelles destinées à effectuer cette recopie. On trouvera, en utilisant une fois les définitions du dictionnaire et en "développant", l'arborescence suivante.

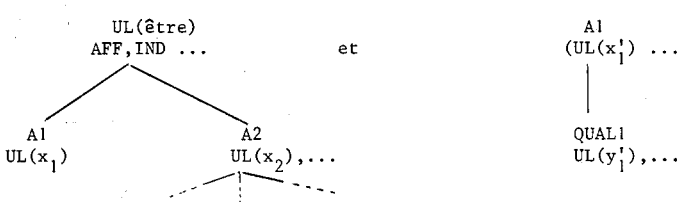


Remarquons que la définition utilisée pour "ripe" doit être obtenue à partir d'un énoncé comme "se dit d'un fruit...". On le considère donc ici comme le prédicat "être mûr". Dans la représentation pivot, on simplifie ceci si l'adjectif qualifie un nom, puisqu'on sait que son "A1" est identique à son "père" (le sommet correspondant au nom qualifié).

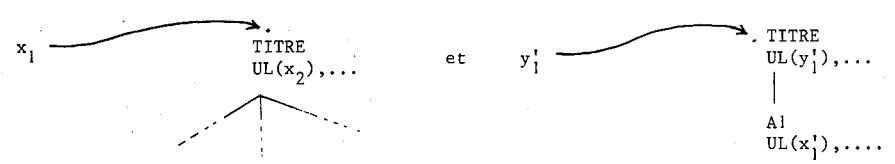
Il est alors simple d'écrire des règles de transformations qui comptent le nombre de coïncidences (ici "fruit") pour chaque candidat, au moyen d'une variable arithmétique, et choisissent le "meilleur". Pour l'organisation et l'écriture des grammaires transformationnelles dans le système du GETA ("CETA"!) on se reportera utilement à [12] . Le type de résolution d'ambiguïtés qu'on peut espérer réaliser ainsi est sans soute analogue à ce qu'atteint le "basic mode" du système de Wilks, puisqu'on vient de voir qu'on peut le simuler à l'aide de ce "dictionnaire pivot" et d'un système de grammaires transformationnelles implémenté pour être efficace sur des volumes importants.

Enfin, on peut remarquer qu'il serait assez malaisé d'introduire automatiquement de nouvelles définitions dans le dictionnaire au fur et à mesure que le texte est traité. Il y a d'ailleurs, outre des difficultés pratiques, des obstacles plus théoriques.

Les difficultés pratiques tiennent à la forme du résultat de l'analyse et à la masse de textes qu'on veut traiter. La forme de l'analyse "pivot" n'est pas directement celle des définitions, il faudrait des règles de transformations spéciales pour isoler les parties de la structure qui peuvent contenir une définition et produire le couple défini-définition sous la forme proposée plus haut. Par exemple, on pourrait supposer que les schémas suivants



donnent lieu à des définitions comme :



Mais la masse de textes qu'on envisage de traiter interdit absolument de redéfinir chaque terme chaque fois qu'il apparaît dans le texte, comme le fait Quillian (qui crée une nouvelle entité pour chaque terme non outil, mais c'est bien sûr la grande majorité).

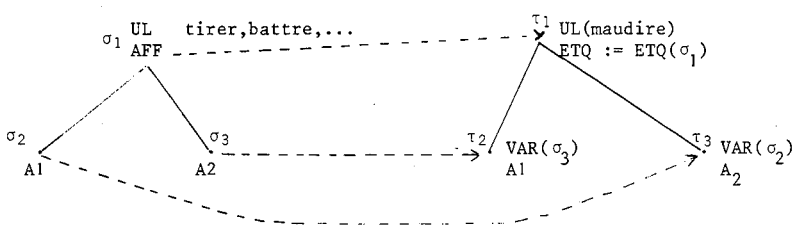
D'autre part, il serait difficile de justifier cette démarche elle-même. Un texte n'est pas habituellement une succession ou une combinaison de définitions. Il construit une structure d'un autre ordre (le "sens" d'une histoire, la "démarche mathématique" dans la preuve d'un théorème); en se servant bien sûr du langage. Habituellement, un texte a rapport avec une situation extérieure, il est écrit par quelqu'un pour quelqu'un dans un certain but, en un mot il est au niveau de L3, alors qu'un dictionnaire de définitions concerne L2.

On pourrait donc imaginer de construire une représentation du "sens" du texte (et ce serait seulement son analyse pivot), à condition de la séparer de la base de données. On peut également concevoir d'augmenter la base de données en introduisant des définitions au cours du traitement, dans un système conversationnel, mais ceci n'est qu'une question d'implémentation.

4.3. INFERENCES ET TRANSFORMATIONS

Dans le paragraphe précédent, on a envisagé l'utilisation d'un dictionnaire de définitions pivot pour résoudre certaines ambiguïtés. On peut penser, et Y. Wilks la bien montré, que cette technique peut être insuffisante : il peut ne pas y avoir de "meilleur" candidat. Effectivement, certaines ambiguïtés semblent ne pouvoir être levées qu'en utilisant des méthodes d'inférence, ou de "déduction faible". Encore une fois, il ne s'agit pas de résoudre une ambiguïté de façon définitive, mais d'une façon assez "vraisemblable" au moment où on traite le fragment de texte en question (phrase, paragraphe...). Il semble en effet qu'on puisse toujours fabriquer un contexte, une situation où la solution choisie sera fausse !

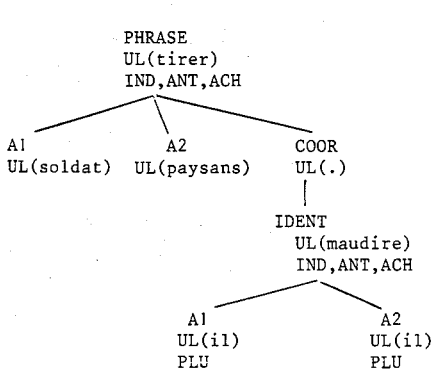
On a déjà remarqué dans la troisième partie l'analogie des "règles du sens commun" ou "inférences" de Y. Wilks avec des règles de transformations d'arborescences utilisées par le système CETA. Il reste donc à indiquer comment employer une règle du type :



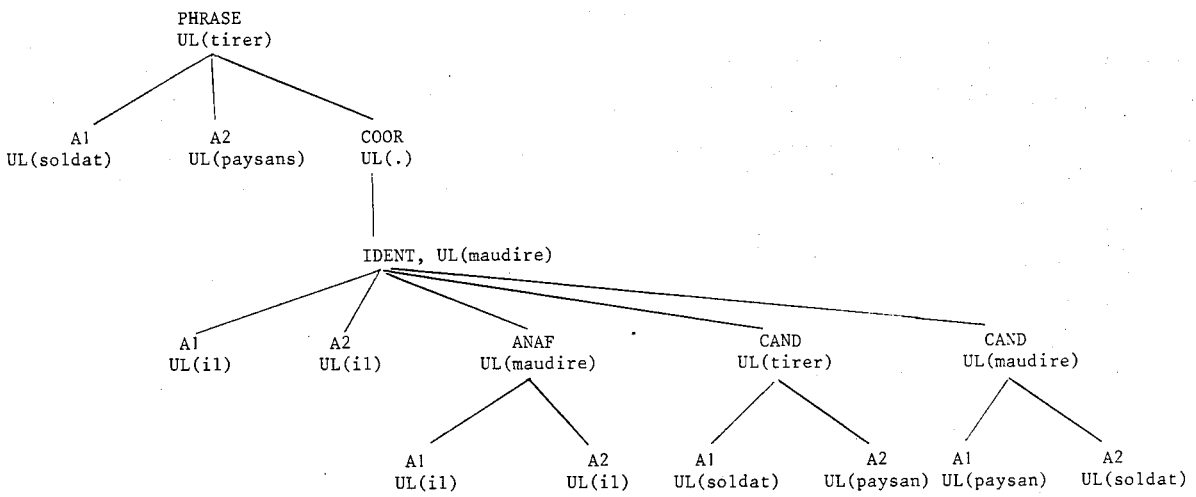
pour résoudre les anaphores dans la phrase "les soldats tirèrent sur les paysans et ils les maudirent !"

Expliquons d'abord brièvement la signification de cette règle : si on trouve dans la structure traitée que A a battu B, on en "déduit" que B maudit A et on transforme la structure en modifiant les variables de la façon indiquée et en raccrochant les "fils" non mentionnés sur le schéma aux extrémités des flèches. σ_1, ..., τ_3 sont des noms de sommets, utilisés seulement au cours de la transformation.

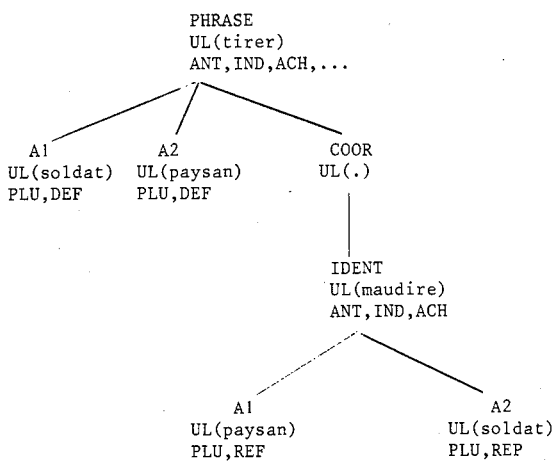
On a donc au départ la structure ci-contre. On peut reprendre la méthode indiquée au paragraphe précédant en utilisant la variable ANTE et en recopiant les parties "candidates" et "anaphores". Pour utiliser les transformations (inférences), on dupliquera chaque structure contenant un schéma avant d'effectuer la transformation, pour conserver la structure originale.



Ainsi, on trouverait la nouvelle structure :



Au moyen d'une nouvelle règle, qui peut être très générale, on peut alors reconnaître la similitude entre la sous-arborescence de racine (d'étiquette) ANAF et la seconde de racine CAND. Les tests sont des conditions tout-à-fait classiques sur les valeurs des variables des différents sommets. La même règle peut alors transformer la structure et donner l'analyse finale :



On a bien sûr pris un cas très simple. Cependant, le système de transformations d'arborescences utilisé permet de définir des stratégies assez compliquées en organisant convenablement les grammaires [54] : elles forment une hiérarchie dont le parcours est un "back-track" contrôlé par des conditions d'entrée et de sortie arbitraires.

4.4. SEMANTIQUE "PROFONDE" ET STRUCTURES RECURSIVES

J'ai essayé de montrer comment on peut résoudre certaines questions par la sémantique en se bornant à utiliser des structures simples à traiter automatiquement : les arborescences (voire des quasi-arborescences). Cependant, ceci conduit parfois à des méthodes "ad hoc" (recopie de sous-arbres, références limitées dans le contexte d'un point, etc.). On peut penser pouvoir y remédier en utilisant des hiérarchies ou des réseaux. Ceci est, semble-t-il, possible et adéquat pour les traitements sémantiques qu'on vient de voir.

Cependant, un problème apparaît dès qu'on veut passer à une représentation de la compréhension, et même à une représentation détaillée du "sens" d'un texte : il faut pouvoir isoler une structure et la traiter comme un élément d'une autre structure. Un exemple simple de ce phénomène est la référence à toute une proposition, voire à un paragraphe (cf.p.V-11). Martemjanov (1973b) propose d'ailleurs une représentation du sens d'un texte au moyen d'un formalisme tel que chaque représentation soit une arborescence, dont tout sommet peut lui-même référer à une représentation, sans cycles possibles. Shank [45] propose un modèle (pour représenter les "conceptualisations") qui peut en admettre. On peut proposer la définition suivante de telles structures "récursives".

DEFINITION

*Soit S_0 un ensemble dénombrable d'*atomes. *On définit alors, pour tout n : $S_{n+1} = (\bigcup_{m \leq n} S_m)^{\mathcal{P}^N_+}$, i.e. l'ensemble des éléments de réseaux [3] dont les sommets sont pris dans les ensembles précédents.*

$\tilde{S} = \bigcup_{n \in \mathbb{N}} S_n$, *est l'*ensemble des "structures récursives" sur S_0.

En toute rigueur, on devrait, comme pour les réseaux, définir une relation d'isomorphisme et passer au quotient. Si on impose des contraintes supplémentaires à la définition de S_{n+1} (par exemple d'être un ensemble d'éléments de réseau arborescents), on les retrouve sur $\tilde{S}$ (par exemple, $\tilde{S}$ est l'ensemble des arborescences récursives).

Si LISP connaît un tel succès en IA, c'est en particulier parce qu'il permet de manipuler aisément de telles structures, via la liste de propriétés des atomes. Il serait intéressant de voir, si, comme les arborescences [12] , on peut trouver une classe d'opérations suffisante pour effectuer les manipulations nécessaires (aux applications en IA dont on a parlé), mais réalisable au moyen d'une classe d'algorithmes de complexité plus faible que celle des algorithmes généraux de parcours de S-graphes.

D'autre part, Keeney (1975) cite plusieurs expériences qui étayent fortement l'hypothèse selon laquelle la compréhension et la mémorisation se feraient en utilisant une structure "profonde" sans rapport avec les réalisations graphiques ou acoustiques. Enfin, un exemple simple, inspiré de la fameuse fonction de substitution de Gödel, illustre bien la nécessité d'une certaine "récursivité" des structures. Supposons en effet que la représentation d'un énoncé inclue, après analyse, la description de ce qu'il dénote. On voit alors que l'énoncé

"le résultat de la substitution de "le résultat de la substitution de "x" par "x" est correct" dans "le résultat de la substitution de "x" par "x" est correct" est correct"

... est correct et se dénote lui-même !

4.5. HEURISTIQUE ET SEMANTIQUE, SYNTAXE ET COMBINATOIRE : DES CONFUSIONS

L1, L2 et L3 ont été définis plus haut comme les niveaux grammatical, conceptuel et pragmatique du langage. La "sémantique" participe donc de L2 et L3, si c'est bien la mise en relation d'un système formel avec un domaine extérieur, statique ou dynamique. Cependant, le langage n'évolue pas in abstracto, c'est un outil pour manier le réel, et, par là-même, il est conduit à associer étroitement certaines propriétés sémantiques à certaines unités de L1, sous forme de traits formels (onomatopées, conjugaisons ou déclinaisons particulières - verbes inchoatifs ou substantifs animés en russe, genres, ...). Si donc on distingue trois types de modèles, N1, N2 et N3, tels que N1 définisse un langage formel que N2 (N3) met statiquement (dynamiquement) en relation avec un domaine extérieur, lui-même formalisé, on comprend qu'ils ne correspondent pas exactement à ces niveaux de langage. En effet, il est souvent possible, et souhaitable pour des raisons d'efficacité, de coder certaines informations d'un niveau de langage donné dans un modèle de niveau moindre. C'est une réduction de ce type qui a été proposée aux 4.2 et 4.3.

D'autre part, dans chaque type de modèle, on peut utiliser des méthodes combinatoires ou heuristiques. D'où un risque supplémentaire de confusion ! Pour fixer les idées, disons qu'une méthode est "combinatoire" si on commence par énumérer toutes les solutions[§] avant de choisir et heuristique si la poursuite ou l'arrêt de la construction d'une solution est guidé par une estimation. Notamment aux U.S.A., à l'enthousiasme initial pour les "méthodes syntaxiques" a succédé un engouement non moins intense pour les "méthodes sémantiques". Il y a là un double amalgame entre syntaxe et combinatoire d'une part, sémantique et heuristique de l'autre. Or, dans le système de Simmons par exemple, on traite les informations "sémantiques" comme les autres, i.e. d'une façon purement combinatoire. Dans celui de Wilks au contraire, on applique une méthode heuristique même au niveau syntaxique. D'ailleurs, il est surprenant de constater que les algorithmes utilisés par les systèmes du type "tout par la sémantique" sont souvent bien plus combinatoires que ceux de systèmes où la syntaxe est réputée essentielle ! On pense au système de Quillian pour le premier cas et à un simple système d'analyse morphologique (ATEF) muni de fonctions de contrôle pour le second.

Pour terminer, espérons que l'utilisation de méthodes plus heuristiques et d'informations plus sémantiques (gnosto-encyclopédiques), en combinaison avec des approches plus classiques, mais, me semble-t-il, nécessaires, permettra d'améliorer de façon substantielle la qualité des traductions produites par les systèmes de TA.

---

§ Cf Ch I : Max $\Phi^*(c)$ pour tout c, quand $\Phi^*(c)$ est fini.

TABLE BIBLIOGRAPHIQUE

[1] Apresjan (1973)
[2] Apresjan (1974)
[3] Apresjan, Zholkovskij & Mel'tchuk (1970)
[4] Arsenteva & Shaljapina (1972)
[5] Barbault & Desclés 51972)
[6] Boitet (1973)
[7] Boitet (1974a)
[8] Boitet (1974b)
[9] Bourguignon, Nedobejkine, Vauquois, Veillon (1968)
[10] Chauché (1971)
[11] Chauché (1972)
[12] Chauché (1974)
[13] Chauché (1975)
[14] Chauché, Guillaume, Quézel-Ambrunaz (1973)
[15] Deransart (1972)
[16] Droste (1973)
[17] Ducrot (1973)
[18] Egli (1974)
[19] Feiblenam (1974)
[20] Feldman & Yakimovsky (1974)
[21] Fillmore (1968)
[22] Goldman (1975)
[23] Hays (1975)
[24] Hunt (1973)
[25] Il'in & al (1969)
[26] Keeney (1975)
[27] Kulagina & Mel'tchuk (1971)
[28] Lavorel (1975)
[29] Ljudskanov (1969)
[30] Ljudskanov (1973)
[31] Lecomte (1974)
[32] Martemjanov (1973a)
[33] Martemjanov (1973b)
[34] Mendenson (1964
[35] Mel'tchuk (1973)
[36] Mel'tchuk (1974)
[37] Mel'tchuk & Kulagina (1967)
[38] Mel'tchuk & Zholkovskij (1971)
[39] Padutcheva (1974)
[40] Quillian (1969)
[41] Quine (1961)
[42] Rouault (1971)
[43] Shaljapina (1972)
[44] Shank (1973)
[45] Shank & Rieger (1973)
[46] Shank & Rieger (1974)
[47] Shaumjan (1971)
[48] Simmons (1973)
[49] Simmons & Slocum (1972)
[50] Tubach (1970)
[51] Vauquois (1967)
[52] Vauquois (1975)
[53] Vauquois, Veillon, Veyrunes (1964)
[54] Vauquois, Veillon, Veyrunes (1965)
[55] Vauquois, Veillon, Veyrunes (1967)
[56] Veillon (1968)
[57] Veillon (1970)
[58] Wilks (1972)
[59] Wilks (1973a)
[60] Wilks (1973b)
[61] Wilks (1975)
[62] Wilks & Herskowits (1973)
[63] Winograd (1971)
[64] Winograd (1973)
[65] Woods (1970)
[66] Zholkovskij & Mel'tchuk (1967)
[67] Zholkovskij & Mel'tchuk (1969)
[68] * GEORGETOWN Report (1963)
[69] * Groupe LEIBNIZ (1975)

BIBLIOGRAPHIE

BIBLIOGRAPHIE

Cette bibliographie comporte, outre des références citées à la fin de chaque chapitre, des références non citées dans le texte. Chacune est indexée selon 16 critères dans une échelle allant de 0 à 9. Le choix des critères est sans doute arbitraire, et l'indexation elle-même assez subjective. J'espère seulement qu'elle pourra aider un lecteur éventuel à utiliser cette bibliographie comme outil de travail.

Les critères et l'échelle choisis sont les suivants :

- AF rapport avec les automates formels
- AL rapport avec l'algorithmique
- AP rapport avec l'apprentissage
- AS rapport avec l'analyse syntaxique
- CP rapport avec la complexité
- IA rapport avec l'intelligence artificielle
- LF rapport avec les langages formels
- LG rapport avec la logique
- LN rapport avec la linguistique
- MA caractère mathématique
- PH caractère méthodologique
- PL rapport avec la psycholinguistique
- PR rapport avec la programmation
- RC rapport avec la récursivité
- SM rapport avec la sémantique
- TA rapport avec la traduction automatique

Echelle	Signification
0	lien nul
1	
2	lien faible
3	
4	lien moyen
5	
6	lien fort
7	
8	lien très fort
9	ouvrage de base

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Aho (1968)
5307 0070 0000 0000
A.V. AHO, "Indexed grammars-an extension of CF-grammar". JACM, V15, N4, Oct. 1968, 647-671.

Aho (1969)
8707 0050 0000 0000.
A.V. AHO, "Nested stack automata". JACM, V16, N3, July 1969, 383-406.

Aho & Ullman (1972)
9909 5080 0000 6300
A.V. AHO & J.D. ULLMAN, "The theory of parsing, translation and compiling". Prentice Hall, 2 vol., 1972.

Amarel (1971)
0570 0800 0030 5050
S. AMAREL, "Representations and modelling in problems of program formation". In Machine Intelligence 6, eds. Meltzer & Mitchie, American Elsevier, New-York, 1971.

Anderaa & Fischer (1967)
8000 0000 0000 0800
S. ANDERAA & P.C. FISCHER, "The solvability of the halting problem for 2-state Post machines". JACM, N4, 677-682, Oct. 1967.

Apresjan (1973)
0000 0000 8000 0070
J.D. APRESJAN, "A description of semantics by means of syntax". Linguistics, n° 96, jan. 1973, 5-32.

Apresjan (1974)
0000 0000 8000 0080
Ю.Д.АПРЕСЯН, "Лексическая семантика".Наука, Москва, 1974.

Apresjan, Zholkovskij, Mel'tchuk (1970)
0000 0000 8000 0006
Ю.Д.АПРЕСЯН, А.К.ЖОЛКОВСКИЙ и И.А.МЕЛЬЧУК, "8 словарных статей толкового-комбинаторного словаря русского языка". Предв. публикация Института Русского Языка АН СССР, №2, Москва, 1970.

Arbib (1969)
9000 7050 0000 0800
M.A. ARBIB, "Theories of abstract automata". Prentice Hall, 1969, 412p.

Arsenteva & Shaljapina (1972)
0000 0000 8000 0006
Н.Г.АРСЕНТЕВА и З.М.ШАЛЯПИНА, "Язык для записи лингвистической информации в автоматическом словаре".МП и ПЛ, Выпуск 16, 113-152, 1972.

Atkinson & Estes (1963)
R.C. ATKINSON & W.K. ESTES , "Stimulus sampling theory". In "Handbook of mathematical psychology", Luce, Bush & Galanter, eds., Wiley, 1963, 121-268.

Azra & Jaulin (1965)
0000 0004 0700 0800
J.P. AZRA & B. JAULIN, "Récursivité". Gauthiers-Villars, 1965.

Barbault & Desclés (1970)
6000 0030 0800 0000
M.C. BARBAULT & J.P. DESCLES, "Approches structurelles et catégorielles des systèmes de transitions et applications à la science du calcul et à la linguistique mathématique". Thèse IIe cycle, janv. 70, Paris.

Barbault & Desclés (1972)
0000 0000 6500 0000
M.C. BARBAULT & J.P. DESCLES, "Transformations formelles et théories linguistiques". Document de Linguistique Quantitative n° 11, Dunod, 1972, 130 p.

Bar-Hillel (1964)
Y. BAR-HILLEL, "Language & Information". Addison Wesley, 1964.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Bar-Hillel (1967)
0000 0000 0050 0006

Y. BAR-HILLEL, "Die Zukunft der maschinellen Übersetzung, oder : Warum Maschinen das Uebersetzen nicht erlernen". Sprache im technischen Zeitalter, 23, 1967.

Bar-Hillel (1971)
0000 0300 3060 0047

Y. BAR-HILLEL, "Some reflections on the present outlook for High Quality Machine Translation". In "Feasibility study on fully automatic high quality translation", RADC-TR-71-295, Univ. of Texas, 1971.

Biermann (1972)
5680 0000 0000 3200

A.W. BIERMANN, "On the inference of Turing machines from sample computations". Artificial Intelligence, 3, 1972, 181-198.

Biermann & Feldman (1972a)
5680 0000 0400 0000

A.W. BIERMANN & J.A. FELDMAN, "On the synthesis of finite-state machines from examples of their behavior". IEEE Trans. Comp., 592-597, June, 1972.

Biermann & Feldman (1972b)
2480 4200 0400 0600

A.W. BIERMANN & J.A. FELDMAN , "A Survey of résults in grammatical inference". In "Frontiers in pattern recognition", Watanabe, ed., Academic Press, New-York, 1972.

Blum (1967b)
0000 9000 0800 0800

M. BLUM, "A machine-independent theory of the complexity of recursive functions". JACM, V14, N2, April 67, 323-336.

Blum (1967b)
0000 9000 0800 0800

M. BLUM, "On the size of machines", I & C 11, 257-265, 1967.

Blum (1971)
0000 9000 0800 0800

M. BLUM, "On effective procedures for speeding-up algorithms". JACM, V18, N2, April 71, 290-305.

Blum & Marques (1973)
0000 8000 0700 0800

M. BLUM & I. MARQUES, "Complexity properties of r.e. sets". The Journal of Symbolic Logic, V 38, N4, 579-593, dec. 1973.

Bobrow (1967)
4607 0000 0000 7000

D.G. BOBROW, "Syntactic theory in computer implementations" . In "Automated language processing, the state of the art", J. Wiley & Sons, New-York, 1967, 215-251.

Bobrow (1968)
0004 0600 2000 7000

D.G. BOBROW, "Natural language input for a computer problem-solving system (STUDENT)". In "Semantic Information Processing", MIT, 1968, 135-215.

Boitet (1972)
0000 0000 2000 5006

C. BOITET, "Transcodages russe-français et russe-anglais utilisant une même analyse morphologique". TR, TAUM, Université de Montréal, juin 1972.

Boitet (1973 a)
0000 0050 0400 0005

C. BOITET, "Astrabondes et dictionnaires". GETA, Grenoble, 1973.

Boitet (1973 b)
4346 0200 4500 3008

C. BOITET, "Towards an adaptative and interactive system in automatic translation". Conf. ICCL, Pise, 1973.

Boitet (1974a)
4527 0060 3500 0305

C. BOITET, "Heuristique et analyse syntaxique prédictive". Doc. GETA, G-2900-A, avril 1974.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Boitet (1974 b)
0000 0500 0040 0087

C. BOITET, "Sémantique et traduction automatique. Quelques remarques". Doc. GETA, G-3000-A, décembre 1974.

Boitet (1976 a)
0200 0600 4000 0087

C. BOITET, "Méthodes sémantiques en TA". TA-Informations, N1, 1976, à paraître.

Boitet (1976 b)
0006 0030 6000 0005

C. BOITET, "Traitements automatiques en syntaxe". In"Langues romanes et traitements automatiques", Doc. de Linguistique Quantitative 29, Dunod, 1976, à paraître.

Boitet & Chauché (1974)
0003 0503 0220 0087

C. BOITET & J. CHAUCHE, "Approches sémantiques pour les modèles d'analyse automatique de langues naturelles". Colloque "Modèles logiques et niveaux d'analyse linguistique", Metz, nov. 1974.

Book (1974 a)
0000 8000 0600 0600

R.V. BOOK, "Comparing complexity classes". JCSS 9, 213-229, 1974.

Book (1974 b)
0000 7040 0600 0500

R.V. BOOK, "Tally languages and complexity classes". I & C 26, 186-193, 1974.

Booth (1967)
0306 0000 6200 4008

A.D. BOOTH, ed., "Machine Translation", Norht-Holland, 1967.

Booth & al (1958)
0004 0000 6400 0005

A.D. BOOTH, L. BRANDWOOD, J.P. CLEAVE, "Mechanical resolution of linguistic problems par ...". Butterworths, London, 1958.

Borodin (1972 a)
0000 9000 0700 0800

A. BORODIN, "Computational complexity and the existence of complexity gaps". JACM, V19, N1, Jan. 1972, 158-174.

Borodin (1972 b)
0000 8000 0700 0800

A. BORODIN, "Computational complexity-Theory and Practice". In "Current trends in the theory of computing", Aho, ed., Prentice-Hall, Englewood Cliffs, N.J., 1972.

Bourguignon, Nedobejkine, Vauquois, Veillon (1968)
0000 0050 6000 0047

C. BOURGUIGNON, N. NEDOBEJKINE, B. VAUQUOIS, G. VEILLON, "Une notation des textes hors des contraintes morphologiques et syntaxiques de l'expression". Conf. ICCL, Stockholm, 1968.

Brainerd & Landweber (1974)
5000 6040 0700 0800

W. BRAINERD & L. LANDWEBER , "Theory of computation". J. Wiley & Sons, New-York, London, 1974, 336p.

Brandt Corstius (1975)
0005 0060 6300 0000

H. BRANDT CORSTIUS, "The proof that languages is not context-free". Première conférence nationale sur l'utilisation de modèles mathématiques et d'ordinateurs en linguistique, Varna, mai 1975.

Brauer & Indermark (1968)
6000 0003 0600 0800

W.BRAUER & K.INDERMARK, "Algorithmen, rekursive Funktionen und formale Sprachen". B.I. Hochschulskripten, 1968, 115p.

Busacker & Saaty (1965)
0000 0000 0700 0000

R.G. BUSACKER & T.L. SAATY, "Finite graphs and networks : an information with applications". McGraw Hill, 1965.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Bush & Mosteller (1955)
0370 0000 0404 0000

R. BUSH & F. MOSTELLER, "Stochastic models for learning", New-York, Wiley, 1955, 366p.

Carroll (1966)
0000 0000 5000 0000

J.B. CARROLL, "An experiment in evaluating the quality of translations". Mechanical Translation, V9, N3 & 4, 55-66 , 1966.

Chauché (1971)
7000 0000 0700 0000

J. CHAUCHE, "Transduction d'arborescences". Thèse 3e cycle, Grenoble, 1971.

Chauché (1972)
0000 0000 0700 0004

J. CHAUCHE, "Arborescences et transformations". Doc. GETA, G-2700-A, Grenoble, avril 1972.

Chauché (1973)
8500 6060 0800 0600

J. CHAUCHE, "Transducteurs composés". Doc. GETA, G-2800-A, Grenoble, avril 1973.

Chauché (1974 a)
4003 3000 0400 6006

J. CHAUCHE, "Un système de contrôle et de transductions d'arborescences : le système CETA". Grenoble, séminaire de programmation, juin 1974.

Chauché (1974 b)
8604 3060 3700 2306

J. CHAUCHE, "Transducteurs et arborescences. Etude et réalisation de systèmes appliqués aux grammaires transformationnelles". Thèse d'Etat, Grenoble, 1974.

Chauché (1975 a)
0502 0030 3000 0204

J. CHAUCHE, "Présentation du système CETA". Doc. GETA, G-3100-A, Grenoble, janvier 1975.

Chauché (1975 b)
3002 0000 0200 5007

J. CHAUCHE, "Les systèmes ATEF et CETA". TA-Informations, N2, 1975. Trad. in AJCL, Microfiche 17, 21-39.

Chauché & al (1973)
7300 0060 3200 3007

J. CHAUCHE, P. GUILLAUME, M. QUEZEL-AMBRUNAZ, "Le système ATEF". Doc. GETA, G-2600-A, Grenoble, 1973.

Chomsky (1957)
4008 0080 9540 0000

N. CHOMSKY, "Syntactic structures". The Hague, 1957.

Chomsky (1963)
3006 0060 3400 0000

N. CHOMSKY, "Formal properties of grammars". In "Handbook of mathematical psychology", Luce, Bush & Galanter, eds, Wiley, 1963, 323-418.

Chomsky (1965)
5006 0070 9550 0020

N. CHOMSKY, "Aspects of the theory of syntax". MIT Press, 1965.

Chomsky (1972)
0000 0000 6066 0050

N. CHOMSKY, "Language and mind". Harcourt Brace Jovanitch, 1972.

Chomsky & Miller (1963)
5006 0060 8300 0000

N. CHOMSKY & G.A. MILLER, "Introduction to the formal analysis of natural languages". In "Handbook of mathematical psychology", Luce, Bush & Galanter, eds. Wiley, 1963, 269-322. Trad. : "L'analyse formelle des langues naturelles", Gauthiers-Villars, 1968.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Church (1944,1956)
0000 0009 0800 0800

A. CHURCH, "Introduction to mathematical logic". Part I : Annals of Math. Studies, 13, Princeton Univ. Press, 1944. Part II : Princeton Univ. Press, 1956, 2 vol.

Church (1936)
0000 0000 0600 0800

A. CHURCH, "An unsolvable problem of elementary number theory". The American Journal of Mathematics, V58, 345-363, 1936. Ou in "The Undecidable", Davis, ed, Raven Press, 1965, 88-107.

Colmerauer (1971)
5506 0050 6000 0008

A. COLMERAUER, "Systèmes-Q". Rapport TAUM 71, Univ. de Montréal, 1971.

Colmerauer & al (1973)
0406 0856 5200 5050

A. COLMERAUER, H. KANOUI, P. ROUSSEL, R. PASERO, "Un système de communication homme-machine en français". Groupe de recherche en IA, UER de Luminy, Univ. d'Aix-Marseille, juin 1973.

Comtet (1970)
0000 0000 0800 0000

L. COMTET, "Analyse combinatoire". PUF, 1970, 2 vol.

Constable (1971)
0000 9060 0800 0800

R.L. CONSTABLE, "Subrecursive programming languages II. On program size". JCSS, 5, 315-334, 1971.

Constable (1972)
0000 9000 0700 0800

R.L. CONSTABLE, "The operator gap". JACM, V19, N1, Jan. 1972, 175-183.

Constable et Borodin (1972)
3000 9060 0800 0800

R.L. CONSTABLE & A.B. BORODIN, "Subrecursive programming languages, Part I : efficiency and program structure". JACM, V19, N3, July 72, 526-568.

Conway (1963)
7608 0040 0400 7000

M.E. CONWAY, "Design of a separable transition-diagram compiler". CACM, V6,N7, 396-408, July 1963.

Cook (1963)
0000 7000 0600 0700

S. COOK, "A hierarchy of nondeterministe time complexities". JCSS, 7, 1973, 343-353.

Cook & Rechow (1973)
8000 8000 0700 0700

S. COOK & R.A. RECHOW, "Time bounded random access machines", JCSS, 7, 354-375, 1973.

Cook & Rozenfeld (1974)
0570 0000 0500 4000

C.M. COOK & A. ROZENFELD, "Some experiments in grammatical inference". In "Procédures informatiques d'apprentissage", 157-172, IRIA, 1974.

Coulon (1975)
4604 0400 0200 6000

D. COULON, "Construction expérimentale d'un modèle informatique pour interpréter des énoncés rédigés en langage naturel". Thèse d'Etat, Univ. Paris VI, janv. 1975.

Courtin (1973)
0500 0000 0000 6004

J. COURTIN, "Un analyseur morphologique et syntaxique pour les langues naturelles". Conf. ICCL, Pise 1973.

Courtin & Voiron (1974-75)
0700 5000 0400 6000

J. COURTIN & J. VOIRON, "Introduction à l'algorithmique et aux structures de données". IUT-B, Grenoble, Dép. Informat., 1974-75.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Crespi-Reghizzi (1970)
0570 0040 0500 4000
S. CRESPI-REGHIZZI, "The mechanical acquisition of precedence grammars". Ph. D. Diss., Univ. of California, 1970.

Crespi-Reghizzi (1974)
0074 6060 0600 0000
S. CRESPI-REGHIZZI, "Non counting languages and learning". In "Procedures informatiques d'apprentissage", 341-348, IRIA, 1974.

Csibi (1974)
0470 6000 0600 0000
S. CSIBI, "On embedding heuristics and including complexity constraints into convergent learning algorithms". In "Frontiers of pattern recognition", S. Watanabe, ed, 99-112, Academic Press, 1972.

Čulik (1973)
5004 0070 4540 0000
K. ČULIK, "Basic problems in the mathematical theory of languages", Linguistics, 118, 5-42, déc. 1973.

Davis (1956)
4000 0000 0700 0800
M.D. DAVIS, "A note on universal Turing machines". In "Automata Studies", 167-177, C.E. Shannon & J. McCarthy, eds, Princeton University Press, 1956.

Davis (1958)
5000 0000 0800 0900
M.D. DAVIS, "Computability and Unsolvability". McGraw Hill, 1958, 210p.

Davis (1965)
6000 0006 0700 0800
M.D. DAVIS, ed., "The undecidable". Raven Press, 1965.

Delavenay (1960)
0000 0000 0050 0007
E. DELAVENAY, "Machine translation of languages : Research and organizational problems". Impact of Science of Society, V10, N1, 26-43, 1960.

Deransart (1972)
0606 0000 2400 6000
P. DERANSART, "Un analyseur syntaxique non-déterministe et des traitements d'arborescences pour aider au développement de la documentation automatique". Thèse Dr Ingénieur, Toulouse, 1972.

Droste (1973)
0000 0004 7050 0060
F.G. DROSTE, "Model theory, logic and linguistics". Linguistics, N105, June 73, 5-34.

Ducrot (1973)
0004 0000 3000 5007
J.M. DUCROT, "Le système TITUS II". Institut Textile de France, sept 73,25p.

Earley (1968)
0609 4050 0600 3000
J. EARLEY, "An efficient context-free parsing algorithm". Ph. D. Thesis, Carnegie-Mellon Univ. , 1968.

Earley (1970)
0609 4050 0400 3000
J. EARLEY, "An efficient context-free parsing algorithm". CACM, V13, N2, Feb. 70, 94-102.

Edmundson (1967)
0000 0030 5400 0033
H.P. EDMUNDSON, "Mathematical models in linguistics and language processing". In "Automated language processing : the state of the art", Harold Borko, ed., J. Wiley & Sons, New-York, 1967, 33-96.

Egli (1974)
0000 0000 6000 0060
U. EGLI, "Ansätze zur Integration der Semantik in die Grammatik". Forschungen / Linguistik und Kommunikationswissenschaft 3, Scriptor (Kronberg), 1974.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Eilenberg (1974)
8000 0080 0800 0000

S. EILENBERG, "Automata, languages and machines", Academic Press, 1974.

Eilenberg & Wright (1970)
7000 0070 0700 0000

S. EILENBERG & J.B. WRIGHT, "Automata in general algebras". I & C 11, 1970, 452-470.

Elcock & al (1971)
0000 0000 0030 7000

E.W. ELCOCK, J.M. FOSTER, P.M.D.-GRAY, J.J. McGREGOR, A.M. MURRAY, "ABSET : a programming language based on sets ; motivations & examples". In "Machine Intelligence", V6, Edinburgh Univ. Press, 1971, 467-494.

Ershov, Narinjani, Mel'chuk (1975)
0000 0500 5000 0040

A. ERSHOV, A. NARINJANI, I. MEL'TCHUK, "RITA- an experimental Man-Computer system on a natural language basis". Advance papers of the IJCAI-75, 387-390, AIL, MIT, 1975.

Feibleman (1974)
0000 0007 0060 0060

J.K. FEIBLEMAN, "Professor Quine and real classes". Notre Dame Journal of Formal Logic, XV, N2, April 1974, 207-224.

Feldman (1972)
2490 6000 0400 0600

J.A. FELDMAN, "Some decidability results on grammatical inference and complexity". I & C 20, 244-262, 1972.

Feldman & al (1969)
0070 6000 0600 0500

J.A. FELDMAN, J. GIPS, J.J. HORNING, S. REDER, "Grammatical complexity and inference". TR-CS-125, Comp. Sc. Dep, Stanford Univ., June 1969, 100p.

Feldman & Shields (1972)
0060 8000 0500 0700

J.A. FELDMAN & P. SHIELDS, "On total complexity and the existence of best programs". TR-CS-253-72, Comp. Sc. Dep., Stanford Univ. , 1972.

Feldman & Yakimovsky (1974)
0070 0800 0000 0000

J.A. FELDMAN & Y. YAKIMOVSKY, "Decision theory and artificial intelligence : I. A semantic-based region analyzer". Artificial Intelligence, V5, N4, 1974, 349-372.

Fillmore (1968)
0000 0000 7000 0000

C. FILLMORE, "The Case for Case" . In "Universals in Linguistics theory", E. Bach & R. Harms, eds, Holt, Rinehart & Winston, New-York, 1968.

Fillmore (1971)
0000 0000 7000 0044

C. FILLMORE, "On a fully developped system of linguistic description". In "Feasibility study on FAHQT", RADC-TR-71-295, 77-94, Univ. of Texas,1971.

Fischer (1965)
8000 5000 0500 0600

P.C. FISHER, "On formalisms for Turing Machines". JACM, V12, N4, 570-580, Oct. 1965.

Fisher (1966)
6000 4000 0600 0700

P.C. FISCHER, "Turing machines with restricted memory access". I & C 9, 364-379, 1966.

Floyd (1967)
6600 0300 0300 4000

R.W. FLOYD, "Nondeterministic algorithms". JACM, V14, N4, oct. 1967, 636-644.

Friant (1966)
0005 0070 0700 0000

J. FRIANT, "Les langages CS". Thèse IIIe cycle, Univ. de Paris, juin 1966.

Friant (1971)
0000 0090 0800 0000

J. FRIANT, "Grammaires de Chomsky. Matrices de règles. Fonctions de poids". Thèse d'Etat, Univ. de Paris, 21 juin 1971.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Friedman & al (1971)
0000 0050 4000 6000

J. FRIEDMAN, T.H. BREDT, R.W. DORAN, B.W. POLLACK, T.S. MARTNER, "A computer model of transformational grammar". Math. linguistics & automatic language processing 9, Elsevier, 1971.

Fry (1970)
0000 0600 0007 0000

D.B. FRY, "Speech reception and perception". In "New horizons in linguistics", J. Lyons, ed., Penguin books, 1970.

Fu (1972)
0340 0700 0300 0000

K.S. FU, "On syntactic pattern recognition and stochastic languages". In "Frontiers of pattern recognition", 113-138, S. Watanabe, ed., Academic Press, 1972.

Garvin (1961)
0000 0000 6003 0008

P.L. GARVIN, "Automatic linguistic analysis, a heuristic problem". First International Congress on MT, 3-16, National Physical Laboratory, Teddington, England, 1961.

Garvin (1967)
0605 0000 5000 0006

P.L. GARVIN, "The Fulcrum syntactic analyzer for Russian". 2° conf. intern. sur le traitement automatique des langues, Grenoble, août 1967.
Trad. russe in "Автоматический перевод", Москва, 1971, 27-41.

Garvin (1971)
0000 0000 3060 0036

P.L. GARVIN, "Operational problems of MT : a position paper". In "Feasibility study of FAHQT" , RADC-TR-71-295, 95-118, Univ. of Texas, 1971.

Gill & Blum (1974)
0000 7000 0800 0600

J. GILL & M. BLUM, "On almost everywhere complex recursive functions". JACM, V21, N3, July 74, 425-435.

Gill & Morris (1974)
0000 6000 0700 0800

J.T. GILL & P.H. MORRIS, "On subcreative sets and S-reducibility". The Journal of Symbolic Logic, V 39, N4, dec. 1974.

Gillogly (1972)
0300 0600 0000 5000

J.J. GILLOGLY, "The Technology Chess program". Artificial Intelligence, V3, N3, 1972, 145-163.

Gladkij & Mel'tchuk (1969)
0500 0030 3500 0000

A.V. GLADKIJ & I.A. MEL'TCHUK, "Tree grammars". Conf. ICCL, AL 32, Sanga Säby, Sweden.

Gladkij & Mel'tchuk (1969)
5004 0060 5600 0003

A.V. GLADKIJ & I.A. MEL'TCHUK, "Eléments de Linguistique Mathématique", Moscou 1969.

Gold (1965)
2400 2000 0600 0800

E.M. GOLD, "Limiting recursion". The Journal of Symbolic Logic, V30, N1, March 1965.

Gold (1967)
2490 0350 0602 0600

E.M. GOLD, "Language identification in the limit". I & C 10, 447-474, 1967.

Gold (1971)
0070 0700 0600 0500

E.M. GOLD, "Universal goal-seekers", I & C 18, 395-403, 1971.

Goldman (1973)
0300 0600 0000 4073

N. GOLDMAN, "Sentence paraphrasing from a conceptual base". Conf. ICCL, Pise, 1973.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Goldman (1975)
0300 0600 0000 4073

N. GOLDMAN, "Sentence paraphrasing from a conceptual base". CACM, V18, N2, 96-106, Feb 1975.

Gross (1972)
3004 0050 6500 0000

M. GROSS, "Mathematical models in linguistics". Prentice Hall, 1972,156p.

Grzegorczyk (1963)
0000 0000 0800 0900

A. GRZEGORCZYK, "Some classes of recursive functions". Rozprawy matematyczne, N4, Institut Matematyczny Polskeij Akademie Nauk, 1-45, 1963.

Guinsburg, Greibach, Harrison (1967)
6007 0060 0800 0000

S.-GUINSBURG, S.A. GREIBACH, M.A. HARRISON, "Stack automata and compiling". JACM, V14, 1967, 172-201.

Guinsburg & Spanier (1968)
4000 0080 0800 0000

S. GUINSBURG & E.H. SPANIER, "Control sets on grammars". Mathematical systems theory, V2, N2, 159-177, 1968.

Harris (1971)
0004 0040 8400 0000

Z.S. HARRIS, "Structures mathématiques du langage". Trad. Fuchs, Dunod, 1971.

Harris et Hofmann (1970)
0000 0000 5000 0006

G. HARRIS & T.R. HOFMANN, "Pidgin translation", META, V15, N2, 1970.

Hartmanis & Hopcroft (1971)
0000 8000 0700 0800

J. HARTMANIS & J.E. HOPCROFT , "An overview of the theory of computational complexity". CACM, V18, N3, July 71, 444-475.

Hartmanis, Lewis & Stearns (1965)
3000 7000 0700 0700

J. HARTMANIS, P.M. LEWIS II & R.E. STEARNS, "Hierarchies of memory limited computations". IEEE Conf. Record on Switching Circuit Theory and Logical Design, Ann Arbor, Michigan, 179-190, 1965.

Hartmanis & Stearns (1964)
0000 6000 0700 0700

J. HARTMANIS & R.E. STEARNS, "Computational complexity of recursive sequences". Proc. fifth Ann Symp. on Switching Circuit Theory and Logical Design, Princeton, New-Jersey, 1964, 82-90.

Hartmanis & Stearns (1965)
3000 8000 0600 0700

J. HARTMANIS & R.E. STEARNS, "On the computational complexity of algorithms". Trans. Amer. Math. Soc., 1965, 285-306.

Hays (1961)
0000 0300 0050 0008

D.G. HAYS, "Research procedures in machine translation". Santa Monica, RAND, 1961. Trad. in "Автоматический перевод", Москва, [illegible]

Hays (1975)
0000 0600 6000 0070

D.G. HAYS, "Cognitive networks". Première conf. nat. sur l'utilisation de modèles mathématiques et d'ordinateurs en linguistique. Varna, Bulgarie, 1975.

Hennie (1965)
4000 5000 0600 0700

F.C. HENNIE, "One-tape, off-line Turing computations". I & C 8, N6, 553-578, 1965.

Hennie (1966)
5000 7000 0500 0600

F.C. HENNIE, "On-line Turing machine computations". IEEE Trans. Elect. Comp., EC-15, 35-44, 1966.

Hennie & Stearns (1966)
6000 7000 0600 0600

F.C. HENNIE & R.E. STEARNS, "Two tape simulation of multitape Turing machines". JACM, V13, N4, 533-546, 1966.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Hérault & Lwoff (1976)
0000 0000 7400 0070

D. HERAULT & LWOFF, "Essai sur le spectre sémantique de quelques langues indo-européennes". Dunod, 1976. A paraître.

Hermes (1965)
6000 0000 0700 0900

H. HERMES, "Enumerability. Decidability". Springer Verlag, Berlin-Heidelberg - New-York, 1965, 245p.

Hibbard (1974)
0005 0070 0700 0000

T.N. HIBBARD, "Context limited grammars". JACM, V21, N3, July, 446-453.

Hockett (1961)
0000 0000 7005 0000

C.F. HOCKETT, "Grammar for the hearer". Structure of language and its mathematical aspects, R. Jakobson, ed., 220-236, 1961. Proc. of Symposia in Applied Math., V XII, Amer. Math. Soc, Providence, Rhode Island.

Hooper (1966)
3000 0000 0600 0700

P.K. HOOPER, "The undecidability of the Turing machine immortality problem". The Journal of Symbolic Logic, V31, 219-234, 1966.

Hopcroft & Ullman (1967)
7000 0000 0700 0500

J.E. HOPCROFT & J.D. ULLMAN, "An approach to an unified theory of automata". BSTJ, V46, N8, 1793-1829, 1967.

Hopcroft & Ullman (1969)
9000 4090 0700 0500

J.E. HOPCROFT & J.D. ULLMAN, "Formal languages and their relation to automata". Addison-Wesley, 1969, 242p.

Horning (1969)
0080 6000 0060 0600

J.J. HORNING, "A study of grammatical inference". TR-CS-139, Comp. Sc. Dep., Stanford Univ., August 1969.

Hunt (1973)
0000 0800 0008 0000

E. HUNT, "The memory we must have". In "Computer models of thought and language", Shank & Colby, eds, 343-371, Freeman & Co, San Francisco, 1973.

Il'in & al (1969)
0003 0600 6000 0050

Г.М.ИЛЬИН, В.М.ЛЕЙНИНА, Т.Н.НИКИТИНА, М.И.ОТКУПЩИКОВА, С.Я.ФИТИАЛОВ, "Семантическая модель текста и система "запрос-ответ" ". НТИ, с. 2, №1, 10-14, 1969. Trad. in Doc. de Ling. Quant. n°10, 51-70, Dunod, 1971.

Il'in & al (1971)
0000 0000 5000 3030

Г.М.ИЛЬИН, В.М.ЛЕЙНИНА, Т.Н.НИКИТИНА, М.И.ОТКУПЩИКОВА, С.Я.ФИТИАЛОВ, "Лингвистический подход к задаче построения информационной системы". Информационные вопросы семиотики, лингвистики и автоматического перевода, Выпуск 2, Москва, 1971, 1-13.

Joshi, Kosarayu, Yamada (1972)
4006 0080 0800 0000

A.K. JOSHI, S.R. KOSARAYU & H.M. YAMADA, "String adjunct grammars". I & C 21, N2, sept. 1972, 93-116 et I & C 21, N3, oct. 72, 235-260.

Kain (1972)
8000 0080 0700 0500

R.Y. KAIN, "Automata theory : machines and languages". McGraw Hill, 1972.

Karp (1972)
0000 6000 0700 0700

R.M. KARP, "Reducibility among combinatorial problems". In "Complexity of computer programming", 85-104, R.E. Miller & J.W. Thatcher, eds, Plenum Press, New-York, London, 1972.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Keeney(1975)
0000 0000 3007 0040

T.J. KEENEY,"Towards a theory of linguistic memory". Lectures notes in Computer Science 22, 159-167, Goos & Hartmanis , eds, Springer, 1975.

King (1959)
0000 0000 0000 6004

G.W. KING, "Word analyzer design data". IBM, Yorktown Heights, New-York, dec. 1959.

Kleene (1952)
0000 0007 0700 0900

S.C. KLEENE, "Introduction to metamathematics". Wolters-Noordhoff, 1952 (reed. 1971).

S. Klein (1966)
0380 0650 5406 0000

S. KLEIN, "Historical change in language using Monte-Carlo techniques". Mechanical Translation, V9, N 3&4 , 67-82, 1966.

S. Klein (1973)
0283 0330 4000 0050

S. KLEIN, "Automatic inference of semantic deep structure rules in generative semantic grammars". Conf. ICCL, Pise, 1973.

W. Klein (1974)
0000 0050 7304 0000

W. KLEIN, "Variation in der Sprache. Ein Verfahren zu ihrer Beschreibung". Skripten/Linguistik und Kommunikationswissenschaft 5, Scriptor, 1974.

Knuth (1965)
7609 6080 0700 3600

D.E. KNUTH, "On the translation of languages from left to right". I & C 8, 607-639, 1965.

Knuth (1971)
6708 4060 0500 4000

D.E. KNUTH, "Top-down syntax analysis". Acta informatica, 1, 77-110, 1971.

Knuth & Bigelow (1967)
6000 0070 0600 6400

D.E. KNUTH & R.H. BIGELOW, "Programming languages for automata". JACM, V14, N4, Oct. 67, 615-635.

Kosaraju (1974)
8000 0060 0800 0400

S.R. KOSARAJU, "One-way stack automaton with jumps". JCSS 9, 164-176, 1974.

Kulagina (1960)
0004 0000 5000 6008

О.С.НУЛАГИНА, "О машинном переводе с французского языка на русский. I словарь, II алгорифм.". Проблемы Кибернетики, 3 и 4, 1960.

Kulagina (1970)
0707 0000 4400 2008

О.С.НУЛАГИНА, "Алгорифм синтаксического анализа в системе французско-русского машинного перевода". Москва, ИПМ, 1970.

Kulagina (1973a)
0000 0000 3060 3008

О.С.НУЛАГИНА, "О машинном переводе текстов на естественных языках". Проблемы Кибернетики, 27, 33-46, 1973.

Kulagina (1973b)
0304 0000 4000 6008

О.С.НУЛАГИНА, "О системе французско-русского машинного перевода ФР-II". Проблемы Кибернетики, 27, 47-62, 1973.

Kulagina & Mel'tchuk (1956)
0003 0000 5200 5007

О.С.НУЛАГИНА и И.А.МЕЛЬЧУК, "Машинный перевод с французского на русский". Вопросы Языкознания, №5, 1956.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Kulagina & Mel'tchuk (1967)
0000 0000 4000 0057

O.S. KULAGINA & I.A. MEL'TCHUL , "AT : some theoretical aspects and design of a translation system". In "Machine Translation", A.D. Booth, ed, 137-173, North-Holland, 1967.

Kulagina & Mel'tchuk (1971)
0003 0040 5350 0008

О.С.КУЛАГИНА и И.А.МЕЛЬЧУН, "Автоматический перевод : краткая история, современное состояние, возможные перспективы". В "Автоматический перевод", 3-26, Москва, 1971.

Kulagina & Vakulovskaja (1959)
0504 0000 4000 6008

O.S. KULAGINA & G.V. VAKULOVSKAJA, "Experimental translation from French into Russian using a "Strela" computer". Problems of Cybernetics, V2, 657-666, Pergamon Press, 1961. Trad. de Проблемы Кибернетики, №2, 1959.

Kulagina & Vakulovskaja (1961-62)
0504 0000 3000 6008

О.С.КУЛАГИНА и Г.В.ВАКУЛОВСКАЯ, "О машинном переводе с французского языка на русский. III Описание программы, IV Результаты экспериментов и анализ ошибок". Проблемы Кибернетики, №2, 1959.

Kuno & Oettinger (1962)
0408 0050 0000 4000

S. KUNO & A.G. OETTINGER, "Multiple-path syntactic analysis". Information Processing 1962, 306-311, North Holland, Amsterdam, 1963.

Kuo (1974)
0000 0000 0000 8000

S.S. KUO, "Assembler language for FORTRAN, COBOL and PL/1 programmers". Addison-Wesley, 1974.

Lacombe (1960)
3000 0000 0800 0900

D. LACOMBE, "La théorie des fonctions récursives et ses applications". Bull. Soc. Math. France, 88, 393-468, 1960.

Landweber & Robertson (1972)
0000 8000 0700 0800

L.H. LANDWEBER & E.L. ROBERTSON, "Recursive properties of abstract complexity classes". JACM, V19, N2, April 72, 296-309.

Lavorel (1975)
0000 0000 6200 0070

P.M. LAVOREL, "Eléments pour un calcul du sens". Doc. de Ling. Quant. 27, Dunod, 1975, 200 p.

Lecomte (1974)
0000 0007 6700 0040

A. LECOMTE, "Essai de formalisation des opérations linguistiques de prédication". Thèse de 3e cycle, Grenoble, 1974.

Lewis, Stearns & Hartmanis (1965)
6005 7060 0600 0600

P.M. LEWIS, R.E. STEARNS & J. HARTMANIS, "Memory bounds for recognition of context-free and context-sensitive languages". IEEE Conf. Rec. on Swithching and Automata Theory, 21-35, Berkeley, California, 1965.

Ljudskanov (1968)
0000 0000 5050 0007

A. LJUDSKANOV, "Is the generally accepted strategy of machine translation research optimal ?". Mechanical Translation, V11, N1 & 2, 1968.

Ljudskanov (1969)
0000 0000 5060 0058

A. LJUDSKANOV, "Traduction humaine et traduction mécanique". Doc. de Ling. Quant. 2 & 4, Dunod, 1969.

Ljudskanov (1973)
0000 0000 6060 0076

A. LJUDSKANOV, "Machine et signification". TA-informations N1, 2-22, 1973.

Lomet (1973)
8000 0000 0700 0000

D.B. LOMET, "A formalization of transition diagrams systems". JACM, V20, N2, April 73, 235-257.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Luce, Bush & Galanter (1963)
3044 0050 5407 0030

D. LUCE, R. BUSH, E. GALANTER, eds, "Handbook of mathematical psychology. "J. Wiley & Sons, New-York, 1963.

Lynch (1974)
0000 0000 0600 0800

N. LYNCH, "Approximations to the halting problem. JCSS 9, 143-150, 1974.

Lyndon (1966)
0000 0008 0600 0000

R.C. LYNDON, "Notes on logic". Van Nostrand, 1966.

McNaughton & Papert (1971)
8000 0040 0700 0000

R. McNAUGHTON & S. PAPERT, "Counter-free automata". MIT Press, research monograph n° 65, 1971.

Mal'cev (1970)
3000 0000 0800 0800

A.I. MAL'CEV, "Algorithms and recursive functions". Wolters-Noordhoff, 1970, 372p.

Manna & Waldinger (1971)
0500 0600 0500 6000

Z. MANNA & R. WALDINGER, "Toward automatic program synthesis". CACM, 14, 151-165, 1971.

Markov (1954)
0000 0005 0800 0900

A.A. MARKOV, "Theory of algorithms". Israel program for scientific translation, Jerusalem, 1962 (trad.)

Martemjanov (1973a)
0000 0000 7000 0050

JU.S. MARTEMJANOV, "Valency-junction-emphasis relations as a language for text description". In "Trends in soviet theoretical linguistics", 335-338, F. Keifer, ed, 1973.

Martemjanov (1973b)
0000 0040 6400 0073

Ю.С.МАРТЕМЯНОВ, "Связный текст - Изложение расчлененного смысла". Институт Русского Языка АН СССР, Предв. Публ, №40, Москва, 1973.

Masterman (1967)
0000 0000 2000 5007

M. MASTERMAN, "Semis-automatic translation from English into French : the system "man-machine thesaurus" ". COLING, Grenoble, 1967, rep. 12. Trad. in "Автоматический перевод", 264-281, Москва, 1971.

Mel'tchuk (1973a)
0000 0000 5060 0006

И.А.МЕЛЬЧУК, "Русский язык в исследованиях по автоматическому переводу. Обзор работ по современному литературному языку за 1966-1969 гг.". Институт Русского Языка АН СССР, Москва, 1973.

Mel'tchuk (1973b)
0000 0000 7000 0073

I.A. MEL'TCHUK, "Linguistics, computational linguistics and Meaning-Text models". Conf. ICCL, Pise, 1973.

Mel'tchuk (1974)
0002 0330 8000 0097

И.А.МЕЛЬЧУК, "Опыт теории лингвистсческих моделей "Смысл-Текст"". Наука, Москва, 1974.

Mel'tchuk & Ravitch (1967)
0000 0000 0060 0008

И.А.МЕЛЬЧУК и Р.Д.РАВИЧ, "Автоматический перевод 1949-1963 (критико-библиографический справочник)". Москва, 1967.

Mel'tchuk & Zholkovskij (1971)
0002 0330 8000 0073

I.A. MEL'TCHUK & A.K. ZHOLKOVSKIJ, "Construction d'un modèle actif de la langue "sens-texte" ". Doc. de Ling. Quant. 10, 11-50, Dunod, 1971. (Trad. de Zholkovskij & Mel'tchuk (1969)).

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Meltzer & Mitchie (1971)
0353 0803 4304 4050

B. MELTZER & D. MITCHIE, eds, "Machine Intelligence 6". Edinburg Univ. Press, 1971, 525p.

Mendelson (1964)
0000 0009 0700 0000

E. MENDELSON, "Introduction to mathematical logic" Van Nostrand, 1964, 300p.

Mercier, Vives, Quinton (1975)
0006 0500 4300 6000

G. MERCIER, R. VIVES & P. QUINTON, "Quelques résultats obtenus en reconnaissance de la parole continue". 6° JE sur la parole, Toulouse, Mai 1975.

Meyer (1972)
0000 8060 0700 0600

A.R. MEYER, "Program size in restricted programming languages". I & C 21, 382-394, 1972.

Meyer & Fischer (1972)
0000 8000 0700 0700

A.R. MEYER & P.C. FISCHER, "Computational seed-up by effective operators". The Journal of Symbolic Logic, V37, N1, March 1972, 55-68.

Mickunas & Schneider (1973)
0006 0030 0000 7000

M.D. MICKUNAS & V.B. SCHNEIDER, "A parser generating system for constructing compressed compilers". CACM, V16, N11, 669-676, nov. 1973.

Minsky (1961)
5000 0000 0700 0800

M. MINSKY, "Recursive unsilvability of Post's problem of 'tag' and other topics in the theory of Turing machines". Annals of mathematics, V74, 437-455, 1971.

Minsky (1967)
7000 0030 0400 0400

M. MINSKY, "Computation : finite and infinite machines". Prentice-Hall, London , 1967.

Minsky (1968)
0300 0800 0400 5070

M. MINSKY, ed, "Semantic information processing". MIT Press, 1968.

Minsky (1972)
0000 0000 0070 0000

M. MINSKY, "Form and content in computer science". JACM, V17, N2, April 70, 197-215.

Minsky & Papert (1969)
5060 0500 0400 0000

M. MINSKY & S. PAPERT, "Perceptrons : an introduction to computational geometry". MIT Press, 1969.

Mounin (1967)
0000 0000 7000 0000

G. MOUNIN, "Les problèmes théoriques de la traduction". Gallimard, Paris, 1967.

Nida (1962)
0000 0000 6000 0070

E. NIDA, "Semantic components". Babel, V8, N4, 175-181, 1962.

Nilsson (1965)
0080 0600 0600 0000

N. NILSSON, "Learning machines". McGraw-Hill, New-York, 1965.

Nilsson (1967)
0660 0800 0600 0000

N. NILSSON, "Problem solving methods in artificial intelligence". McGraw-Hill, New-York, 1967.

Ollongren (1974)
7000 0070 0700 6000

A. OLLONGREN, "Definition of programming languages by interpreting automata". Academic Press, 1974.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Ouspenskij (1966)
2000 0003 0700 0800

V.A. OUSPENSKIJ, "Leçons sur les fonctions calculables". Hermann, 1966, 412p.

Owings (1973)
0000 0000 0800 0800

J.C. OWINGS, Jr, "Diagonalization and the recursion theorem". Notre-Dame Journal of Formal Logic, XIV, N1, Janv. 1973, 95-99.

Owings (1975)
0000 0060 0700 0800

J.C. OWINGS, Jr, "Splitting a context-sensitive set". JCSS, V10, N1, Feb. 1975, 83-87.

Padutcheva (1974)
0000 0000 8000 0080

Е.В.ПАДУЧЕВА, "О семантике синтаксиса". Наука, Москва, 1974.

Pager (1969)
0000 7000 0700 0800

D. PAGER, "On the problem of finding minimal programs for tables". I & C 14, 550-554, 1969.

Pager (1970)
0000 7000 0600 0800

D. PAGER, "On the efficiency of algorithms". JACM, V17, N4, Oct. 70, 704-714.

Pager (1974)
0000 7000 0700 0800

D. PAGER, "Further results on the problem of finding minimal length programs for decision tables". JACM, V21, N2, April 74, 207-212.

Pankowicz (1967)
0000 0000 0000 0008

Z. PANKOWICZ, "Commentary on the ALPAC report". March 1967.

Panov (1960)
0000 0000 0000 4008

JU. D. PANOV, "Automatic translation", Pergamon Press, 1960.

Peizer & Olmstedt (1969)
0280 0000 3406 0000

D.B. PEIZER & D.L. OLMSTEDT, "Modules of grammar acquisition". Language, V45, N1, 1969, 60-96.

Pendergraft (1971)
0000 0300 5050 0067

E.D. PENDERGRAFT , "Meaning revisited". In "Feasibility study of FAHQT", RADC-TR-71-295, 189-323, Univ. of Texas, 1971.

Pentonnen (1974)
0000 0080 0800 0000

M. PENTONNEN, "One-sided and two-sided context in formal grammars". I & C 25, 371-392, 1974.

Peters & Ritchie (1969)
0006 0080 0700 0000

P.S. PETERS & R.W. RITCHIE, "Context-sensitive immediate constituent analysis. Context-free languages revisited". ACM Symposium on theory of computing, 1969, 1-8.

Peters & Tabory (1969)
0003 0070 5600 0000

P.S. PETERS & R. TABORY, "Ambiguity, completeness and restriction problems in the syntax-based approach to computational linguistics". Linguistics 46, 54-76, 1969.

Petrick (1965)
0806 0040 0600 7000

S.R. PETRICK, "A recognition procedure for transformational grammars". Ph. D. Thesis, MIT Dep. of modern languages, Cambridge, Mass. 1965.

Pognan (1975)
0500 0200 7200 6030

P. POGNAN, "Analyse morpho-syntaxique automatique du discours scientifique tchèque". Doc. de Ling. Quant. 20, Dunod, 1975, 262 p.

Poitou (1963)
0000 0000 0800 0000

G. POITOU, "Introduction à la théorie des catégories". Cours Univ. de Lille, 1963-64.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Post (1936)
0000 0000 0600 0800

E. POST, "Finite combinatory processes - formulation I". The Journal of Symbolic Logic, V1, 103-105, 1936.

Post (1947)
0000 0000 0700 0800

E. POST, "Recursive unsolvability of a problem of Thue". The Journal of Symbolic logic, V12, 1-11, 1947. Also in "The Undecidable", M. Davis, ed, Raven Press, 1965, 292-303.

Postal (1964)
0000 0060 7300 0000

P. POSTAL, "Limitations of phrase structure grammars". In "The structure of language", J.A. Fodor & J.J. Katz, eds, Englewood Cliffs, 1964.

Pour-El & Putnam (1965)
0000 0006 0700 0800

M.B. POUR-EL & H. PUTNAM, "Recursively enumerable classes and their applications to recursive sequences of formal theories". Archiv. Math. Logik und Grundlagenforschung 8, 104-121, 1965.

Quézel-Ambrunaz & Guillaume (1973)
3000 0040 2000 6007

M. QUEZEL-AMBRUNAZ & P. GUILLAUME , "Analyse automatique de textes par un système d'états finis". Conf. ICCL, Pise 1973.

Quillian (1968)
0303 0800 3000 6070

R. QUILLIAN, "Semantic memory". In "Semantic information processing", MIT Press, 1968, 216-270.

Quillian (1969)
0203 0800 3004 6070

R. QUILLIAN, "The teachable language comprehender : a simulation program and a theory of language". CACM, V12, N8, Aug. 1969, 459-476.

Quine (1961)
0000 0008 0660 0070

W.V.O. QUINE, "From a logical point of view". Harvard Univ. Press, 1961.

Quinton (1975)
0508 0470 3700 3000

P. QUINTON, "Analyse syntaxique ascendante et reconnaissance de la parole", comm. pers. , 1975.

Quinton & Caylux (1974)
0005 0060 0200 6000

P. QUINTON & B. CAYLUX, "Système conversationnel d'analyse syntaxique", 5e JE sur la parole, GALF, Orsay, mai 1974.

Raphael (1968)
0000 0800 0000 6070

B. RAPHAEL, "SIR : semantic information retrieval". In "Semantic information processing", MIT Press, 1968.

Reingold (1972a)
0000 8000 0700 0700

E.M. REINGOLD, "Establishing lower bounds on algorithms, a survey", Proc. 1972 Spring Joint Computer Conf., AFIPS, V40, 471-481, 1972.

Reingold (1972b)
0000 8000 0700 0700

E.M. REINGOLD, "On the optimality of some sets of algorithms". JACM, V19, N4, Oct. 72, 649-659.

Riesbeck (1973)
0000 0700 3007 5060

C.K. RIESBECK, "Expectation as a basic mechanism of language comprehension". Conf. ICCL, Pise, 1973.

Riesbeck (1974)
0002 0700 4005 4002

C.K. RIESBECK, "Computational understanding : analysis of sentences and context". WP-4, Istituto per gli Studi Semantici e Cognitivi, Castagnola, 1974.

Robinson (1950)
0000 0000 0800 0800

J. ROBINSON, "General recursive functions". Proc. Amer. Math. Soc., V1, 707-718, 1950.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Rogers (1959)
0000 0000 0800 0800

H. ROGERS, Jr, "Computing degrees of unsolvability". Mathematische Annalen, V138, 125-140, 1959.

Rogers (1967)
2200 0006 0800 0900

H. ROGERS, Jr, "Theory of recursive functions and effective computability". McGraw-Hill, 1967, 482 p.

Rosenzweig (1964)
0000 0000 7000 0003

В.Ю.РОЗЕНЦВЕЙГ, "Перевод и трансформация". В "Трансформационный метод в структурной лингвистике", Москва, 1964, 87-97.

Rosenzweig (1971)
0000 0000 7300 0004

V. JU. ROSENZWEIG, "Les modèles dans la linguistique soviétique". Moscou, 1971.

Rosenzweig (1974)
0000 0000 6000 0037

V. JU. ROSENZWEIG, ed, "Machine translation and applied linguistics". In "Soviet papers in formal linguistics", F. Kiefer, ed., 2 vol, Athenaion, 1974.

Rouanet (1967)
0080 0000 0506 0000

H. ROUANET, "Les modèles stochastiques d'apprentissage". Gauthier-Villars, Paris, 1967.

Rouault (1971)
0000 0005 6600 0070

J. ROUAULT, "Approche formelle des problèmes liés à la sémantique des langues naturelles". Thèse d'Etat, Grenoble, nov. 1971.

Rustin (1973)
0000 8000 0700 0700

R. RUSTIN, ed. "Computational complexity". Courant computer science, symp. 7 , Algorithmics Press, 1973.

Ruwet (1967)
3002 0050 7200 0000

N. RUWET, "Introduction à la grammaire générative". Recherches en Sciences Humaines 22, Plon, Paris, 1967, 448p.

Salkoff (1973)
0407 0040 7400 5004

M. SALKOFF, "Une grammaire en chaîne du français". Dunod, Paris, 1973.

Salomaa (1969a)
0000 0070 0600 0000

A. SALOMAA, "Probabilistic and weighted grammars". I & C 15, 529-544, 1969.

Salomaa (1969b)
9000 4040 0700 0000

A. SALOMAA, "Theory of automata". Pergamon Press, 1969.

Salomaa (1973)
7000 0090 0700 0500

A. SALOMAA, "Formal languages". Academic Press, 1973.

Sandewall (1971)
0000 0500 5600 0070

E. SANDEWALL, "Representing natural language information in predicate calculus". In "Machine Intelligence 6", Meltzer & Mitchie, eds, American Elsevier, New-York, 1971.

Santos (1969)
6000 0000 0700 0800

E.S. SANTOS, "Probabilistic Turing machines and computability". Proc. Amer. Math. Soc. 22, 704-710, 1969.

Santos (1972)
7000 0070 0600 0000

E.S. SANTOS, "Probabilistic grammars and automata". I & C 21, 27-47, 1972.

Santos & Wee (1968)
8000 0000 0600 0700

E.S. SANTOS & W.G. WEE, "General formulation of sequential machines". I & C 12, 5-10, 1968.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Savitch (1970)
3000 7000 0600 0600

W.J. SAVITCH, "Relationship between nondeterministic and deterministic tape complexities". JCSS 4, 177-192, 1970.

Savitch (1973)
7000 7000 0600 0600

W.J. SAVITCH, "Maze recognizing automata and nondeterministic tape complexity". JCSS 7, 389-403, 1973.

Savitch (1975)
5000 0070 0700 0000

W.J. SAVITCH, "Some characterizations of Lindenmayer systems in terms of Chomsky-type grammars and stack machines". I & C 27, 37-60, 1975.

Schreider (1966)
0000 7000 6600 0000

Ю.Д.ШРЕЙДЕР, "Характеристики комплексности структуры текста". НТИ, №7, 34-41, 1966.

Schreider (1973)
0000 0006 5400 0000

JU. A. SCHREIDER, "On the contrast between the concepts 'language model' and 'mathematical model' ". In "Mathematical models of language", Soviet papers in formal linguistics, F. Kiefer, ed, Athenäum, 1973. Trad. de "Модели в лингвистике и в математике", Математическая лингвистика, 63-84, Москва, 1973.

Schubert (1974)
0000 6000 0700 0700

L.K. SCHUBERT, "Iterated limiting recursion and the program minimization problem". JACM, V21, N3, July 1974, 436-445.

Schuler (1974)
0000 0070 0700 2600

P.F. SCHULER, "Weakly context-sensitive languages as model for programming languages". Acta Informatica 3, 155-170, 1974.

Schuler (1975)
0000 0070 0700 0600

P.F. SCHULER, "WCS-analysis of the context-sensitive". Acta Informatica 4, 359-371, 1975.

Shaljapina (1972)
0000 0000 6000 0055

З.М.ШАЛЯПИНА, "Семантические элементы как основа лексикографического описания: общие принципы - формальный аппарат". МП и ПЛ, Выпуск 15, 54-98, Москва, 1972.

Shaljapina (1973)
0000 0000 7000 0055

З.М.ШАЛЯПИНА, "Англо-русский многоаспектный автоматический словарь". МП и ПЛ, Выпуск 17, 7-68, Москва, 1973.

Shamir (1962)
0040 0050 0600 0600

E. SHAMIR, "A remark on discovery algorithms for grammars". I & C 5, 246-251, 1962-A.

Shank (1973a)
0001 0800 2005 2080

R.C. SHANK, "Identification of conceptualizations underlying natural language". In "Computer models of thought and language", 187-247, Shank & Colby, eds, Freeman & Co, San Francisco, 1973.

Shank (1973b)
0000 0600 3005 0070

R.C. SHANK, "Causality and reasoning", WP-1, Istituto per gli studi semantici e cognitivi, Castagnola, 1973.

Shank (1974a)
0000 0700 4007 2070

R.C. SHANK, "Is there a semantic memory ?" WP-3, Istituto per gli studi semantici e cognitivi, Castagnola, 1974.

Shank (1974b)
0000 0700 4000 5063

R.C. SHANK, "Understanding paragraphs". WP-6, Istituto per gli studi semantici e cognitivi, Castagnola, 1974.

Shank & Colby (1973)
0003 0800 6006 6070

R.C. SHANK & K.M. COLBY, eds, "Computer models of thought and language". Freeman & Co, 1973, 454p.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Shank & Rieger (1973)
0001 0700 2002 4080

R.C. SHANK & C.J. RIEGER III, "Inference and conceptual memory". Conf. ICCL, Pise , 1973.

Shank & Rieger (1974)
0000 0800 2005 0080

R.C. SHANK & C.J. RIEGER III, "Inference and the computer understanding of natural language". Artificial Intelligence 5, 373-412, 1974.

Shannon (1956)
6000 4000 0500 0700

C.E. SHANNON, "A universal Turing machine with 2 states". In "Automata studies", 157-169, C.E. Shannon & J.C. McCarthy, eds, Princeton Univ. Press, 1956.

Shaumjan (1965)
0000 0000 8400 0040

С.К.ШАУМЯН, "Структурная лингвистика". Москва, 1965. Ed. augm. "Principles of structural linguistics", Miller, trad., Mouton, 1971.

Shepherdson & Sturgis (1963)
8000 0000 0500 0800

J.C. SHEPHERDSON & H.E. STURGIS, "Computability of recursive functions". JACM, V10, 217-255, 1963.

Simmons (1970)
0003 0700 4000 6050

R.F. SIMMONS, "Natural language question-answering systems". CACM, V13, 15-31, 1970.

Simmons (1973)
6504 0600 3036 4060

R.F. SIMMONS, "Semantic networks : their computation and use for understanding english sentences", in "Computer models of thought and language", 63-113, Shanl & Colby, eds, Freeman & Co , 1973.

Simmons & Slocum (1972)
0000 0700 2000 4060

R.F. SIMMONS & J. SLOCUM, "Generating english discourse from semantic networks". CACM, V15, N10, Oct. 72, 891-905.

Simon (1974)
0080 2440 0600 4000

J.C. SIMON, ed, "Procédures informatiques d'apprentissage". Séminaire DRME, IP, IRIA, OTAN, Bonas, Gers, 1974.

Slagle (1970)
0600 0700 0400 6000

J.R. SLAGLE, "Heuristic search programs". In "Theoretical approaches to non-numerical problem-solving", 246-273, R. Banarji & M. Mesarovic, eds, Springer, New-York, 1970.

Slagle & Dixon (1969)
0600 0700 0400 6000

J.R. SLAGLE & J.K. DIXON, "Experiments with some programs that search game trees". JACM, V16, N2, April 1969, 189-207.

Slagle & Lee (1971)
0600 0600 0400 4000

J.R. SLAGLE & R.C. LEE, "Application of game tree searching techniques to sequential pattern recognition". CACM, V14, N2, Feb. 1971, 103-110.

Solomonoff (1959)
4060 0600 0500 0000

R.J. SOLOMONOFF, "A progress report on machines to learn to translate languages and retrieve information". Advances in documentation and library sciences, V3, Part 2, 941-953, Interscience, New-York, 1959.

Solomonoff (1960)
0070 0630 2500 0000

R.J. SOLOMONOFF, "The mechanization of linguistic learning". Proc. Second Int. Congr. Cybernetics (1958), Namur, Belgium, 180-193, 1960.

Solomonoff (1964)
4080 0040 0600 0000

R.J. SOLOMONOFF, "A formal theory of inductive inference". I : I & C 7, 1-22, 1964. II : I & C 7, 224-254, 1964.

Stachowitz (1971)
0000 0200 5060 0008

R. STACHOWITZ, "Requirements for MT : problems, solutions, prospects". In "Feasibility study of FAHQT", 409-576, RADC-TR-71-295, Univ. of Texas, 1971.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Sternberg (1063)
0070 0000 0400 0000
S. STERNBERG, "Stochastic learning theory". In "Handbook of mathematical psychology", Luce, Bush & Galanter, eds, Wiley, 1963, 1-120.

Sugita (1968)
0006 0000 5200 6008
S. SUGITA, "A study of mechanical translation from english into japanese". Doct. th., Kyoto Univ., 1968.

Thorne & al (1968)
4008 0030 5300 6000
J. THORNE, P. BRATLEY & H. DEWAR, "The syntactic analysis of English by machine". In "Machine Intelligence 3", American Elsevier, New-York, 1968.

Thouin (1975)
0000 0000 0000 5006
B. THOUIN, "Systèmes informatiques pour l'analyse et la génération de langues naturelles". DEA inf., Grenoble, sept. 1975.

Thue (1914)
0000 0004 0600 0700
A. THUE, "Probleme über Veränderungen von Zeichenreihen nach gegebenen Regeln". Skr. Videnskab. Selsk. I, 10, 1914.

Tseitin & al (1966)
0305 0000 5300 6007
Б.М.ЛЕЙКИНА, Т.Н.НИКИТИНА.М.И.ОТКУПЩИКОВА, С.Я.ФИТИАЛОВ, Г.С.ЦЕЙТИН, "Система автоматического перевода, разрабатываемая в группе лингвистики ВЦ ЛГУ". НТИ, №1, 40-50, 1966.

Tseitin (1971)
0707 0060 0600 0004
Г.С.ЦЕЙТИН, "Алгорифм для упрощенного синтаксического анализа". Проблемы Кибернетики, №24, Москва, 1971.

Tsypkin (1971)
5680 0000 0600 0000
YA. Z. TSYPKIN, "Adaptation and learning in automatic systems". Academic Press, 1971.

Tsypkin (1973)
0680 0000 0700 0000
YA. Z. TSYPKIN, "Foundations in the theory of learning systems". Academic Press, 1973.

Tubach (1970)
0666 0000 0400 6000
J.P. TUBACH, "Reconnaissance automatique de la parole . Etude et réalisation fondée sur les niveaux acoustique, morphologique et syntaxique." Thèse d'Etat, Grenoble, 1970.

Turing (1936-37)
0000 0008 0700 0700
A. TURING, "On computable numbers, with an application to the Entscheidungsproblem". Proc. London Math. Soc., S2, V42, 230-265, 1936-37. Also in "The Undecidable", M. Davis, ed, Raven Press, 1965.

Uhr (1973)
0760 0800 0246 0040
L. UHR, "Pattern recognition, learning and thought", Prentice-Hall, 1973.

Vaissière (1971)
0000 0600 2002 2000
J. VAISSIERE, "Contribution à la synthèse par règles du français". Thèse IIIe cycle, Grenoble, 1971.

Van Caneghem (1971)
0000 0000 7000 6006
M. VAN CANEGHEM, "Morphographie générative du français". Thèse, Dép. Inf. UdM , Montréal, 1971.

Vauquois (1966)
0000 0000 0050 0006
B. VAUQUOIS, "Présentation du CETA du CNRS". TA-Informations, N1, 1-18, 1966.

Vauquois (1967)
0004 0000 0050 5008
B. VAUQUOIS, "Le système de TA du CETA". Conf. sur la TA, Erevan, 1967.

Vauquois (1968a)
0605 0070 5300 5008
B. VAUQUOIS, "A survey of formal grammars and algorithms for recognition and transformation in mechanical translation". IFIP Congress 1968., North-Holland.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Vauquois (1968b)
0004 0060 4000 5057

B. VAUQUOIS, "Structures profondes et traduction automatique . Le système du CETA". Revue Roumaine de Linguistique, 13, 105-130, 1968.

Vauquois (1969)
3000 0070 0600 0300

B. VAUQUOIS, "Calculabilité des langages". Cours, Grenoble, 1969.

Vauquois (1975)
4406 0060 4350 0009

B. VAUQUOIS, "La traduction automatique à Grenoble". Doc. de Ling. Quant.24, Dunod, 1975, 184p.

Vauquois, Veillon, Veyrunes (1964)
0005 0060 5000 0048

B. VAUQUOIS, G. VEILLON, J. VEYRUNES, "Application des grammaires formelles aux modèles linguistiques en TA". Colloque de Prague, sept. 1964.

Vauquois, Veillon, Veyrunes (1965)
0506 0000 4300 0048

B. VAUQUOIS, G. VEILLON, J. VEYRUNES, "Syntax and interpretation". Conf. ICCL, New-York, 1965. Also in Mechanical Translation, V9, N2, 44-54, 1966.

Vauquois, Veillon, Veyrunes (1967)
0300 0040 0000 6007

B. VAUQUOIS, G. VEILLON, J. VEYRUNES, "Un métalangage de grammaires transformationnelles". Conf. ICCL, Grenoble, 1967.

Veillon (1968)
0000 0000 5000 0068

G. VEILLON, "Le langage pivot du CETA". TA-informations, N1, 1968.

Veillon (1970)
3706 0050 0600 6009

G. VEILLON, "Modèle et algorithmes pour la traduction automatique". Thèse d'Etat, Grenoble, 1970.

Veillon & Veyrunes (1964)
0607 0040 0300 6004

G. VEILLON & J. VEYRUNES, "Etude de la réalisation pratique d'une grammaire context-free et de l'algorithme associé". TA-informations, V5, 1964.

Vivier (1974)
5685 0230 0500 3000

G. VIVIER, "Essai d'inférence de grammaires pour des langages faciles à apprendre". Grenoble, séminaire de programmation, déc. 1974.

Vuillemin (1974a)
0000 0070 0700 0050

J. VUILLEMIN, "Correct and optimal implementation of recursion in a simple programming language". JCSS 9, 332-354, 1974.

Vuillemin (1974b)
0000 0070 0800 3050

J. VUILLEMIN , "Syntaxe, sémantique et axiomatique d'un langage de programmation simple". Thèse d'Etat, Univ. Paris VI, sept. 1974.

Wall (1972)
4000 0060 3500 0000

R. WALL, "Introduction to mathematical linguistics". Prentice-Hall, 1972.

Wang (1957)
4000 0000 0600 0800

H. WANG, "A variant to Turing's theory of computability". JACM 4, 63-92, 1957.

Watanabe (1972)
0000 0800 0300 4000

S. WATANABE, ed, "Frontiers of pattern recognition". Academic Press, 1972, 602p.

Werner (1973)
4000 8000 0800 0800

G. WERNER, "Problèmes de décidabilité et d'effectivité en théorie de la complexité". Thèse d'Etat, Grenoble, 1973.

Wharton (1974)
0280 3000 0500 0600

R.M. WHARTON, "Approximate language identification". I & C 26, 236-255, 1974.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Wilks (1968)
0003 0600 3000 0075

Y. WILKS, "On-line semantic analysis of english texts". Mechanical Translation, V11, N3 & 4, 59-72, 1968.

Wilks (1972)
0200 0700 3072 0076

Y. WILKS, "Grammar, meaning and the machine analysis of language". Routledge, London, 1972, 198p.

Wilks (1973a)
0200 0700 3072 0076

Y. WILKS, "An artificial intelligence approach to machine translation". In "Computer models of thought and language", 114-151, Shank & Colby, eds, Freeman & Co, 1973.

Wilks (1973b)
0200 0700 3002 0080

Y. WILKS, "Preference semantics". Stanford AI Laboratory, AIM-206, Stan-CS-73-377, July 1973.

Wilks (1975)
0200 0700 3002 0076

Y. WILKS, "An intelligent analyzer and understander of english". CACM, V18, N5, 264-274, May 1975.

Wilks & Herskowits (1973)
0200 0700 3002 4076

Y. WILKS & A. HERSKOWITS, "An intelligent analyzer and generator for natural language". Conf. ICCL, Pise, 1973.

Wilks & King (1976)
0300 0800 3042 6076

Y. WILKS & M. KING, "Semantics, Preference and Inference. A full description of a system and a program". WP-18, Istituto per gli studi semantici e cognitivi, Geneva, 1976.

Winograd (1971)
0504 0800 5060 5070

T. WINOGRAD, "Procedures as a representation for data in a computer program for understanding natural language". AI-TR-17, MIT, Cambridge, Mass., Janv. 71.

Winograd (1973)
0200 0700 3064 2070

T. WINOGRAD, "Procedural model of language understanding". In "Computer models of thought and language", 152-186, Shank & Colby, eds, Freeman & Co, San Francisco, 1973.

Wlodarczyk (1974)
0000 0060 7400 0060

H. WLODARCZYK, "La grammaire générative applicative de S.K. Saumjan". Langages, 33, 15-64, mars 1974.

Woods (1969)
7207 4000 4500 6000

W.A. WOODS, "Augmented transition networks for natural language analysis". TR-CS-1, Harvard Univ., dec. 1969.

Woods (1970)
5307 3000 4000 3000

W.A. WOODS, "Transition network grammars for natural language analysis". CACM, V13, N10, Oct. 70, 591-606.

Woods (1972)
6007 0000 0500 7000

W.A. WOODS, "An experimental parsing system for transition network grammars". BBN report N° 2362, May 1972.

Woods (1975)
3003 0700 4200 0070

W.A. WOODS, "Syntax, semantics and speech". BBN report n° 3067, AI report n° 27, April 1975. Also in "Speech recognition : invited papers presented at the IEEE Symposium", D.R. Reddy, ed, Academic Press, 1975.

Woods & al (1972)
3003 0700 3000 6000

W.A. WOODS, R.M. KAPLAN & B. NASH-WEBER, "The Lunar Sciences natural language information system-Final report". BBN report n° 2378, June 1972.

Yates (1962)
0000 0000 0800 0800

C.E.M. YATES, "R.e. sets and retracing functions". Zeitschrift für mathematische Logik und Grundlagen der Mathematik, V8, 331-345, 1962.

AF CP LN PR
AL IA MA RC
AP LF PH SM
AS LG PL TA

Yngve (1961)
0000 0000 0060 0006

V. YNGVE, "MT at the MIT". Proc. Nat. Symp. on MT, Prentice-Hall, 1961, 126-132.

Yngve (1964)
0000 0000 5050 0006

V. YNGVE, "Theorie und Praxis der maschinellen Sprachübersetzung". Zeitschrift für Phonetik, V17, N1, 1964, 103-117. Trad. in "Автоматический перевод", 214-234, Москва, 1971.

Yngve (1957)
0000 0000 0050 0007

V. YNGVE, "A framework for syntactic translation". Mechanical Translation, V14, N3, dec. 1957.

Young (1969)
0000 5000 0700 0800

P.D. YOUNG, "Toward a theory of enumerations". JACM, V16, N2, April 69, 328-348.

Young (1971)
0000 7000 0600 0600

P.D. YOUNG, "A note on "axioms" for computational complexity and computation of finite functions". I & C 19, 377-386, 1971.

Zadeh (1965)
0000 0000 0600 0000

L.A. ZADEH, "Fuzzy sets". I & C 8, 338-353, 1965.

Zampolli (1973)
0000 0000 4060 4000

A. ZAMPOLLI, "L'automatisation de la recherche lexicographique, état actuel et tendances nouvelles". META, V12, N1&2, 1973.

Zholkovskij & Mel'tchuk (1967)
0000 0000 8000 0085

А.К.ЖОЛКОВСКИЙ и И.А.МЕЛЬЧУК, "О семантическом синтезе". Проблемы Кибернетики, №19, 117-238, Москва, 1967. Trad. in TA-informations, N2, 1970.

* ALPAC (1966)
0000 0000 0070 0008

ALPAC REPORT, "A report by the Automatic Language Processing Advisory Committee". Languages & machines. Computers in translation and linguistics. Nat. Academy of Sciences, Nat. Research Council, Washington D.C., 1966.

* Georgetown (1963)
0405 0020 6200 5008

McDONALD & DOSTERT, MT Research Project, General report 1952-1963, N.30, June 1963, Georgetown Univ.

* Groupe LEIBNIZ (1975)
0000 0050 6000 0048

GROUPE LEIBNIZ, "Projet de représentation des structures de phrase au niveau du transfert". Lugano, mars 1975.

* PANKOWICZ (1973)
0000 0000 0000 0008

Z.B. PANKOWICZ, technical evaluator, "User's evaluation of machine translation. Georgetown MT system (1963-1973)". RADC-TR-73-239, Univ. of Texas at Austin, 1973.

* RADC (1971)
0002 0000 6070 2028

ROME AIR DEVELOPMENT CENTER, "Feasibility study on fully automatic high quality translation (FAHQT)". Final technical report, 1971, 2 vol., RADC-TR-71-295, Univ. of Texas.

* TAUM (1971)
0606 0060 7000 7008

A. COLMERAUER, B. HARRIS, R. KITTREDGE, J. DANSEREAU, M. VAN CANEGHEM, "Rapport TAUM 71", Projet TAUM, UdM , Montréal, janvier 1971.

Thèse Boitet

ERRATA

[7↑ ligne 7 à partir du bas
↑ au milieu
/xxx/yy/ changer xxx en yy

Page.

vi 7↑ /Porcédures/Procédures/
vii ↑ /vers a/vers/

I-6 ↑ /classe de fonctions/classe $\mathcal{C}$ de fonctions/
I-16 5↓ /utilise/utilisent/
I-32 Prop. 8, (3) /f récursive/f récursive et Im f $\neq \omega$/
I-38 2↓ /$h(p) = \operatorname{Max} F^{p}(a_{\cdot})$/$h(0)=0$ et $h(p+1) = \operatorname{Max} F\{0,\dots,h(p)\}$/
I-39 3↓ /λ_n/λn/
Déf. 27, 2↓ /$n \in R^2$/$h \in R^2$/
I-43 Déf. 35, 1↓ /$\delta=(\varepsilon,\tilde{\eta},\gamma,Q)$ est/$\delta=(\varepsilon,\tilde{\eta},\gamma,Q)$, est/
I-3 9↑ /peut/peu/
I-8 Déf. 10, 1↓ à 4↓ /$W=(W_i)$/$W=(w_i)$/
I-9 3↑ /$1+\underset{i\leq n}{\operatorname{Max}}\ e_n^{(n)}(i)$/$1+\underset{i\leq n}{\operatorname{Max}}\ \varphi_{e_n(i)}^{(n)}$/
7↑ /≥/≤/
11↑ /C w_n/$f(n)$ w_n/
II-11 1↑ /{b,1}/{b,1/
4↑ /ρ_0/e_0/
↑ /$= h(x)]$/$= \varphi_{h(x)}]$/
II-12 Th. 4, 2↓ /P(i,j,ē) D/P(i,j,ē) ≡ D/
Schéma, 3↓ /- iFA/+iFA/
Echanger les deux cadres inférieurs droits ↳ EF ↰ PR
III-24 ↑ /stratégie/recherche/
IV-28 15↓ /ou/où/
III-31 1↑ /$\mid g(x)$/$\mid^{g(x)}$/
III-36 ↑ /$[q',j]\ q'$/$[q',j] \mid q'$/
III-37 7↑ (Schéma) /B(c)/b(c)/
IV-5 /MONIT → CETA/MONIT → CETA/
IV-11 5e /MERPHO, ARBOR/MORPHO, ARBOR/ et /SYGMØR/SYGMOR/
V-8 3↑ /inférieurs ou égaux/relatifs de valeur absolue inférieure ou égale/
V-21 3.3.3 1↓ /est/et/
V-25 ↑ (Schéma) /A2/A1/

Dernière page d'une thèse

VU

Grenoble, le

Le Président de la thèse

VU, et permis d'imprimer,

Grenoble, le 24 mars 1976

Le Président de l'Institut
National Polytechnique

Le Président de l'Université
Scientifique et Médicale