Transducteurs et arborescences : études et réalisations de systèmes appliquées aux grammaires transformationnelles

Jacques Chauché

N° D'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR ES SCIENCES MATHEMATIQUES

par

Jacques CHAUCHÉ

TRANSDUCTEURS & ARBORESCENCES

Etudes et réalisations de systèmes

appliquées aux grammaires transformationnelles

Thèse soutenue le 17 décembre 1974 devant la commission d'examen :

Monsieur	J. KUNTZMANN	Président
Messieurs	A. COLMERAUER	Examinateurs
	B. VAUQUOIS	
	G. VEILLON	
	J.M. ZEMB	

N° D'ordre

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

LE GRADE DE DOCTEUR ES SCIENCES MATHEMATIQUES

par

Jacques CHAUCHÉ

TRANSDUCTEURS & ARBORESCENCES

Etudes et réalisations de systèmes

appliquées aux grammaires transformationnelles

Thèse soutenue le 17 décembre 1974 devant la commission d'examen :

Monsieur	J. KUNTZMANN	Président
Messieurs	A. COLMERAUER	Examinateurs
	B. VAUQUOIS	
	G. VEILLON	
	J.M. ZEMB	

AVANT - PROPOS

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

M. Michel SOUTIF Président
M. Gabriel CAU Vice Président

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

PROFESSEURS TITULAIRES

MM.	ANGLES D'AURIAC Paul	Mécanique des fluides
	ARNAUD Georges	Clinique des maladies infectieuses
	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	Pédiatrie
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BEZES Henri	Chirurgie générale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Pathologie médicale
	BOUCHERLE André	Chimie et Toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologie
	CALAS François	Anatomie
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et Toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Oto-Rhino-Laryngologie
	CHATEAU Robert	Thérapeutique
	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
	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de Dermatologie et Syphiligraphie
	FAU René	Clinique neuro-psychiatrique

MM.	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
	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
	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mle	LUTZ Elisabeth	Mathématiques Pures
	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Seméiologie médicale
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	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
	REULOS René	Physique industrielle
	RINALDI Renaud	Physique
	ROGET Jean	Clinique de pédiatrie et de puériculture
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie
	SOUTIF Michel	Physique générale
	TANCHE Maurice	Physiologie
	TRAYNARD Philippe	Chimie générale
	VAILLANT François	Zoologie
	VALENTIN Jacques	Physique Nucléaire
	VAUQUOIS Bernard	Calcul électronique
Mme	VERAIN Alice	Pharmacie galénique
M.	VERAIN André	Physique
MM.	VEYRET Paul	Géographie
	VIGNAIS Pierre	Biochimie médicale
	YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM.	ASCARELLI Gianni	Physique
	CHEEKE John	Thermodynamique
	GILLESPIE John	I.S.N.
	ROCKAFELLAR Ralph	Mathématiques appliquées
	WOHLFARTH Erich	Physique du solide

PROFESSEURS SANS CHAIRE

Mle	AGNIUS-DELORD Claudine	Physique pharmaceutique
	ALARY Josette	Chimie analytique
MM.	BELORIZKY Elie	Physique
	BENZAKEN Claude	Mathématiques appliquées
	BERTRANDIAS Jean-Paul	Mathématiques appliquées
	BIAREZ Jean-Pierre	Mécanique
Mme	BONNIER Jane	Chimie générale
MM.	BRUGEL Lucien	Energétique
	CARLIER Georges	Biologie végétale
	CONTE René	Physique
	DEPASSEL Roger	Mécanique des Fluides
	GAUTHIER Yves	Sciences biologiques
	GAUTRON René	Chimie
	GIDON Paul	Géologie et Minéralogie
	GLENAT René	Chimie organique
	HACQUES Gérard	Calcul numérique
	HOLLARD Daniel	Hématologie
	HUGONOT Robert	Hygiène et Méd.Préventive
	IDELMAN Simon	Physiologie animale
	JANIN Bernard	Géographie
	JOLY Jean-René	Mathématiques pures
	JULLIEN Pierre	Mathématiques appliquées
Mme	KAHANE Josette	Physique
MM.	KUHN Gérard	Physique
	LUU-DUC-Cuong	Chimie Organique
	MAYNARD Roger	Physique du solide
	MULLER Jean-Michel	Thérapeutique
	PERRIAUX Jean-Jacques	Géologie et minéralogie
	PFISTER Jean-Claude	Physique du solide
Mle	PIERY Yvette	Physiologie animale
MM.	REBECQ Jacques	Biologie (CUS)
	REVOL Michel	Urologie
	REYMOND Jean-Charles	Chirurgie générale
	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIBILLE Robert	Construction Mécanique
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	AMBROISE-THOMAS Pierre	Parasitologie
	ARMAND Yves	Chimie
	BEGUIN Claude	Chimie organique
Mme	BERIEL Héléne	Pharmacodynamique
M.	BILLET Jean	Géographie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BOUCHET Yves	Anatomie
	BRODEAU François	Mathématiques (IUT B)
	BUISSON Roger	Physique
	BUTEL Jean	Orthopédie
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)

MM.	COHEN-ADDAD Jean-Pierre	Spectrométrie physique
	COLOMB Maurice	Biochimie médicale
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FONTAINE Jean-Marc	Mathématiques Pures
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GRIFFITHS Michaël	Mathématiques Appliquées
	GROS Yves	Physique (stag.)
	GROULADE Joseph	Biochimie médicale
	GUITTON Jacques	Chimie
	IVANES Marcel	Electricité
	JALBERT Pierre	Histologie
	KRAKOWIAK Secha	Mathématiques appliquées
Mme	LAJZEROWICZ Jeannine	Physique
MM.	LEROY Philippe	Mathématiques
	LOISEAUX Jean-Marie	Physique Nucléaire
	MACHE Régis	Physiologie végétale
	MAGNIN Robert	Hygiène et Médecine préventive
	MARECHAL Jean	Mécanique
	MARTIN-BOUYER Michel	Chimie (CUS)
	MICHOULIER Jean	Physique (I.U.T. "A")
Mme	MINIER Colette	Physique
MM.	MICOUD Max	Maladies infectieuses
	NEGRE Robert	Mécanique
	PARAMELLE Bernard	Pneumologie
	PECCOUD François	Analyse (IUT B)
	PEFFEN René	Métallurgie
	PELMONT Jean	Physiologie animale
	PERRET Jean	Neurologie
	PHELIP Xavier	Rhumatologie
	RACHAIL Michel	Médecine interne
	RACINET Claude	Gynécologie et obstétrique
	RAYNAUD Hervé	M.I.A.G.
	RENAUD Maurice	Chimie
Mme	RENAUDET Jacqueline	Bactériologie
M.	RICHARD Lucien	Botanique
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROMIER Guy	Mathématiques (IUT B)
	SHOM Jean Claude	Chimie Générale
	STIEGLITZ Paul	Anesthésiologie
	STOEBNER Pierre	Anatomie pathologique
	VROUSOS Constantin	Radiologie

MAITRES DE CONFERENCES ASSOCIES

MM.	CRABBEE Pierre	C.E.R.M.O.
	CABOT	Mathématiques appliquées
	CURRIE Jan	Mathématiques appliquées

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

MM.	BARGE Michel	Neuro-chirurgie
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	DENIS Bernard	Cardiologie
	KOLODIE Lucien	Hématologie
	RAMBAUD Pierre	Pédiatrie
	ROCHAT Jacques	Hygiène et hydrologie

Fait à Saint Martin d'Hères, JANVIER 1974.

A ma femme

Les travaux qui ont fait l'objet de cette thèse ont été effectués dans l'équipe du Groupe d'Etudes pour la Traduction Automatique (G.E.T.A.) du Laboratoire d'Informatique de l'U.E.R. "I.R.M.A.".

Monsieur KUNTZMANN, Professeur à l'Université Scientifique et Médicale de Grenoble, m'a fait l'honneur de présider le jury. Je le remercie vivement pour l'intérêt qu'il a toujours manifesté à l'égard de ces travaux.

Monsieur VAUQUOIS, Professeur à l'Université Scientifique et Médicale de Grenoble, a dirigé l'ensemble de ces travaux. Je le remercie profondément pour m'avoir aidé par ses nombreux conseils en suivant de très près l'évolution de ces travaux.

Monsieur COLMERAUER, Professeur à l'Université d'Aix Marseille, a bien voulu s'intéresser à ce travail, critiquer la rédaction provisoire et faire partie du jury. Je le prie de trouver ici l'expression de toute ma reconnaissance.

Monsieur VEILLON, Professeur à l'Institut National Polytechnique de Grenoble, a bien voulu faire partie du jury. Je le remercie vivement.

Monsieur ZEMB, Professeur à l'Université de Paris, a accepté de se pencher sur ce travail. Je le prie de trouver ici l'expression de ma gratitude.

Je tiens enfin à remercier les membres du Groupe d'Etudes pour la Traduction Automatique avec lesquels j'ai travaillé et notamment
Messieurs GUILLAUME et QUEZEL-AMBRUNAZ qui ont participé activement à la programmation des systèmes.
Mesdemoiselles ALLOSIO et CURINIER qui ont dactylographié la rédaction provisoire et mes nombreuses corrections en vue de la rédaction définitive.
Les membres du service de reproduction du C.I.C.G. qui ont terminé la réalisation matérielle de ce travail.

SOMMAIRE

VII - LE SYSTEME C.E.T.A.

A- *Principe du système*

B- *Présentation des différents composants*

C- *Traitement informatique*

D- *Carte syntaxique du langage*

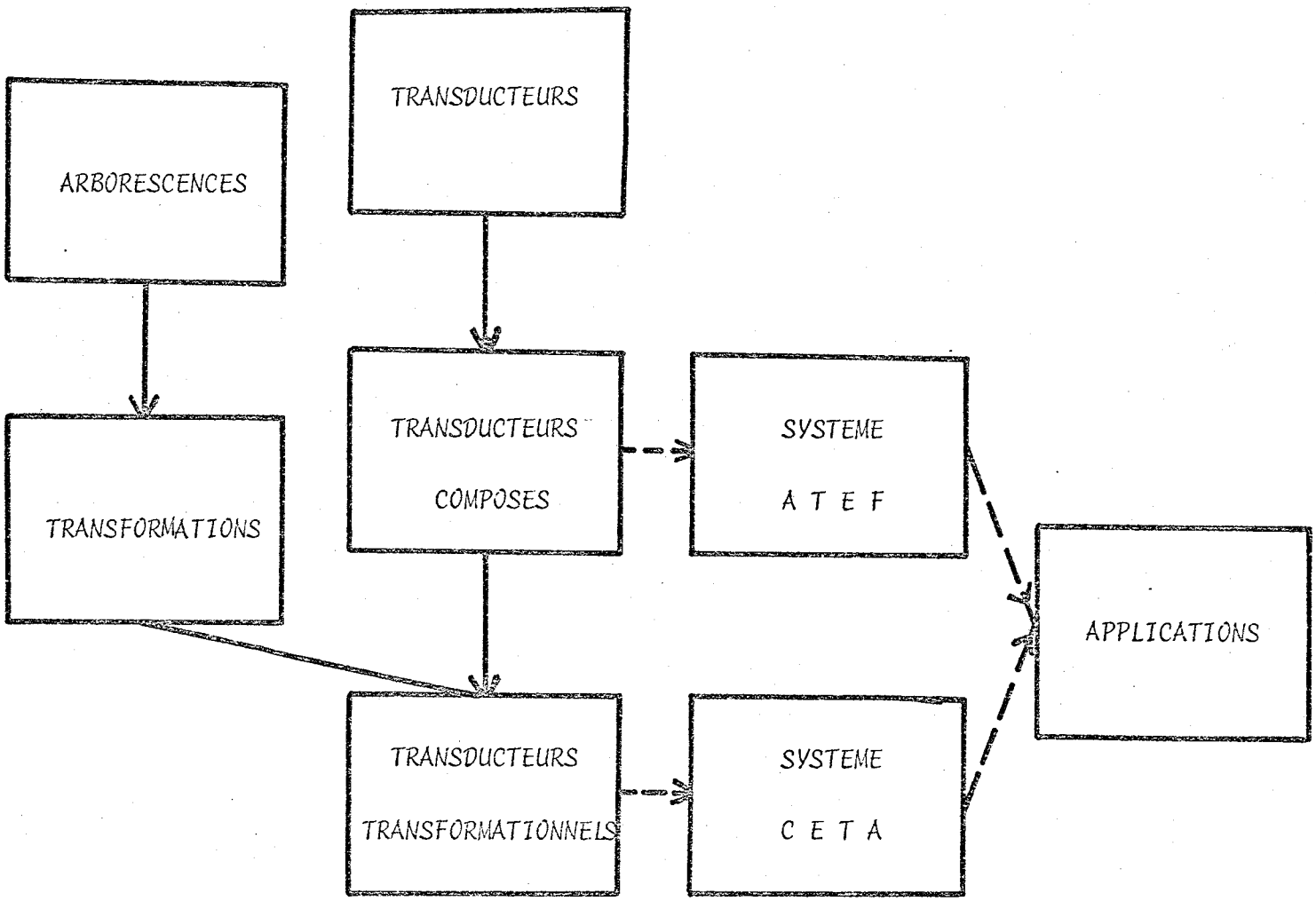
VIII - APPLICATIONS

A- *Application du système A.T.E.F.*

B- *Application du système C.E.T.A.*

- CONCLUSION

- BIBLIOGRAPHIE


ARBORESCENCES
TRANSDUCTEURS
TRANSFORMATIONS
TRANSDUCTEURS
COMPOSES
SYSTEME
A T E F
APPLICATIONS
TRANSDUCTEURS
TRANSFORMATIONNELS
SYSTEME
C E T A

Le développement de l'informatique a fait apparaître le problème de la communication homme-machine. De plus en plus des langages évoluës sont utilisés pour cette communication et chaque problème nécessite un langage particulier. L'accès à l'utilisation de l'informatique pour des personnes non spécialisées nécessite une évolution importante donnée par l'emploi de langues naturelles comme moyen de communication. Les problèmes posés par l'utilisateur de langues naturelles lors de la communication homme-machine se retrouvent très amplifiés en traduction automatique, le domaine d'étude étant alors beaucoup plus large. Des problèmes du même ordre sont d'ailleurs présents lors de l'élaboration d'un système de calcul formel : le langage de communication est plus restreint, mais les opérations nécessaires sont souvent plus complexes. Les différentes études réalisées jusqu'à présent (Müllner [43]) font apparaître ces problèmes comme liés à la manipulation de chaînes et d'arborescences. Pour aborder ces types de manipulations deux solutions apparaissent naturellement. L'une consiste à l'emploi d'un langage de programmation approprié. L'algorithme d'analyse et de transduction est alors défini dans ce langage. Ainsi le programme détermine deux éléments distincts qui sont le langage ou la langue analysée, et l'algorithme d'analyse et de transduction. Un des langages de programmation les plus utilisés dans ce domaine à l'heure actuelle est certainement le langage "LISP". Il permet la manipulation d'arborescences de façon aisée, ces éléments étant le principal support employé. Ce langage a permis la réalisation d'analyseur de langues naturelles et de systèmes de calculs formels. Une approche différente de ces problèmes est donnée par la réalisation de systèmes réalisant un algorithme (Colmerauer [23,24] , Veillon [77] , Winograd [79] , Woods [80,81]). L'utilisateur peut alors définir au moyen d'un langage une donnée particulière à cet algorithme. La particularité de ces systèmes est surtout donnée par l'incapacité pour l'utilisateur de modifier l'algorithme défini dans le système. Par contre, l'avantage de tels systèmes provient du fait que les données nécessaires à l'analyse deviennent indépendantes de l'algorithme programmé. La donnée de ces systèmes est souvent une grammaire syntagmatique (Ginsburg [32,35] , Gladky et Mel'čuk [38] , Gross et Lentin [39] , Hopcroft et Ullman [46] , Salomaa [70] , Vauquois [76]).
Plusieurs classes de grammaires ont été étudiées en fonction de la reconnaissance possible des langages qu'elles engendrent. A chacune de ces classes correspond une classe particulière d'algorithmes de reconnaissance . Un support privilégié de la description de ces algorithmes est donné par l'étude des automates (Arbib [9] , Edenberg et Wright [27] , Hanison [40] , Kain [46] , Minsky [58] , Salomaa [67]) . Ces automates réalisent des fonctions et l'étude

de la classe des fonctions simulée par un automate donne un moyen de comparaison entre ces diverses machines. L'automate le plus général, la machine de Turing (Fischer [29,31] , Hennie [42]) permet de simuler une classe de fonctions dont les algorithmes associés ne sont pas tous programmables en raison de l'indécidabilité de leurs prédicats d'arrêt. (Davis [25] , Hermers [43] , Mal'cev [53] , Mendelson [56] , Rogers [63]). A partir de cet automate, diverses restrictions ont été imposées à son fonctionnement de façon à définir des classes de fonctions acceptables (Fischer [30] , Harrison et Schkolnick [8], Harrison et Havel [39]). La classe des machines la plus employée en analyse est certainement celle de l'automate à pile, acceptant les langages hors contexte (Mc Naughton et Papert [52] , Parickh [58] , Schutzenberger [68]). La définition des langages hors contexte impose des restrictions trop sévères. Divers extensions ou définitions (Aho [2] , Fischer [26] , Meyer [54] , Yoshi, Kosaraju et Yamada [79,80]) donnent accès à des classes de langages plus importantes. A l'intérieur de la classe des langages hors contextes, des algorithmes de reconnaissance comme l'algorithme de Coke ou l'algorithme d'Earley (Earley [26]) permettent la construction d'une structure syntaxique d'une phrase donnée. La donnée d'un tel algorithme est alors une grammaire hors contexte et une phrase à analyser.

Les systèmes décrits ici empruntent leurs définitions aux deux types de modèles, la grammaire décrite contenant un ensemble de fonctions algorithmiques.

Le deuxième problème est celui de la manipulation des arborescences (Berge [12] , Kuntzmann [48]). En considérant la structure syntaxique provenant de l'analyse d'un mot, divers systèmes de manipulations ont été étudiés (Aho et Ullman [1,3,4] , Lewis et Stearn [49] , Petrone [59] , Thatcher [69, 71, 72]). Une arborescence peut aussi représenter une structure sur les éléments d'un vocabulaire (Barbault et Descles [10, 11] , Pair et Quere [57] , Quere [60]) définissent ainsi un moyen d'analyse et de transduction. La définition de grammaires transformationnelles sans liaison étroite avec la structure d'un langage fait apparaître la notion de vocabulaire stratifié (Brainerd [13,14,15] , Levy et Joshi [53] , Rosen [62] , Rounds [63,64]). Les arborescences ainsi décrites sont ordonnées et le remplacement d'une sous-arborescence (règle de transformation) s'effectue avec des contraintes sévères (sous-arborescence feuille et complète).

Une étude des transformations d'arborescences non ordonnées est présentée par Gladky et Mel'cuk [34] . Cette étude ne considère que des arborescences non orientées et ne fournit pas d'algorithmes permettant une réalisation pratique d'un système. De plus, la transformation d'ensemble de sous-arborescences (Ginsburg et Partee [32], Veillon, Veyrune et Vauquois [74]) n'est pas abordée. La définition donnée ici s'inspire très largement des différentes définitions précédentes. Une généralisation de la définition des domaines d'arbres, le traitement des arborescences orientées, non orientées ou partiellement orientées est réalisable avec le même algorithme. Cette étude permet également de séparer le traitement de la structure (arborescence) du traitement de l'étiquette dans le cas où celle-ci est étiquetée. L'étude des différentes grammaires transformationnelles permet d'obtenir des grammaires d'analyse reconnaissant des ensembles récursifs d'arborescences. Construire un système à partir d'une telle définition nécessite l'emploi d'algorithmes permettant la réalisation de ces transformations. Les arborescences, étiquetées ou non, seront représentées par un langage de parenthèses (Knuth [45] , Mc Naughton [51]). Les transformations d'arborescences correspondront alors à des transformations de mots. Ces transformations de mots peuvent être prises en compte par des automates. Le support des algorithmes décrit ici sera donc établi par l'étude des transducteurs (Aho et Ullman [5] , Aho, Hopcroft et Ullman [4] , Ginsburg et Hopcroft [36] , Ibarra [47], Ginsburg et Rose [31], Thatcher [69, 70, 71]). Définir un ensemble de transducteurs réalisant un ensemble de fonctions ne peut conduire à des systèmes pratiques. L'écriture des tables de transitions serait fastidieuse et les tables générées trop importantes. Aussi les systèmes construits feront appel aux transducteurs universels (Arbib [9]). Les grammaires transformationnelles définies seront simulées par des transducteurs à pile composés et universels. La décidabilité de ces transducteurs est donnée par leurs constructions et le langage nécessaire à cette construction peut être identique à celui qui représente les arborescences. Le système C.E.T.A. permet la définition et l'application de ces transducteurs, réalisant ainsi une grammaire transformationnelle. De même que pour le système A.T.E.F., une certaine influence de l'algorithme est effectuée lors de l'écriture des grammaires. Cette influence permet une accélération de l'algorithme d'analyse ou de transfert sans avoir les contraintes présentées par les langages de programmation.

Les deux systèmes ont pour support externe deux langages hors contexte. Deux phases distinctes sont donc réalisées : la compilation des données définissent le transducteur à simuler et l'application du transducteur ainsi défini sur

un élément d'entrée. La compilation des données algorithmiques à partir du langage de définition s'effectue par des algorithmes classiques (Aho et Ullman [6,7] , Knuth [46,47]).

L'application de ces systèmes n'est pas limitée à l'analyse et la transduction des langues et des langages. Leurs définitions s'effectuent dans un cadre très général. Le vocabulaire choisi pour des éléments du système appartient pour une large part à celui employé en analyse des langues naturelles ne doit pas faire penser à cette seule utilisation. Le système A.T.E.F. définit un transducteur régulier quelconque et le système C.E.T.A. une grammaire transformationnelle. L'application de ce dernier système au calcul formel ou au traitement d'arborescences étiquetées nécessite aucune modification. Actuellement, ces systèmes sont implantés à l'Université de Grenoble et sont utilisés par le Groupe d'Etudes pour la Traduction Automatique pour l'étude des langues naturelles.

CHAPITRE I

TRANSDUCTEURS

INTRODUCTION

Le support théorique des traitements automatiques est donné par l'étude des automates, machines abstraites permettant d'exprimer un algorithme. Les transducteurs forment une classe particulière de ces automates. Un transducteur est un automate qui lit un ensemble de mots d'entrées et écrit un ensemble de mots de sorties. Les mots d'entrées sont définis sur un vocabulaire appelé vocabulaire d'entrée et les mots de sorties sur un vocabulaire appelé vocabulaire de sortie (qui peut éventuellement être le même que celui de l'entrée). Ainsi, un transducteur sera défini comme un automate comprenant plusieurs têtes de lecture, plusieurs têtes d'écriture, et une unité de contrôle caractérisée par un état. Les têtes de lecture ou d'écriture fonctionnent indépendamment l'une de l'autre et forment un élément essentiel des configurations du transducteur qui représentent l'état de ce transducteur à un instant donné. Une configuration est déterminée par l'ensemble des mots présents sur les bandes d'entrées, les symboles de ces mots repérés par les têtes de lecture, l'état interne de l'automate, et l'ensemble des mots présents sur les bandes de sorties. L'évolution des configurations est définie par la fonction de transition. Cette fonction de transition détermine, en fonction des éléments repérés par les têtes de lecture et l'état interne, les différentes opérations à effectuer :

- modification des symboles repérés par les têtes de lecture
- déplacement des têtes de lecture
- changement d'état interne
- écriture de mots de sorties

La configuration initiale d'un transducteur implique que les têtes de lecture se trouvent toutes en têtes des différents mots d'entrées, que l'état interne appartienne à un sous-ensemble des états (appelés ensemble des états initiaux), et que les mots de sorties soient tous égaux au mot vide. L'arrêt d'un transducteur est obtenu lorsque la fonction de transition n'est pas définie pour une configuration donnée. Pour représenter un élément de sortie cette configuration doit être telle que l'état interne de l'automate appartienne à un sous-ensemble des états possibles (appelé ensemble des états finaux). Le résultat du transducteur est alors donné par l'état interne et l'ensemble des mots présents sur les bandes de sorties. La notion de langage transcrit détermine le comportement du transducteur sur un ensemble de mots donnés.

Un problème intéressant de la théorie des transducteurs consiste à comparer leurs comportements ou leurs résultats. Ces comparaisons sont réalisées par l'existence de relation d'équivalence ou de préordre (Arbib [9]). La première relation dite relation de simulation indique la possibilité pour un transducteur d'effectuer (simuler) toutes les transitions possibles d'un autre transducteur. Cette relation est une relation de préordre. Les relations d'équivalences entre deux transducteurs expriment que leurs résultats sont équivalents pour un ensemble de mots donnés. En imposant des restrictions sur la façon d'obtenir ces résultats identiques on définit une équivalence forte. Enfin, la dernière relation est une relation de similitude qui exprime qu'un calcul effectué par un transducteur dans un environnement donné est réalisé par le second dans un autre environnement. Toutes ces relations sont dites simples lorsque l'entrée des transducteurs comprend un seul mot. Evidemment les définitions les plus générales de transducteurs conduisent à la machine de Turing qui possède, en particulier, un prédicat d'arrêt indécidable. Pour obtenir des automates plus simples, on impose au fonctionnement des transducteurs généraux différentes restrictions qui peuvent se diviser en deux groupes :

Les premières restrictions apportées concernent le fonctionnement du transducteur sur des bandes de sorties. Ces restrictions ne modifient pas les propriétés du prédicat d'arrêt du transducteur auquelles elles sont appliquées.

Les deuxièmes restrictions apportées concernent le fonctionnement du transducteur sur les bandes d'entrées. Ces restrictions modifient les propriétés du prédicat d'arrêt. En particulier, nous nous intéresserons toujours par la suite à deux types de transducteurs particuliers : les transducteurs réguliers (dérivant directement des automates d'états finis), et les transducteurs à piles (dérivant directement des automates à piles).

Ces deux restrictions conduisent à des transducteurs dont le prédicat d'arrêt est toujours vérifié. Les deux groupes de restrictions ne sont pas mutuellement exclusifs et des restrictions sur le fonctionnement des bandes de sorties pourront être apportées pour les transducteurs à pile ou régulier.

Ce chapitre consiste à rappeler les propriétés et les résultats obtenus sur ces transducteurs. D'autres définitions de "machine" ont été proposées pour atteindre différents objectifs tels que la reconnaissance de classes particulières de langage. Un tour d'horizon de ces différentes "machines" peut être trouvé dans KAIN [48].

NOTATIONS

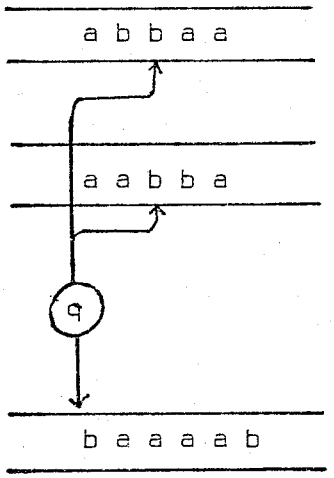
Soit V un ensemble fini non vide, $V = \{a_1, \ldots, a_n\}$

- L'opération de concaténation consiste à juxtaposer deux éléments.
- V^* désigne le monoïde libre engendré par V et l'opération de concaténation ".".
- ε désigne le mot vide
- Les éléments de V^* sont appelés des mots et un langage est un ensemble de mots : partie de V^*.
- Le miroir d'un mot W est un mot $\widetilde{W}$ défini de la façon suivante :
  - $\widetilde{\varepsilon} = \varepsilon$
  - $\widetilde{\sigma W} = \widetilde{W}.\sigma$

A - DEFINITION

Un transducteur est un automate possédant plusieurs bandes de lectures et d'écriture. Sur chaque bande de lecture, une tête de lecture peut se déplacer indépendamment du mouvement des autres. Sur chaque bande se sortie, un mot résultant est construit pas à pas. La construction s'effectue indépendamment sur chaque bande de sortie par concaténation d'un préfixe ou d'un suffixe. Le transducteur possède différents états. La lecture simultanée des bandes d'entrées permet de faire évoluer l'état du transducteur. Cette évolution s'effectue avec une modification des caractères des bandes d'entrée, d'un mouvement de chaque tête de lecture et de la construction du mot de chaque bande de sortie.

Exemple :



$((\sigma,\sigma), q) \rightarrow ((\sigma,\sigma), q', (G,N), (\sigma,G))$

$((\sigma,\sigma'),q') \rightarrow ((\sigma,\sigma'), q, (N,D), (\sigma',D)$

Lorsque le transducteur se trouve dans l'état q et que le même symbole se trouve devant les deux têtes de lecture, il écrit le symbole de la première bande en préfixe du mot de sortie, ne change pas les symboles d'entrées, déplace seulement la tête de lecture de la première bande vers la gauche et passe dans l'état q'. Dans cet état, le transducteur écrit en suffixe le symbole de la deuxième bande, déplace la tête de lecture de la deuxième bande vers la droite et revient dans l'état q.
Ainsi, avec ce transducteur, les deux mots W et $\widetilde{W}$ sont mis sous la forme W $\widetilde{W}$ en outre. La progression ne s'effectue que dans le cas où W et $\widetilde{W}$ sont sur les bandes d'entrée.

* Définition

Un transducteur est un automate qui analyse un mot d'entrée et construit un mot de sortie. Il est défini par :

$T = (V, V_E, V_S, Q, \delta, q_0, F, n, m)$ où :

- V est un ensemble fini, le vocabulaire général dont un des éléments est le blanc (noté b).
- V_E est un ensemble fini, le vocabulaire d'entrée : $V_E \subset V$
- V_S est un ensemble fini, le vocabulaire de sortie : $V_S \subset V$
- Q est l'ensemble des états. Si Q est infini, le transducteur est dit d'ordre infini
- Q_0 est l'ensemble des états initiaux : $Q_0 \subseteq Q$
- F est l'ensemble des états finaux : $F \subseteq Q$
- n est un entier de N^+, la puissance d'entrée
- m est un entier de N^+, la puissance de sortie
- δ est la fonction de transition :

$$V^n \times Q \rightarrow \mathcal{P}_f (V^n \times Q \times \{G,N,D\}^n \times (V_S^*)^m \times \{G,D\}^m)$$

si $\mathcal{P}_f (A)$ représente l'ensemble des parties finies de A.

* Configuration

Un transducteur est un automate qui lit n bandes d'entrées, change d'état, change les symboles d'entrées et écrit m bandes de sorties. Une configuration est un état transitoire d'un transducteur. La fonction δ induit une relation $\vdash_T$ sur l'ensemble des configurations de T.

Formellement une configuration est un élément de $(V^*)^{2n} \times Q \times (V_S^*)^m$
La relation $\vdash$ est définie par :

$$((W_{1_1}\sigma_{1_1}, \ldots, W_{1_n}\sigma_{1_n}, \sigma_{2_1}W_{2_1}, \ldots, \sigma_{2_n}W_{2_n}), q, (\Gamma_1, \ldots, \Gamma_n)) \vdash_T$$

$$((W'_{1_1}, \ldots, W'_{1_n}, W'_{2_1}, \ldots, W'_{2_n}), q', (\Gamma'_1, \ldots, \Gamma'_n)) \text{ si et seulement si}$$

$$((\sigma'_1, \ldots, \sigma'_n), q, (M_1, \ldots, M_n), (\gamma_1, \ldots, \gamma_m), (M'_1, \ldots, M'_m)) \in \delta((\sigma_{2_1}, \ldots, \sigma_{2_n}), q)$$

et $W'_{1_i} = W_{1_i}\sigma_{1_i}$ $\wedge$ $W'_{2_i} = \sigma'_i W_{2_i}$ si $M_i = N$

$$- W'_{1_i} = W_{1_i}\,\sigma_{1_i}\sigma'_i \quad \Lambda \quad W'_{2_i} = \begin{cases} W_{2_i} & \text{si } W_{2_i} \neq \varepsilon \\ & \text{si } M_i = D \\ \not b & \text{sinon} \end{cases}$$

$$- W'_{1_i} = \begin{cases} W_{1_i} & \text{si } W_{1_i} \neq \varepsilon \\ \not b & \text{sinon} \end{cases} \quad \Lambda \quad W'_{2_i} = \sigma_{1_i}\,\sigma'_i\,W_{2_i} \quad \text{si } M_i = G$$

$$- \Gamma'_i = \begin{cases} \Gamma_i\gamma_i & \text{si } M'_i = D \\ \gamma_i\Gamma_i & \text{si } M'_i = G \end{cases}$$

* Configuration d'arrêt

Une configuration A est une configuration d'arrêt s'il n'existe pas de configuration A' tel que $A \overset{}{\underset{T}{\vdash}} A'$

Une configuration d'arrêt est notée $[(W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q,(\Gamma_1,\ldots,\Gamma_m)]$

* Configuration initiale

Une configuration $((W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q,(\Gamma_1,\ldots,\Gamma_m))$ est une configuration initiale si et seulement si :

$$((W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q,(\Gamma_1,\ldots,\Gamma_m)) \in \{\not b\}^n \times (I_n(V_E^*))^n \times Q \times \{\varepsilon\}^m$$

Où $I_n(V_E^*) = V_E^+ \cup \{\not b\}$ définie comme l'extension naturelle aux parties de V_E^* de la fonction I_n' :

$$I_n'(\varepsilon) = \not b \quad , \quad I_n'(W) = W \quad \forall\, W \neq \varepsilon \quad , \quad W \in V_E^*$$

NOTATION :

$$|(W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q,(\Gamma_1,\ldots,\Gamma_n)|$$

Une configuration initiale est simple si et seulement si :

$$(W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q,(\Gamma_1,\ldots,\Gamma_n) \in \{\not b\}^n \times I_n(V_E^*) \times \{b\}^{n-1} \times Q \times \{\varepsilon\}^m$$

Caractéristiques d'un transducteur

Les caractéristiques du transducteur T sont définies de la façon suivante :

- vocabulaire général : $\rho_V(T) = V$
- vocabulaire d'entrée : $\rho_{V_E}(T) = V_E$

-vocabulaire de sortie : $\rho_{V_S}(T) = V_S$

-indice caractéristique de sortie $\rho_{I_S}(T) = 1$

- états initiaux : $\rho_{Q_0}(T) = Q_0$

- états finaux : $\rho_F(T) = F$

- puissance d'entrée : $\rho_E(T) = n$

de sortie : $\rho_S(T) = n$

*Mot transcrit par un transducteur T

Soit la relation $\overset{*}{\underset{T}{\vdash}}$, l'extension transitive de la relation $\underset{T}{\vdash}$

Un n-uple de mots $(\mu_1,\ldots,\mu_n)$ est transcrit en un ensemble de couples de $F \times (V_S^*)^n$ désigné par $T(q,(\mu_1,\ldots,\mu_n))$. $T(q,(\mu_1,\ldots,\mu_n))$ est défini par :

$\forall\, q \in \rho_{Q_0}(T)$, $\forall\, (\mu_1,\ldots,\mu_n) \in (V_E^*)^n$:

$$T(q,(\mu_1,\ldots,\mu_n)) = \{(q',W') \mid (\not b^n, I_n(\mu_1),\ldots,I_n(\mu_n)),q,\varepsilon^m \mid \overset{*}{\underset{T}{\vdash}} [W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}),q',W'] \;,\; W' \in (V_S^*)^m,\; q' \in F\}$$

*Mot transcrit simplement par un transducteur T

Un mot μ est transcrit simplement en un ensemble de couples de $F \times (V_S^*)$ désigné par $T(q,\mu)$. $T(q,\mu)$ est défini par :

$\forall\, q \in \rho_{Q_0}(T)$, $\forall\, \mu \in V_E^*$

$$T(q,\mu)=\{(q',W') \mid (\not b^n, I_n(\mu), \not b^{n-1}),q,\varepsilon^m \mid \overset{*}{\underset{T}{\vdash}} [W_{1_1},\ldots,W_{1_n},W_{2_1},\ldots,W_{2_n}), q',W'], q' \in F, W' \in (V_S^*)^m\}$$

*Langage transcrit

1 - Définitions

*On appelle langage produit de puissance n tout élément de $\mathcal{P}((V_E^*)^n)$

*π'_2 est l'application projection de $Q\times(V_S^*)^m$ dans $(V_S^*)^m$ définie par : $\pi'_2(q,W)=W \quad \forall\,(q,W) \in Q\times(V_S^*)^m$

*π_2 est l'extension naturelle aux parties de $Q\times(V_S^*)^m$ de la fonction π'_2

(extension naturelle : $\Pi_2(A) = \underset{X \in A}{\cup} \Pi'_2(X)$)

2 - Langage transcrit

Un langage produit E de puissance n est transcrit en un langage produit de puissance m désigné par T(E).

T(E) est défini par :

$$T(E)=\pi_2 \left(\bigcup_{q \in Q_0} \bigcup_{W \in E} T(q, W) \right)$$

3 - Langage transcrit simplement

Un langage E de V_E^* est transcrit simplement en un langage produit de puissance m désigné par T(E).

T(E) est défini par :

$$T(E)=\pi_2 \left(\bigcup_{q \in Q_0} \bigcup_{W \in E} T(q, W) \right)$$

Exemple

1. Soit $V=\{\not b, 1\}$, $V_E=\{1\}$, $V_S=\{1\}$. Un nombre entier X est représenté dans V_E^* ou V_S^* par un mot de longueur (X + 1). Le transducteur T_1 défini ci-après réalise la fonction $\lambda x[2x]$:

$$T_1=(V, V_E, V_S, \{q_0, q_1, q_2\}, \delta_1, \{q_0\}, \{q_2\}, 1, 1)$$

Où δ_1 est défini par :

$$\delta_1(1, q_0)=\{(1, q_1, D, \varepsilon, D)\}$$
$$\delta_1(1, q_1)=\{(1, q_1, D, 11, D)\}$$
$$\delta_1(\not b, q_1)=\{(\not b, q_2, N, 1, D)\}$$

Tout mot de V_E^+ est transcrit en un mot W' tel que $\ell(W')=2\ell(W)-1$

Le transducteur T_2 défini ci-après réalise la fonction $\lambda xy\,[x + y]$

$$T_2=(V, V_E, V_S, \{q_0, q_1\}, \delta_2, \{q_0\}, \{q_1\}, 2, 1)$$

δ_2 est défini par :

$\delta_2((1,1),q_0) = \{((1,1),q_1,D,1,D)\}$

$\delta_2((1,1),q_1) = \{((1,1),q_1,D,11,D)\}$

$\delta_2((1,\dot{b}),q_1) = \{((1,\dot{b}),q_1,D,1,D)\}$

$\delta_2((\dot{b},1),q_1) = \{((\dot{b},1),q_1,D,1,D)\}$

2. Soit $V = \{\dot{b},a,a',a'',c,c'\}$, $V_E = \{a,a'\}$, $V_S = \{c,c'\}$

Considérons le langage de Dyck $\mathcal{D}$ sur V_E^* défini par la grammaire suivante :

$G=(\{S\},\{a,a'\},S,\ \{\ S\to SS,\ S\to aa',\ S\to aSa'\})$

Alors le transducteur T_3 défini ci-après transcrit un mot W en un mot W' si et seulement si ce mot appartient à $\mathcal{D}$.

$T_3 = (V,V_E,V_S,\{q_0,q_1,q_2,q_3\},\delta_3,\{q_0\},\{q_3\},1,1)$

Où δ_3 est défini par :

$\delta_3(a,q_0) = \{(a,q_0,D,c,D)\}$

$\delta_3(a',q_0) = \{(a'',q_1,G,c',D)\}$

$\delta_3(a'',q_1) = \{(a'',q_1,G,\varepsilon,D)\}$

$\delta_3(a,q_1) = \{(a'',q_2,D,\varepsilon,D)$

$\delta_3(a'',q_2) = \{(a'',q_2,D,\varepsilon,D)\}$

$\delta_3(a,q_2) = \{(a,q_0,D,c,D)\}$

$\delta_3(a',q_2) = \{(a'',q_1,G,c',D)\}$

$\delta_3(\dot{b},q_2) = \{(\dot{b},q_3,G,\varepsilon,D)\}$

$\delta_3(a'',q_3) = \{(a'',q_3,G,\varepsilon,D)\}$

$\delta_3(a,q_3) = \{(a'',q_0,N,\varepsilon,D)\}$

Alors $T(q_0,W)=\emptyset \quad \forall\ W \notin \mathcal{D}$

$T(q_0,W)=\{(q_3,h(W))\}\ \forall\ W \in \mathcal{D}$ ou h est défini par :

$h'(a) = c$, $h'(a') = c'$

et $h(\varepsilon) = \varepsilon$, $h(\sigma W)=h'(\sigma)\ .\ h(W)$

B - RELATIONS ENTRE TRANSDUCTEURS

Soient deux transducteurs T_1 et T_2 définis par :

$$T_1=(V_1, V_{E_1}, V_{S_1}, Q_1, \delta_1, Q_{0_1}, F_1, n_1, m_1)$$

$$T_2=(V_2, V_{E_2}, V_{S_2}, Q_2, \delta_2, Q_{0_2}, F_2, n_2, m_2)$$

DEFINITION E_1 : SIMULATION

Le transducteur T_1 simule le transducteur T_2 si et seulement si :

- $\exists$ une application récursive h injective de $(\rho_{V_E}(T_2)^*)^{\rho_E(T_2)}$ dans $(\rho_{V_E}(T_1)^*)^{\rho_E(T_1)}$
- $\exists$ une application récursive h' surjective de $(\rho_{V_S}(T_1)^*)^{\rho_S(T_1)}$ dans $(\rho_{V_S}(T_2)^*)^{\rho_S(T_2)}$
- $\exists$ une application récursive f injective de $\rho_{Q_0}(T_2)$ dans $\rho_{Q_0}(T_1)$
- $\exists$ une application récursive f' surjective de $\rho_F(T_1)$ dans $\rho_F(T_2)$
- $\forall (q,W) \in \rho_{Q_0}(T_2) \times (\rho_{V_E}(T_2)^*)^{\rho_E(T_2)}$

$$(f'(q'),h'(W')) \in T_2(q,W) \Longleftrightarrow (q',W') \in T_1(f(q),h(W))$$

Le transducteur T_1 simule le transducteur T_2 si pour chaque calcul effectué par le transducteur T_2 il existe un calcul image correspondant effectué par le transducteur T_1

NOTATION :

$$T_2 \subset T_1$$

DEFINITION E_2 : EQUIVALENCE

Les deux transducteurs T_1 et T_2 sont dits équivalents si et seulement si :

- $\rho_E(T_1)=\rho_E(T_2)=n$, $\rho_S(T_1)=\rho_S(T_2)=m$
- $V_E=\rho_{V_E}(T_1) \cap \rho_{V_E}(T_2)$

- $\pi_2(\bigcup_{q_i \in \rho_{q_0}(T_1)} T_1(q_i,W)) = \pi_2(\bigcup_{q_j \in \rho_{q_0}(T_2)} T_2(q_j,W))$ $\forall W \in (V_E^*)^n$

- $\forall q \in \rho_{q_0}(T_1)$, $W \in (\rho_{V_E}(T_1)^*)^n - (V_E^*)^n$: $T_1(q,W)=\emptyset$

- $\forall q \in \rho_{q_0}(T_2)$, $W \in (\rho_{V_E}(T_2)^*)^n - (V_E^*)^n$: $T_2(q,W)=\emptyset$

NOTATION :

$T_1 \sim T_2$

Les deux transducteurs T_1 et T_2 sont équivalents s'ils fournissent le même langage à partir du même mot d'entrée.

DEFINITION E_3 : EQUIVALENCE FORTE

Les deux transducteurs T_1 et T_2 sont dits fortement équivalents si et seulement si :

- $\rho_E(T_1)=\rho_E(T_2)=n$, $\rho_S(T_1)=\rho_S(T_2)=m$

- $V_E=\rho_{V_E}(T_1) \cap \rho_{V_E}(T_2)$

- $\exists$ une application récursive biunivoque f de $\rho_{Q_0}(T_1)$ sur $\rho_{Q_0}(T_2)$

- $\exists$ une application récursive biunivoque h de $\rho_F(T_1)$ sur $\rho_F(T_2)$

- $\forall q \in \rho_{Q_0}(T_1)$, $W \in (V_E^*)^n$:

  $(q',W') \in T_1(q,W) \Longleftrightarrow (h(q'),W') \in T_2(f(q),W)$

- $\forall q \in \rho_{Q_0}(T_1)$, $W \in (\rho_{V_E}(T_1)^*)^n - (V_E^*)^n$: $T_1(q,W)=\emptyset$

- $\forall q \in \rho_{Q_0}(T_2)$, $W \in (\rho_{V_E}(T_2)^*)^n - (V_E^*)^n$: $T_2(q,W)=\emptyset$

NOTATION :

$T_1 \approx T_2$

DEFINITION E_4 : SIMILITUDE

Les deux transducteurs T_1 et T_2 sont dits similaires si et seulement si :

- $\exists$ une application récursive biunivoque h de de $(\rho_{V_E}(T_2)^*)^{\rho_E(T_2)}$ sur $(\rho_{V_E}(T_1)^*)^{\rho_E(T_1)}$

- $\exists$ une application récursive biunivoque f de $\rho_{Q_0}(T_2)$ sur $\rho_{Q_0}(T_1)$

- $\exists$ une application récursive biunivoque f' de $\rho_F(T_2)$ sur $\rho_F(T_1)$

- $\exists$ une application récursive biunivoque h' de $(\rho_{V_S}(T_1)^*)^{\rho_S(T_1)}$ sur $(\rho_{V_S}(T_2)^*)^{\rho_S(T_2)}$

- $\forall (q,W) \in \rho_{Q_0}(T_2) \times (\rho_{V_E}(T_2)^*)^{\rho_E(T_2)}$

$$(q',W') \in T_2(q,W) \iff (f'(q'),h'(W')) \in T_1(f(q),h(W))$$

NOTATION :

$$T_1 \# T_2$$

PROPRIETES

1. Les relations d'équivalence, d'équivalence forte et de similitude sont des relations d'équivalence.

2. La relation de simulation est une relation de préordre, et elle induit une relation de préordre sur l'ensemble des classes d'équivalences de la relation de similitude.

La réflexivité de la relation de simulation est déduite directement de la définition et si deux transducteurs sont similaires, l'un simule l'autre.

La transitivité de la relation de simulation est déduite de la propriété de la composition des applications injectives.

3. La relation d'équivalence forte raffine la relation d'équivalence

$T_1 \approx T_2 \Longrightarrow T_1 \sim T_2$

C - RELATIONS SIMPLES

Soient deux transducteurs T_1 et T_2 définis par :

$T_1 = (V_1, V_{E_1}, V_{S_1}, Q_1, \delta_1, Q_{0_1}, F_1, n_1, m_1)$

$T_2 = (V_2, V_{E_2}, V_{S_2}, Q_2, \delta_2, Q_{0_2}, F_2, n_2, m_2)$

DEFINITION S1 : SIMULATION SIMPLE

Le transducteur T_1 simule simplement le transducteur T_2 si et seulement si :

- $\exists$ une application récursive h, injective de $\rho_{V_E}(T_2)^*$ dans $\rho V_E(T_1)^*$
- $\exists$ une application récursive h', surjective de $(\rho_{V_S}(T_1)^*)^{\rho_S(T_1)}$ dans $(\rho_{V_S}(T_2)^*)^{\rho_S(T_2)}$
- $\exists$ une application récursive f, injective de $\rho_{Q_0}(T_2)$ dans $\rho_{Q_0}(T_1)$
- $\exists$ une application récursive f', surjective de $\rho_F(T_1)$ dans $\rho_F(T_2)$
- $\forall\, q \in \rho_{Q_0}(T_2)$, $W \in \rho_{V_E}(T_2)^*$:

$$(f'(q'),h'(W')) \in T_2(q,W) \Longleftrightarrow (q',W') \in T_1(f(q),h(W))$$

NOTATION :

$$T_2 \underset{S}{\subset} T_1$$

DEFINITION S2 : EQUIVALENCE SIMPLE

Les deux transducteurs T_1 et T_2 sont dits simplement équivalents, si et seulement si :

- $\rho_S(T_1) = \rho_S(T_2) = m$
- $V_E = \rho_{V_E}(T_1) \cap \rho_{V_E}(T_2)$
- $\Pi_2\left(\bigcup_{q_i \in \rho_{Q_0}(T_1)} T_1(q_i,W)\right) = \Pi_2\left(\bigcup_{q_j \in \rho_{Q_0}(T_2)} T_2(q_j,W)\right), \forall\, W \in V_E^*$
- $\forall\, q \in \rho_{Q_0}(T_1)$, $W \in \rho_{V_E}(T_1)^* - V_E^*$: $T_1(q,W) = \emptyset$
- $\forall\, q \in \rho_{Q_0}(T_2)$, $W \in \rho_{V_E}(T_2)^* - V_E^*$: $T_2(q,W) = \emptyset$

NOTATION :

$$T_1 \tilde{S} T_2$$

DEFINITION S3 : EQUIVALENCE SIMPLE ET FORTE

Les deux transducteurs T_1 et T_2 sont dits simplement et fortement équivalents, si et seulement si :

- $\rho_S(T_1) = \rho_S(T_2) = m$
- $V_E = \rho_{V_E}(T_1) \cap \rho_{V_E}(T_2)$
- $\exists f$, application récursive biunivoque de $\rho_{Q_0}(T_1)$ sur $\rho_{Q_0}(T_2)$
- $\exists h$, application récursive biunivoque de $\rho_F(T_1)$ sur $\rho_F(T_2)$
- $\forall q \in \rho_{Q_0}(T_1)$, $W \in V_E^*$:

$$(q',W') \in T_1(q,W) \Longleftrightarrow (h(q'),W') \in T_2(f(q),W)$$

- $\forall q \in \rho_{Q_0}(T_1)$, $W \in \rho_{V_E}(T_1)^* - V_E^*$: $T_1(q,W) = \emptyset$
- $\forall q \in \rho_{Q_0}(T_2)$, $W \in \rho_{V_E}(T_2)^* - V_E^*$: $T_2(q,W) = \emptyset$

NOTATION :

$$T_1 \tilde{\tilde{S}} T_2$$

DEFINITION S4 : SIMILITUDE SIMPLE

Les deux transducteurs T_1 et T_2 sont dits simplement similaires, si et seulement si :

- $\exists$ une application récursive h, biunivoque de $\rho_{V_E}(T_2)^*$ sur $\rho_{V_E}(T_1)^*$
- $\exists$ une application récursive h', biunivoque de $(\rho_{V_S}(T_2)^*)^{\rho_S(T_2^E)}$ sur $(\rho_{V_E}(T_1)^*)^{\rho_S(T_1)}$
- $\exists$ une application récursive f, biunivoque de $\rho_{Q_0}(T_2)$ sur $\rho_{Q_0}(T_1)$
- $\exists$ une application récursive f', biunivoque de $\rho_F(T_2)$ sur $\rho_F(T_1)$

- $\forall q \in \rho_{Q_0}(T_2)$, $W \in \rho_{V_E}(T_2)^*$:

$$(q',W') \in T_2(q,W) \iff (f'(q'),h'(W') \in T_1(f(q),h(W))$$

NOTATION :

$$T_1 \underset{S}{\#} T_2$$

PROPRIETES

1 - Les relations d'équivalence simple, d'équivalence simple et forte et de similitude sont des relations d'équivalences.

2 - La relation de simulation simple est une relation de pré-ordre et elle induit une relation de pré-ordre sur l'ensemble des classes d'équivalence de la relation de similitude simple.

3 - La relation d'équivalence simple et forte raffine la relation d'équivalence simple.

4 - Toute relation de simulation, d'équivalence, d'équivalence forte ou de similitude entre deux transducteurs implique la relation simple correspondante entre ces deux transducteurs.

D - P R O P R I E T E

THEOREME T1

Pour tout transducteur T de puissance d'entrée supérieure à un, il existe un transducteur T' de puissance d'entrée égale à un, simplement et fortement équivalent.

E - T R A N S D U C T E U R S P A R T I C U L I E R S

Un transducteur général est trop puissant pour être le support de système programmable. Les restrictions abordées ici permettent d'atteindre des transducteurs pouvant être simulés par programme sur ordinateur. Les trois premières restrictions n'affectent pas la puissance des transducteurs, alors que les deux dernières constituerons la base des transducteurs utilisés par la suite.

1 - TRANSDUCTEUR DIRECT

* Définition

Un transducteur T est direct si son mouvement de sortie est constant. Un transducteur est direct si et seulement si :

$$\delta : Q \times V^n \rightarrow \mathcal{P}(V^n \times Q \times \{G,N,D\}^n \times (V_S^*)^n \times M) \quad \text{et} \quad M \in \{G,D\}^m$$

De plus, il est direct à gauche si $M \in \{G\}^m$,

direct à droite si $M \in \{D\}^m$.

* Théorème T2

Pour tout transducteur T, il existe un transducteur direct T simplement et fortement équivalent.

2 - TRANSDUCTEURS DETERMINISTES

α) Transducteur faiblement déterministe

Un transducteur T est dit faiblement déterministe si et seulement si :

$$[\mathrm{Card}(\delta(\sigma,q)) \leqslant 1 \quad \forall\, q \in Q,\ \sigma \in V^n]$$

β) Transducteur déterministe

Un transducteur T est dit déterministe si et seulement si :

$$[\mathrm{Card}(\delta(\sigma,q)) \leqslant 1 \quad \forall\, q \in Q,\ \sigma \in V^n]$$

$$\mathrm{Card}(Q_0) = 1$$

REMARQUE :

On ne peut pas parler de l'équivalence d'un transducteur déterministe et d'un transducteur non déterministe, car les éléments transcrits ne sont pas comparables. Dans le premier cas, l'ensemble transcrit à partir d'un mot et d'un état est toujours de cardinal inférieur ou égal à 1. Dans le second cas, l'ensemble transcrit à partir d'un mot et d'un état est de cardinal quelconque (fini ou dénombrable).

3 - TRANSDUCTEUR ELEMENTAIRE

* Définition

Un transducteur T est élémentaire si il écrit à chaque pas un mot de longueur au plus égale à 1 sur chaque bande de sortie.
T est élémentaire si et seulement si :

$$T = (V, V_E, V_S, Q, \delta, Q_0, F, n, m)$$

$$\delta : V^n \times Q \rightarrow \mathcal{P}\left(V^n \times Q \times \{G,N,D\}^n \times (V_S \cup \{\varepsilon\})^m \times \{G,D\}^m\right)$$

* Théorème T3

Pour tout transducteur T, il existe un transducteur T' élémentaire fortement équivalent.

4 - TRANSDUCTEURS REGULIERS

Un transducteur régulier est un transducteur beaucoup moins puissant, mais dont l'intérêt pratique est évident. Son fonctionnement nécessite une lecture seulement de la bande d'entrée.

Un transducteur T est régulier si et seulement si :

- $T = (V, V_E, V_S, Q, \delta, Q_0, F, n, m)$
- $\delta : V^n \times Q \rightarrow \mathcal{P}(V^n \times Q \times \{N,D\}^n \times (V_S^*)^m \times \{G,D\}^m)$

Propriété

1 - Pour tout transducteur régulier T, il existe un transducteur régulier T' de puissance d'entrée égale à un, simplement et fortement équivalent.

2 - Pour tout transducteur régulier T, il existe un transducteur régulier élémentaire T' fortement équivalent.

DEMONSTRATION

1 - La démonstration est immédiate, dans ce cas, les autres bandes d'entrée ne servent à rien et peuvent être ignorées.

2 - La démonstration du cas général (Théorème T3) conserve le mouvement des têtes de lectures.

5 - TRANSDUCTEURS A PILES

Un transducteur à pile est une restriction du transducteur général donnant un système plus puissant que le transducteur régulier.

Un transducteur à pile est défini par :

$T = (V, V_E, V_S, V_p, Q, \delta, Q_0, F, n, m)$

Où seul V_p et δ diffèrent de la définition générale :

- $V_p \subset V$ est le vocabulaire de pile

- δ : fonction de transition :

$$V^n \times V_p \cup \{\varepsilon\} \times Q \rightarrow \mathcal{P}(V^n \times V_p^* \times Q \times \{N,D\}^n \times (V_S^*)^m \times \{G,D\}^m)$$

Une configuration d'un transducteur à pile est un élément de $(V^*)^n \times (V^*)^n \times V_p^* \times Q \times V_S^*$. La fonction de transition δ induit une relation $\vdash_T$ sur l'ensemble des configurations de la même manière que précédemment :

$$((W_{1_1}\sigma_{1_1},\ldots,W_{1_n}\sigma_{1_n},\sigma_{2_1}W_{2_1},\ldots,\sigma_{2_n}W_{2_n}),W_p\sigma_p,q,\Gamma_1,\ldots,\Gamma_m) \vdash_T$$

$$((W'_{1_1},\ldots,W'_{1_n},W'_{2_1},\ldots,W'_{2_n}),W'_p,q',\Gamma'_1,\ldots,\Gamma'_m)$$

si et seulement si :

$$((\sigma'_1,\ldots,\sigma'_n),W'',q',(M_1,\ldots,M_n),\gamma_1,\ldots,\gamma_m,(M'_1,\ldots,M'_m))$$

$$\in \delta((\sigma_{2_1},\ldots,\sigma_{2_n}),\sigma_p,q)$$

et

$$\begin{cases} \sigma_p = \varepsilon \quad \text{si } W_p\sigma_p = \varepsilon \\ \sigma_p \in V \quad \text{sinon} \end{cases}$$

$$W'_p = W_p W''$$

$$\forall_j : \begin{cases} W'_{1_j} = W'_{1_j}\sigma_{1_j}\sigma'_j \quad \text{et} \quad W'_{2_j} = \begin{cases} W_{2_j} \text{ si } W_{2_j} \neq \varepsilon \\ \text{Đ sinon} \end{cases} \quad \text{si } M_j = D \\ W'_{1_j} = W_{1_j}\sigma_{1_j} \quad \text{et} \quad W'_{2_j} = \sigma'_j W_{2_j} \quad \text{si } M_j = N \end{cases}$$

$$\forall_j : \begin{cases} \Gamma'_j = \Gamma_j\gamma_j \quad \text{si } M'_j = D \\ \Gamma'_j = \gamma_j\Gamma_j \quad \text{si } M'_j = G \end{cases}$$

De la même manière, une configuration A est une configuration d'arrêt si et seulement si il n'existe pas de configuration A' tel que $A \vdash_T A'$. (Notation [A]).

Une configuration initiale est un élément de :

$$\{b\}^n \times (V_E^*)^n \times \{\varepsilon\} \times Q_0 \times \{\varepsilon\}^m \qquad \text{(Notation } |A|\text{)}.$$

Soit $\overset{*}{\underset{T}{\vdash\!\!-\!\!-}}$ l'extension transitive de la relation $\overset{}{\underset{T}{\vdash\!\!-\!\!-}}$.

Un élément $W \in (V_E^*)^n$ est transcrit en un ensemble $T(q,W)$ de couples de $Fx(V_S^*)$ à partir de l'état q, si $T(q,W)$ est défini par :

$$T(q,W) = \{(q',W') \mid\mid b^n,W,\varepsilon,q,\varepsilon^m \overset{*}{\underset{T}{\vdash\!\!-\!\!-}} [W'',W_p,q',W'],q' \in F\}$$

Cette définition permet de considérer un transducteur à pile comme un transducteur classique et de pouvoir obtenir, par exemple, l'équivalence d'un transducteur à pile et d'un transducteur.

Théorème :

Pour tout transducteur à pile, il existe un transducteur fortement équivalent.

Cette propriété est bien connue. Il suffit de construire un transducteur dont la première bande d'entrée comportera quatre pistes. Les deux premières pistes serviront à placer les éléments relatifs à la première bande d'entrée du transducteur simulé et les deux dernières pistes seront utilisées comme une pile.

Propriété :

1 - Pour tout transducteur à pile T, il existe un transducteur à pile élémentaire fortement équivalent.

2 - Pour tout transducteur à pile T, de puissance d'entrée n, il existe un transducteur produit à pile T' de puissance d'entrée égale à 1, simplement et fortement équivalent.

Le fait d'ajouter une pile ne change rien à la démonstration générale, il faut seulement que la pile reste inchangée quand le transducteur équivalent T' ne simule pas un pas de T.

CHAPITRE II

TRANSDUCTEURS COMPOSES

INTRODUCTION

Dans le cas général, un transducteur calcule une fonction partielle. Ce genre de transducteur n'a pas d'intérêt pratique car son prédicat d'arrêt est indécidable. Les transducteurs restreints ne permettent que le calcul de fonctions élémentaires et, pour la réalisation pratique d'un système, le transducteur programmé ne doit pas nécessiter l'évaluation de son prédicat d'arrêt. Ces différentes conditions, apparemment contradictoires, conduisent à la notion de composition. Trois types de composition seront abordés dans ce chapitre.

La première composition étudiée est dite composition substitutive et se présente comme une extension de la composition de machine de Turing étudiée dans Hermès [45]. Un transducteur composé substitutivement peut être assimilé à un réseau de transducteurs. Chaque transducteur forme un point de ce réseau. Le passage d'un point du réseau à un autre correspond au passage de l'information résultante du calcul du premier transducteur (point de départ) sur l'entrée du second transducteur (point d'arrivée). Remarquons les deux propriétés de cette composition :

a) La composition des transducteurs généraux n'augmente pas leur puissance.

b) La composition de transducteurs restreints est équivalente à un transducteur général dès que le réseau comporte un cycle.

En conséquence, nous définissons deux types particuliers de compositions : la composition linéaire et la composition ultra-linéaire. Ces deux compositions forment des réseaux sans cycle et la composition ultra-linéaire est équivalente à un réseau arborescent. Ces deux types de composition conservent les propriétés de décidabilité des transducteurs élémentaires. En particulier, dans le cas où tous les transducteurs élémentaires ont un prédicat d'arrêt toujours vérifié, le transducteur composé résultant aura un prédicat d'arrêt toujours vérifié.

On étudie la simulation des fonctions récursives primitives par ce type de transducteur afin d'avoir une idée précise de leurs possibilités. La composition linéaire ou ultra-linéaire est trop faible pour pouvoir simuler un schéma de récursion primitive. Néanmoins, on obtient la classe de fonction fermée par substitution régulière et contenant les fonctions de base suivante : nulle, successeur, choix d'argument, somme et différence propre.

Un deuxième type de composition est la composition récursive. Le calcul d'un transducteur composé récursivement s'effectue suivant un schéma similaire à un schéma de récursion. Le résultat du calcul pour mot de la forme σW, où σ est un symbole du vocabulaire et W un mot différent du mot vide, est défini par l'application de la formule : $T(\sigma W) = T(\sigma T(W))$.

Cette composition récursive préserve la décidabilité du prédicat d'arrêt des transducteurs élémentaires. Ainsi, ces deux compositions conduisent à des transducteurs qui calculent des fonctions totales dans le cas où chaque transducteur élémentaire calcule lui-même une fonction totale. Les transducteurs à pile ou régulier, composés linéairement ou récursivement, peuvent simuler des fonctions récursives primitives.

Dans les chapitres suivants, les systèmes construits seront définis à partir de transducteurs à pile ou régulier, composés linéairement et récursivement.

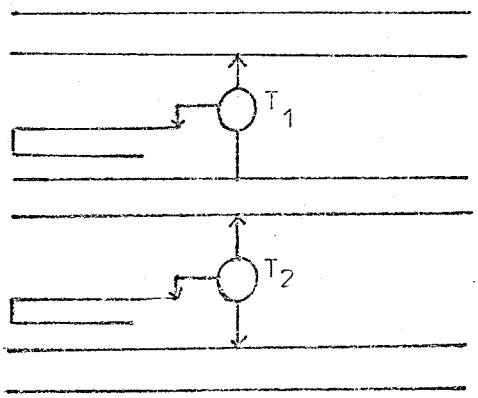
L'étude de transducteur comme élément de définition d'un système programmable implique la définition de transducteurs universels pour une classe de fonctions donnée. Ainsi, par exemple, la définition d'analyseur syntaxique se comporte comme un transducteur universel pour un ensemble défini de langages. La donnée de ces systèmes est alors un couple composé d'une grammaire syntagmatique définissant un accepteur particulier et le mot du langage à analyser. Ici, l'action est décomposée en deux temps, car la donnée définissant l'accepteur ou le transducteur particulier ne permet son fonctionnement immédiat. Un premier transducteur appelé transducteur constructeur permet de définir à partir d'un mot donné l'automate particulier que l'on veut appliquer. Cet automate ou transducteur ainsi construit sera alors appliqué sur le second mot d'entrée. L'essemble des transducteurs formés par un transducteur constructeur et les transducteurs construits forment un transducteur composé universellement. Le langage nécessaire au transducteur constructeur définit une famille de transducteurs constructibles.

A - COMPOSITION SUBSTITUTIVE

Afin d'obtenir des systèmes décidables de transducteurs, les transducteurs élémentaires seront composés. Dans le cas de la composition des transducteurs généraux, la puissance de l'automate obtenu est inchangée. Mais, dans le cas de transducteurs moint puissants, la composition permet d'obtenir des systèmes complexes et décidables. La décidabilité d'un tel transducteur est donnée par sa construction. Cette propriété est importante car un système programmé simulant de tels transducteurs n'a pas à évaluer un prédicat d'arrêt. La composition substitutive est une généralisation, pour les transducteurs, de l'opération de substitution effectuée pour les fonctions. Le résultat d'un transducteur est fourni comme entrée à un second transducteur et ainsi de suite. Certains transducteurs peuvent également ne pas transmettre leurs résultats. Ces transducteurs fournissent alors le résultat final des calculs. Dans le cas particulier de la composition cyclique, le transducteur obtenu est trop puissant pour avoir un intérêt pratique puisqu'il est équivalent à un transducteur général.
Tous les systèmes étudiés par la suite auront donc comme base des transducteurs composés linéairement, c'est-à-dire, non cycliques. La composition ultra-linéaire est l'image pour les transducteurs de l'opération de substitution effectuée pour les fonctions. Les transducteurs composés se placent dans le cadre général des transducteurs, seuls leurs modes de fonctionnement diffèrent.

Exemple :

Soit le langage $\{a^n b^n c^n \mid n \geqslant 1\}$. La construction du transducteur à pile composé reconnaissant ce langage est donnée par le schéma suivant :



Le premier transducteur à pile analyse $a^n b^n c^*$ et transcrit le langage $b^* c^*$. Le second transducteur analyse alors le langage $b^n c^n$.

1 - DEFINITION

α) Un transducteur est un transducteur composé d'ordre un ($\rho_0(T) = 1$).

β) Soit $T_1,\ldots,T_n$, n transducteurs composés d'ordre $n_1,\ldots,n_n$.

Ils sont composables si et seulement si :

- $\forall_i,j : 1\leq i, j\leq n : i\neq j \Longrightarrow (\rho_{V_S}(T_i)\subset\rho_{V_E}(T_j) \wedge \rho_S(T_i) = \rho_E(T_j)) \vee$

$$(\rho_F(T_i) \cap \rho_{Q_0}(T_j) = \emptyset)$$

- $\exists k : \forall_i : (\rho_F(T_i) - \bigcup_{j\neq i}\rho_{Q_0}(T_j) = \emptyset) \vee (\rho_S(T_i) = k)$

- $\exists_i : (\rho_F(T_i) - \bigcup_{j\neq i}\rho_{Q_0}(T_j)) \neq \emptyset$

Une suite ordonnée de transducteurs composables est un transducteur composé $T = C_{T_1,\ldots,T_n}$.

Une suite de transducteurs composables conduit à un transducteur composé où chaque transducteur effectue un calcul sur le résultat d'un autre. Les conditions de compositions expriment les restrictions suivantes :

- Un transducteur ne peut fournir un mot d'entrée à un second transducteur que si le vocabulaire de sortie du premier est compatible avec le vocabulaire d'entrée du second. Cette compatibilité doit aussi être étendue entre les puissances de sortie et d'entrée des deux transducteurs.

- Un transducteur ne peut fournir le résultat du transducteur composé que si sa puissance de sortie est égale à k.

- Au moins un transducteur peut fournir un résultat.

Les caractéristiques d'un transducteur composé $C_{T_1,\ldots,T_n}$ sont définies par :

- vocabulaire général : $\rho_V(T) = \bigcup_{i=1}^{n} \rho_V(T_i)$

- vocabulaire d'entrée : $\rho_{V_E}(T) = \rho_{V_E}(T_1)$

- indices caractéristiques de sortie :

$$\rho_{I_S}(T) = \{i \mid \rho_F(T_i) - \bigcup_{j\neq i} \rho_{Q_0}(T_j) \neq \emptyset\}$$

- vocabulaire de sortie : $\rho_{V_S}(T) = \bigcup_{j\in\rho_{I_S}(T)} \rho_{V_S}(T_j)$

- états unitiaux : $\rho_{Q_0}(T) = \rho_{Q_0}(T_1)$

- états finaux : $\rho_F(T) = \bigcap_{i=1}^{n} (\rho_F(T_i) - \bigcup_{j\neq i} \rho_{Q_0}(T_j))$

- états généraux : $\rho_Q(T) = \bigcap_{i=1}^{n} \rho_Q(T_i)$

- puissance d'entrée : $\rho_E(T) = \rho_E(T_1)$

- puissance de sortie : $\rho_S(T) = \{\rho_S(T_i) \mid \rho_F(T_i) - \bigcup_{j\neq i} \rho_{Q_0}(T_j) \neq \emptyset\}$

- ordre : $\rho_0(T) = \sum_{i=1}^{n} \rho_0(T_i)$

Mot transcrit par un transducteur composé T :

Un mot W de $(\rho_{V_E}(T)^*)^{\rho_E(T)}$ est transcrit en un ensemble de couples $T(q,W)$ (ou $C_{T_1,\ldots,T_n}(q,W)$) par le transducteur T à partir de l'état q, élément de $\rho_{Q_0}(T)$, si $T(q,W)$ est défini de la façon suivante :

$(q',W') \in C_{T_1,\ldots,T_n}(q,W)$ si et seulement si il existe une suite finie $(i_1,(q_1,W_1)),\ldots,(i_k,(q_k,W_k))$, tel que :

- $W_1 = W$, $q_1 = q$, $i_1 = 1$, $W_k = W'$, $q_k = q'$

- $i_j \leqslant n \ \forall_j$

- $\forall_{j<k}$: $[(q_{j+1},W_{j+1}) \in T_{i_j}(q_j,W_j)] \wedge [q_{j+1} \in \rho_{Q_0}(T_{i_{j+1}})] \wedge [i_j \neq i_{j+1}]$

- $q_k \notin \rho_{Q_0}(T_h) \ \forall\ h\leqslant n,\ h\neq i_{k-1}$

Cette définition implique les propriétés suivantes :

- Un transducteur composé se comporte du point de vue externe comme un transducteur.

- Les relations de simulation, d'équivalence, d'équivalence forte et de similitude ont été définies pour l'ensemble des transducteurs composés et conservent toutes leurs propriétés.

2 - TRANSDUCTEURS COMPOSES RESTREINTS

a) Transducteurs composés déterministes

α) Un transducteur composé est dit faiblement déterministe si et seulement si chaque transducteur qui le compose est faiblement déterministe.

β) Un transducteur composé $C_{T_1,\ldots,T_n}$ est dit déterministe si et seulement si :

- chaque transducteur qui le compose est faiblement déterministe.
- le transducteur T_1 est déterministe.

b) Transducteurs composés réguliers

Un transducteur composé est dit "régulier" si et seulement si chaque transducteur qui le compose est régulier.

c) Transducteur composé à pile

Un transducteur composé est dit "à pile" si et seulement si :

- chaque transducteur le composant est soit un transducteur régulier, soit un transducteur à pile.

- au moins un des transducteurs est un transducteur à pile.

γ) Transducteur composé cyclique

Un transducteur composé $C_{T_1,\ldots,T_n}$ est dit cyclique si et seulement si :

- Soit un des transducteurs T_i le composant est un transducteur composé cyclique.

- Soit il possède la propriété :

$$\exists_{i,j} : i>j,\ 1\leqslant i,\ j\leqslant n : F_i \cap Q_{0_j} \neq \emptyset$$

δ) Transducteur linéaire

Un transducteur composé non cyclique est dit linéaire.

Un transducteur composé $C_{T_1,\ldots,T_n}$ est ultra-linéaire si et seulement si il est linéaire et possède la propriété :

- Chaque transducteur le composant est soit un transducteur composé d'ordre un, soit un transducteur ultra linéaire.

$$- \forall_{i,j}\ \rho_F(T_i) \cap \rho_{Q_0}(T_j) \neq \emptyset \Longrightarrow \forall_h \neq i,j : \rho_F(T_h) \cap \rho_{Q_0}(T_j) = \emptyset$$

La composition ultra linéaire est la composition arborescente.

3 - PROPRIETES

* Réduction simple

Pour tout transducteur composé $C_{T_1,\ldots,T_n}$ tels que les derniers transducteurs ne peuvent être "atteints", il existe un transducteur composé d'ordre inférieur fortement équivalent :

$$\forall\ C_{T_1,\ldots,T_n} : \exists k<n : \forall\ l,h,\ l\leqslant k \wedge h>k :$$

$$\rho_F(T_l) \cap \rho_{Q_0}(T_h) = \emptyset \Longrightarrow C_{T_1,\ldots,T_k} \approx C_{T_1,\ldots,T_n}$$

Théorème II 1

Pour tout transducteur composé d'ordre n, il existe un transducteur composé d'ordre un, fortement équivalent.

4 - PROPRIETES DES TRANSDUCTEURS REGULIERS

Les transducteurs réguliers peuvent conduire à des systèmes pratiques. Ils peuvent être effectivement programmés car leurs prédicats d'arrêt est décidable. Le système A.T.E.F. est une application directe de ces transducteurs.

a) Conservation des langages

Un transducteur régulier direct à droite est appelé "generalized sequential machine". Un transducteur régulier direct à gauche est équivalent à une opération réalisée par une "generalized sequential machine" suivie d'une opération miroir. D'où la propriété.

Théorème II 2

Toute classe de langage fermée pour la substitution, l'opération miroir et l'intersection avec un langage régulier est fermée pour la transduction directe.

Toute classe de langage fermée pour la substitution et l'intersection avec un langage régulier est fermée pour la transduction directe à droite.

Corollaire T 2

La classe des langages réguliers et des langages hors contexte est fermée pour la transduction directe.

Théorème II 3

La classe des langages réguliers et des langages hors-contexte n'est pas fermée pour la transduction.

DEMONSTRATION

Soit le transducteur régulier T défini par :

$$T = (V, V_E, V_S, Q, \delta, Q_0, F, 1, 1)$$

Où : $V = V_E = V_S = \{a, b, c\}$

$$Q = Q_0 = F = \{q_0\}$$

$$\delta : \delta(q_C, a) = (a, q_O, D, a, G)$$
$$\delta(q_O, b) = (b, q_O, D, b, D)$$
$$\delta(q_O, c) = (c, q_O, D, c, D)$$

Alors : $T((ba)^+) = \{a^n b^n \mid n>0\}$

$$T(\{(ba)^n c^n \mid n>0\}) = \{a^n b^n c^n \mid n>0\}$$

b) Transducteurs composés réguliers cycliques

Lorsqu'un transducteur composé est cyclique, il devient trop puissant pour avoir un intérêt pratique. Un transducteur composé régulier cyclique est équivalent à un transducteur général.

Théorème II 4

Pour tout transducteur T de puissance égale à un, il existe un transducteur composé régulier, cyclique, direct à droite et d'ordre deux le simulant.

DEMONSTRATION

On suppose T élémentaire. Si T n'est pas élémentaire, il existe T' fortement équivalent et élémentaire.

Soit $T = (V, V_E, V_S, Q, \delta, Q_O, F, 1, 1)$

Alors, le transducteur régulier est défini par $C_{T_1 T_2}$ *où*

$$T_1 = (V_1, V_{E_1}, V_{S_1}, Q_1, \delta_1, Q_{O_1}, F_1, 1, 1)$$

$$T_2 = (V_1, V_{E_1}, V_{S_1}, Q_2, \delta_2, Q_{O_2}, F_2, 1, 1)$$

$$V_{E_1} = \{(\sigma_1, \sigma_2) \mid \sigma_1 \in V \cup \{\psi, *\},\ \sigma_2 \in V_S \cup \{*, \psi\}\}$$

$V_1 = V_{E_1} \cup \{\hat{b}\}$ *On suppose que* $\psi, * \in V$

$$Q_1 = Q \cup \{\rho_q \mid q \in Q\} \cup \{f_q \mid q \in Q\} \cup \{(q, M) \mid q \in Q, M \in \{G, N, D\}\}$$

$$\cup \{(q, M, \sigma, M') \mid q \in Q, M \in \{G, N, D\}, \sigma \in V_S, M' \in \{G, D\}\}$$

$Q_{O_1} = Q$

$$F_1 = \{f_q \mid q \in Q\} \cup \{(q, M) \mid q \in Q, M \in \{G, N, D\}\} \cup \{(q, M, \sigma, G) \mid$$
$$q \in Q, M \in \{G, N, D\}, \sigma \in V_S\}$$

Le symbole ψ représente la position de la tête de lecture simulée. T_1 lit jusqu'à cette marque et mémorise le pas à effectuer avec le symbole suivant. Le mot correspondant à l'entrée du transducteur simulé est entouré de deux symboles "".*

$$\delta_1 : ((\sigma_1,\sigma_2),q,D,(\sigma_1,\sigma_2),D) \in \delta_1((\sigma_1,\sigma_2),q) \forall q \in Q, (\sigma_1,\sigma_2) \in V_1, \sigma_1 \neq \psi$$

$$((\psi,\sigma_2),\rho_q,D,(\psi,\sigma_2),D) \in \delta_1((\psi,\sigma_2),q) \forall q \in Q, (\psi,\sigma_2) \in V_1$$

Simulation d'un pas de T :

$$((\sigma_1,\sigma_2),(q',M),D,(\sigma'_1,\sigma_2),D) \in \delta_1((\sigma_1,\sigma_2),\rho_q) \forall \rho_q \in Q_1, (\sigma_1,\sigma_2) \in V_1$$

tel que : $(\sigma'_1,q',M,\varepsilon,M') \in \delta(\sigma_1,q)$

$$((\sigma_1,\sigma_2),(q',M,\sigma,M'),(\sigma'_1,\sigma_2),D) \in \delta_1((\sigma_1,\sigma_2),\rho_q) \forall \rho_q \in Q_1, (\sigma_1,\sigma_2) \in V_1$$

tel que : $(\sigma'_1,q',M,\sigma,M') \in \delta(\sigma_1,q)$ *et* $\sigma \neq \varepsilon$

$$((\sigma_1,\sigma_2),f_q,D,(\sigma_1,\sigma_2),D) \in \delta_1((\sigma_1,\sigma_2),\rho_q) \forall \rho_q \in Q_1, (\sigma_1,\sigma_2) \in V_1$$

tel que $\delta(\sigma_1,q)$ *ne soit pas défini.*

Le changement de caractère effectué, T_1 écrit complètement l'élément de sortie :

$$(\sigma,(q,M),D,\sigma,D) \in \delta_1(\sigma,(q,M)) \forall \sigma \in V_{E_1}, (q,M) \in Q_1$$

$$(\sigma,(q,M,\sigma',M'),D,\sigma,D) \in \delta_1(\sigma,(q,M,\sigma',M')) \forall \sigma \in V_{E_1}, (q,M,\sigma',M') \in Q_1$$

et tel que $\sigma \neq (*,*) \wedge M' \neq D$

$$((*,*),(q',M),D,(b,\sigma),(*,*),D) \in \delta_1((*,*),(q',M,\sigma,D))$$

$$(\sigma,f_q,D,\sigma,D) \in \delta_1(\sigma,f_q) \forall \sigma \in V_{E_1}, f_q \in Q_2$$

Dans le cas où la transition est définie, le transducteur T_1 s'arrête dans un état de la forme (q,M) ou (q,M,σ,G). Il donne alors le contrôle à T_2 qui effectue les opérations suivantes :

- *écriture de la sortie*
- *mouvement de la tête de lecture*
- *passage du contrôle à* T_1.

$$Q_2 = Q_{O_2} \cup \{((q,M),\sigma) \mid q\in Q, M\in\{G,N,D\}, \sigma\in V_{E_1}\} \cup Q \cup \{D_q \mid q\in Q\}$$

$$Q_{O_2} = \{(q,M) \mid q\in Q, M\in\{G,N,D\}\} \cup \{(q,M,\sigma,G) \mid q\in Q, M\in\{G,N,D\}, \sigma\in V_S\}$$

$$F_2 = Q$$

δ_2 :

- *Ecriture immédiate de la sortie gauche :*

$$((*,*),((q,M),(\bar{b},\sigma)),D,(*,*),D)\in\delta_2((*,*),(q,M,\sigma,G))$$

$$\forall (q,M,\sigma,G)\in Q_{O_2}$$

- *Recherche de la tête de lecture :*

$$(\sigma,((q,M),\sigma),D,\varepsilon,D)\in\delta_2(\sigma,(q,M))\ \forall\sigma\in V_{E_2},\ (q,M)\in Q_{O_2}$$

$$(\sigma,((q,M),\sigma),D,\sigma',D)\in\delta_2(\sigma,((q,M),\sigma'))\ \forall\sigma\in V_{E_2},\ ((q,M),\sigma')\in Q_2$$

et $\sigma = (\sigma_1,\sigma_2), \sigma_1 \neq \psi$

- *Modification de la tête de lecture*

$$((\psi,\sigma_2),q,D,\sigma'(\psi,\sigma_2),D)\in\delta_2((\psi,\sigma_2),(q,N),\sigma'))\ \forall(\psi,\sigma_2)\in V_{E_2},$$

$$((q,N),\sigma')\in Q_2$$

- *Mouvement à gauche :*

$$((\psi,\sigma_2),q,D,(\psi,\sigma'_2)(\sigma'_1,\sigma_2),D)\in\delta_2((\psi,\sigma_2),((q,G),(\sigma'_1,\sigma'_2))$$

$$\forall(\psi,\sigma_2)\in V_{E_2},((q,N),(\sigma'_1,\sigma'_2))\in Q_2$$

- *Débordement à gauche :*

$$((\psi,\sigma_2),q,D,(*,*)(\psi,\psi)(\bar{b},\sigma_2),D)\in\delta_2((\psi,\sigma_2),((q,G),(*,*))$$

$$\forall((q,G),(*,*))\in Q_2,(\psi,\sigma_2)\in V_{E_2}$$

- *Mouvement à droite :*

$((\psi,\sigma_2),((q,D),(\psi,\sigma_2)),D,\sigma',D) \in \delta_2((\psi,\sigma_2),((q,D),\sigma'))$

$\forall (\psi,\sigma_2) \in V_{E_2}, ((q,D),\sigma') \in Q_2$

$((\sigma_1,\sigma_2),D_q,D,(\sigma_1,\sigma'_2)(\psi,\sigma_2),D) \in \delta_2((\sigma_1,\sigma_2),((q,D),(\psi,\sigma'_2))$

$\forall (\sigma_1,\sigma_2) \in V_{E_2}, ((q,D),(\psi,\sigma'_2)) \in Q_2$

$(\sigma,q,D,\sigma,D) \in \delta_2(\sigma,D_q) \forall \sigma \in V_{E_2}, \sigma \neq (*,*), D_q \in Q_2$

- *Débordement à droite :*

$(\sigma,q,D,(\not{b},\psi)\sigma,D) \in \delta_2(\sigma,D_q) \forall D_q \in Q_2, \sigma = (*,*)$

Une fois la transition terminée, T_2 *recopie le reste de l'entrée et passe le contrôle à* T_1 *:*

$(\sigma,q,D,\sigma,D) \in \delta_2(\sigma,q) \forall \sigma \in V_{E_2}, q \in Q.$

Nous avons alors les propriétés suivantes :

Soit h la fonction définie par :

. $h'(\sigma) = (\sigma,\psi) V\sigma \in V_{E_1}$

. $h(\varepsilon) = \varepsilon$

. $h(\sigma.W) = h'(\sigma).h(W)$

Lorsqu'un mot d'entrée est de la forme $(*,*)(\psi,\psi)h(W).(*,*)$*, seul le transducteur* T_2 *peut éventuellement s'arrêter dans un état final qui ne soit pas un état initial de l'autre transducteur. A chaque entrée du transducteur* T_1*, si l'état initial est q et le mot d'entrée* $(*,*)W_1(*,*)$ *nous avons :*

- $\pi_1(W_1) = W'_1 \psi W'_2$

- $\pi_2(W_1) = W''_1$

$\pi_1(W_1)$ *représente le mot d'entrée du transducteur T où* ψ *est un marqueur de la position de la tête de lecture.*

$\pi_2(W_1)$ *représente le mot de sortie du transducteur T.*

Soit f la fonction définie par :

. $f(\sigma) = \sigma \ \forall \sigma \in V_S$

. $f'(\psi) = \varepsilon$

. $f(\varepsilon) = \varepsilon$

. $f(\sigma.W) = f'(\sigma).f(W) \forall \sigma \in V_S \cup \{\psi\}, W \in (V_S \cup \{\psi\})^*$

La simulation du transducteur T par le transducteur composé $C_{T_1T_2}$ *est définie par les fonctions suivantes :*

- $h_1 : (\rho_{V_E}(T)^*)^{\rho_E(T)} \rightarrow (\rho_{V_E}(C_{T_1T_2})^*)^{\rho_E(C_{T_1T_2})}$

$h_1(W) = (*,*).(\psi,\psi).h(W).(*,*)$

h étant une injection, h_1 *l'est également.*

- $h'_1 : (\rho_{V_S}(C_{T_1T_2})^*)^{\rho_S(C_{T_1T_2})} \rightarrow (\rho_{V_S}(T)^*)^{\rho_E(T)}$

$h'_1(\sigma.W) = h''_1(\sigma).h'_1(W)$

$h'_1(\varepsilon) = \varepsilon$

$h''_1((*,*)) = \varepsilon$

$h''_1(\sigma) = f'(\pi_2(\sigma)) \forall \sigma \neq (*,*)$

- $f_1 : \rho_{Q_0}(T) \rightarrow \rho_{Q_0}(C_{T_1T_2})$

$f_1(q) = q \ \forall q \in Q_0$

- $f'_1 : \rho_F(C_{T_1T_2}) \rightarrow \rho_F(T)$

$f'_1(f_q) = q \ \forall f_q \in Q_1$

Corollaire II 4

Pour tout transducteur T de puissance égale à un, il existe un transducteur composé régulier, cyclique, direct à droite et d'ordre quatre fortement équivalent.

DEMONSTRATION

Dans la construction du transducteur composé simulant le transducteur T du théorème précédent, les fonctions peuvent être toutes effectuées par deux transducteurs d'états finis, T'_1 et T'_2 réalisant les fonctions suivantes :

$$T'_1 : (q,W) \rightarrow (f_1(q),h_1(W)) \text{ et } \rho_{Q_0}(T'_1) = \rho_{Q_0}(T)$$

$$T'_2 : (q,W) \rightarrow (f'_1(f_q),h'_1(W)) \text{ et } \rho_F(T'_2) = \rho_F(T)$$

L'équivalence forte est alors donnée par des applications identiques.

c) Transducteurs réguliers composés linéaires

Les transducteurs composés linéaires sont équivalents aux transducteurs réguliers.

Théorème II 5

Pour tout transducteur régulier composé d'ordre deux, linéaire et direct à droite, il existe un transducteur régulier fortement équivalent.

DEMONSTRATION

Il suffit de faire fonctionner les deux transducteurs en même temps. Soit $C_{T_1T_2}$ un transducteur composé régulier linéaire, direct à droite et d'ordre deux. Sans nuire à la généralité, on suppose que chaque T_i est élémentaire.

$$T_i = (V_i, V_{E_i}, V_{S_i}, \delta_i, Q_{O_i}, F_i, n_i, m_i)$$

Alors T est donné par :

$$T = (V, V_E, V_S, \delta, Q_O, F, n, m)$$

Où :

$$V = \rho_V(T_1) \cup \rho_V(T_2)$$

$$V_E = \rho_{V_E}(C_{T_1T_2})$$

$$Q_O = \rho_{Q_O}(C_{T_1T_2})$$

$$F = \rho_F(C_{T_1T_2}) \cup \{(q_1,(q_1,q_2)) \mid q_1 \in \rho_F(T_1) \cap \rho_{Q_0}(T_2), q_2 \in \rho_F(T_2)\}$$

$$n = \rho_E(C_{T_1T_2})$$

$$m = \rho_S(C_{T_1T_2})$$

$$Q = \rho_{Q_{0_1}} \cup \{(q_1,(q_2,q'_2))) \mid q_1 \in Q_{0_1}(T_1), q_2, q'_2 \in \rho_Q(T_2) \cup \{\varepsilon\}\}.$$

Fonctionnement simultané

- *Etat de départ*

$(\sigma,(q'_1,(q_2,q'_2)),D,\sigma',D) \in \delta(\sigma,q)$

$\forall \sigma \in V_E, q \in Q_0$ *et tel que :*

- *Soit* $q_2 = q'_2 = \varepsilon \wedge (\sigma,q'_1,D,\sigma',D) \in \delta_1(\sigma,q) \wedge (\rho_S(T_1) = \rho_S(T_2))$

- *Soit* $q_2 \in \rho_{Q_0}(T_2), (\sigma,q'_1,D,\sigma'',D) \in \delta_1(\sigma,q)$

$$\text{et} \begin{cases} \sigma' = \varepsilon^m, q'_2 = q_2 \text{ si } \sigma'' = \varepsilon \\ (\sigma'',q'_2,D,\sigma',D) \in \delta_2(\sigma'',q_2) \text{ sinon.} \end{cases}$$

- *Etat courant*

$(\sigma,(q_1,(q_2,q''_2)),D,\sigma',D) \in \delta(\sigma,(q_1,(q_2,q'_2)))$ $\forall \sigma \in V_E, q \in \rho_Q(T_1)$

tel que soit $q_2 = q'_2 = \varepsilon, q_2 = q''_2 = \varepsilon, (\sigma,q'_1,D,\sigma',D) \in \delta_1(\sigma,q_1)$

soit $(\sigma,q'_1,D,\sigma'',D) \in \delta_1(\sigma,q_1)$

$$\text{et} \begin{cases} \sigma' = \varepsilon, q''_2 = q'_2 \text{ si } \sigma'' = \varepsilon \\ (\sigma'',q''_2,D,\sigma',D) \in \delta_2(\sigma'',q'_2) \text{ sinon.} \end{cases}$$

Remarque :

Seul T_2 doit être direct à droite.

Corollaire

Pour tout transducteur composé régulier linéaire et direct à droite, il existe un transducteur composé régulier d'ordre un, direct à droite, fortement équivalent.

5 - PROPRIETE DES TRANSDUCTEURS A PILES

- Les transducteurs à piles ne conservent pas les langages [33].

- La réduction des transducteurs à piles composés ne peut être faite dans le cas général. Ainsi, il est simple de trouver le transducteur composé d'ordre deux qui accepte le langage : $\{a^n b^n c^n \mid n \geq 1\}$. Il est bien évident qu'il n'existe pas de transducteur à pile d'ordre un qui accepte ce seul langage.

- Les transducteurs à piles composés fortement déterministes réalisent des fonctions définies par :

$$\mathcal{F}_{C_{T_1,\ldots,T_n}} : (\rho_{V_E}(C_{T_1,\ldots,T_n})^*)^{\rho_E(C_{T_1,\ldots,T_n})} \rightarrow$$

$$(\rho_{V_S}(C_{T_1,\ldots,T_n})^*)^{\rho_S(C_{T_1,\ldots,T_n})}$$

$$\forall W \in \rho_{V_E}(C_{T_1,\ldots,T_n})^*, \mathcal{F}_{C_{T_2,\ldots,T_n}}(W) = \pi_2(C_{T_1,\ldots,T_n}(q_0,W)),$$

$$q_0 \in \rho_{q_0}(C_{T_1,\ldots,T_n})$$

De plus, le prédicat suivant est toujours déterminé par le calcul de $\mathcal{F}$:

$\mathcal{P}(W) \Longleftrightarrow \mathcal{F}_{C_{T_1,\ldots,T_n}}(W)$ est toujours défini si $C_{T_1,\ldots,T_n}$ est défini.

- Les transducteurs composés réalisent des fonctions définies de la façon suivante :

a) Fonction associée à un transducteur composé

Soit $C_{T_1,\ldots,T_n}$ un transducteur composé. Il réalise la fonction $\mathcal{F}_{C_{T_1,\ldots,T_n}}$, si $\mathcal{F}_{C_{T_1,\ldots,T_n}}$ est définie par :

$$\mathcal{F}_{C_{T_1,\ldots,T_n}} : (\rho_{V_E}(C_{T_1,\ldots,T_n})^*)^{\rho_E(C_{T_1,\ldots,T_n})} \rightarrow$$

$$\mathcal{P}((\rho_{V_S}(C_{T_1,\ldots,T_n})^*)^{\rho_S(C_{T_1,\ldots,T_n})})$$

$$\forall W \in (\rho_{V_E}(C_{T_1,\ldots,T_n})^*)^{\rho_E(C_{T_1,\ldots,T_n})}$$

$$\mathcal{F}_{C_{T_1,\ldots,T_n}}(W) = \bigcup_{q \in \rho_{Q_0}(C_{T_1,\ldots,T_n})} \pi_2(C_{T_1,\ldots,T_n}(q,W))$$

b) Propriétés

- Pour tout transducteur composé linéaire, à pile ou régulier, la fonction associée est une fonction totale.

c) Relation entre transducteurs à piles et fonctions récursives

α) Simulation d'une fonction par un transducteur composé

Un transducteur composé simule une fonction récursive φ si l'on a l'association suivante :

$$\rho_{V_E}(C_{T_1,\ldots,T_n}) = \{1\}$$

$\forall X \in N$, $\bar{X}$ est la représentation de X dans le langage $\rho_{V_E}(C_{T_1,\ldots,T_n})^*$ et définie par :

- $\bar{0} = 1$
- $\overline{S(X)} = \bar{X}.1$
- Un élément de N^k est réprésenté par un élément de $(\{1\}^*)^k$.
- Une fonction récursive $\varphi : N^k \rightarrow N^l$ est simulée par $C_{T_1,\ldots,T_n}$ si et seulement si :

$$\rho_{V_E}(C_{T_1,\ldots,T_n}) = \rho_{V_S}(C_{T_1,\ldots,T_n}) = \{1\}$$

$$\rho_E(C_{T_1,\ldots,T_n}) = k,\ \rho_S(C_{T_1,\ldots,T_n}) = 1$$

$\forall (x_1,\ldots,x_k)\in N^k$:

$$\varphi(x_1,\ldots,x_k) = (x'_1,\ldots,x'_p) \Longleftrightarrow \mathcal{F}_{C_{T_1,\ldots,T_n}}((\bar{x}_1,\ldots,\bar{x}_k)) = \{(\bar{x}'_1,\ldots,\bar{x}'_l)\}$$

d) Relation entre transducteurs à piles et fonctions récursives

Soit C_1 la classe des fonctions définies par :

- $N(x)\in C_1 \quad N(x) = 0, \forall_x$
- $S(x)\in C_1 \quad S(x) = x+1, \forall_x$
- $U_i^n (x_1,\ldots,x_n)\in C_1 \quad U_i^n (x_1,\ldots,x_n) = x_i$
- $X + Y \in C_1$
- $X \dot{-} Y \in C_1$
- C_1 est fermée pour la substitution régulière.

Théorème II 6

La classe de fonctions C_1 est contenue dans la classe des fonctions simulables par un transducteur régulier composé linéairement et déterministe.

DEMONSTRATION

Les transducteurs réalisant les cinq fonctions sont des transducteurs réguliers.

- *N(x) :*

$$T_1 = (V_1,\{1\},\{1\},\{q_0,q_1\},\delta_1,\{q_0\},\{q_1\},1,1)$$

$$V_2 = \{1,\flat\}$$

$$\delta_1 : (1,q_1,D,1,D)\in\delta_1(1,q_0)$$

$$(1,q_1,D,\varepsilon,D)\in\delta_1(1,q_0)$$

- *S(x) :*

$$T_2 = (V_1,\{1\},\{1\},\{q_0,q_1\},\delta_2,\{q_0\},1,1)$$

$$\delta_2 : (1,q_0,D,1,D)\in\delta_2(1,q_0)$$

$$(\flat,q_1,D,1,D)\in\delta_2(\flat,q_0)$$

- $U_i^n(x)$:

$$T_3 = (V_1,\{1\},\{1\},\{q_0\},\delta_3,\{q_0\},\{q_0\},n,1)$$

$$\delta_3 : (\sigma,q_0,D,1,D)\in\delta_3(\sigma,q_0)\ \forall\sigma\in V_1^n : \pi_i(\sigma) = 1$$

- $X + Y$:

$$T_4 = (V_1, \{1\}, \{1\}, \{q_0, q_1\}, \delta_4, \{q_0\}, \{q_1\}, 2, 1)$$

δ_4 : $((1,1), q_1, D, 1, D) \in \delta_4((1,1), q_0)$

$((1,\check{b}), q_1, D, 1, D) \in \delta_4((1,\check{b}), q_0)$

$((\check{b},1), q_1, D, 1, D) \in \delta_4((\check{b},1), q_0)$

$((1,1), q_1, D, 11, D) \in \delta_4((1,1), q_1)$

$((1,\check{b}), q_1, D, 1, D) \in \delta_4((1,\check{b}) q_1)$

$((\check{b},1), q_1, D, 1, D) \in \delta_4((\check{b},1), q_1)$

- $X \dot{-} Y$:

$$T_5 = (V_1, \{1\}, \{1\}, \{q_0, q_1\}, \delta_5, \{q_0\}, \{q_1\}, 2, 1)$$

δ_5 : $((1,1), q_1, D, 1, D) \in \delta_5((1,1), q_0)$

$((1,\check{b}), q_1, D, 1, D) \in \delta_5((1,1), q_0)$

$((\check{b},1), q_1, D, 1, D) \in \delta_5((\check{b},1), q_0)$

$((1,1), q_1, D, \varepsilon, D) \in \delta_5((1,1), q_1)$

$((1,\check{b}), q_1, D, 1, D) \in \delta_5((1,\check{b}), q_1)$

$((\check{b},1), q_1, D, \varepsilon, D) \in \delta_5((\check{b},1), q_1)$

Fermeture pour la substitution régulière

A chaque transducteur définissant une fonction, on associe le transducteur comportant une bande ou plusieurs bandes supplémentaires en entrée et en sortie, et qui, tout en effectuant les calculs de la fonction, recopie les bandes supplémentaires sans modifications.

Il existe un transducteur régulier réalisant une application de V^k dans $V^{k \times l}$ définie par :

$$\forall \sigma \in N^k \qquad \sigma \rightarrow \sigma^l$$

Fermeture pour la substitution régulière

Soit $h^{(n)}$ définie par :

$$h^{(n)} = g^{(p)}(f_1^{(n)}, \ldots, f_p^{(n)})$$

Alors, si g et f sont simulables par un transducteur régulier linéaire, h l'est également.

De chaque transducteur composé réalisant $f_1,\ldots,f_p$ on déduit le transducteur T_{f_i} réalisant la fonction à partir de la ième bande, et laissant inchangées les $(i-1)^{ième}$ premières bandes et les bandes comprises entre les $(i+n)^{ième}$ et la $((p-i)x\ n+i)^{ième}$.

Si - T_1 réalise la duplication : $\sigma \rightarrow \sigma^p$

- T_{f_i} réalise les fonctions précédemment décrites

- T_g réalise la fonction g

alors, h est simulé par $C_{T_1 T_{f_1},\ldots,T_{f_p} T_g}$

d) Simulation linéaire

α) Définition

Un transducteur composé T est simple si et seulement si :

- $\rho_E(T) = \rho_S(T) = 1$
- chaque transducteur le composant est un transducteur composé simple.

β) Simulation simple d'une fonction par un transducteur composé

La représentation simple d'un élément de N^k est ici linéaire et appartient au langage défini sur $\{1,*\}^*$ par :

$$(x_1,\ldots,x_k)\in N^k \Longrightarrow \overline{(x_1,\ldots,x_k)} = \bar{x}_1 * \ldots * \bar{x}_k$$

Une fonction $\varphi : N^k \rightarrow N^l$ est simulée simplement par le transducteur composé T, si et seulement si :

$\forall\ (x_1,\ldots,x_k)\in N^k$:

$$\varphi(x_1,\ldots,x_k) = (x'_1,\ldots,x'_l) \Longleftrightarrow \mathcal{F}_T(\overline{(x_1,\ldots,x_k)}) = \overline{(x'_1,\ldots,x'_l)}$$

Théorème II 7

La classe des fonctions C_1 est contenue dans la classe des fonctions simulables simplement par un transducteur à pile linéaire, simple et déterministe.

DEMONSTRATION

Les transducteurs réalisant N(x) et S(x) sont les mêmes que précédemment.

- $\cup_i^n(x_1,\ldots,x_n)$:

$$T'_3 = (\{1,\flat,*\},\{1,*\},\{1\},\{q_1,\ldots,q_n\},\delta'_3,\{q_1\},\{q_n\},1,1)$$

$$\delta'_3 : (1,q_j,D,\varepsilon,D)\in\delta'_3(1,q_j)\ \forall\ j\neq i$$

$$(*,q_{j+1},D,\varepsilon,D)\in\delta'_3(*,q_j)\ \forall\ j\neq n$$

$$(1,q_i,D,1,D)\in\delta'_3(1,q_i)$$

- $X + Y$:

$$T'_4 = (\{1,\flat,*\},\{1,*\},\{1\},\{q_0,q_1,q_2\},\delta'_4,\{q_0\},\{q_2\},1,1)$$

$$\delta'_4 : (1,q_0,D,1,D)\in\delta'_4(1,q_0)$$

$$(*,q_1,D,\varepsilon,D)\in\delta'_4(*,q_0)$$

$$(1,q_2,D,\varepsilon,D)\in\delta'_4(1,q_1)$$

$$(1,q_2,D,1,D)\in\delta'_4(1,q_2)$$

- $X \dashv Y$: *Seul ce transducteur est un transducteur à pile*

$$T'_5 = (\{1,\flat,*\},\{1,*\},\{1,*\},\{1\},\{q_0,q_1,q_2\},\delta'_5,\{q_0\},\{q_1\},1,1)$$

$$\delta'_5 : (1,1,q_0,D,\varepsilon,D)\in\delta'_5(1,\phi,q_0)$$

$$(1,11,q_0,D,\varepsilon,D)\in\delta'_5(1,1,q_0)$$

$$(*,1,q_1,D,1,D)\in\delta'_5(*,1,q_0)$$

$$(1,\varepsilon,q_1,D,\varepsilon,D)\in\delta'_5(1,1,q_1)$$

$$(1,\varepsilon,q_1,D,1,D)\in\delta'_5(1,\phi,q_1)$$

$$(\flat,\varepsilon,q_1,N,1,D)\in\delta'_5(\flat,1,q_1)$$

Fermeture pour la substitution régulière

A chaque transducteur T réalisant une fonction, on associe le transducteur similaire T' défini sur le vocabulaire $\{1,*,\alpha,\beta\}$ et tel qu'il recopie toute la bande d'entrée jusqu'au premier caractère "α" compris et ensuite effectue la même transition que T.

Soit le transducteur à pile D réalisant la fonction :

$\sigma_1 \beta \sigma_2 \rightarrow \sigma_1 \beta \sigma_2 \alpha \sigma_1$

Soit le transducteur à pile E réalisant la fonction :

$\sigma_1 \alpha \sigma_2 \rightarrow \sigma_1 * \sigma_2$

Soit le transducteur régulier S réalisant la fonction :

$\sigma_1 \beta \sigma_2 \rightarrow \sigma_2$

Alors, si $h^{(n)}$ *est une fonction définie par :*

$h^{(n)}(x_1,\ldots,x_n) = g^{(p)}\ (f_1^{(n)}(x_1,\ldots,x_n),\ldots,f_p^{(n)}(x_1,\ldots,x_n))$

Où chaque f_i *et* g *sont des fonctions simulables par un transducteur à pile composé,* $h^{(n)}$ *est simulable par le transducteur* T*, si* T *est défini par :*

$$T = C_{D.T_{f_1}\,EDT_{f_2}\ldots EDT_{f_p}\,EST_g}$$

B - COMPOSITION RECURSIVE

1 - DEFINITION

a) Concaténation produit :

Soient deux mots W_1 et W_2, éléments de $(V^*)^k$. W est dit être résultant de la concaténation de W_1 et W_2 si et seulement si :

- $W \in (V^*)^k$
- $\forall_{i \leq k}, \pi_i(W) = \pi_i(W_1).\pi_i(W_2)$

NOTATION :

$W_1.W_2$

Le préfixe d'un mot W de puissance k est défini par :

$$pr(W) = W' \iff \forall_{i \leq k}, (\pi_i(W') = \varepsilon \iff \pi_i(W) = \varepsilon) \wedge (\pi_i(W') = \sigma \iff \pi_i(W) = \sigma W"_i, \sigma \in V, W"_i \in V^*)$$

Le suffixe d'un mot W de puissance k est défini par :

$$suf(W) = W" \iff W = pr(W).W"$$

b) Composition récursive

Deux transducteurs composés T_1 et T_2 sont composables récursivement si et seulement si :

- $\rho_{Q_0}(T_1) \subset \rho_{Q_0}(T_2)$
- $\rho_S(T_1) = \rho_S(T_2) = \rho_E(T_2) = \rho_E(T_1)$
- $\rho_{V_S}(T_1) \subset \rho_{V_E}(T_2), \rho_{V_S}(T_2) \subset \rho_{V_E}(T_2), \rho_{V_E}(T_1) \subset \rho_{V_E}(T_2)$

On appelle transducteur composé récursif, deux transducteurs composés, composables récursivement (Notation $R_{T_1T_2}$).

* Les caractéristiques du transducteur $T = R_{T_1T_2}$ sont :

- vocabulaire général : $\rho_V(T) = \rho_V(T_1) \cup \rho_V(T_2)$
- vocabulaire d'entrée : $\rho_{V_E}(T) = \rho_{V_E}(T_1)$

- vocabulaire de sortie : $\rho_{V_S}(T) = \rho_{V_S}(T_1) \cup \rho_{V_S}(T_2)$

- Indices caractéristiques de sortie : $\rho_{I_S}(T) = \{1,2\}$

- états initiaux : $\rho_{Q_0}(T) = \rho_{Q_0}(T_1)$

- états finaux : $\rho_F(T) = \rho_F(T_1) \cup \rho_F(T_2)$

- états généraux : $\rho_Q(T) = \rho_Q(T_1) \cup \rho_Q(T_2)$

- puissance d'entrée : $\rho_E(T) = \rho_E(T_1) = \rho_E(T_2)$

- puissance de sortie : $\rho_S(T) = \rho_S(T_1) = \rho_S(T_2)$

- ordre : $\rho_0(T) = 2$

* Mot transcrit par un transducteur composé récursivement :

Un mot W de $(\rho_{V_E}(T)^*)^{\rho_E(T)}$ est transcrit en un ensemble de couples $T(q,W)$ (ou $R_{T_1T_2}(q,W)$ par le transducteur T, à partir de l'état q, élément de $\rho_{Q_0}(T)$, si $T(q,W)$ est défini par :

$(q',W') \in R_{T_1T_2}(q,W)$ si et seulement si :

- $pr(W) = W \wedge (q',W') \in T_1(q,W)$

- $pr(W) \neq W \wedge (q',W') \in T_2(\pi_1(T(q,suf(W))), pr(W).\pi_2(T(q,suf(W))))$

Cette définition implique :

- Un transducteur composé récursivement se comporte du point de vue externe comme un transducteur.

- Les relations de simulation, d'équivalence, d'équivalence forte et de similitude ont été définies pour l'ensemble des transducteurs composés (substitutivement ou récursivement) et conservent toutes leurs propriétés.

- Si $R_{T_1T_2}$ et $R_{T'_1T'_2}$ sont tels que :

  - T'_1 est fortement équivalent à T_1

  - T'_2 est fortement équivalent à T_2

Alors $R_{T'_1T'_2}$ est fortement équivalent à $R_{T_1T_2}$

$T'_1 \approx T_1,\ T'_2 \approx T_2 \Longrightarrow R_{T'_1T'_2} \approx R_{T_1T_2}$

2 - TRANSDUCTEURS COMPOSES RESTREINTS

a) Transducteurs composés déterministes

α) Un transducteur composé récursivement $R_{T_1T_2}$ est dit faiblement déterministe si et seulement si les transducteurs composés T_1 et T_2 sont faiblement déterministes.

β) Un transducteur composé récursivement $R_{T_1T_2}$ est dit déterministe si et seulement si les transducteurs composés T_1 et T_2 sont déterministes.

b) Transducteurs composés réguliers

Un transducteur composé récursivement $R_{T_1T_2}$ est dit "régulier" si et seulement si les transducteurs composés T_1 et T_2 sont réguliers.

c) Transducteurs composés à pile

Un transducteur composé récursivement $R_{T_1T_2}$ est dit "à pile" si et seulement si il possède la propriété :

- les transducteurs composés T_1 et T_2 sont soit des transducteurs composés réguliers, soit des transducteurs composés à pile.
- Au moins un des deux est un transducteur à pile.

3 - PROPRIETE

Théorème II 8

Pour tout transducteur composé récursivement, il existe un transducteur composé d'ordre un fortement équivalent.

DEMONSTRATION

Pour montrer cette propriété, il suffit de construire un transducteur composé substitutivement qui soit fortement équivalent au transducteur composé récursivement $R_{T_1T_2}$. Le théorème II 1 complète alors la démonstration.

4 - PROPRIETE DES TRANSDUCTEURS REGULIERS COMPOSES

La classe des transducteurs réguliers composés linéairement ou récursivement est une classe de transducteurs simulant des fonctions totales par construction. Cette classe contient un sous-ensemble important de fonctions totales : les fonctions récursives primitives.

Théorème II 9

La classe des fonctions récursives primitives est contenue dans la classe des fonctions simulables par un transducteur régulier déterministe composé linéairement et récursivement.

DEMONSTRATION

Il suffit de montrer que la classe des fonctions simulables par un transducteur régulier composé linéairement ou récursivement est fermée pour le schéma de récursion primitive.

Soit $f_1^{(n)}$ *et* $f_2^{(n+2)}$ *deux fonctions simulables par un transducteur régulier composé linéairement ou récursivment.*

Alors, la fonction $f^{(n+1)}$ *définie par :*

$$f^{(n+1)}(0,x_1,\ldots,x_n) = f_1^{(n)}(x_1,\ldots,x_n)$$

$$f^{(n+1)}(S(x),x_1,\ldots,x_n) = f_2^{(n+2)}(x,f^{(n+1)}(x,x_1,\ldots,x_n),x_1,\ldots,x_n)$$

est simulable par un transducteur régulier composé linéairement et récursivement. Si T_1 *et* T_2 *sont les transducteurs simulant* f_1 *et* f_2*, le transducteur simulant f sera de la forme* $C_{T.R_{T'_1T'_2}T'}$ *où :*

. T'_1 *et* T'_2 *sont les transducteurs simulant le schéma de récursion,*

. *T et T' deux transducteurs permettant de placer les arguments d'entrée et de sortie dans une forme convenable. La composition de* $T,R_{T_1T_2}$ *et T' est ultra-linéaire, c'est-à-dire, que nous avons :*

- $\rho_F(T) = \rho_{Q_0}(R_{T_1T_2})$, $\rho_{V_S}(T) = \rho_{V_S}(R_{T_1T_2})$

- $\rho_F(R_{T_1T_2}) = \rho_{Q_0}(T')$, $\rho_{V_S}(R_{T_1T_2}) = \rho_{V_E}(R_{T_1T_2}) = \rho_{V_E}(T')$

- $\rho_F(T) \cap \rho_{Q_0}(T') = \emptyset$

- $\rho_F(R_{T_1T_2}) \cap Q_0(T) = \emptyset$

- $\rho_F(T') \cap \rho_{Q_0}(T) = \emptyset$

- $\rho_F(T') \cap \rho_{Q_0}(R_{T_1T_2}) = \emptyset$

T_1 et T_2 ont une puissance d'entrée égale à n et n+2 respectivement et une puissance de sortie égale à un. T'_1 et T'_2 auront une puissance d'entrée égale à leur puissance de sortie et ayant pour valeur n+2.

La première bande de sortie, ainsi que les bandes 3 à n+2 seront systématiquement recopiées sur la sortie pour tous les symboles "1".

La deuxième bande aura comme sortie le résultat du clacul. Pour obtenir un transducteur ayant cette propriété, il suffit d'ajouter, à chaque transducteur élémentaire, (n+1) bandes de sorties et de transformer toutes les transitions suivant le schéma :

$$(\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{n+2},q',M,\sigma',M') \in \delta((\sigma_1,\sigma_2,\sigma_3,\ldots\sigma_{n+2}),q)$$

$$\Longrightarrow((\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{n+2}),q',M,(\sigma'_1,\sigma',\sigma'_3,\ldots,\sigma'_{n+2}),$$

$$(D,M',D,\ldots,D)) \in \delta'((\sigma_1,\sigma_2,\sigma_3,\ldots,\sigma_{n+2}),q)$$

Où $\sigma'_i = \varepsilon$ *si* $\sigma_i \neq 1$, $\sigma'_i = \sigma_i$ *sinon*

Afin d'obtenir la valeur de toutes les composantes nécessaires au calcul de f_1, le mot d'entrée sera transformé de façon à avoir chaque mot de longueur constante.

Un élément $(x,x_1,\ldots,x_n)$ est codé sur $(\{1,*\})^{n+1}$ par $(W,W_1,\ldots,W_n)$. Il sera transformé en un mot de $(\{1,*\})^{n+2}$ par le transducteur T, de façon à avoir la configuration suivante :

$$k = l(W)$$
$$k_i = l(W_i)$$
$$k_m = max\ \{k_i\}$$

$T : (W, W_1, \ldots, W_n) \rightarrow (W.*^{k_m}, *^k.*^{k_m}, *^k.W_1.*^{k_m - k_1}, \ldots,$

$*^k.W_p.*^{k_m - k_p}, \ldots, *^k W_n *^{k_m - k_n})$ *avec la convention* $*^o = \varepsilon$.

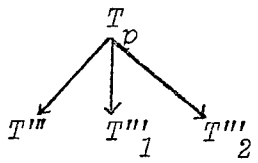
Ainsi au pas de récursion sur le premier symbole de W, nous aurons le mot : $(\sigma, *, W_1, \ldots, W_n)$ *et un pas intermédiaire :* $(W', W'', W_1, \ldots, W_n)$ *où W' est un suffixe de W et W" le résultat des précédents calculs. (Les transducteurs* T'_1 *et* T'_2 *ne recopient que les symboles "1").*

On suppose T_1 *et* T_2 *tel que* $\rho_Q(T_1) \cap \rho_Q^{\cdot}(T_2) = \emptyset$

(Dans le cas contraire, on construit les transducteurs T'_1 *et* T'_2 *à partir de* T''_1 *et* T_2 *tel que* $T''_1 \approx T_1$ *et* $\rho_Q(T''_1) \cap \rho_Q(T_2) = \emptyset$*).*

T'_1 *est un transducteur tel qu'il recopie tous les symboles "1" d'entrée sur la sortie. (*T'_1 *est inopérant).*

T'_2 *contient les deux transducteurs composés* T_1 *et* T_2*.*
Il a donc le schéma :



T_p *donne le contrôle à* T''' *dans le cas où le mot d'entrée contient un symbole "*$*$*" sur la première bande.*

T_p *donne le contrôle à* T'''_1 *dans le cas où le mot d'entrée ne contient pas de symbole "*$*$*" sur la première bande et contient un symbole "*$*$*" sur la seconde.*

T_p *donne le contrôle à* T'''_2 *dans le cas où le mot d'entrée ne contient pas de symbole "*$*$*" sur les deux premières bandes.*

T''' *recopie simplement le mot d'entrée sur la sortie.*

T'''_1 *est déduit de* T_1 *de façon à effectuer le calcul et recopier les autres bandes d'entrées.*

T'''_2 est déduit de T_2 de façon à effectuer le calcul et recopier également les autres bandes d'entrées.

A chaque pas de T'_2, nous avons à la sortie

$$(W, W', W_1, \ldots, W_n) \rightarrow (W, f_2(W, W', W_1, \ldots, W_n), W_1, \ldots, W_n)$$

Où $f_2(W, W', W_1, \ldots, W_n)$ est employé pour $\pi_2(T_2(q_0, (W, W', W_1, \ldots, W_n))$

Ainsi, $R_{T_1 T_2}$ réalise le schéma de récursion.

T' permet d'écrire le résultat; sa puissance de sortie est égale à un et il recopie seulement la deuxième bande d'entrée.

5 - PROPRIETE DES TRANSDUCTEURS A PILES COMPOSES

La classe des transducteurs à piles composés linéairement et récursivement est une classe de transducteurs simulant également des fonctions totales. Cette classe contient un ensemble de transducteurs permettant de simuler simplement un sous-ensemble important des fonctions totales, les fonctions récursives primitives.

Théorème II 10

La classe des fonctions récursives primitives est contenue dans la classe des fonctions simulables simplement par un transducteur à pile simple et déterministe, composé linéairement et récursivement.

DEMONSTRATION

De même que pour le théorème II 9, seule la fermeture pour le schéma de récursion primitive est nécessaire.

Soit $f_1^{(n)}$ et $f_2^{(n+2)}$ deux fonctions simulables simplement par un transducteur à pile simple et déterminsite, composé linéairement et récursivement. Alors, la fonction $f^{(n+1)}$ définie par le schéma de récursion primitive est simulable simplement par un transducteur à pile simple et déterministe, composé linéairement et récursivement.

$$f^{(n+1)}(0, x_1, \ldots, x_n) = f_1^{(n)}(x_1, \ldots, x_n)$$

$$f^{(n+1)}(S(x), x_1, \ldots, x_n) = f_2^{(n+2)}(x, f^{(n+1)}(x, x_1, \ldots, x_n), x_1, \ldots, x_n)$$

Si T_1 et T_2 sont les transducteurs simulant f_1 et f_2, le transducteur simulant f sera de la forme $C_{TR_{T'_1 T'_2} T'}$, où :

T'_1 et T'_2 sont des transducteurs simulant le schéma de récursion, T et T' deux transducteurs permettant de placer les arguments d'entrée et de sortie dans une forme convenable.

La composition de $T, R_{T'_1 T'_2}$, et T' est ultra linéaire, nous avons :

- $\rho_F(T) = \rho_{Q_O}(R_{T'_1 T'_2})$, $\rho_{V_S}(T) = \rho_{V_E}(R_{T_1 T_2})$

- $\rho_F(R_{T_1 T_2}) = \rho_{Q_O}(T')$, $\rho_{V_S}(R_{T_1 T_2}) = \rho_{V_E}(R_{T_1 T_2}) = \rho_{V_E}(T')$

- $\rho_F(T) \cap \rho_{Q_O}(T') = \emptyset$

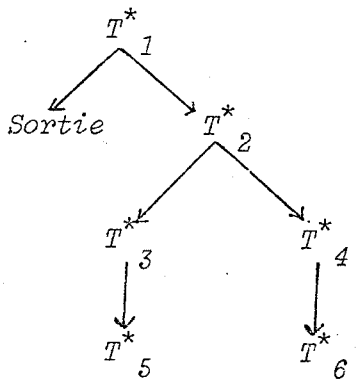
Le transducteur T augmente simplement le nombre de composantes en recopiant l'entrée. Il opère suivant le schéma :

$$W*W_1*,\ldots,*W_n \xrightarrow[T]{} W**W_1*,\ldots,*W_n$$

Le transducteur T'_1 recopie simplement l'entrée sur la sortie. Il réalise la fonction :

$$W \xrightarrow[T'_1]{} W \quad \forall\, W \in \{1,*\}^*$$

Le transducteur T'_2 est composé de plusieurs transducteurs. Il réalise le schéma de récursion. Sa composition est ultra linéaire et peut être décrite par le schéma suivant :



T'_2 effectue la fonction qui suit le schéma suivant.

$W{*}{*}W_1{*} \ldots {*}W_n \rightarrow W{*}f_1(W_1,\ldots,W_n){*}W_1{*} \ldots {*}W_n$

et $W{*}W'{*}W_1{*} \ldots {*}W_n \rightarrow W{*}f_2(W,W',W_1,\ldots,W_n){*}W_1{*} \ldots {*}W_n$

Où $f_1(W_1,\ldots,W_n)$ est employée pour $\pi_2(T_1(q_0,(W_1,\ldots,W_n))$

et $f_2(W,W',W_1,\ldots,W_n)$ est employée pour $\pi_2(T_2(q_0,(W,W',W_1,\ldots,W_n))$

On suppose que T_1 et T_2 n'ont aucun état commun. (Dans le cas contraire, la construction de T'_2 s'effectuerait à partir de T_1 et T''_2 où : $T''_2 \approx T_2$, T_1 et T''_2 ayant aucun état commun).

On suppose de plus que le symbole \$ n'appartient pas aux vocabulaires de T_1 et T_2.

Les différents transducteurs composant T'_2 effectuent les transitions suivantes :

- T^*_1 : recopie l'entrée sur la sortie et s'arrête dans son état initial dans le cas où le mot d'entrée contient moins de (n+2) symboles "*" ou si le préfixe de ce mot est un symbole "*".
Dans le cas contraire, T^*_1 donne le contrôle au transducteur T^*_2.

- T^*_2 : ce transducteur est un transducteur composé. Suivant le préfixe du mot d'entrée, il effectue plusieurs opérations.

  - Si l'entrée de T^*_2 est de la forme |**W, alors, il effectue les opérations suivantes :

    α) $|{*}{*}W \rightarrow |{*}W{*}\$W$

    β) donne le contrôle à T^*_3

  - Si l'entrée de T^*_2 n'est pas de la forme |**W, alors, l'entrée de T^*_2 est de la forme $W{*}W'{*}W''$ où $W,W' \in \{1\}^*$.
    Il effectue alors les opérations suivantes :

    α) $W{*}W'{*}W'' \rightarrow W{*}W''{*}\$W{*}W'{*}W''$

    β) donne le contrôle à T^*_4

- T^*_3 : *ce transducteur est déduit du transducteur* T_1 *en étant opérant seulement à partir du symbole "$". De plus, il recopie tous les symboles d'entrée qui précèdent le symbole "$". Il réalise la fonction suivante :*

$$W\$W' \rightarrow Wf_1(W')$$

- T^*_5 : *ce transducteur réordonne le mot d'entrée. Il est composé et réalise la fonction suivante :*

$$W*W_1* \ldots *W_n*W' \rightarrow W*W'*W_1* \ldots *W_n$$

De plus, il s'arrête dans l'état initial de T^*_1.

- T^*_4 : *ce transducteur est déduit du transducteur* T_2 *en étant opérant seulement à partir du symbole "$". Il recopie tous les symboles d'entrée qui précèdent le symbole "$". Il réalise la fonction suivante :*

$$W\$W' \rightarrow Wf_2(W')$$

- T^*_6 : *identique au transducteur* T^*_5, *ce transducteur réalise la fonction suivante :*

$$W*W_1* \ldots *W_n*W' \rightarrow W*W'*W_1* \ldots *W_n$$

Ainsi, $R_{T'_1 T'_2}$ *réalise le schéma de récursion primitive. Le transducteur T' extrait le résultat final ; il réalise la fonction :*

$$W*W'*W_1* \ldots *W_n \rightarrow W'$$

C - COMPOSITION UNIVERSELLE

Lorsque un transducteur composé peut réaliser une fonction donnée, comme par exemple, la simulation d'une transformation arborescente, il est nécessaire d'avoir un algorithme permettant d'obtenir à partir d'un langage les états de ce transducteur. Ceci peut être obtenu par un transducteur. Lors de la réalisation pratique d'un système, deux choix sont possibles. Le premier consiste à obtenir l'ensemble des transducteurs nécessaires au système et de les faire fonctionner à chaque appel des fonctions qu'ils simulent. Cette façon a l'invonvénient de prendre beaucoup de place en mémoire, puisque pour chaque fonction à simuler, il faut une table de transition d'un transducteur. L'autre possibilité est de conserver un langage de représentation de la fonction, nécessairement beaucoup moins important qu'une table de transition. L'algorithme programmé simulera alors un transducteur composé universellement. A chaque appel d'une fonction, si cette fonction n'est pas construite (table de transition du transducteur), le transducteur "constructeur" établiera préalablement à la simulation de cette fonction la table de transition. Cette deuxième possibilité a l'inconvénient de nécessiter un peu plus de temps de calcul que la première. Ce temps de calcul sera réduit si le transducteur constructeur est de type simple (régulier ou transducteur à pile).

De plus, une bonne gestion de l'emplacement réservé aux tables de transition permet de palier à cet inconvénient tout en gardant les avantages très importants que sont le gain de place très important et la plus grande simplicité du compilateur. Celui-ci doit alors transcrire un langage de représentation externe en un langage de représentation interne ; les deux langages étant très proches l'un de l'autre.

1 - TRANSDUCTEUR CONSTRUCTEUR

α) Définitions particulières

* Configuration

Soit $\{T_1,\ldots,T_n\}$ une suite de transducteurs. Une configuration sur $\{T_1,\ldots,T_n\}$ est définie de la façon suivante :

- T_i est une configuration sur la suite $\{T_i\}$. Sa représentation linéaire est : $rep(T_i) = \langle i \rangle$.

- Si $T'_1,\ldots,T'_k$ sont des configurations sur les suites $\{T_{1_1},\ldots,T_{1_{m_1}}\},\ldots,\{T_{k_1},\ldots,T_{k_{m_k}}\}$ alors, $C^{C}_{T'_1,\ldots,T'_k}$ est une configuration sur la suite $\{T_{1_1},\ldots,T_{1_{m_1}},\ldots,T_{k_1},\ldots,T_{k_{m_k}}\}$.

Sa représentation linéaire est :

$$rep(C_{T'_1,\ldots,T'_k}) = C.\langle .rep(T'_1),\ldots,rep(T'_k).\rangle$$

- Si T et T' sont des configurations sur les suites $\{T_{1_1},\ldots,T_{1_m}\}$ et $\{T_{2_1},\ldots,T_{2_p}\}$ alors, $R_{TT'}$, est une configuration sur la suite $\{T_{1_1},\ldots,T_{1_m},T_{2_1},\ldots,T_{2_p}\}$. Sa représentation linéaire est :

$$rep(R_{TT'}) = R.\langle .rep(T).rep(T').\rangle$$

* Configuration compatible

- Si $\{T_i\}$ est une suite de transducteurs ne comprenant qu'un seul élément (transducteur composé d'ordre 1), toute configuration sur cette suite est compatible.

- Si $T'_1,\ldots,T'_k$ sont des configurations compatibles sur les suites $\{T_{1_1},\ldots,T_{1_{m_1}}\},\ldots,\{T_{k_1},\ldots,T_{k_{m_k}}\}$, la configuration $C_{T'_1,\ldots,T'_k}$ sur la suite $\{T_{1_1},\ldots,T_{1_{m_1}},\ldots,T_{k_1},\ldots,T_{k_{m_k}}\}$ est une configuration compatible si et seulement si :

. $\forall_{i,j} : 1 \leqslant i,j \leqslant k : i \neq j \Longrightarrow$

$(\rho_{V_S}(T'_i) \subset \rho_{V_E}(T'_j) \wedge \rho_S(T'_i) = \rho_E(T'_j)) \vee (\rho_F(T'_i) \cap \rho_{Q_0}(T'_j) = \emptyset)$

. $\exists h : \forall_i : (\rho_F(T'_i) - \bigcup_{j \neq i} \rho_{Q_0}(T'_j) = \emptyset) \vee (\rho_S(T'_i) = h)$

. $\exists i : \rho_F(T'_i) - \bigcup_{j \neq i} \rho_{Q_0}(T'_j) \neq \emptyset$

- Si T et T' sont des configurations compatibles sur les suites $\{T_{1_1},\ldots,T_{1_m}\}$ et $\{T_{2_1},\ldots,T_{2_p}\}$ alors, la configuration $R_{TT'}$, sur la suite $\{T_{1_1},\ldots,T_{1_m},T_{2_1},\ldots,T_{2_p}\}$ est compatible si et seulement si :

. $\rho_{Q_0}(T) \subset \rho_{Q_0}(T')$

. $\rho_S(T) = \rho_S(T') = \rho_F(T') = \rho_E(T)$

. $\rho_{V_S}(T) \subset \rho_{V_E}(T'), \rho_{V_S}(T') \subset \rho_{V_E}(T'), \rho_{V_E}(T) \subset \rho_{V_E}(T')$.

Une configuration compatible sur une suite de transducteurs $\{T_1,\ldots,T_n\}$ est une composition de ces transducteurs. L'ensemble des configurations sur une suite de transducteurs donnés est un ensemble fini noté C^n.

* Transition

Soit T un transducteur, $T = (V,V_E,V_S,Q,\delta,Q_0,F,n,m)$. Une transition de T est un élément de δ, c'est-à-dire, un élément de l'ensemble : $V^n \times Q \times V^n \times Q \times \{G,N,D\} \times (V_S^*)^m \times \{G,D\}^m$. La propriété suivante est définie par extension

$$\delta = \{(\sigma,q,\sigma',q',M,W,M') \mid (\sigma',q',M,W,M') \in \delta(\sigma,q)\}.$$

Soit un symbole "*" n'appartenant pas à V et soit un ensemble fini V_Q tel que : $V_Q \cap (V \cup \{*\}) = \emptyset$. Une représentation linéaire de δ est définie par un mot sur $(V \cup V_Q \cup \{*\} \cup \{G,N,D\})^*$, appartenant au langage L_δ :

$$L_\delta = (\{*\}.V^n.V_Q^*.V^n.V_Q^*.\{G,N,D\}^n.\{*\}.(V_S^*.\{*\})^m.\{G,D\}^m.\{*\})^*$$

Si h est une application injective de Q dans V_Q^*, la représentation de δ par h est définie par :

$rep_h(\delta)$:

. $\delta = \delta' \cup \{(\sigma,q,\sigma',q',M,(W_1,\ldots,W_m),(M'_1,\ldots,M'_m))\} \wedge$

$\{(\sigma,q,\sigma',q',M,(W_1,\ldots,W_m),(M'_1,\ldots,M'_m))\} \cap \delta' = \emptyset \Rightarrow$

$rep_h(\delta) = *.\sigma.h(q).\sigma'.h(q').M.*.W_1.* \ldots *.W_m.*.M'_1 \ldots M'_m.*.$

$rep(\delta')$

. $rep_h(\emptyset) = \varepsilon$

* Transducteur

Soit T un transducteur, $T = (V,V_E,V_S,Q,\delta,Q_0,F,n,m)$.

Une représentation linéaire de T est définie par un mot sur $(V \cup V_Q \cup \{*\} \cup \{G,N,D\})^*$, appartenant au langage L_T :

$$L_T = \{*\}.(V_Q^*.\{*\})^*.V_E^*.\{*\}.V_S^*.\{*\}.(V_Q^*.\{*\})^*.L_\delta$$

Si h est une application injective de Q dans V_Q^*, la représentation de T par h est définie par :

$$rep_h(T) = \{*\}.rep_h(Q_0).\{*\}.rep(V_E).\{*\}.rep(V_S).\{*\}.rep_h(F).\{*\}.rep_h(\delta)$$

Avec :

- Si Q' est un sous-ensemble de Q, $rep_h(Q')$ est défini par :

  . $rep_h(\emptyset) = \varepsilon$

  . $rep_h(\{q\}) = h(q)$

  . $rep_h(\{q,q'\} \cup Q'') = h(q).\{*\}.rep_h(\{q'\} \cup Q'') \Longleftrightarrow$

  $Q'' \cap \{q,q'\} = \emptyset$

- Si V' est un sous-ensemble de V, rep(V') est défini par :

  . $rep(\emptyset) = \varepsilon$

  . $rep(\{\sigma\} \cup V'') = \sigma.rep(V'') \Longleftrightarrow \{\sigma\} \cap V'' = \emptyset$

β) Définition

* Un transducteur T est un transducteur constructeur si et seulement si :

- $\exists V_Q, V'$:

. $V_Q \cup V' \cup \{G,N,D\} \cup \{*\} \subset \rho_{V_S}(T)$

. $V_Q \cap V' = \emptyset$, $V_Q \cap \{*\} = \emptyset$, $V' \cap \{*\} = \emptyset$

- $\rho_F(T) = Q'XC^m \wedge m = \rho_S(T)$

* Transducteur transcrit par un transducteur constructeur

Un mot W de $(V_E^*)^n$ est transcrit en un transducteur T' par le transducteur T à partir de l'état q, élément de Q_0, si T' est défini de la façon suivante :

. $\exists (q',W') \in T(q,W)$

. $\exists \{T_1,\ldots,T_n\}$ et une application injective $h : \bigcup_{i=1}^{n} Q_i \to V_Q^*$:

$\forall_i$: $rep_h(T_i) = \pi_i(W')$

. $q' = (q'',T')$ et T' est une configuration compatible de $\{T_1,\ldots,T_n\}$.

NOTATION :

$T(q,W) \to T'$

2 - TRANSDUCTEUR UNIVERSEL COMPOSE

α) Fonction d'adaptation

Soit V un vocabulaire donné. Une fonction d'adaptation est une application A_m^n de $(V^*)^n$ dans $(V^*)^m$ défini de la façon suivante :

- $\forall_i$: $i \leq \min(m,n)$: $\pi_i(A_m^n(W)) = \pi_i(W)$

- $n<m \implies \forall_i$: $n<i\leq m$: $\pi_i(A_m^n(W)) = \varepsilon$

β) Définition

Un transducteur universel composé $\mathcal{U}_T$ est un transducteur constructeur.

Un couple de mots (u,v) est transcrit en un couple (q',W) par le transducteur universel $\mathcal{U}_T$ à partir de l'état q, si et seulement si :

. $u \in (\rho_{V_E}(T)^*)^{\rho_E(T)}$

. $T' \in T(q,u) \wedge \pi_1(q) \in \rho_{Q_0}(T')$

. $\exists\, s : v \in (\rho_{V_E}(T')^*)^s$ et $(q', W) \in T'(\pi_1(q), \hat{A}^s{}_{\rho_E(T')}(v))$

NOTATION :

$$\mathcal{U}_T(q,(u,v))$$

3 - FAMILLE CONSTRUCTIBLE

α) Définition

Une famille de transducteurs $\{T_i\}_{i \in I}$ est constructible relativement au langage L, si et seulement si :

. $\exists$ un transducteur universel $\mathcal{U}_T$ tel que :

$$\forall_{i \in I} : \exists W \in L,\ q \in \rho_{Q_0}(T) : T(q,W) \to T_i$$

Une famille de transducteurs constructibles sera dite R-constructible (p-constructible, G-constructible), si le transducteur constructeur engendrant cette famille est régulier (transducteur à pile, général).

β) Propriété

Théorème U.1

La famille de tous les transducteurs d'un ordre n donné fixé est R-constructible.

DEMONSTRATION

Immédiat d'après la définition. Le transducteur constructeur est alors l'identité et il a autant d'états initiaux que de configurations possibles sur une suite d'ordres n.

CHAPITRE III

ARBORESCENCES

INTRODUCTION

La définition des arborescences doit répondre aux objectifs suivants :

- avoir une vue globale de l'arborescence
- pouvoir sélectionner simplement une sous-arborescence
- permettre une définition des transformations à partir de sous-arborescences
- obtenir les mêmes définitions pour différents types d'arborescences (orientées, non orientées, partiellement orientées).

Différentes définitions ont été proposées pour l'étude des grammaires transformationnelles. La notion de domaine d'arbre développée par Brainerd [15] et plus généralement par Rosen [62] nous semble la plus propice à une définition d'arborescence en vue de l'étude des grammaires transformationnelles. Une arborescence est alors un ensemble de mots formés sur le vocabulaire constitué par N^+ (ensemble des entiers naturels non nuls). Une arborescence sera donc définie à partir d'un langage sur le monoïde u engendré par N^+ et l'opération de concaténation "." de deux entiers. Cette concaténation diffère de celle nécessaire à l'écriture d'un entier dans une base donnée. Ainsi, les deux mots 1.1 et 11 sont différents : le premier mot représente la concaténation de deux éléments "1" de N^+, le second, l'élément 11 de N^+. Pour représenter un domaine d'arbre (Rosen [62]), un langage A de u doit posséder les propriétés de fermeture suivantes :

- Tout préfixe d'un mot de A doit être également un mot de A :

$$X.W \in A \Longrightarrow X \in A$$

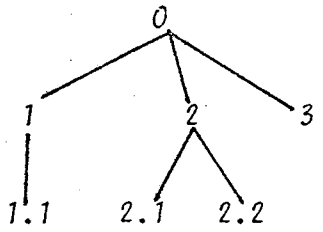
- La présence dans A d'un mot de la forme $W.j$ implique la présence dans A des mots de la forme $W.l$ où l est inférieur à j :

$$W.j \in A,\ j<l \Longrightarrow W.l \in A$$

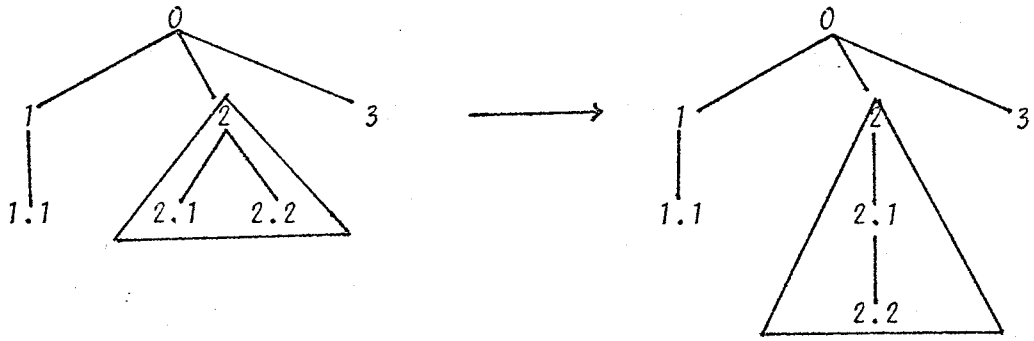
Tout langage représentant un domaine d'arbre contient le mot vide qui sera noté "0". Ainsi, la donnée du langage A définit globalement un domaine d'arbre. Ce langage associe un mot à chaque sommet de l'arborescence et la formation des mots indique les relations constructives. Tous les descendants d'un mot W admettent ce mot comme préfixe et ces descendants propres n'admettent pas

de préfixe propre plus long. Le langage suivant définit un domaine d'arbre :

$$A = \{0,1,1.1,2,2.1,2.2,3\}$$



La notion de sous-arborescence apparaît alors comme celle de sous-langage formé de tous les mots ayant un même préfixe. Le sous-langage ainsi déterminé définit un domaine d'arbre A' par effacement de ce préfixe sur chaque mot. Les grammaires transformationnelles construites sur cette définition sont similaires aux grammaires syntagmatiques définies pour les langages : le remplacement d'une sous-arborescence correspond au remplacement d'un mot :



Cette définition est insuffisante pour construire un système transformationnel général, car les arborescences traitées sont obligatoirement ordonnées, la notion de sous-arborescence est trop restrictive et le vocabulaire d'étiquettage des arborescences est un vocabulaire stratifié.

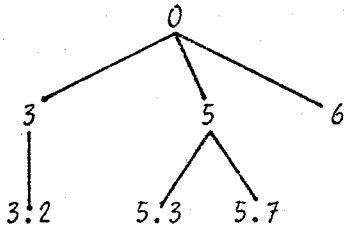
Aussi, en vue de traiter ce cas général, nous proposons la notion d'élément d'arborescence. A cet effet, la deuxième condition définissant un domaine d'arbre sera supprimée. Un élément d'arborescence est défini comme un langage A de u possédant la propriété de fermeture suivante :

- Tout préfixe d'un mot de A doit être également un mot de A.

Cette définition est plus générale que celle des domaines d'arbres. En effet, soit par exemple, l'élément A'_1 défini ci-après. Cet élément est un élément d'arborescence, mais n'est pas un domaine d'arbre.

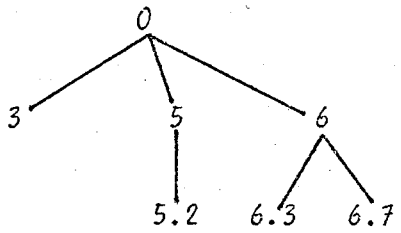
$$A'_1 = \{0,3,3.2,5,5.3,5.7,6\}$$

$A'_1 = \{C, 3, 3.2, 5, 5.3, 5.7, 6\}$



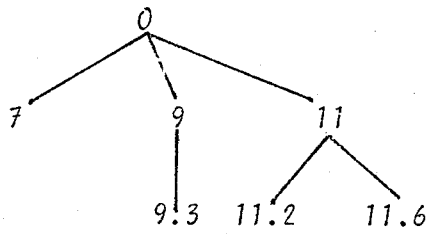
La nécessité de considérer des arborescences non orientées conduit à considérer comme arborescence un ensemble de ces éléments. Ainsi, l'élément d'arborescence A_2 défini ci-après doit définir la même arborescence que l'élément A_1 :

$A_2 = \{0, 3, 5, 5.2, 6, 6.3, 6.7\}$



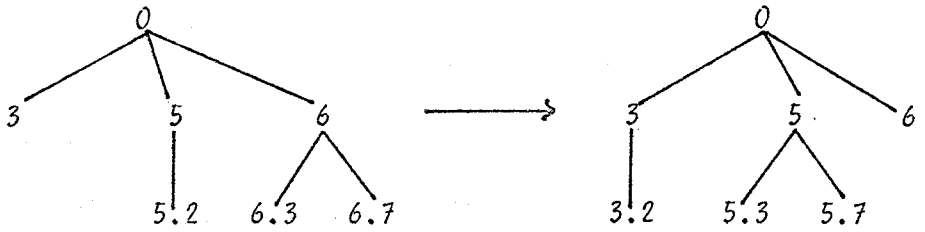
La structure arborescente est donnée par le graphe de points et ne doit pas dépendre du choix des mots associés aux différents points. Ainsi, l'élément A_3 doit également définir la même arborescence que l'élément A_1 ou A_2 :

$A_3 = \{0, 7, 9, 9.3, 11, 11.2, 11.6\}$



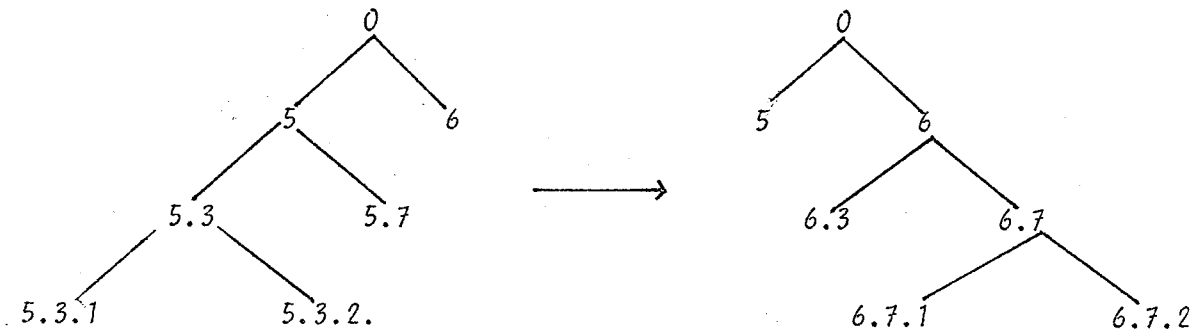
Une arborescence sera donc définie comme une classe d'équivalence d'éléments d'arborescences. Cette classe d'équivalence sera introduite par l'existence d'une fonction d'équivalence permettant de passer d'un élément de la classe à un autre élément. Cette fonction doit réaliser des permutations des éléments de N^+ pour chaque point de l'arborescence. Ainsi, la

permutation associée à la racine conduit à la transformation suivante :



Chaque sommet d'un élément d'arborescence est caractérisé par un mot de u. Donc, une fonction d'équivalence sera donnée par une application de chaque mot de u sur une permutation de N^+. La modification par cette fonction d'un entier à l'intérieur d'un mot s'effectue par l'application d'une permutation sur cet élément. Cette permutation ne dépend que du préfixe maximum de ce mot ne contenant que cet élément. Ainsi, dans l'application suivante, le passage du mot 5.3.2 au mot 6.7.1 suit le schéma :

5.3.2 → 5.3.1 (Permutation associée à 5.3)
5.3.1 → 5.7.1 (Permutation associée à 5)
5.7.1 → 6.7.1 (Permutation associée à 0)

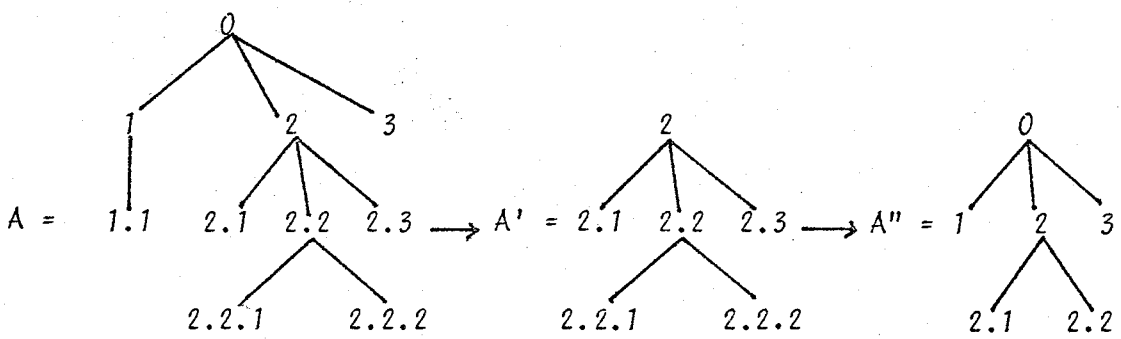


Les arborescences ainsi définies sont non orientées et deviennent indépendantes des mots de u choisis pour leurs représentations. Dans cette définition, les domaines d'arbre jouent un rôle particulier et sont appelés éléments simples. Une arborescence admet plusieurs éléments simples et pour tout élément d'arborescence, il est possible de construire une fonction d'équivalence transformant cet élément en un élément simple. On appelle fonction canonique toute fonction qui permet d'obtenir à partir d'un élément d'arborescence quelconque un élément simple équivalent.

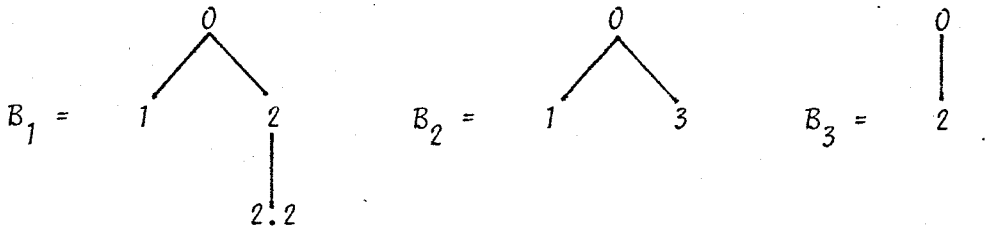
La définition des sous-arborescences s'effectue de la même façon que précédemment. Cette définition est moins restrictive et permet de définir une notion plus générale : une sous-arborescence est obtenue à partir d'un

sous-langage A' d'un élément d'arborescence. Ce sous langage contient tous les mots ayant un même préfixe X. En effaçant dans A' ce préfixe, on obtient naturellement un nouvel élément d'arborescence A".
Tout élément d'arborescence contenu dans A" est alors un représentant d'une sous-arborescence de A.

En conséquence, les sous-arborescences que nous définissons ne contiennent pas tous les sommets dépendants d'un point appartenant à la sous-arborescence. L'obtention d'une sous-arborescence peut être résumée par le schéma suivant :



Alors :



sont des représentants de sous-arborescence de A.

Certaines sous-arborescences possèdent des propriétés particulières et seront recherchées pour effectuer certaines transformations. Toutes ces sous-arborescences font référence aux points de l'arborescence principale. La définition d'un point d'une arborescence ne peut être effectuée que par l'intermédiaire d'un élément d'arborescence. Un point d'un élément d'arborescence A est un mot de ce langage A. La notion de point d'une arborescence se déduit alors par application d'une fonction d'équivalence. Le point dont le mot associé est "0" dans tous les éléments d'arborescences d'une arborescence donnée est appelé racine de cette arborescence. Un point dont le mot associé dans un élément d'arborescence A n'est le préfixe d'aucun autre mot de A est appelé feuille de cette arborescence.

La notion de "sous-arborescence sommet" et "sous-arborescence feuille" est déduite des propriétés de leurs points respectifs. Une sous-arborescence sommet a sa racine qui coïncide avec la racine de l'arborescence. Une sous-arborescence feuille est telle que chaque feuille de la sous-arborescence est également une feuille de l'arborescence. La notion de sous-arborescence complète permet d'obtenir un test d'égalité. Une sous-arborescence est complète lorsque tout point de la sous-arborescence qui n'est pas une feuille de cette sous-arborescence n'admet pas d'autres descendants directs que ceux de la sous-arborescence. Alors, une sous-arborescence A possédant les trois propriétés précédentes est égale à cette arborescence.

L'étude d'ensemble de sous-arborescences conduit à comparer leurs positions relatives. Deux sous-arborescences sont étrangères lorsqu'elles n'ont aucun point commun. Parmi les sous-arborescences étrangères, on distingue les sous-arborescences dépendantes des sous-arborescences indépendantes. Une sous-arborescence est dépendante d'une autre sous-arborescence si sa racine dépend d'une feuille de la seconde. La notion de coupe (Ginsburg and Partee [34]) se présente alors comme un ensemble de sous-arborescences étrangères indépendantes et complètes.

L'orientation des arborescences est introduite par une restriction de la fonction d'équivalence. Une fonction d'équivalence est dite ordonnée sur un élément d'arborescence si elle est croissante sur les trois éléments de A. L'orientation partielle est obtenue par des fonctions d'équivalences croissantes sur un sous-ensemble de A. Les classes d'équivalences obtenues définissent alors les sous-arborescences orientées ou partiellement orientées. Les sous-arborescences orientées ou partiellement orientées sont définies de la même façon que précédemment.

L'utilisation pratique d'un système transformationnel implique la manipulation d'arborescences étiquetées. Chaque point de l'arborescence contient alors une information appelée étiquette. L'ensemble des informations possibles constitue un vocabulaire V. Un élément d'arborescences étiquetées est constitué par un couple (A, ρ) formé d'un élément d'arborescence A et d'une fonction ρ de A dans V. Les fonctions d'équivalences sur les arborescences étiquetées opèrent de la même façon que précédemment. En transformant un élément d'arborescences A en un élément A', elles font passer la fonction ρ à une fonction ρ', permettant à chaque point de conserver son étiquette, quel que soit le représentant de u choisi. Cette propriété est naturellement

obtenue par la recherche d'un diagramme commutatif. Toutes les définitions précédentes sur les arborescences et sous-arobrescences se transposent dans le cas des arborescences étiquetées. Ainsi, la construction d'un système basé sur cette définition est indépendante de l'ensemble des étiquettes choisies et le traitement de la structure s'opère indépendamment de celui des étiquettes.

R A P P E L S

GRAPHE : couple $G = (X,u)$ où :
X est un ensemble de points $X = \{X_1,\ldots,X_n\}$
U est une famille d'éléments du produit cartésien $X \times X = \{(x,y) \mid x \in X,\ y \in X\}$.

ARC : élément (x,y) de U. x et y sont les extrémités de cet arc, x étant l'extrémité initiale et y l'extrémité terminale.

CHAINE : séquence $\mu = (u_1,\ldots,u_p)$ d'arcs de G, telle que chaque arc de la séquence ait une extrémité commune avec l'arc précédent et l'autre extrémité commune avec l'arc suivant.

GRAPHE CONNEXE : graphe tel que pour toute paire x,y de points distincts, il existe une chaîne $\mu(x,y)$ reliant ces deux points.

ARBRE : graphe tel que tout couple de sommet est relié par une chaîne et une seule.

CHEMIN : chaîne $\mu = (u_1,\ldots,u_p)$ tel que pour tout arc u_i l'extrémité terminale de u_i coïncide avec l'extrémité initiale de u_{i+1} $(i<p)$. L'extrémité initiale de u_1 est alors l'extrémité initiale du chemin.

RACINE : point tel que tout autre sommet du graphe puisse être atteint par un chemin issu de ce point.

ARBORESCENCE : arbre muni d'une racine.

Le but du présent chapitre est d'étudier une extension de la représentation des arborescences définies pour les grammaires transformationnelles (Brainerd [15] , Rosen [62]).
Cette extension est rendue nécessaire pour tenir compte des différents types d'arborescences (orientées, non orientées, partiellement orientées).

N O T A T I O N S

N^+ : ensemble des entiers naturels non nuls, $\{1,\ldots,n,\ldots\}$

u : le monoïde libre engendré par N^+ et l'opération de concaténation "." de deux entiers

"0" est l'élément neutre de ce monoïde

g^* : groupe des permutations récursives de N^+

$\mathcal{F}$: ensemble des applications récursives de u dans g^*

$\leqslant$: ordre naturel de N^+

$\lesssim$: ordre défini sur u par $a \lesssim b \iff \exists X \in u : a.X = b$
(a est préfixe de b)

ℓ : longueur d'un mot, application de u dans N^+ définie par :
- $\ell(0) = 1$
- $\ell(W.X) = \ell(W) + 1 \quad \forall X \in N^+, W \in u.$

A - D E F I N I T I O N S

1 - ELEMENT D'ARBORESCENCE

Un élément d'arborescence est une partie finie A de $\mathcal{P}(u)$ fermée à gauche pour $\leqslant$. (Si un mot appartient à A, tous ses préfixes appartiennent aussi à A).

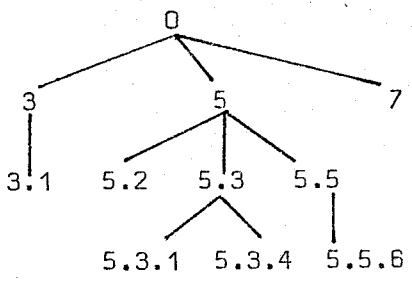
A est un élément d'arborescence si et seulement si :

- $A \in \mathcal{P}(u)$
- $Card(A) \in N^+$
- $X \in A,\ Y \in u,\ Y \leqslant X \Longrightarrow Y \in A$.

2 - EXEMPLE

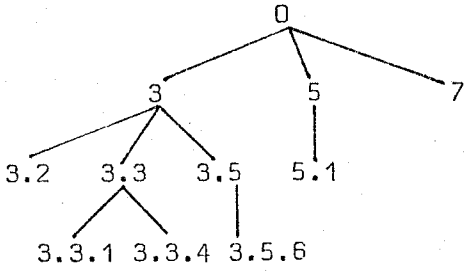
Un élément d'arborescence représente une arborescence. La relation d'ordre est directement donnée par la relation .

A = {0,3,3.1,5,5.2,5.3,5.3.2,5.3.4,5.5,5.5.6,7}

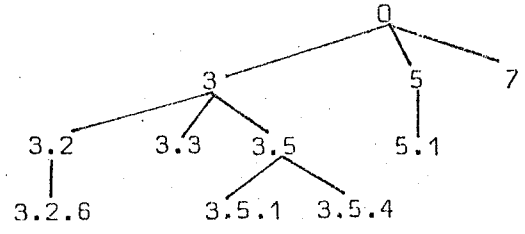


Afin d'obtenir des arborescences non orientées, il est nécessaire d'obtenir une équivalence entre plusieurs représentations ; ainsi, les éléments dépendants de "0" peuvent être permutés.

A' = {0,3,3.2,3.3,3.3.1,3.3.4,3.5,3.5.6,5,5.1,7}



A partir de A, indépendamment de la permutation opérée sur les dépendants de 0, les dépendants de 5 peuvent être également permutés :



Tous ces éléments d'arborescences, A, A', A", doivent être équivalents. Cette équivalence est donnée par la fonction d'équivalence définie ci-après.

3 - FONCTION D'EQUIVALENCE

Soit $f \in \mathcal{F}$: on définit une fonction ψ_f dite fonction d'équivalence, comme l'extension naturelle aux parties de $\mathcal{U}$ de la fonction ψ'_f définie par :

$$- \psi'_f(0) = 0$$

$$- \psi'_f(W.X) = \psi'_f(W)\ f_W(X) \quad \forall\ W \in \mathcal{U},\ X \in N^+$$

où l'on note f_W au lieu de $f(W)$, permutation récursive de N^+.

4 - PROPRIETES

Toute fonction d'équivalence :

a) préserve la longueur

b) est une permutation de $\mathcal{U}$.

DEMONSTRATION

a) On démontre par récurrence sur la longueur des mots que $l(\psi'_f(W)) = l(W)$.

- *la propriété est vraie pour n = 1 :*
  $l(\psi'_f(0)) = l(0) = 1$

- *supposons la propriété vraie pour tout mot de longueur n.*

Soit W' un mot de longueur n+1 :

$W' = W.X,\ x \in N^+,\ W \in \mathcal{U} \ \wedge\ l(W) = n$

$l(\psi'_f(W')) = l(\psi'_f(W.X)) = l(\psi'_f(W).f_W(X))$

$f_W(X) \in N^+ \Longrightarrow l(\psi'_f(W).f_W(X)) = l(\psi'_f(W)) + 1 = l(W) + 1 = l(W.X)$

b) ψ'_f *est une permutation de* $\mathcal{U}$.

INJECTION

On montre par récurrence sur la longueur des mots que :

$$\psi'_f(W) = \psi'_f(W') \Longrightarrow W = W'$$

- ψ'_f préservant la longueur, nécessairement $l(W) = l(W')$

Il existe un seul mot de longueur 1 : 0

$0 = \psi'_f(W) = \psi'_f(W') \Longrightarrow W = W' = 0$

Supposons la propiiété vraie pour tous les mots de longueur inférieure ou égale à n. Soit W et W' deux mots de longueur n+1 tel que $\psi'_f(W) = \psi'_f(W')$, nous avons :

$l(W) = l((W') = n+1 \Longrightarrow W = W_1.X,\ W' = W'_1.X',\ W_1, W'_1 \in \mathcal{U},\ X, X' \in N^+$

$\psi'_f(W) = \psi'_f(W_1.X) = \psi'_f(W_1).f_{W_1}(X)$

$\psi'_f(W') = \psi'_f(W'_1.X') = \psi'_f(W'_1).f_{W'_1}(X')$

$\psi'_f(W) = \psi'_f(W') \Longrightarrow \psi'_f(W_1).f_{W_1}(X) = \psi'_f(W'_1).f_{W'_1}(X')$

$\psi'_f(W_1).f_{W_1}(X)$ et $\psi'_f(W'_1).f_{W'_1}(X')$ étant deux mots de $\mathcal{U}$ et

$f_{W_1}(X)$, $f_{W'_1}(X')$ deux éléments de N^+, nécessairement $\psi'_f(W_1) = \psi'_f(W'_1)$

et $f_{W_1}(X) = f_{W'_1}(X')$

W_1 et W'_1 sont deux mots de longueur inférieure ou égale à n, d'après l'hypothèse :

$\psi'_f(W_1) = \psi'_f(W'_1) \Longrightarrow W_1 = W'_1$

On a donc :

$f_{W_1}(X) = f_{W_1}(X')$

f_{W_1} étant une permutation de N^+, on a nécessairement X = X' et W = W'.

SURJECTION

f_W étant une permutation de N^+, pour tout W on a :

$$\bigcup_{X \in N^+} \{\psi'_f(W.X)\} = \bigcup_{X \in N^+} \{\psi'_f(W).X\}$$

Car $\psi'_f(W.X) = \psi'_f(W).f_W(X)$ et $\bigcup_{X \in N^+} f_W(X) = N^+$

On montre par récurrence sur la longueur des mots que $\psi_f(\mathcal{U}) = \mathcal{U}$.

L'ensemble des mots de longueur inférieure ou égale à n est stable pour ψ_f.

- La propriété est vraie pour 1, $\psi_f(\{0\}) = \{0\}$.

- Supposons que la propriété soit vraie pour les mots de longueur inférieure ou égale à n.

Soit : - D_1 l'ensemble des mots de longueur inférieure ou égale à n.

- D_2 l'ensemble des mots de longueur inférieure ou égale à n+1.

$$D_2 = D_1 \cup \{W.X \mid W \in D_1 \wedge X \in N^+\}$$

$$\psi_f(D_2) = \psi_f(D_1) \cup \psi_f(\{W.X \mid W \in D_1 \wedge X \in N^+\})$$

Il suffit de montrer la propriété :

$$\bigcup_{W \in D_1} \bigcup_{X \in N^+} \{\psi_f(W.X)\} = \bigcup_{W \in D_1} \bigcup_{X \in N^+} \{W.X\}$$

Car, d'après l'hypothèse de récurrence : $\psi_f(D_1) = D_1$

Donc $\bigcup_{W \in D_1} \psi'_f(W) = \bigcup_{W \in D_1} W$

et $\psi'_f(W.X) = \psi'_f(W).f_W(X)$

f_W étant une permutation de N^+ : $\bigcup_{X \in N^+} \{f_W(X)\} = \bigcup_{X \in N^+} \{X\}$

Donc $\bigcup_{W \in D_1} \bigcup_{X \in N^+} \{\psi'_f(W.X)\} = \bigcup_{W \in D_1} \bigcup_{X \in N^+} \{W.X\}$

D'où : $\psi_f(D_2) = D_2$

REMARQUE :

Toute permutation de u n'est pas de ce type, car une permutation ne préserve pas toujours la longueur.
ψ'_δ et ψ_δ seront confondus dans tout ce qui suit.

5 - LEMME 1

La relation : $A \sim B \iff \exists f \in \mathcal{F} : B = \psi_f(A)$, est une relation d'équivalence sur $\mathcal{P}(u)$.

DEMONSTRATION

On notera toujours par la suite ψ_I *la fonction définie par :*

$\psi_I(W) = W \quad \forall\, W \in \mathcal{U}$.

Cette fonction est donnée par ψ_f *telle que* $f_W = I \ \forall W \in \mathcal{U}$

REFELXIVE

$$\psi_I(A) = A \implies A \sim A$$

SYMETRIE

ψ_f étant une bijection de $\mathcal{U}$ sur $\mathcal{U}$, cela entraîne l'existence de ψ_f^{-1}. On définit alors f' par : $f'_W = f^{-1}_{\psi_f^{-1}(W)} \quad \forall\, W \in \mathcal{U}$.

Si f_W est une permutation récursive, f'_W est également une permutation récursive d'après la thèse de Church.

On a alors :

$$\psi_f,(\psi_f(W.X)) = \psi_f,(\psi_f(W).f_W(X))$$

$$= \psi_f,(\psi_f(W)).f^{-1}_{\psi_f^{-1}(\psi_f(W))}(f_W(X))$$

$$= \psi_f,(\psi_f(W)).f_W^{-1}(f_W(X))$$

$$= \psi_f,(\psi_f(W)).X$$

On en déduit de façon évidente :

$$\psi_f,\circ\psi_f = \psi_I$$

Soit $B = \psi_f(A)$

$$\psi_f,(B) = \psi_f,(\psi_f(A)) = A$$

$$A \sim B \implies B = \psi_f(A) \implies A = \psi_f,(B) \implies B \sim A.$$

TRANSITIVE

$A \sim B \Longrightarrow B = \psi_f(A)$

$B \sim C \Longrightarrow C = \psi_g(B)$

Alors, il existe h tel que $C = \psi_h(A)$. h est définie par :

$h(W) = g(\psi_f(W)).f(W)$

h est une permutation récursive d'après la thèse de Church.

On a :

$$\psi_h(W.X) = \psi_h(W).h_W(X) = \psi_h(W).g_{\psi_f(W)}(f_W(X))$$

Mais : $\psi_g(\psi_f(W.X)) = \psi_g(\psi_f(W).f_W(X))$

$$= \psi_g(\psi_f(W)).g_{\psi_f(W)}(f_W(X))$$

Donc : $\psi_h(W.X) = \psi_g(\psi_f(W.X)) \Longleftrightarrow \psi_h(W) = \psi_g(\psi_f(W))$

Comme $\psi_h(0) = \psi_g(\psi_f(0))$ on a :

$$\psi_h(W) = \psi_g(\psi_f(W)) \quad \forall\, W \in \mathcal{U}.$$

$C = \psi_g(B) \Longrightarrow C = \psi_g(\psi_f(A)) = \psi_h(A) \Longrightarrow A \sim C$

6 - DEFINITION DES ARBORESCENCES

Une arborescence est une classe d'équivalence d'éléments d'arborescence. La définition est compatible avec la relation d'équivalence, car :

- Si A est finie, alors $\psi_f(A)$ est finie.
- Si A est fermée pour $\lneq$, alors $\psi_f(A)$ est également fermée pour $\lneq$.

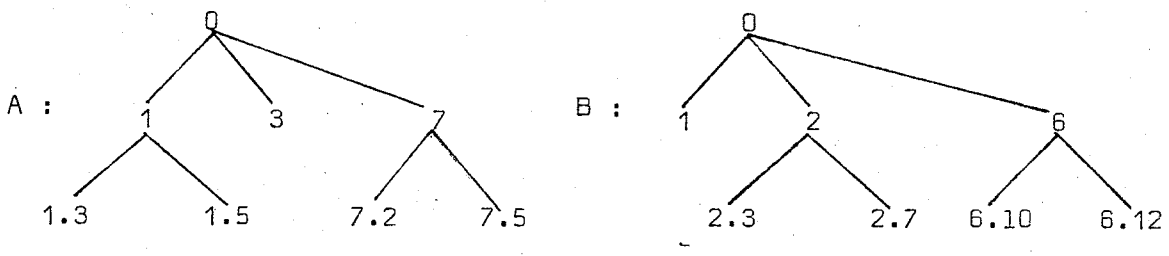
NOTATION :

{A}

Exemple d'arborescence

Soit {A} = {0,1,1.3,1.5,3,7,7.2,7.5}

{B} = {0,1,2,2.3,2.7,6,6.10,6.1.2}



Nous avons {A} = {B}

Une fonction ψ_f telle que B = ψ_f(A) est donnée par :

$f_0(1) = 2,\ f_0(2) = 3,\ f_0(3) = 1,\ f_0(6) = 7,\ f_0(7) = 6$

$f_0(X) = X \quad \forall\ X \notin \{1,2,3,6,7\}$

$f_1(5) = 7,\ f_1(7) = 5,\ f_1(X) = X \quad \forall\ X \notin \{5,7\}$

$f_7(2) = 6,\ f_7(5) = 12,\ f_7(6) = 2,\ f_7(12) = 5,\ f_7(X) = X,$

$\forall\ X \notin \{2,5,6,12\}$

$f_W = I \quad \forall\ W \notin \{0,1,7\}$

7 - ELEMENT SIMPLE

Un élément d'arborescence A est un élément simple si et seulement si :

$\forall\ x.i.y \in A, x, y \in u, i \in N^+ : j \leqslant i \Longrightarrow x.j \in A$

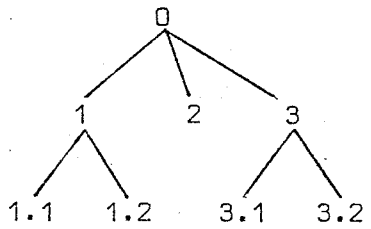
A étant fermée pour la relation $\leqslant$, cette définition est équivalente à :

$\forall\ x.i \in A,\ j \leqslant i \Longrightarrow x.j \in A$

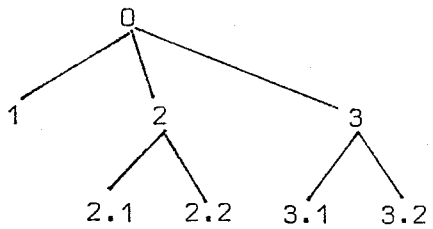
Exemple d'élément simple

Soit {A} l'arborescence définie dans l'exemple précédent. Les éléments suivants sont tous des éléments simples de A :

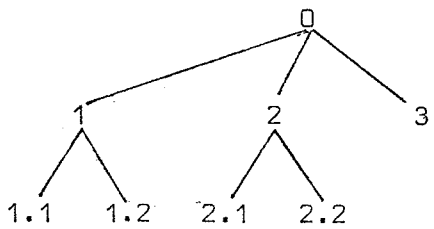
$A_1 = \{0,1,1.1,1.2,2,3,3.1,3.2\}$



$A_2 = \{0,1,2,2.1,2.2,3,3.1,3.2\}$



$A_3 = \{0,1,1.1,1.2,2.1,2.2,3\}$



8 - THEOREME 1

Il existe une fonction récursive ξ de $\mathcal{P}(\mathcal{U}) \to \mathcal{F}$ tel que pour tout élément d'arborescence A, $\psi_{\xi(A)}(A)$ soit un élément simple.

DEMONSTRATION

Un élément d'arobrescence étant une partie finie, les fonctions suivantes sont définies et récursives :

$$\alpha_X = \min_{i \in N^+} \{i \mid X.i \in A\}$$

$$\beta_X = \max_{i \in N^+} \{i \mid X.i \in A\}$$

$$\delta_X = \begin{cases} 1 \text{ si } i = \alpha_X \vee X.i \notin A \\ \delta_X(j) + 1 \text{ si } j<i \wedge X.i, X.j \in A \wedge (\forall h,\ j<h<i,\ X.h \notin A) \end{cases}$$

$\gamma_X = \min_{i \in N^+} \{ i \mid X.i \notin A \}$

On définit alors $f = \xi(A)$ *par :*

$\forall X : \nexists j : X.j \in A,\ f_X = I$

$\forall X : \exists j : X.j \in A,$

$$f_X(i) = \begin{cases} 1 \text{ si } i = \alpha_X \\ f_X(j) + 1 \text{ si } X.i,\ X.j \in A \wedge j<i \wedge (\forall h,\ j<h<i,\ X.h \notin A) \\ \delta_X(\beta_X) + 1 \text{ si } i = \gamma_X \\ f_X(j) + 1 \quad X.i,\ X.j \notin A \wedge j<i \wedge (\forall h,\ j<h<i,\ X.h \in A) \end{cases}$$

f_X *est une bijection* $\forall X$:

Si X *est tel que :* $\nexists i : X.i \in A$ *alors* $f_X = I$

Sinon :

$\forall i - i > \beta_X,\ f_X(i) = i$:

$i > \beta_X \Longrightarrow$ - $\exists\, \delta_X(\beta_X)$ *mots de la forme* $X.j$ *tel que* $X.j \in A \wedge j<i$

- $\exists i - \delta_X(\beta_X)$ *mots de la forme* $X.j$ *tel que* $X.j \notin A \wedge j<i$

$i - (\delta_X(\beta_X) + 1)$ *étant le nombre d'éléments qui n'appartiennent pas à* A *entre* $X.1$ *et* $X.i$ *excepté* $X.\gamma_X$.

Alors $f_X(i) = \delta_X(\beta_X) + 1 + i - \delta_X(\beta_X) - 1$

$f_X(i) = i$

- $i < \beta_X$, f_X *réalise une permutation des entiers entre 1 et* β_X *en conservant l'ordre des éléments de* A.

$i \leqslant j,\ X.i \in A,\ X.j \in A \Longrightarrow f_X(i) \leqslant f_X(j)$

f_X *est une bijection pour tout* X *et on a :*

$\psi_{\xi(A)}(A)$ *est équivalent à* A

$\psi_{\xi(A)}(A)$ *est un élément simple :*

$X.i \in A$ *alors*

$\psi_{\xi(A)}(X.i) = \psi_{\xi(A)}(X).f_X(i)$ *avec* $f_X = \xi(A)(X)$

$i = \alpha_X \Longrightarrow f_X(i) = 1$

$i \neq \alpha_X \Longrightarrow$ *si* $j<f_X(i)$, *d'après la définition de* ξ, *il existe* h *tel que*

$f_X(h) = j \wedge h<\beta_X \wedge X.h \in A$

donc $\psi_{\xi(A)}(X.h) = \psi_{\xi(A)}(X).f_X(h) = \psi_{\xi(A)}(X).j \in \psi_{\xi(A)}(A)$

$\psi_{\xi(A)}(X).f_X(i) \in \psi_{\xi(A)}(A) \Longrightarrow \forall j \leqslant f_X(i), \psi_{\xi(A)}(X).j \in \psi_{\xi(A)}(A)$

donc $\psi_{\xi(A)}(A)$ *est un élément simple.*

9 - LEMME 2

Le nombre des éléments simples d'une arborescence est fini.

DEMONSTRATION

Soit A un élément simple de {A}. A est un élément d'arborescence, donc A est fermée à gauche pour $\leqslant$. *Nous allons construire un élément simple A' tel que tout élément simple de {A} soit contenu dans A'. Les fonctions suivantes sont définies :*

$$n_l^A = max\ \{l(x) \mid x \in A\}$$

$$n_r^A = max(\{1\} \cup \{i \mid X.i \in A,\ X \in \mathcal{U},\ i \in N^+\})$$

Soit l'élément simple A' défini par :

$$A' = \{0\} \cup \{X.i.Y \mid l(X.i.Y) \leqslant n_l^A,\ i \leqslant n_r^A,\ X,\ Y \in A'\}$$

On a $A \subset A'$:

$X \in A$: $*l(X) = 1 \Longrightarrow X = 0 \Longrightarrow X \in A'$

$*$*Supposons* $l(X) = n \Longrightarrow X \in A'$ *soit* $X \in A$ *et* $l(X) = n + 1$, $X = Y.i$ *avec* $Y \in A$ *et* $l(Y) = n$.

$Y.i \in A \Longrightarrow i \leqslant n_r^A,\ l(Y.i) \leqslant n_l^A,\ Y \in A'$

donc par construction de A' : $Y.i \in A'$

Si A_1 est un élément simple de $\{A\}$, il existe une fonction f, telle que $A_1 = \psi_f(A)$. Soit $X.i \in A$, alors, il existe au plus n_r^A mots de même longueur ayant le même préfixe. (Sinon $\exists k > n_r^A : X.k \in A$)

$$\psi_f(X.i) = \psi_f(X).f_X(i)$$

ψ_f étant une bijection, il existe au plus n_r^A mots ayant pour préfixe $\psi_f(X)$ dans A_1. Comme A_1 est un élément simple, donc fermé à gauche pour $\leqslant$, on a :

$$f_X(i) \leqslant n_r^A \quad \forall\ i : X.i \in A$$

La démonstration par récurrence que $\psi_f(A) \subset A'$ est maintenant simple.

$$X = 0 \Longrightarrow \psi_f(X) = 0 \Longrightarrow \psi_f(X) \in A'.$$

Si $l(X) = n \wedge X \in A \Longrightarrow \psi_f(X) \in A'$ soit $X.i \in A$ tel que $l(X.i) = n+1$

$$\psi_f(X.i) = \psi_f(X).f_X(i) \quad X.i \in A \Longrightarrow f_X(i) \leqslant n_r^A$$

$$\psi_f(X) \in A' \text{ par hypothèse} \Longrightarrow \psi_f(X).f_X(i) \in A'$$

Donc $\psi_f(X.i) \in A'$

$$A_1 = \psi_f(A) \subset A'$$

Tout élément simple de $\{A\}$ est contenu dans A'. Il y a donc un nombre d'éléments simples de $\{A\}$ au plus égal au nombre de parties de A' fermées à gauche pour $\lesssim$ et $\leqslant$ et ayant le même nombre d'éléments qu'un représentant de A. Comme A' est fini, ce nombre est fini.

10 - FONCTION CANONIQUE

On appelle fonction canonique toute fonction de l'ensemble des éléments d'arborescences dans lui même possédant la propriété.

h canonique $\Longleftrightarrow \forall$ A élément d'arborescence :

h(A) $\sim$ A $\wedge$ h(A) est un élément simple.

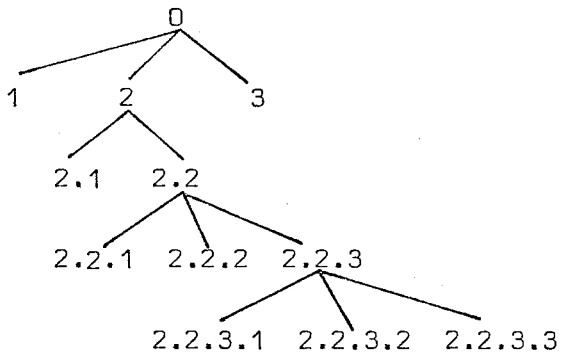
La fonction $\psi_{\xi(A)}(A)$ est une fonction canonique dont on ne réfèrera pas la suite par γ :

$$\gamma(A) = \psi_{\xi(A)}(A) \quad \forall \text{ élément d'arborescence A.}$$

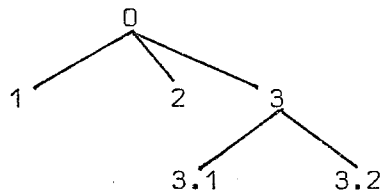
B - S O U S - A R B O R E S C E N C E S

Les relations construites sur les arborescences s'obtiennent à partir des relations sur des sous-arborescences. Une sous-arborescence est une arborescence présente dans l'arborescence donnée. Cette propriété est facilement donnée pour la fonction d'effacement qui permet de considérer l'ensemble des points ayant le même préfixe, c'est-à-dire, le même ancêtre. Soit par exemple l'arborescence A définie par :

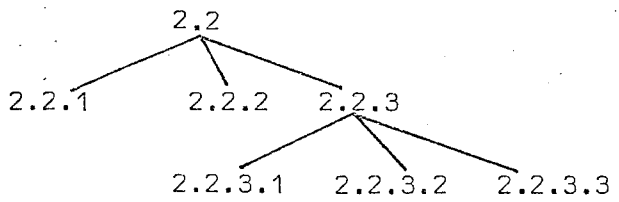
A = {0,1,2,2.1,2.2,2.2.1,2.2.2,2.2.3,2.2.3.1,2.2.3.2,2.2.3.3,3}



L'arborescence B' = {0,1,2,3,3.1,3.2} est une sous-arborescence de A



Il suffit de considérer par exemple, l'ensemble des descendants du point 2.2 :



Par effacement du préfixe 2.2, sur tous les mots de cet ensemble, on obtient un élément d'arborescence qui contient l'élément d'arborescence B.

1 - FONCTION D'EFFACEMENT

Une fonction d'effacement notée $\mathcal{E}_X$ est l'extension naturelle aux parties de u d'une fonction $\mathcal{E}'_X$ définie par :

$$\mathcal{E}'_X(X.Y.) = Y \quad \forall\ X.Y \in u.$$

$$\mathcal{E}'_X(Y) = 0 \quad \forall\ Y \in u \wedge Y \not\leqslant X$$

2 - PROPRIETE

$\forall\ f \in \mathcal{F}$, $A \in \mathcal{P}(u)$, on a $\mathcal{E}_{\psi_f(X)}(\psi_f(A)) \sim \mathcal{E}_X(A)$

DEMONSTRATION

On construit f' telle que $\psi_{f'}(\mathcal{E}_X(A)) = \mathcal{E}_{\psi_f(X)}(\psi_f(A))$

f' est définie par :

$$f'_W = f_{X.W} \quad \forall\ W \in \mathcal{E}_X(A)$$

$$f'_W = I \quad \forall\ W \in \mathcal{E}_X(A)$$

Cette fonction est récursive car A étant un ensemble fini, $\mathcal{E}_X(A)$ *est également fini.*

On montre alors sur la longueur des mots que :

$$\psi_{f'}(W) = \mathcal{E}_{\psi_f(X)}(\psi_f(X.W)) \quad \forall\ X.W \in A$$

La propriété est vraie pour 1 :

$$\psi_{f'}(0) = \mathcal{E}_{\psi_f(X)}(\psi_f(X)) = 0$$

Si la propriété est vraie pour tout mot de longueur inférieure ou égale à n, soit $W.i \in A$, *de longueur n+1 :*

$$\psi_{f'}(W.i) = \psi_{f'}(W).f'_W(i) = \psi_{f'}(W).f_{X.W}(i)$$

Mais $\psi_f(X.W.i) = \psi_f(X.W).f_{X.W}(i)$

$\psi_f(X.W)$ *a pour préfixe* $\psi_f(X)$ *donc*

$$\psi_{f'}(W.i) = \mathcal{E}_{\psi_f(X)}(\psi_f(X.W.i)) \Longleftrightarrow \psi_{f'}(W) = \mathcal{E}_{\psi_f(X)}(\psi_f(X.W))$$

$\forall\ W$ *tel que* $X.W \in A$ *on a* $\psi_{f'}(W) = \mathcal{E}_{\psi_f(X)}(\psi_f(X.W))$

donc $\forall\ W \in \mathcal{E}_X(A) : \psi_{f'}(W) = \mathcal{E}_{\psi_f(X)}(\psi_f(X.W))$

et $\forall\ X.W \in A : \psi_f(X).\psi_{f'}(W) = \psi_f(X.W)$

Alors :

$W \in \mathcal{E}_X(A) \iff X.W \in A$

$X.W \in A \iff \psi_f(X.W) \in \psi_f(A)$

$\iff \psi_f(X).\psi_{f'}(W) \in \psi_f(A)$

$\iff \psi_{f'}(W) \in \mathcal{E}_{\psi_f(X)}(\psi_f(A))$

donc $\mathcal{E}_{\psi_f(X)}(\psi_f(A)) \sim \mathcal{E}_X(A)$

3 - LEMME 3

Si A est un élément simple, alors :

$\forall\ X \in A$, $\mathcal{E}_X(A)$ est un élément simple.

DEMONSTRATION

$Y.i \in \mathcal{E}_X(A) \iff X.Y.i \in A$

$j \leqslant i \implies X.Y.j \in A \implies Y.j \in \mathcal{E}_X(A)$

donc $\mathcal{E}_X(A)$ *est un élément simple*

4 - SOUS-ARBORESCENCE

Une arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si :

$\exists f \in \mathcal{F}$, $A \in \{A\}$, $B \in \{B\}$, $X \in A$ tel que : $B \subset \psi_f(\mathcal{E}_X(A))$

Cette définition est cohérente, c'est-à-dire, que si cette propriété est vérifiée pour deux éléments B et A de {B} et {A}, elle est vérifiée pour tous les éléments de {B} et de {A}.

Preuve :

Soit $B' \in \{B\}$, $A' \in \{A\}$. Alors, il existe f_1 et f_2 telles que :

$$B' = \psi_{f_1}(B),\quad A' = \psi_{f_2}(A)$$

Comme $\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A)) \sim \mathcal{E}_X(A)$, $\exists f_3$ tel que :

$$\psi_{f_3}(\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A))) = \mathcal{E}_X(A)$$

Alors :

$$\psi_f(\psi_{f_3}(\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A)))) = \psi_f(\mathcal{E}_X(A)) \supset B$$

et :

$$\psi_{f_1}(\psi_f(\psi_{f_3}(\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A))))) \supset \psi_{f_1}(B)$$

$$A' = \psi_{f_2}(A),\ X' = \psi_{f_2}(X),\ B' = \psi_{f_1}(B) \text{ et } \psi_{f'} = \psi_{f_1} \circ \psi_f \circ \psi_{f_3}$$

Alors :

$$B' \subset \psi_{f'}(\mathcal{E}_{X'}(A'))$$

5 - LEMME 4

Une arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si :

$$\forall\, A \in \{A\},\ B \in \{B\},\ \exists X \in \psi_{\xi(A)}(A),\ f \in \mathcal{F} \text{ tel que :}$$

$$\psi_{\xi(B)}(B) \subset \psi_f(\mathcal{E}_X(\psi_{\xi(A)}(A)))$$

DEMONSTRATION

Evident d'après la cohérence de la définition.

6 - COROLLAIRE 4

Une arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si :

$$\forall\, B \in \{B\},\ A \in \{A\},\ \exists X \in \psi_{\xi(A)}(A),\ f \in \mathcal{F} \text{ tel que :}$$

$$\psi_f(\psi_{\xi(B)}(B) \subset \mathcal{E}_X(\psi_{\xi(A)}(A))$$

7 - LEMME 5

Une arborescence {B} est une sous-arborescence de {A} si et seulement si, il existe deux éléments simples B et A et un point X tels que :

$B \in \{B\}$, $A \in \{A\}$, $X \in A$, $B \subset \mathcal{E}_X(A)$

DEMONSTRATION

D'après le lemme 4, l'arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si, il existe deux éléments simples B et A, un point X et une fonction f tels que :

$$B \subset \psi_f(\mathcal{E}_X(A))$$

ψ_f peut être égale à ψ_I en dehors des éléments de $\mathcal{E}_X(A)$. D'après le lemme 3, $\mathcal{E}_X(A)$ est encore un élément simple.

Soit f' définie par :

$$f'_{X.W} = f_W \quad \forall\ W$$

$$f'_W = I \quad \forall\ W : W \leqslant X \vee X \not\leqslant W$$

Alors, $\psi_{f'}(A)$ est un élément de {A} tel que :

$$\mathcal{E}_X(\psi_{f'}(A)) = \psi_f(\mathcal{E}_X(A))$$

On considère $A' = \psi_{f'}(A)$

Si A' est un élément simple, la propriété est démontrée. Sinon, il existe deux éléments de $\mathcal{E}_X(A)$ tels que :

$$\psi_f(Y.i) = \psi_f(Y).h$$

$$\psi_f(Y.j) = \psi_f(Y).k$$

Avec h, $h + 1 \neq k$, $h \leqslant k$ et : - $\forall\ l$, $h < l < k$ $\quad \psi_f(Y).l \notin \mathcal{E}_X(A')$

- $\forall\ l$, $l \leqslant h$ $\quad \psi_f(Y).l \in \mathcal{E}_X(A')$

nécessairement, $l > h \Longrightarrow \psi_f(Y).l \notin B$ car B est un élément simple et $B \subset \psi_f(\mathcal{E}_X(A)) = \mathcal{E}_X(A')$.

On considère alors f'' définie par :

$f''_W = I \quad \forall W \not\leq X \vee W \neq X.Y$

$f''_{X.Y}(l) = l \quad \forall l \neq j \quad l \neq f_Y^{-1}(h+1)$

$f''_{X.Y}(j) = h+1$

$f''_{X.Y}(f_Y^{-1}(h+1)) = k$

Alors, $B \subset \mathcal{E}_X(A') \iff B \subset \mathcal{E}_X(\psi_{f''}(A'))$ *car* $X.j,\ Y.h+1 \notin B$

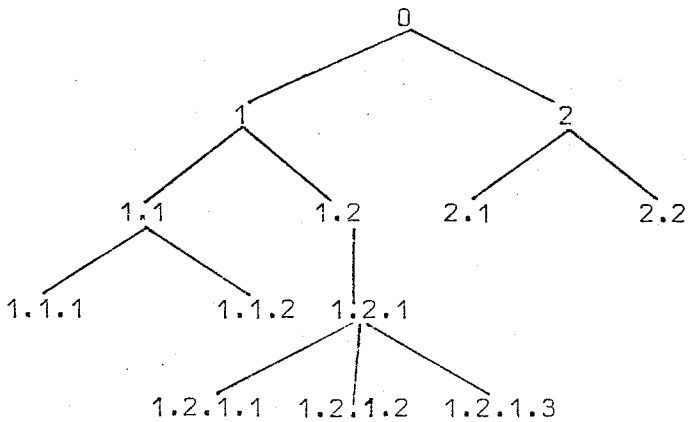
$\psi_{f''}(A')$ *a un point de moins qui ne satisfait pas la propriété d'élément simple. Comme A' est fini, de proche en proche, il existe f''' tel que* $\psi_{f'''}(A')$ *soit élément simple et*

$B \subset \mathcal{E}_X(A') \iff B \subset \mathcal{E}_X(\psi_{f'''}(A'))$.

8 - EXEMPLE

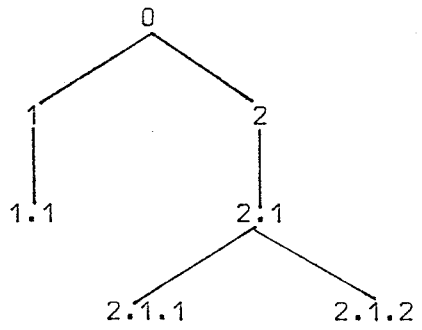
Soit l'arborescence {A} définie par :

A = {0,1,1.1,1.1.1,1.1.2,1.2,1.2.1,1.2.1.1,1.2.1.2,1.2.1.3, 2,2.1,2.2}



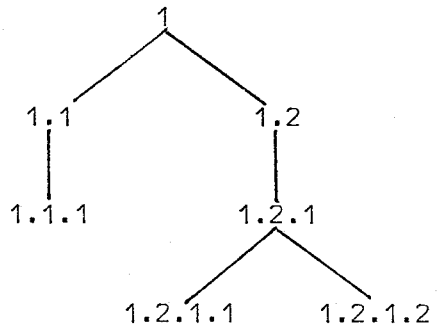
et l'arborescence {B} telle que :

B = (0,1,1.1,2,2.1,2.1.1,2.1.2)



Les deux éléments A et B sont des éléments simples et {B} est une sous-arborescence de A. En effet, soit :

A' = (1,1.1,1.1.1,1.2,1.2.1,1.2.1.1,1.2.1.2)



Alors : $A' \subset A$ et $\mathcal{E}_1(A') = B$

C - SOUS - ARBORESCENCES REMARQUABLES

1 - POINT D'UNE ARBORESCENCE

a - Point associé à un élément d'arborescence

Soit A un élément d'arborescence. Tout mot X appartenant à A est un point associé à l'élément d'arborescence A. (Not (X, A) où (X_A)).
A est alors l'élément de référence pour X.

b - Point d'une arborescence

Soit (X, A) un élément d'arborescence et un point associé. Ce point engendre le point de {(X, A)} de l'arborescence A par :

$$(X, A), (X', A') \in \{(X, A)\} \iff \exists f \in \mathcal{F} : A' = \psi_f(A) \wedge X' = \psi_f(X)$$

PROPRIETE

Soit {(X, A)} un point d'une arborescence et (X, A) le point de l'élément d'arborescence générateur A. Alors, pour tout élément d'arborescence A' équivalent à A, il existe un point associé X' tel que :

$$(X', A') \in \{(X, A)\}$$

La propriété est évidente d'après la définition de {A},

$A \sim A' \implies \exists f : A' = \psi_f(A)$

$X \in A \implies X' = \psi_f(X) \in A'$ et (X', A') forme le couple désiré.

2 - FEUILLE D'UNE ARBORESCENCE

Un point X associé à l'élément d'arborescence A est une feuille si et seulement si : $\forall\ Y \in A,\ Y \neq X \implies X \not\leq Y$.

Un point {(X, A)} d'une arborescence {A} est une feuille si et seulement si $\exists$ un élément $(X', A') \in \{(X, A)\}$ tel que X', soit une feuille de A'.

Cette définition est cohérente car si X est une feuille de A, pour tout f, $\psi_f(X)$ est une feuille de $\psi_f(A)$.

3 - SOUS-ARBORESCENCES PARTICULIERES

a - Sous-arborescence sommet

Une arborescence {B} est une sous-arborescence sommet de l'arborescence {A} si et seulement si :

$\exists f \in \mathcal{F}$, $A \in \{A\}$, $B \in \{B\}$ tels que $B \subset \psi_f(A)$

b - Sous-arborescence feuille

Une arborescence {B} est une arborescence feuille de {A} si on a la propriété :

Soit $f \in \mathcal{F}$, $B \in \{B\}$, $A \in \{A\}$, $X \in A$ tels que : $B \subset \psi_f(\mathcal{E}_X(A))$

Alors, Y est une feuille de B $\iff$ X.Y est une feuille de A.

c - Sous-arborescence complète

Une arborescence {B} est une sous-arborescence complète de {A} si on a la propriété :

Soit $f \in \mathcal{F}$, $B \in \{B\}$, $A \in \{A\}$, $X \in A$ tels que : $B \subset \psi_f(\mathcal{E}_X(A))$

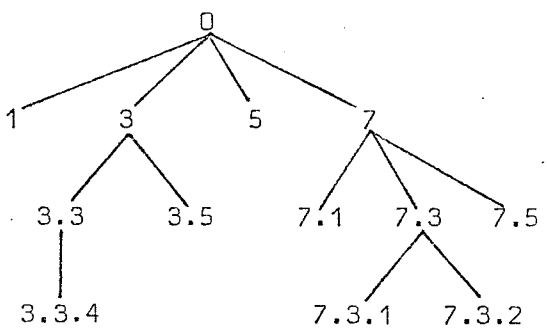
Alors : $Y \in \psi_f(\mathcal{E}_X(A)) \wedge Y \notin B \Longrightarrow \exists X' \in B : X' \leqslant Y \wedge X'$ est une feuille de B.

PROPRIETE

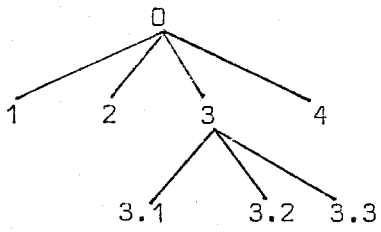
Si une sous-arborescence {B} de l'arborescence {A} est à la fois sous-arborescence sommet, feuille et complète, les deux arborescences sont égales.

Exemple

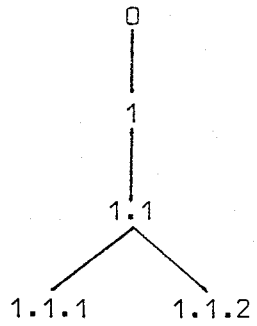
Soit A = {0,1,3,5,7,3.3,3.5,3.3.4,7.1,7.3,7.5,7.3.1,7.3.2}



B = {0,1,2,3,4,3.1,3.2,3.3}

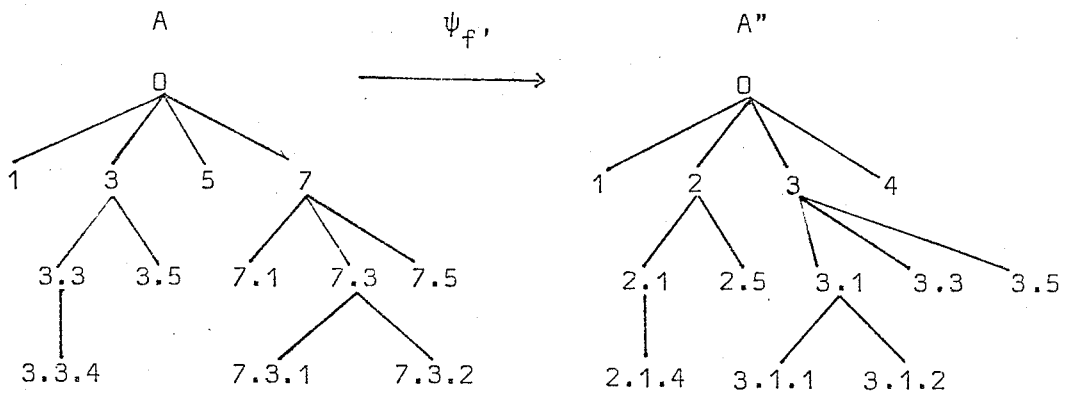


C - {0,1,1.1,1.1.1,1.1.2}



Alors l'arborescence {B} est une sous-arborescence sommet et complète de l'arborescence {A}, l'arborescence {C} est une sous-arborescence sommet et feuille de l'arborescence {A}.

Nous avons :



Avec f défini par f_0 : (1 → 1, 3 → 2, 7 → 3, 5 → 4, 2 → 5, 4 → 6,

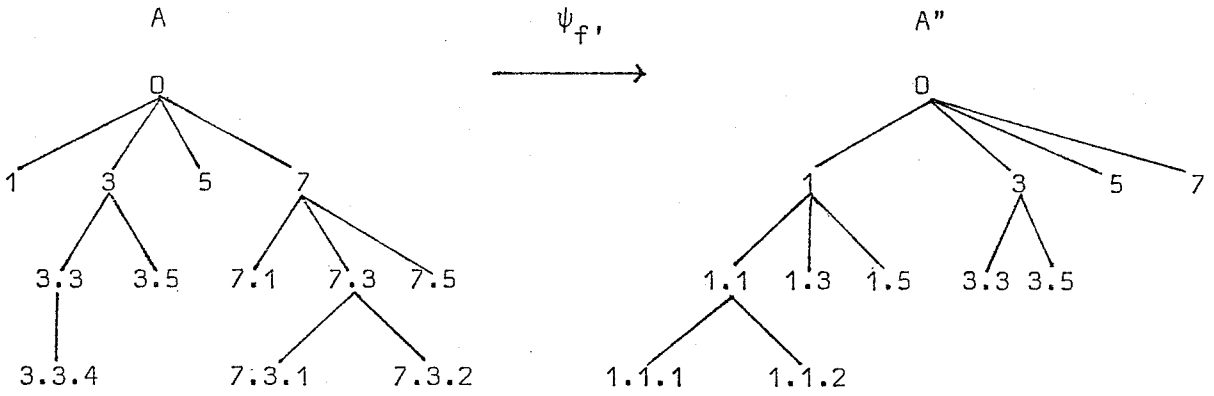
6 → 7, X → X ∀ X>7)

f_3 = (3 → 1, 1 → 3, X → X, ∀ X ≠ 1,3)

$f_7 = f_3$

f_W = I ∀ W ≠ 0,3,7

Alors, B⊂A'.



Avec f' définie par : f'_0 : (1 → 7, 7 → 1, X → X ∀ ≠ 1,7)

f'_7 : (1 → 3, 3 → 1, X → X ∀ X ≠ 1,3)

f_W = I ∀ W ≠ 0,7

Alors, C⊂A".

4 - POINTS ADJACENTS D'UNE ARBORESCENCE

a - Points adjacents d'un élément d'arborescence

Deux points X et Y de l'élément d'arborescence A sont adjacents si et seulement si :

$$(X = 0 \wedge (Y = Y'.j,\ j\in N^+ \Longrightarrow \forall\ k<j,\ Y'.k\notin A) \vee$$

$$(X\neq 0 \wedge (X = X'.i,\ i\in N^+ \Longrightarrow (Y = X'.j \wedge \forall\ k : i<k<j,\ X'.k\notin A) \vee$$

$$(Y = 0 \wedge \forall\ k>i : X'.k\notin A)$$

NOTATION :

$Adj(A,X,Y)$

b - Sous-arborescence à points adjacents

* descendant conditionnel d'un point X

Définition : $\pi^C_X(Y) = \begin{cases} Y & \text{si } Y = 0 \\ X.Y & \text{si } Y \neq 0 \end{cases}$

* sous-arborescence à points adjacents

Soit une arborescence {B}, sous-arborescence de l'arborescence {A}. Cette sous-arborescence admet le couple de points $(\{Y_B\},\{Y'_B\})$ comme points adjacents si et seulement si :

$\exists A\in\{A\},\ B\in\{B\},\ X\in A$:

- $B\subset \mathcal{E}_X(A)$
- $Y_B\in\{Y_B\},\ Y'_B\in\{Y'_B\} \Longrightarrow Adj(A,\pi^C_X(Y),\ \pi^C_X(Y'))$

Une sous-arborescence {B} de l'arborescence {A} est munie de l'ensemble de points adjacents $\{(\{Y_B\},\{Y'_B\})\}$ si et seulement si :

$$\exists A\in\{A\},\ B\in\{B\},\ X\in A :$$

- $B\subset\mathcal{E}_X(A)$
- $\forall\ (\{Y_B\},\{Y'_B\})\in\{(\{Y_B\},\{Y'_B\})\},$
  $Y_B\in\{Y_B\}\wedge Y'_B\in\{Y'_B\}\Longrightarrow Adj(A,\pi^C_X(Y),\pi^C_X(Y'))$

5 - POSITIONS RELATIVES DE DEUX SOUS-ARBORESCENCES

Soit {A} une arborescence et {B} et {C} deux sous-arborescences de {A}.

α) Les deux sous-arborescences {B} et {C} de {A} sont étrangères si et seulement si :

$$\exists A\in\{A\},\ B\in\{B\},\ C\in\{C\},\ f,\ f'\in\mathcal{F},\ X,\ X'\in A$$

- $B\subset\psi_f(\mathcal{E}_X(A))$
- $C\subset\psi_{f'}(\mathcal{E}_{X'}(A))$
- $\{X.\psi_f^{-1}(B)\}\cap\{X'.\psi_{f'}^{-1}(C)\}=\emptyset$

β) Deux arborescences sont étrangères et indépendantes si et seulement si :

$$\exists A\in\{A\},\ B\in\{B\},\ C\in\{C\},\ f,\ f'\in\mathcal{F},\ X,\ X'\in A \text{ tel que}$$

- $B\subset\psi_f(\mathcal{E}_X(A))$
- $C\subset\psi_{f'}(\mathcal{E}_{X'}(A))$
- $\{X.\psi_f^{-1}(B)\}\cap\{X'.\psi_{f'}^{-1}(C)\}=\emptyset$
- $X\not\leq X'\ \wedge\ X'\not\leq X$

γ) Deux arborescences étrangères qui ne sont pas indépendantes sont dites dépendantes. De plus, si {B} et {C} sont deux sous-arborescences étrangères dépendantes de {A}, {C} est dite dépendante de {B} si et seulement si :

$$\exists A\in\{A\},\ B\in\{B\},\ C\in\{C\},\ f,\ f'\in\mathcal{F},\ X,\ X'\in A \text{ tels que :}$$

- $B\subset\psi_f(\mathcal{E}_X(A))$
- $C\subset\psi_{f'}(\mathcal{E}_{X'}(A))$
- $\{X.\psi_f^{-1}(B)\}\cap\{X'.\psi_{f'}^{-1}(C)\}=\emptyset$
- $X\leq X'$

Le point de dépendance est le point X" défini par :

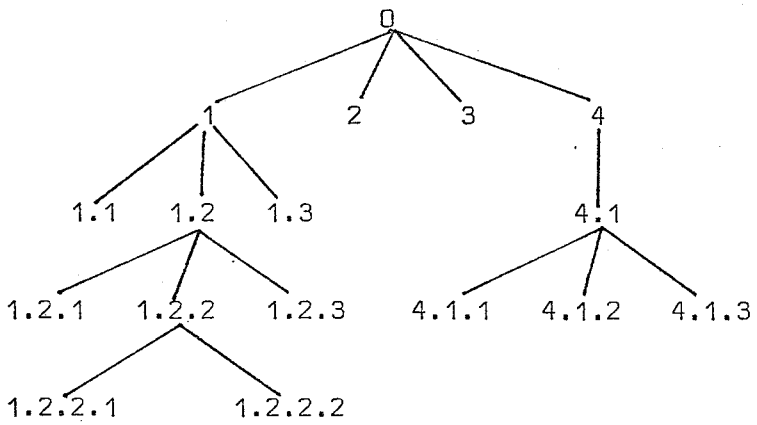
{(X",A)} et

$X \leqslant X'' \leqslant X' \wedge \psi_f \mathcal{E}_X(X")$ est une feuille de B.

Exemples

Soit l'arborescence {A} telle que :

A = {0,1,1.1,1.2,1.2.1,1.2.2 , 1.2.2.1,1.2.2.2,1.2.3,1.3,
2,3,4,4.1,4.1.1,4.1.2,4.1.3}

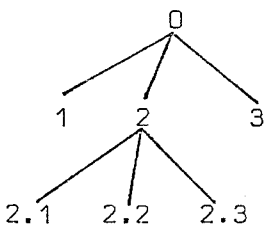


et les trois sous-arborescences {C},{D},{E} définies par :

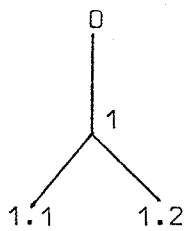
C = {0,1,2,2.1,2.2,2.3,3}

D = {0,1,1.1,1.2}

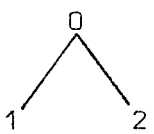
E = {0,1,2}



C



D



E

Les sous-arborescences {C} et {D} sont étrangères

$X = 1 \qquad X' = 4$

$\psi_f = \psi_I \qquad \psi_f = \psi_I$

$C \subset \mathcal{E}_1(A) = \{0,1,2,2.1,2.2,2.2.1,2.2.2,2.3,3\}$

$D \subset \mathcal{E}_4(A) = \{0,1,1.1,1.2,1.3\}$

Les sous-arborescences {D} et {E} sont dépendantes

$X = 0 \qquad X' = 1.2.2$

$D \subset \mathcal{E}_0(A)$

$E \subset \mathcal{E}_{1.2.2}(A) = \{0,1,2\}$ Alors, le point de dépendance est $\{(1.2,A)\}$

δ) Deux sous-arborescences étrangères sont dites permutables si et seulement si :

$\exists A \in \{A\}, B \in \{B\}, C \in \{C\}, f, f', f'' \in \mathcal{F}, X, X' \in A$

tel que :

- $B \subset \psi_f(\mathcal{E}_X(A))$
- $C \subset \psi_{f'}(\mathcal{E}_{X'}(A))$
- $\{X.\psi_f^{-1}(B)\} \wedge \{X'.\psi_f^{-1}(C)\} = \emptyset$
- $X \not\leqslant X' \wedge X' \not\leqslant X$
- $\psi_{f'}(\mathcal{E}_{X'}(A)) = \psi_{f''}(\psi_f(\mathcal{E}_X(A)))$

(ou : $\psi_{f'}(\mathcal{E}_{X'}(A)) \sim \psi_f(\mathcal{E}_X(A))$)

6 - ENSEMBLE DE SOUS-ARBORESCENCES

La généralisation des sous-arborescences étrangères est immédiate.

α) Un ensemble $\{A_1, \ldots, A_n\}$ de sous-arborescences d'une arborescence {A} est un sous-ensemble étranger si et seulement si :

$\exists A \in \{A\}, A_i \in \{A_i\}, f_i \in \mathcal{F}, X_i \in A, i = 1,\ldots,n$

- $A_i \subset \psi_{f_i}(\mathcal{E}_{X_i}(A))$
- $\forall i,j, i \neq j, \{X_i.\psi_{f_i}^{-1}(A_i)\} \cap \{X_j.\psi_{f_j}^{-1}(A_j)\} = \emptyset$

β) Un sous-ensemble étranger de sous-arborescences est indépendant si il possède la propriété :

$$\exists A \in \{A\},\ A_i \in \{A_i\},\ f_i \in \mathcal{F}_i,\ X_i \in A,\ i = 1,\ldots,n$$

$$- A_i \subset \psi_{f_i}(\mathcal{E}_{X_i}(A))$$

$$- \forall i,j,i \neq j : - \{X_i . \psi_{f_i}^{-1}(A_i)\} \cap \{X_j . \psi_{f_j}^{-1}(A_j)\} = \phi$$

$$- X_i \not\leq X_j \wedge X_j \not\leq X_i$$

γ) Un sous-ensemble de sous-arborescences est complet si et seulement si :

- ce sous-ensemble est étranger et indépendant
- il possède la propriété

$$\exists A \in \{A\},\ A_i \in \{A_i\},\ f_i \in \mathcal{F}_i,\ X_i \in A_i,\ i = 1,\ldots,n$$

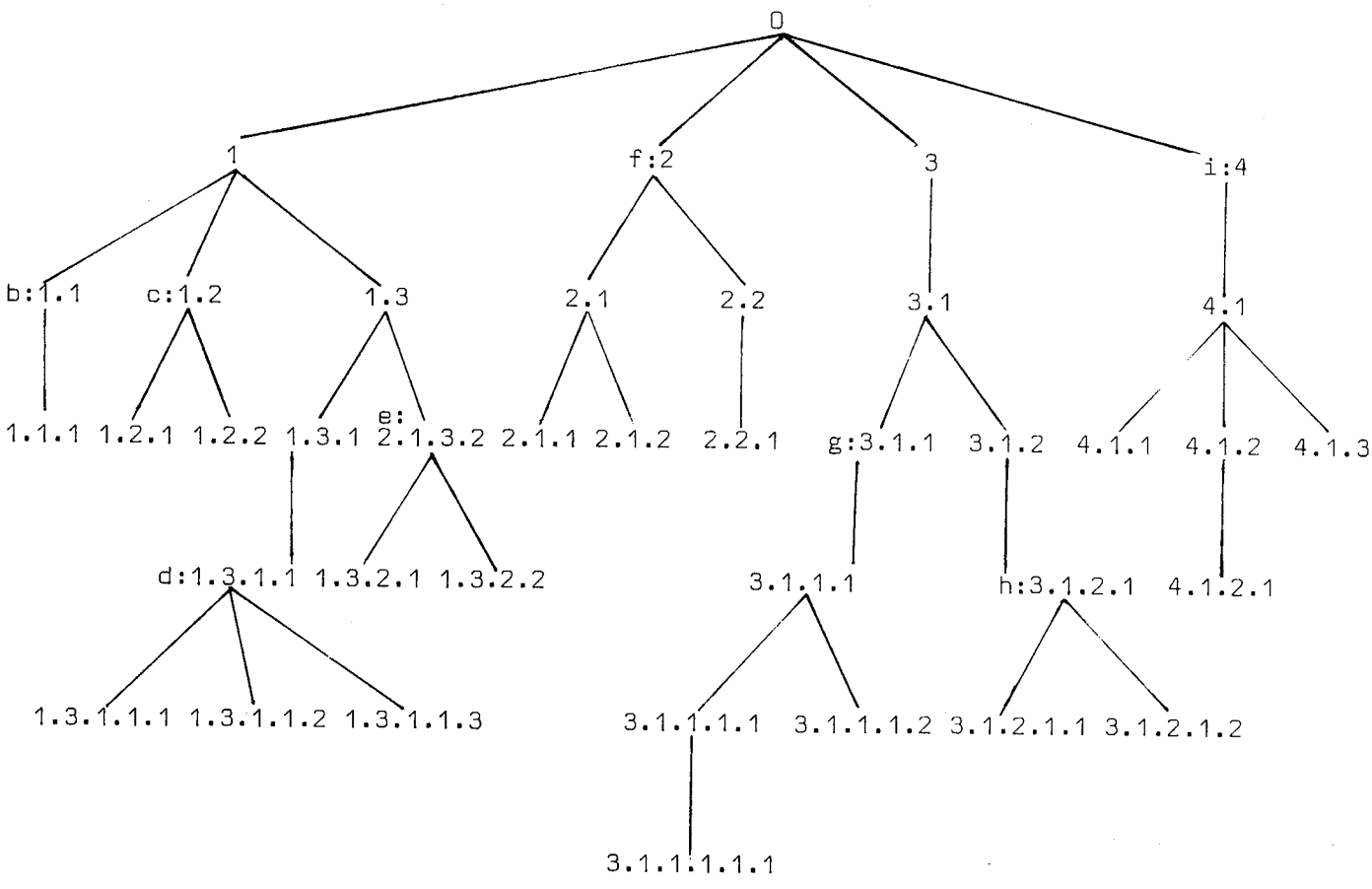
$$- A_i \subset \psi_{f_i}(\mathcal{E}_{X_i}(A))$$

$$- \forall i,j,i \neq j : - \{X_i . \psi_{f_i}^{-1}(A_i)\} \cap \{X_j . \psi_{f_j}^{-1}(A_j)\} = \phi$$

$$- X_i \not\leq X_j \wedge X_j \not\leq X_j$$

$$- \forall\ X \in A : \exists i : (X \leq X_i) \vee (\mathcal{E}_{X_i}(X) \in A_i)$$

Exemple



A = {0,1,1.1,1.1.1,1.2,1.2.1,1.2.2,1.3,1.3.1,1.3.1.1,1.3.1.1.1,
1.3.1.1.2,1.3.1.1.3,1.3.2,1.3.2.1,1.3.2.2,2,2.1,2.1.1,2.1.2,
2.2,2.2.1,3,3.1,3.1.1,3.1.1.1,3.1.1.1.1,3.1.1.1.1.1,3.1.1.1.2,
3.1.2,3.1.2.1,3.1.2.1.1,3.1.2.1.2,4,4.1,4.1.1,4.1.2,4.1.2.1,
4.1.3}

L'ensemble des sous-arborescences {B}, {C}, {D}, {E}, {F}, {G}, {H}, {I} réalise un sous-ensemble complet.

B = {0,1} D = {0,1,2,3} F = {0,1,1.1,1.2,2,2.1}

C = {0,1,2} E = {0,1,2}

G = {0,1,1.1,1.1.1,1.2} I = {0,1,1.1,1.2,1.2.1,1.3}

H = {0,1,2}

Sur la représentation de A, les points X_i sont notés par les lettres minuscules correspondant à l'arborescence $\{A_i\}$.

D - ARBORESCENCES ORIENTEES

Les arborescences orientées sont des arborescences telles que pour tout point d'une arborescence, l'ensemble de ses descendants directs est totalement ordonné.

1 - DEFINITION

Soit $A \in (\mathcal{P})$, une fonction d'équivalence ψ_f est dite ordonnée sur A (notée ψ_f^A) si et seulement si :

$$\forall\ W, W.i, W.j \in A,\ f_W(i) \leqslant f_W(j) \Longleftrightarrow i \leqslant j$$

PROPRIETES

a) $\forall\ f \in \mathcal{F}$ il existe au moins un ensemble $A \in \mathcal{P}(u)$ tel que ψ_f^A soit une fonction d'équivalence ordonnée.

b) $\forall\ A \in \mathcal{P}(u)$, ψ_I^A est une fonction d'équivalence ordonnée

c) si ψ_f^A et $\psi_{f'}^{\psi_f(A)}$ sont deux fonctions d'équivalence ordonnées, alors $(\psi_{f'} \circ \psi_f)^A$ est une fonction d'équivalence ordonnée sur A

d) si ψ_f^A est une fonction d'équivalence ordonnée sur A, alors elle est ordonnée sur toute partie B de A.

e) si ψ_f^A est une fonction d'équivalence ordonnée sur A, alors ψ_f^{-1} est une fonction d'équivalence ordonnée sur $\psi_f(A)$

f) la relation : $A \approx B \Longleftrightarrow f \in \mathcal{F}$ telle que :

- ψ_f^A soit ordonnée sur A
- $B = \psi_f^A(A)$

est une relation d'équivalence. (Appelée équivalence forte).

DEMONSTRATION

a) évident, il suffit de prendre $A = \{0\}$

b) par définition $I_W(i) = i \quad \forall\, W \in u,\ i \in N^+$

c) on a l'équivalence :

$$W.i \in A \iff \psi_f(W).f_W(i) \in \psi_f(A)$$

donc la propriété :

$$\forall\, W,\ W.i,\ W.j \in A,\ f_W(i) \leqslant f_W(j) \iff i \leqslant j$$

entraîne :

$$W,\ W.i,\ W.j \in A \iff \psi_f(W),\ \psi_f(W).f_W(i),\ \psi_f(W).f_W(j) \in \psi_f(A)$$

$$\implies f'_{\psi_f(W)}(f_W(i)) \leqslant f'_{\psi_f(W)}(f_W(j)) \iff f_W(i) \leqslant f_W(j) \iff i \leqslant j$$

d) évident

e) les deux équivalences :

- $W,\ W.i,\ W.j \in A \implies (f_W(i) \leqslant f_W(j) \iff i \leqslant j)$

- $W,\ W.i,\ W.j \in A \iff \psi_f(W),\ \psi_f(W).f_W(i),\ \psi_f(W).f_W(j) \in \psi_f(A)$

entraînent :

$$\forall\, \psi_f(W),\ \psi_f(W).f_W(i),\ \psi_f(W).f_W(j) \in \psi_f(A),$$

$$i = f^{-1}_{\psi_f^{-1}(\psi_f(W))}(f_W(i)) \leqslant f^{-1}_{\psi_f^{-1}(\psi_f(W))}(f_W(j)) = j$$

$$\iff f_W(i) \leqslant f_W(j)$$

f) la relation est une relation d'équivalence :

Réflexive : ψ_I *est une fonction d'équivalence ordonnée pour tout* A.

Symétrie : $A \sim B \iff B = \psi_f(A)$. ψ_f^{-1} *est une fonction d'équivalence ordonnée sur* $\psi_f(A)$, *donc sur* B, *et* $\psi_f^{-1}(B) = \psi_f^{-1}(\psi_f(A)) = A \implies B \sim A$.

Transitive

$A \sim B \iff B = \psi_f(A)$

$B \sim C \iff C = \psi_{f'}(B)$

Mais, ψ'_f est une fonction d'équivalence ordonnée sur B, donc sur $\psi_f(A)$ et $\psi_{f'} \circ \psi_f$ est une fonction d'équivalence $\psi_{f''}$ ordonnée sur A, telle que $C = \psi_{f''}(A)$.

2 - ARBORESCENCE ORIENTEE

Une arborescence orientée est une classe d'équivalence d'élément d'arborescence pour la relation d'équivalence définie ci-dessus. La classe de A sera notée $\{A\}_o$.

Propriété immédiate : $\{A\}_o \subset \{A\}$.

3 - LEMME 6

$\psi_{\xi(A)}$ est une fonction d'équivalence ordonnée sur A.

DEMONSTRATION

D'après la définition de $\xi(A)$:

$X.i, X.j \in A, i \leqslant j \Longrightarrow \exists h : i \leqslant h \leqslant i$ *et* $\xi(A)(X)(h) = \xi(A)(X)(i) + 1$

Si $j = h \Longrightarrow \xi(A)(X)(i) \leqslant \xi(A)(X)(j)$

Sinon : $\xi(A)(X)(i) \leqslant \xi(A)(X)(j) \iff \xi(A)(X)(h) \leqslant \xi(A)(X)(j)$

Donc, $\psi_{\xi(A)}$ est une fonction ordonnée sur A.

4 - LEMME 7

Il existe un et un seul élément simple d'une arborescence orientée.

DEMONSTRATION

Soit une arborescence orientée $\{A\}_o$, il existe toujours au moins un élément simple, car $A \in \{A\}_o$, $\psi_{\xi(A)}(A) \in \{A\}_o$ et $\psi_{\xi(A)}(A)$ est un élément simple.

Supposons qu'il existe deux éléments simples A et A' de $\{A\}_o$.

$A,\ A' \in \{A\}_o \Longrightarrow \exists f : A' = \psi_f(A)$

Soit $n_W = max\{h \mid W.h \in A\}$

A' étant un élément simple, ψ_f *une injection, il existe exactement* n_W *éléments de préfixe* $\psi_f(W)$ *dans A'. Donc* f_W *est une permutation de* N^+ *tel que :*

$i \leqslant j \leqslant h \Longrightarrow f_W(i) \leqslant f_W(j)$

et $i \leqslant h \Longrightarrow f_W(i) \leqslant h$

donc $f_W(i) = I_W(i)\ \forall\ i \leqslant h$

et $A' = A$

5 - LEMME 8

Soit $A \in \mathfrak{P}(u)$ et ψ_f^A une fonction d'équivalence ordonnée sur A, alors $\mathcal{E}_X(A)$ est fortement équivalent à $\mathcal{E}_{\psi_f(X)}(\psi_f(A))$.

DEMONSTRATION

On construit f' ordonnée sur $\mathcal{E}_X(A)$ *telle que :*

$$\psi_{f'}(\mathcal{E}_X(A)) = \mathcal{E}_{\psi_f(X)}(\psi_f(A))$$

f' est définie par :

$$f'_W = f_{X.W} \quad \forall\ W \in \mathcal{E}_X(A)$$

$$f'_W = I \quad \forall\ W \in \mathcal{E}_X(A)$$

Les propriétés de la fonction d'effacement impliquent :

$$\psi_{f'}(\mathcal{E}_X(A)) = \mathcal{E}_{\psi_f(X)}(\psi_f(A)) \quad \textit{(Propriété I.B.2)}$$

$\psi_{f'}$ *est ordonnée sur* $\mathcal{E}_X(A)$ *:*

$$W.i \in \mathcal{E}_X(A) \Longleftrightarrow X.W.i \in A$$

$$W.i,\ W.j \in \mathcal{E}_X(A) \wedge i \leqslant j \Longleftrightarrow X.W.i,\ X.W.j \in A \wedge i \leqslant j$$

$$\Longleftrightarrow f_{X.W}(i) \leqslant f_{X.W}(j) \Longleftrightarrow f'_W(i) \leqslant f'_W(j)$$

Comme ψ_f *est ordonnée sur A,* $\psi_{f'}$ *est ordonnée sur* $\mathcal{E}_X(A)$.

6 - FONCTION D'EQUIVALENCE PARTIELLEMENT ORDONNEE

Soit A un élément d'arborescence et A' un sous-ensemble de A. Une fonction d'arborescence est partiellement ordonnée sur A relativement à A' si elle est ordonnée sur A'.

Lorsque l'on considère un sous-ensemble A' d'un élément d'arborescence A comme élément orienté, on considère le couple (A,A') (Not. $A_{A'}$).

La fonction π_0 est définie par $\pi_0(A_{A'}) = A \ \forall (A,A')$.

La relation $A_{A'} \sim B_{B'} \iff \exists f$ tel que :

- $B = \psi_f(A)$
- $B' = \psi_f(A')$
- ψ_f est partiellement ordonné sur A relativement à A'

est une relation d'équivalence sur l'ensemble des couples (A,A') tel que A soit un élément d'arborescence et A' un sous-ensemble de A.

DEMONSTRATION

- *Réflexive : ψ_I est ordonnée sur A, donc partiellement ordonnée sur A relativement à tout sous-ensemble A' de A.*

- *Symétrie : les deux équivalences :*

  - $W, W.i, W.j \in A' \Longrightarrow (f_W(i) \leqslant f_W(j) \iff i \leqslant j)$
  - $W, W.i, W.j \in A' \iff \psi_f(W), \psi_f(W).f_W(i), \psi_f(W).f_W(j) \in A'$

  impliquent la propriété : ψ_f^{-1} est ordonnée sur $\psi_f(A')$

  et donc $A_{A'} \sim B_{B'} \iff \exists f$:
  - $B = \psi_f(A)$
  - $B' = \psi_f(A')$
  - ψ_f *est partiellement ordonnée sur A relativement à A'.*

  donne : $\psi_f^{-1}(B) = \psi_f^{-1}(\psi_f(A)) = A$

  $\psi_f^{-1}(B') = \psi_f^{-1}(\psi_f(A') = A'$

  ψ_f^{-1} est partiellement ordonnée sur B relativement à $\psi_f(A') = B'$
  donc $B \sim A$

- Transitive :

$A_{A'} \sim B_{B'} \Longleftrightarrow B = \psi_f(A),\ B' = \psi_f(A'),$ *ψ_f est partiellement ordonnée sur A relativement à A'*

$B_{B'} \sim C_{C'} \Longleftrightarrow C = \psi_{f'}(A),\ C' = \psi_f(B'),$ *ψ_f est partiellement ordonnée sur B relativement à B'*

Alors, $\psi_{f'} \circ \psi_f$ *est partiellement ordonnée sur A relativement à A'*

et tel que $C = \psi_{f'} \circ \psi_f(A)$

$C' = \psi_{f'} \circ \psi_f(A')$

7 - ARBORESCENCE PARTIELLEMENT ORIENTEE

Une arborescence partiellement orientée est une classe d'équivalence de couple (A,A') pour la relation définie ci-dessus.

Not $\{A_{A'}\}_\circ$

PROPRIETE

$\{A\}_\circ \subset \{\pi_0(A_{A'})\}_\circ \subset \{A\}$.

D - SOUS - ARBORESCENCES ORIENTEES ET PARTIELLEMENT ORIENTEES

1 - SOUS-ARBORESCENCES ORIENTEES

Une arborescence orientée $\{B\}_o$ est une sous-arborescences orientée de l'arborescence orientée $\{A\}_o$ si et seulement si :

$\exists A \in \{A\}_o$, B $\{B\}_o$, $f \in \mathcal{F}$, $X \in A$:

- ψ_f est ordonnée sur B
- $\psi_f(B) \subset \mathcal{E}_X(A)$

Cette définition est cohérente, c'est-à-dire, que si cette propriété est vérifiée pour deux éléments B et A de $\{B\}_o$ et $\{A\}_o$, elle est vérifiée pour tous les éléments B' de $\{B\}_o$ et A' de $\{A\}_o$.

PREUVE

Soit $B' \in \{B\}_o$, $A' \in \{A\}_o$, alors $\exists f_1$, $f_2 \in \mathcal{F}$ telles que :

$B = \psi_{f_1}(B')$ et ψ_{f_1} est ordonnée sur B'

$A = \psi_{f_2}(A')$ et ψ_{f_2} est ordonnée sur A'

D'après le lemme 8, $\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A)) \simeq \mathcal{E}_X(A)$, $\exists \psi_{f_2}$ ordonnée sur $\mathcal{E}_X(A)$ tel que :

$$\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A)) = \psi_{f_3}(\mathcal{E}_X(A))$$

donc

$$\psi_{f_3}(\psi_f(B)) \subset \psi_{f_3}(\mathcal{E}_X(A))$$

$$\psi_{f_3}(\psi_f(B)) \subset \mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A))$$

comme : $B = \psi_{f_1}(B')$:

$$\psi_{f_3}(\psi_f(\psi_{f_1}(B'))) \subset \mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A))$$

Alors, $\psi_{f_3} \circ \psi_f \circ \psi_{f_1}$ est une fonction ordonnée sur B', ψ_f, et :

$X' = \psi_{f_2}(X)$ est un élément de A'

$A' = \psi_{f_2}(A)$

donc $\psi_{f'}(B') \subset \mathcal{E}_X(A')$

CONSEQUENCES

LEMME 9

Une arborescence orientée $\{B\}_\circ$ est une sous-arborescence orientée de l'arborescence orientée $\{A\}_\circ$ si et seulement si :

$$\forall B \in \{B\}_\circ,\ A \in \{A\}_\circ,\ \exists X \in \psi_{\xi(A)}(A),\ f \in \mathcal{F} :$$

- ψ_f est une fonction ordonnée sur $\psi_{\xi(B)}(B)$

- $\psi_f(\psi_{\xi(B)}(B)) \subset \mathcal{E}_X(\psi_{\xi(A)}(A))$

2 - SOUS-ARBORESCENCES PARTIELLEMENT ORIENTEES

L'arborescence partiellement orientée $\{B_{B'}\}_\circ$ est une sous-arborescence partiellement orientée de l'arborescence partiellement orientée $\{A_{A'}\}_\circ$ si et seulement si :

$$\exists A_{A'} \in \{A_{A'}\}_\circ,\ B_{B'} \in \{B_{B'}\}_\circ,\ X \in A,\ f \in \mathcal{F}:$$

- ψ_f est une fonction d'équivalence partiellement ordonnée sur B relativement à B'

- $\psi_f(B) \subset \mathcal{E}_X(A)$

- $\psi_f(B') \subset \mathcal{E}_X(A')$

La définition est cohérente, car si deux éléments $B_{B'}$ et $A_{A'}$ de $\{B_{B'}\}_\circ$ et $\{A_{A'}\}_\circ$ possèdent la propriété, tous les éléments de $\{B_{B'}\}_\circ$ et $\{A_{A'}\}_\circ$ la possèdent.

PREUVE

Soit $B''_{B''} \in \{B_B, \}_\circ$ et $A''_{A''} \in \{A_A, \}_\circ$, $\exists f_1, f_2 \in \mathcal{F}$ telles que :

$B = \psi_{f_1}(B'')$, $B' = \psi_{f_1}(B''')$ et ψ_{f_1} est ordonnée sur B'''

$A = \psi_{f_2}(A'')$, $A' = \psi_{f_2}(A''')$ et ψ_{f_2} est ordonnée sur A'''

Comme $\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A)) \sim \mathcal{E}_X(A)$, $\exists f_3$ ordonnée sur $\mathcal{E}_X(A')$ telle que :

$$\mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A'')) = \psi_{f_3}(\mathcal{E}_X(A)) \text{ et } \mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A''')) = \psi_{f_3}(\mathcal{E}_X(A'))$$

$$\psi_{f_3}(\psi_f(B)) \subset \psi_{f_3}(\mathcal{E}_X(A))$$

et $\psi_{f_3}(\psi_f(B')) \subset \psi_{f_3}(\mathcal{E}_X(A'))$ car $\psi_f(B') \subset \mathcal{E}_X(A')$

Alors : $B = \psi_{f_1}(B'')$ et $B' = \psi_{f_1}(B''')$

donc $\psi_{f_3}(\psi_f(\psi_{f_1}(B''))) \subset \psi_{f_3}(\mathcal{E}_X(A))$

$$\psi_{f_3}(\psi_f(\psi_{f_1}(B'''))) \subset \psi_{f_3}(\mathcal{E}_X(A'))$$

d'où $\psi_{f_3}(\psi_f(\psi_{f_1}(B''))) \subset \mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A''))$

et $\psi_{f_3}(\psi_f(\psi_{f_1}(B'''))) \subset \mathcal{E}_{\psi_{f_2}(X)}(\psi_{f_2}(A'''))$

Alors $\psi_{f'} = \psi_{f_3} \circ \psi_f \circ \psi_{f_1}$, $X' = \psi_{f_2}(X)$ et

$$\psi_{f'}(B'') \subset \mathcal{E}_{X'}(A'')$$

$$\psi_{f'}(B''') \subset \mathcal{E}_{X'}(A''')$$

CONSEQUENCE

LEMME 10

Une arborescence partiellement orientée $\{B_{B'}\}_o$ est une sous-arborescence partiellement orientée de l'arborescence partiellement orientée $\{A_{A'}\}_o$ si et seulement si :

$$\forall\, B_{B'} \in \{B_{B'}\}_o,\ A_{A'} \in \{A_{A'}\}_o,\ \exists X \in \psi_{\xi(A)}(A)\ f \in \mathcal{F} :$$

- ψ_f est ordonnée sur $\psi_{\xi(B)}(B')$

- $\psi_f(\psi_{\xi(B)}(B)) \subset \mathcal{E}_X(\psi_{\xi(A)}(A))$

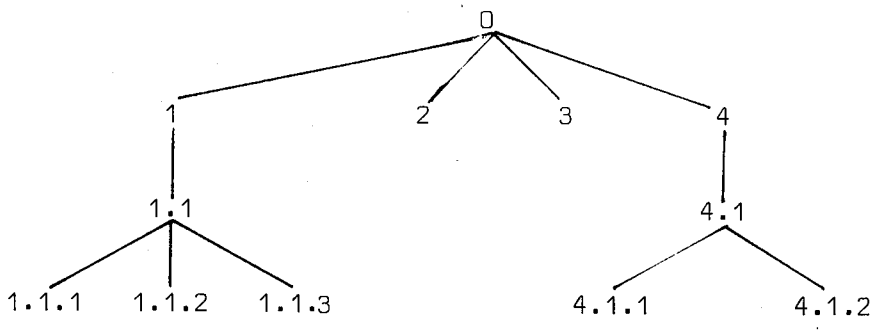
- $\psi_f(\psi_{\xi(B)}(B')) \subset \mathcal{E}_X(\psi_{\xi(A)}(A'))$

3 - EXEMPLE

Soit l'arborescence partiellement orientée $\{A_{A'}\}_o$ définie par

A = {0,1,1.1,1.1.1,1.1.2,1.1.3,2,3,4,4.1,4.1.1,4.1.2}

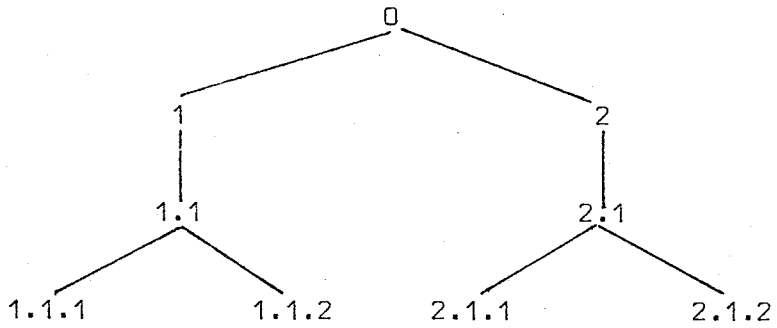
A' = {1,2,3,4}



l'arborescence partiellement orientée $\{B_B,\}_o$ définie par :

B = {0,1,1.1,1.1.1,1.1.2,2,2.1,2.1.1,2.1.2}

B' = {1,2}

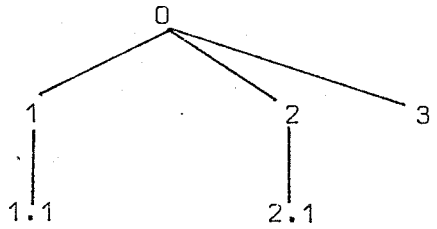


est une sous-arborescence partiellement orientée de $\{A_A,\}_o$.

L'arborescence partiellement orientée $\{C_C,\}_o$ définie par :

C : {0,1,1.1,2,2.1,3}

C' = {1,2,3}



n'est pas une sous-arborescence partiellement orientée de $\{A_A,\}_o$.

E - ARBORESCENCES ETIQUETEES

En traitement informatique, les arborescences ne forment que la structure des informations. Chaque information est alors associée à un point de l'arborescence. Une arborescence étiquetée peut être orientée, partiellement orientée ou quelconque. Une information est attachée à un point de l'élément d'arborescence et donc dépend de lui.

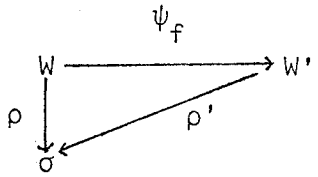
1 - DEFINITION

Soit V un vocabulaire fini. Un élément d'arborescence étiqueté est un couple (A,ρ) où A est un élément d'arborescence et ρ une application de A dans V.

Deux éléments d'arborescences étiquetés (A,ρ) et (A',ρ') sont équivalents si et seulement si :

$$\exists f \in \mathcal{F} : - A' = \psi_f(A)$$

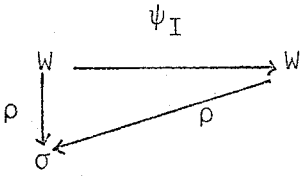
- $\forall W \in A$ le diagramme suivant est commutatif



Cette relation est une classe d'équivalence : il suffit de vérifier la propriété pour la fonction d'étiquetage ρ.

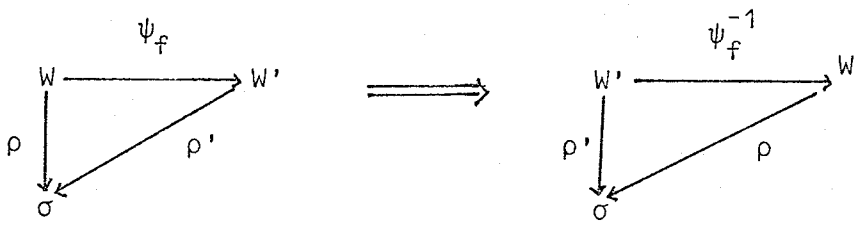
Réflexive :

$$(A,\rho) \sim (A,\rho)$$



Symétrique :

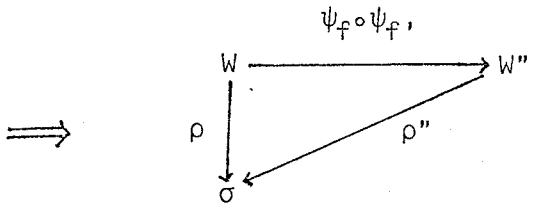
$$(A,\rho)\sim(A',\rho') \Longrightarrow (A',\rho')\sim(A,\rho)$$



Transitive :

$(A,\rho)\sim(A',\rho')$

$(A',\rho')\sim(A'',\rho'')$



Les propriétés de ψ_f n'interfèrent pas avec les propriétés de ρ et donc tous les résultats obtenus sur les arborescences sont maintenus avec les arborescences étiquetées.

DEFINITION

L'ensemble des fonctions d'équivalences est étendu de la façon suivante :

$$\psi_f((A,\rho)) = (A',\rho') \Longleftrightarrow (A,\rho)\sim(A',\rho').$$

DEFINITION

Une arborescence étiquetée est une classe d'équivalence d'élément de la forme (A,ρ) pour la relation définie ci-dessus.

NOTATION

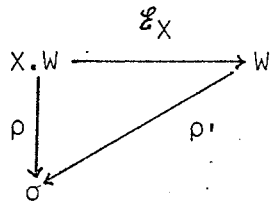
$$\{(A,\rho)\},\{(A,\rho)\}_o,\{(A_{A'},\rho)\}_o$$

2 - SOUS-ARBORESCENCES ETIQUETEES

α) Fonction d'effacement

Soit $\mathcal{E}_X$ une fonction d'effacement définie précédemment. On étend cette fonction aux éléments de la forme (A,ρ) par la définition suivante :

$$\mathcal{E}_X((A,\rho)) = (A',\rho') \iff \begin{cases} - \ \mathcal{E}_X(A) = A' \\ - \text{ le diagramme suivant commute } \forall\ W\in A' \end{cases}$$



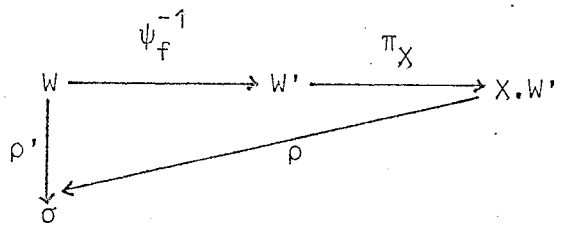
β) Sous-arborescence

Une arborescence étiquetée $\{(B,\rho')\}$ est une sous-arborescence étiquetée de l'arborescence $\{(A,\rho)\}$ si et seulement si :

$$\exists (B,\rho')\in\{(B,\rho')\},(A,\rho)\in\{(A,\rho)\},X\in A,f\in\mathcal{F}:$$

- $B\subset\psi_f(\mathcal{E}_X(A))$

- le diagramme suivant commute $\forall\ W\in B$.



où π_X est la fonction définie par :

$$\pi_X(W) = X.W \quad \forall\ W\in\mathcal{U}$$

Toutes les propriétés des sous-arborescences orientées, partiellement orientées, ou quelconques, se conservent pour les arborescences étiquetées. En particulier :

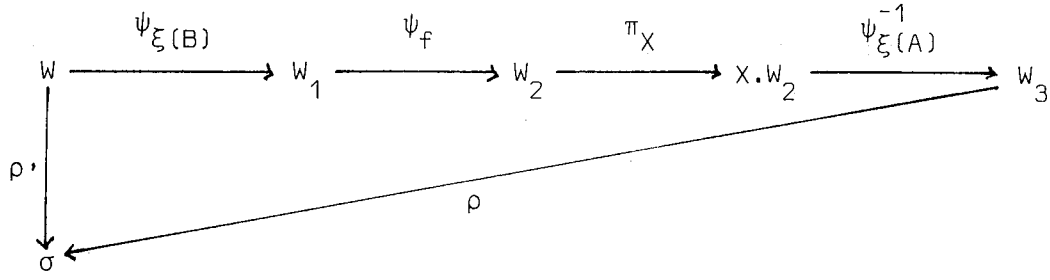
LEMME :

L'arborescence étiquetée $\{(B,\rho')\}$ est une sous-arborescence de l'arborescence étiquetée $\{(A,\rho)\}$ si et seulement si :

$$\forall\ (B,\rho')\in\{(B,\rho)\},(A,\rho)\in\{(A,\rho)\},\exists X\in\psi_{\xi(A)}(A),\ f\in\mathcal{F}\ \text{tel que :}$$

- $\psi_f(\psi_{\xi(B)}(B))\subset\mathcal{E}_X(\psi_{\xi(A)}(A))$

- le diagramme suivant commute pour tout élément W de B

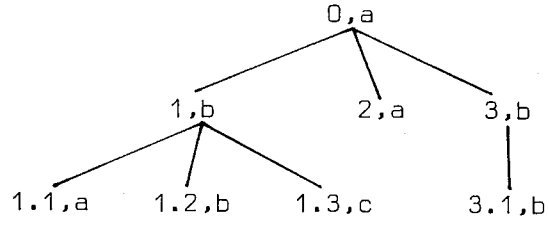


Exemple

Soit V = {a,b,c} et soit l'arborescence étiquetée (A,ρ) définie par :

A = {0,1,2,3,1.1,1.2,1.3,3.1}

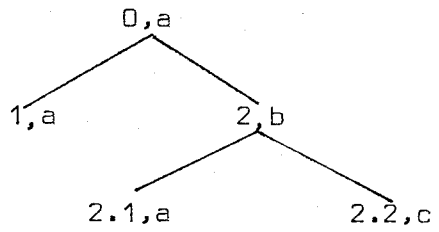
ρ = (0 → a, 1 → b, 2 → a, 3 → b, 1.1 → a, 1.2 → b, 1.3 → c, 3.1 → b)



Alors l'arborescence étiquetée (B,ρ') définie par :

$B = \{0,1,2,2.1,2.2\}$

$\rho' = (0 \to a,\ 1 \to a,\ 2 \to b,\ 2.1 \to a,\ 2.2 \to c)$



est une sous-arborescence étiquetée de l'arborescence étiquetée $\{(A,\rho)\}$

Nous avons : $X = 0$

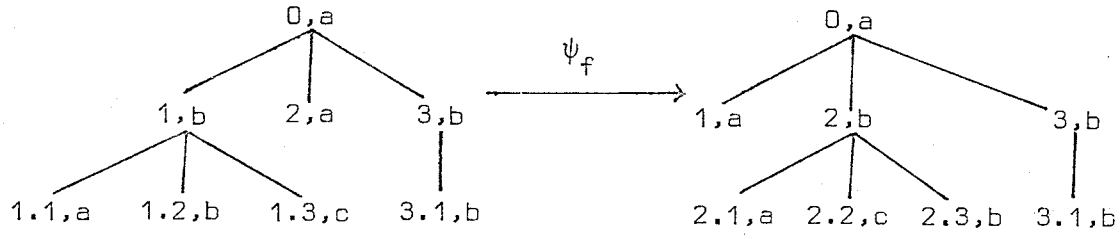
ψ_f définie par :

$f_0 : 1 \to 2,\ 2 \to 1,\ X \to X\ \forall\ X \neq 1,2$

$f_1 : 2 \to 3,\ 3 \to 2,\ X \to X\ \forall\ X \neq 2,3$

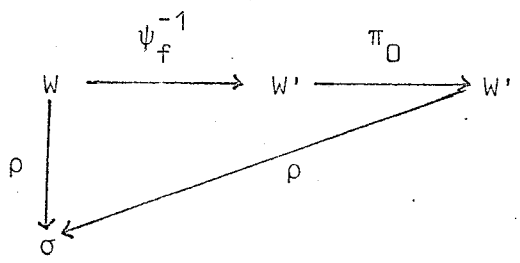
$f_X : I\ \forall\ X \neq 0,1$

$\mathcal{E}_X(A) = A$ et



- $B \subset \psi_f(\mathcal{E}_X(A))$

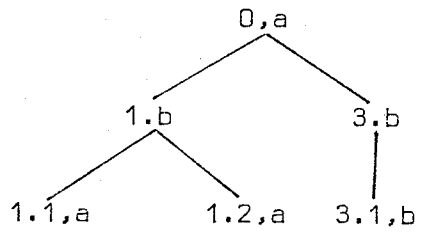
- le diagramme suivant est commutatif $\forall\ W \in B$



L'arborescence étiquetée {(C,ρ")} définie par :

C = {0,1,1.1,1.2,3,3.1}

ρ" = (0 → a, 1 → b, 1.1 → a, 1.2 → a, 3 → b, 3.1 → b)



est telle que :

L'arborescence {C} est une sous-arborescence de l'arborescence {A} .

L'arborescence étiquetée {(C,ρ")} n'est pas une sous-arborescence étiquetée de l'arborescence étiquetée {(A,ρ)}.

REMARQUE

Toutes les définitions des sous-arborescences particulieres, telles qu sous-arborescence sommet, feuille, etc, donnent naissance à des définitions identiques dans le cas des sous-arborescences étiquetées.

CHAPITRE IV

TRANSFORMATIONS

INTRODUCTION

Les précédentes définitions des arborescences et sous-arborescences permettent d'aborder la notion de transformation à partir de la modification d'une sous-arborescence. Les transformations d'arborescences sont définies pour des arborescences non étiquetées. Cette définition a l'avantage de séparer le traitement d'un point de son étiquette, cette confusion conduisant à des transformations sur des alphabets stratifiés (Brainerd [14], Rosen [68], Rounds [68]). La définition d'une transformation élémentaire d'arborescence non orientée est similaire aux transformations définies par Gladky et Melcuk [37]. Enfin, les transformations simultanées de plusieurs sous-arborescences ordonnées permettent d'obtenir un système transformationnel similaire à celui défini par Ginsburg et Partee [34].

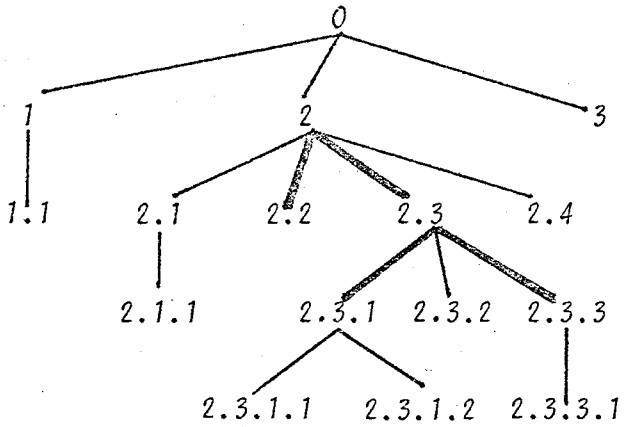
Une sous-arborescence {A} étant localisée dans une arborescence {C}, nous considérons une partition de l'ensemble des points de {C}. Cette partition est définie par rapport aux deux éléments d'arborescence A et C qui permettent de définir la position de cette sous-arborescence, et est constituée par les ensemble suivants :

a) L'ensemble des points qui n'appartiennent pas à la sous-arborescence et ne dépendent pas de ses points. Si X est le mot de l'élément d'arborescence C associée à la racine de la sous-arborescence, cet ensemble est noté $\mathcal{S}_C(x)$.

b) Les points de la sous-arborescence

c) Les points qui n'appartiennent pas à la sous-arborescence et dépendent de sa racine. Ce dernier ensemble est encore divisé en autant de classes qu'il y a de sommets dans la sous-arooorescence. Si un mot W représente un point de cette sous-arborescence, la classe associée à ce point est formée des points dont les mots associés ont ce mot W pour préfixe et n'admettent pas de préfixe plus long appartenant à la sous-arborescence. On appelera $A_A(Y)$ le préfixe et $R_A(Y)$ le suffixe d'un mot Y appartenant à cet ensemble.

Soit, par exemple, les arborescences {A} et {C} définies ci-après. L'arborescence {A} est une sous-arborescence de {C} et caractérise une partition :



C = {0,1,1.1,2,2.1,2.1.1,2.2,2.3,2.3.1,2.3.1.1,2.3.1.2,2.3.2, 2.3.3,2.3.3.1,2.4,3}

A = {0,2,3,3.1,3.3}

Alors, différents ensembles réalisant la partition sont définis par :

- $\mathcal{S}_C(2)$ = {0,1,1.1,3}
- *Points de la sous-arborescence* : {2,2.2,2.3,2.3.1,2.3.3}
- *Points dépendants de la racine de la sous-arborescence :*
  - *ensemble associé à 2* : {2.1,2.1.1,2.4}
  - *ensemble associé à 2.2* : ∅
  - *ensemble associé à 2.3* : {2.3.2}
  - *ensemble associé à 2.3.1* : {2.3.1.1,2.3.1.2}
  - *ensemble associé à 2.3.3* : {2.3.3.1}

Une transformation d'arborescence est définie par deux éléments d'arborescence et une fonction qui applique les points de la première dans l'ensemble des points de la seconde. Cette transformation sera notée (A,B,τ), où A et B sont deux éléments d'arborescence et τ une fonction de A dans B. Une arborescence {C} sera transformée en une arborescence {C'} par cette transformation si l'arborescence {A} est une sous-arborescence de {C}. Alors, la suite des opérations suivantes permet de définir le transformé :

L'ensemble des points de $\mathcal{S}_C(X)$ est inchangé. Ainsi, une transformation laisse inchangée la partie de l'arborescence indépendante de la sous-arborescence reconnue.

L'élément B est substitué à l'élément A.

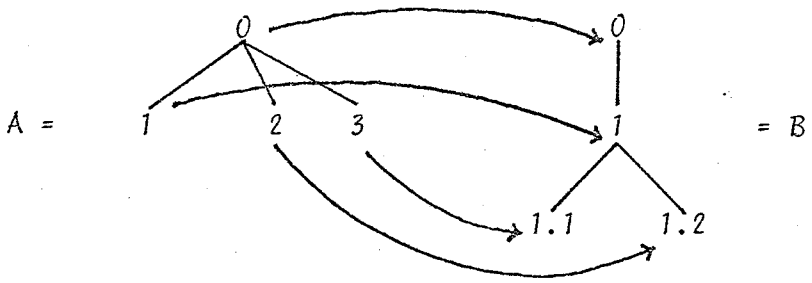
Chaque point de l'arborescence {C} n'appartenant pas à la sous-arborescence {A} est transformé par l'intermédiaire de la fonction τ appliquée au point de référence de la classe de ce point. Ainsi, la fonction τ définit la nouvelle position par rapport à B de l'ensemble des points dépendants d'un point de la sous-arborescence A.

Soit, par exemple, la transformation (A,B,τ) définie par :

$A = \{0,1,2,3\}$

$B = \{0,1,1.1,1.2\}$

$\tau = \{0 \to 0,\ 1 \to 1,\ 2 \to 1.2,\ 3 \to 1.1\}$

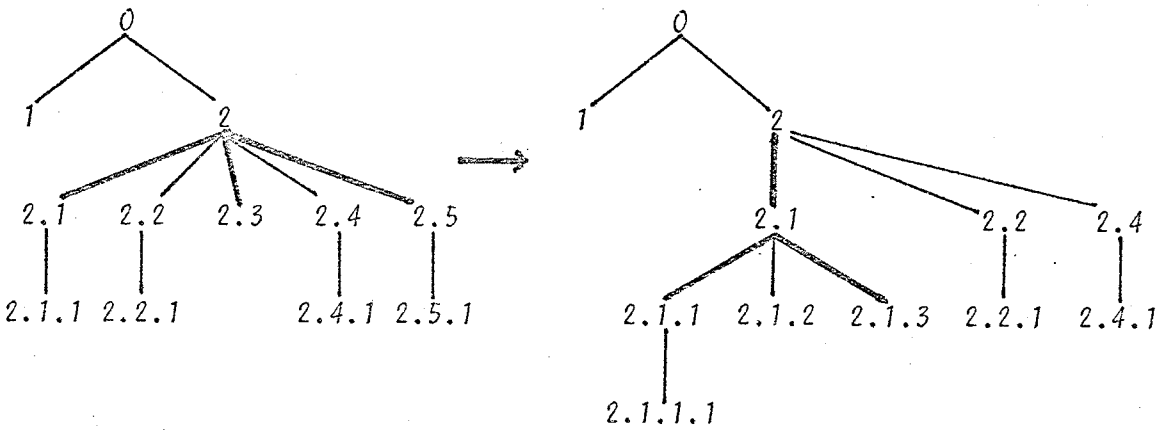


Alors, l'arborescence {C} définie par :

$C = \{0,1,2,2.1,2.1.1,2.2,2.2.1,2.3,2.4,2.4.1,2.5,2.5.1\}$

est transformée en l'arborescence {C'} :

$C' = \{0,1,2,2.1,2.1.1,2.1.1.1,2.1.2,2.1.3,2.2,2.2.1,2.4,2.4.1\}$



Naturellement, avec cette transformation et cette arborescence, le résultat n'est pas unique et dépend du choix de la sous-arborescence. Une notion importante pour l'étude de la décidabilité de l'arrêt d'une suite d'applications de transformations est donnée par la définition de transformation sélective. Dans une transformation sélective, les points de la sous-arborescence source de la transformation doivent appartenir à un ensemble sélectionné. Ce dernier se modifie à chaque pas de transformation.

Lorsqu'une même arborescence comporte plusieurs sous-arborescences étrangères (n'ayant aucun point commun), plusieurs transformations peuvent s'effectuer simultanément. Ainsi, nous généralisons la notion de transformation et les transformations précédentes seront dites élémentaires. La définition des arborescences orientées ou partiellement orientées est similaire à celle des arborescences, aussi leurs transformations se déduisent directement des transformations d'arborescences. Toutefois, une information supplémentaire est apportée par une transformation ordonnée ou partiellement ordonnée. Cette information permet de modifier l'ordre des descendants d'un point de la sous-arborescence source et est constituée par une relation d'ordre. Deux points de l'arborescence qui deviennent dépendants d'un même point de la sous-arborescence résultante devront avoir un ordre compatible avec celui des points associés à leurs classes respectives.

Le cas des transformations d'arborescences étiquetées ajoute au cas précédent le traitement des étiquettes. Une transformation étiquetée comprend pour chaque point de la sous-arborescence source une application de l'ensemble des étiquettes dans lui-même. Ces applications définissent les nouvelles étiquettes de l'arborescence résultante à partir des étiquettes de l'arborescence source. Cette modification d'étiquettes concerne seulement les points appartenant à la sous-arborescence cible. Les autres points conservent la même étiquette au cours de la transformation.

Pour pouvoir s'effectuer, certaines transformations imposent des conditions sur la position des sous-arborescences reconnues. Ces transformations portent le nom de la restriction imposée , transformation sommet, feuille, complète, suivant qu'il s'agit de transformer des sous-arborescences ayant respectivement l'une de ces trois propriétés. Lorsqu'une transformation comprend plusieurs sous-arborescences, la dépendance d'une sous-arborescence par rapport à une autre peut être préalablement définie. Ce type de transformation est nommé transformation hiérarchisée.

La définition de transformation conduit naturellement à la construction de grammaire transformationnelle. Une grammaire transformationnelle est constituée par un ensemble de transformations. L'application d'une grammaire transformationnelle à une arborescence {C} conduit à une arborescence {C'}, si, il existe une séquence finie d'arborescences $\{C_j\}$, dont chacune d'elle est déduite de la précédente par l'application d'une règle de la grammaire, et que les éléments initiaux et finaux de cette séquence sont respectivement {C} et {C'} ; en outre, aucune règle de la grammaire n'est applicable à {C'}. Cette définition est très générale et comprend comme cas particulier les grammaires syntagmatiques. Le prédicat d'arrêt d'une telle grammaire est indécidable.

Afin d'obtenir des grammaires transformationnelles puissantes et décidables, nous apportons les restrictions suivantes :

- Chaque pas d'une transformation applique toutes les transformations accessibles à des sous-arborescences étrangères et indépendantes.

- Dans une grammaire exhaustive, l'application d'une règle ne se fait qu'une seule fois. Le nombre de pas maximum avec une telle grammaire est donc égal au nombre de règles de la grammaire.

- Une restriction moins sévère est obtenue avec les grammaires sélectives définies à partir des règles de transformations sélectives. Une telle grammaire s'arrête si à chaque pas le nombre de sommets susceptibles d'être transformés est décroissant. La différence entre le nombre de sommets accessibles après et avant la transformation est appelée indice caracréristique des règles sélectives.

A l'aide de ces différentes grammaires, on peut naturellement construire des grammaires de reconnaissance. L'ensemble des arborescences reconnu par une grammaire sélective d'indice caractéristique négatif est récursif. Au chapitre VII on donne en exemple la grammaire de reconnaissance du langage $\{a^n\ b^n\ c^n \mid n \geq 1\}$.

A - D E F I N I T I O N S

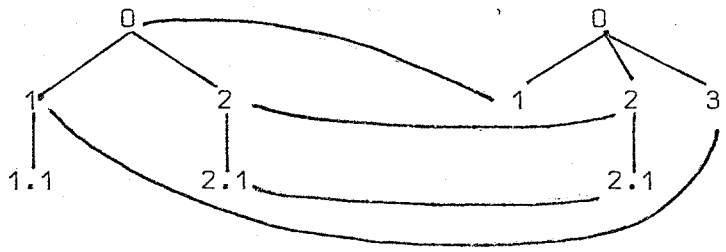
Une transformation d'arborescence est définie par deux arborescences A et B, et d'une fonction d'application τ de l'ensemble des points de l'une dans l'ensemble des points de l'autre. Une arborescence sera transformée en une autre arborescence C' par la transformation définie par (A,B,τ) si A est une sous-arborescence de C. Alors, un éclatement des différents éléments caractéristiques permettra de construire l'élément C'.

Exemple :

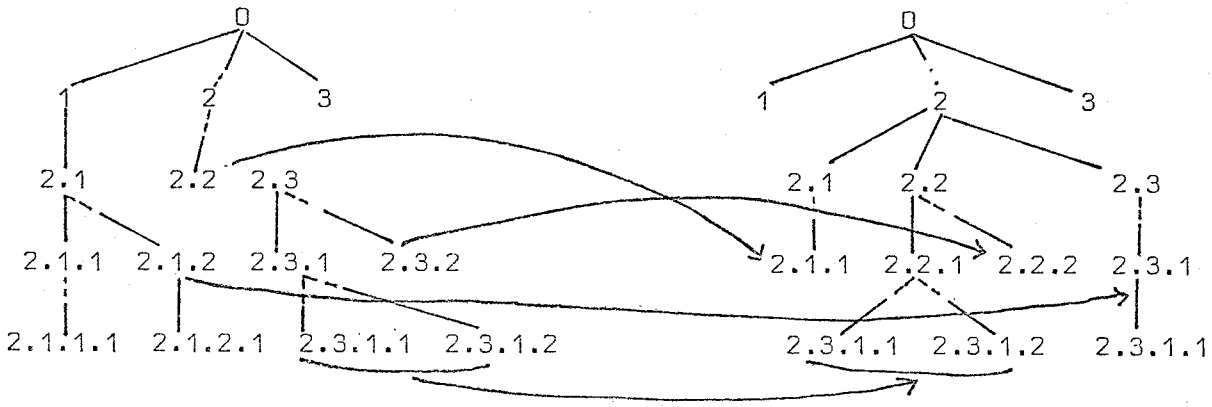
Soit (A,B,τ) la transformation définie par :

$A = \{0,1,1.1,2,2.1\}$, $B = \{0,1,2,2.1,3\}$

$\tau = \{0 \to 1,\ 1 \to 3,\ 2 \to 2,\ 2.1 \to 2.1\}$



L'arborescence $C = \{0,1,1.1,2,2.1\}$ est transformée en l'arborescence $C' = \{0,1,2,2.1,3\}$



Les différents ensembles remarquables d'une transformation sont d'abord définis. L'ensemble des points ne dépendant pas du point racine de la sous-arborescence A de C est appelé sommet. Pour chaque point de la sous-arborescence A, il faut considérer l'ensemble de ces descendants qui n'appartiennent pas à A. Cet élément sera donné par le descendant propre.
Enfin, une arborescence dépendant d'un point sera appelée la descendance de ce point.

1 - FONCTIONS PARTICULIERES

* Ancêtre par rapport à un ensemble donné

Soit X un élément de $\mathcal{U}$ et B une partie de $\mathcal{U}$. L'ancêtre de X par rapport à B défini par :

$$\mathcal{A}_B(X) = Y \Longleftrightarrow Y \leqslant X \wedge Y \in B \wedge (\forall Y' : Y \leqslant Y' \leqslant X \wedge Y' \neq Y \Longrightarrow Y' \notin B)$$

∘ Propriété :

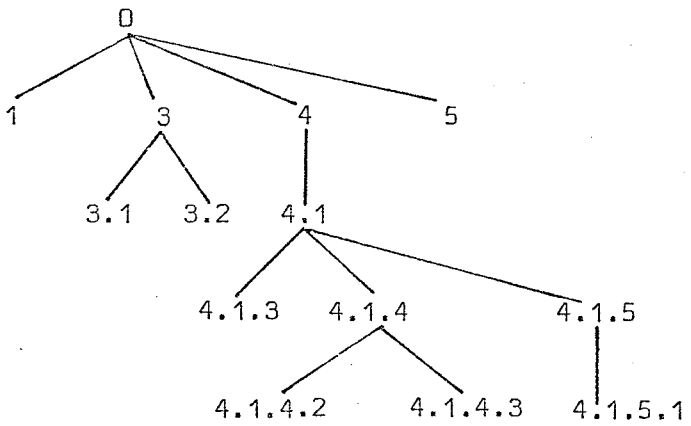
- $\mathcal{A}_B(X)$ est définie pour tout élément X et tout élément d'arborescence B.

- Si $X \in B$, $\mathcal{A}_B(X) = X$

∘ Exemple

Soit A = {0,1,3,3.1,3.2,4,4.1,4.1.3,4.1.4,4.1.4.2,4.1.4.3, 4.1.5,4.1.5.1,5}

B = {4,4.1,4.1.3,4.1.4}



Alors $\mathcal{A}_B(4.1.4.3) = 4.1.4$, $\mathcal{A}_B(4.1.5) = 4.1$, $\mathcal{A}_B(4.1.5.1) = 4.$

* Descendant propre par rapport à un ensemble donné

Soit X un élément de $\mathcal{U}$ et B une partie de $\mathcal{U}$. Le descendant propre de X par rapport à B est défini par :

$$R_B(X) = Y \Longleftrightarrow X = \mathcal{A}_B(X).Y$$

∘ Exemple

Dans l'exemple précédent, nous avons :

$R_B(4.1.4.3) = 3$, $R_B(4.1.5) = 5$, $R_B(4.1.5.1) = 5.1$

* Sommet d'un élément X par rapport à un ensemble donné

Soit X un élément de $\mathcal{U}$ et B une partie de $\mathcal{U}$. Le sommet de X par rapport à B est défini par :

$$S_B(X) = \{y \mid y \in B \wedge X \not\in Y\}$$

∘ Propriété

Si B est un élément d'arborescence et $X \not\in B$: $S_B(X) = B$

∘ Exemple

Dans l'exemple précédent,

$S_A(4) = \{0,1,3,3.1,3.2,5\}$

* Descendance d'un ensemble B par rapport à un élément X

Soit B une partie de $\mathcal{U}$ et X un élément de $\mathcal{U}$. La descendance de B par rapport à X est définie par :

$$\pi_X(B) = \{X.Y \mid y \in B\}$$

∘ Propriété

$\forall X \in \mathcal{U}, B \in \mathcal{P}(\mathcal{U}) : \mathcal{E}_X(\pi_X(B)) = B$

∘ Exemple

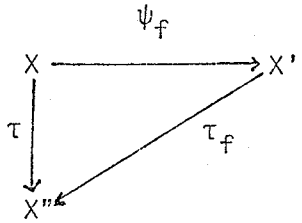
Soit A et B définies précédemment et soit B' définie par :

$B' = \mathcal{E}_4(A)$

Alors $B \subset \pi_4(B')$

* Fonction permutante d'un ensemble A dans un ensemble B

Soit τ une fonction d'une partie A dans une partie B de $\mathcal{U}$. Pour toute fonction $f \in \mathcal{F}$, on appelle fonction permutante par rapport à τ et f la fonction τ_f rendant le diagramme suivant commutatif pour tout élément X de A :



* Descendance extérieure d'un point X par rapport à un ensemble donné A dans un ensemble B

Soit X un élément de $\mathcal{U}$, A et B deux parties de $\mathcal{U}$ tels que : $X \in B \wedge B \subset A$.

La descendance extérieure du point X par rapport à A dans B est définie par :

$$D_B^X(A) = \{y \mid y \in A \wedge \exists i \in N^+ : X.i \leqslant y \wedge X.i \notin B\}$$

∘ Exemple

Soit A et B les éléments définis précédemment, alors :

$D_B^{4.1}(A) = \{4.1.5, 4.1.5.1\}$, $D_B^{4.1.4}(A) = \{4.1.4.2, 4.1.4.3\}$

2 - TRANSFORMATION ELEMENTAIRE D'ELEMENT D'ARBORESCENCE

Une transformation élémentaire d'élément d'arborescence est définie par la donnée de deux éléments d'arborescences A et B d'une fonction τ de A dans B. (Notation : (A,B,τ) ou $A \overset{\tau}{\rightarrow} B$).

Soit C un élément d'arborescence. Il est transformé en un élément d'arborescence C' par la transformation $A \overset{\tau}{\rightarrow} B$ enracinée en X et séparée par f si et seulement si :

- $X \in C \wedge A \subset \mathcal{E}_X(C)$

- $f \in \mathcal{F} \wedge A' = \psi_f(A) \wedge \forall Y,\ Y' \in \psi_f(\mathcal{E}_X(C))$:

  * $R_{A'}(Y) \neq 0 \Longrightarrow B \cap \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\} = \emptyset$

  * $R_{A'}(Y) \neq 0 \wedge R_{A'}(Y') \neq 0 \Longrightarrow$

    $(\tau_f(\mathcal{A}_A(Y)).R_A(Y) = \tau_f(\mathcal{A}_A(Y')).R_A(Y')) \Longrightarrow Y = Y'$

- $C' = S_C(X) \cup \pi_X(B \cup (\bigcup_{Y \in \psi_f(\mathcal{E}_X(C))} \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\}))$

NOTATION :

$$C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{X,f} C'$$

L'élément d'arborescence C est transformé en l'élément d'arborescence C' par les transformations (A,B,τ) si et seulement si :

$$\exists f \in \mathcal{F},\ X \in C : C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{X,f} C'$$

NOTATION :

$$C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{} C'$$

REMARQUE :

Si $C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{} C' \wedge A \neq \mathcal{E}_X(C),\ \exists C'' : C'' \sim C \wedge C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{} C'' \wedge C'' \neq C'.$

PREUVE

Si C est transformé en C' et si ψ_f réalise la séparation de la transformation, il existe $\psi_{f'}$, égale à ψ_f pour tous les éléments de A et de valeurs différentes sur au moins un point de $\mathcal{E}_X(C)$-A qui réalise également la séparation de cette transformation. La transformation $C \vdash\!\!\!\xrightarrow[(A,B,\tau)]{X,f'} C''$ donne l'élément C" cherché.

* Image d'un point de C par la transformation (A,B,τ)

Soit $X' \epsilon C \wedge C' : C \xrightarrow[(A,B,\tau)]{X,f} C'$

$X'_C \xrightarrow[(A,B,\tau)]{} X''_{C'}$, si et seulement si X" est défini par :

$X' \not\leqslant X \Longrightarrow X' = X''$

$X' \leqslant X \wedge X' = X.Y \Longrightarrow X'' = X.\tau_f(A_{\psi_f(A)}(\psi_f(Y)).R_{\psi_f(A)}(\psi_f(Y)))$

NOTATION :

$X''_{C'} = im(X',C,C')$

* Image d'un ensemble de points de C par la transformation (A,B,τ)

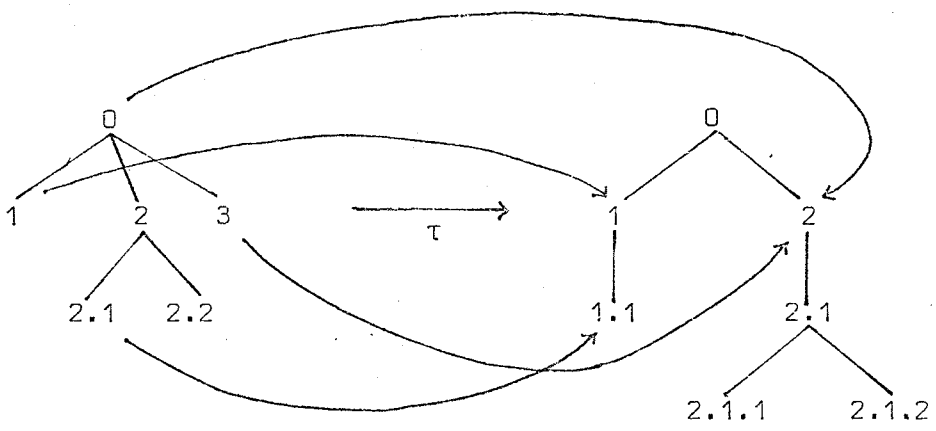
$im(\{X'\}, C, C') = \{im(X', C, C') \mid X' \epsilon \{X'\}\}$

∘ Exemple

Soit la transformation (A,B,τ) définie par :

$A = \{0,1,2,2.1,2.2,3\}$

$B = \{0,1,1.1,2,2.1,2.1.1,2.1.2\}$

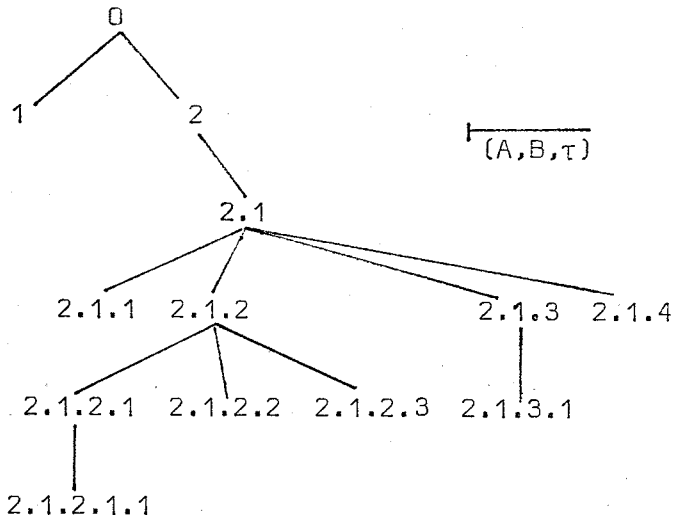


$\tau = \{0 \to 2,\ 1 \to 1,\ 3 \to 2,\ 2.1 \to 1.1\}$

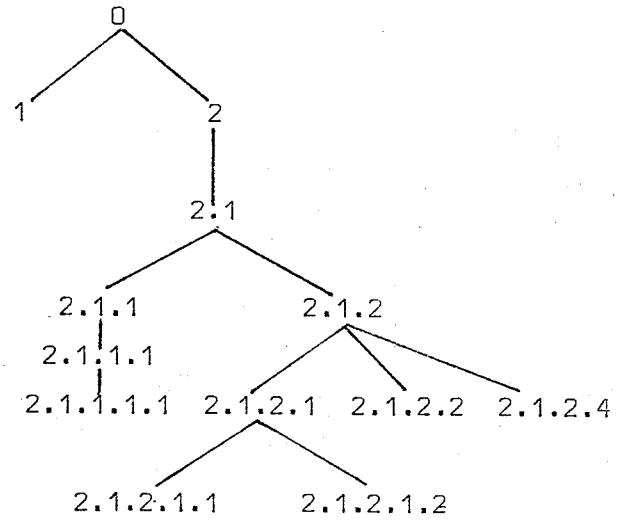
L'élément d'arborescence C est transformé en un élément C', où C et C' sont définis par :

C = {0,1,2,2.1,2.1.1,2.1.2,2.1.2.1,2.1.2.1.1,2.1.2.2,2.1.2.3, 2.1.3,2.1.3.1,2.1.4}

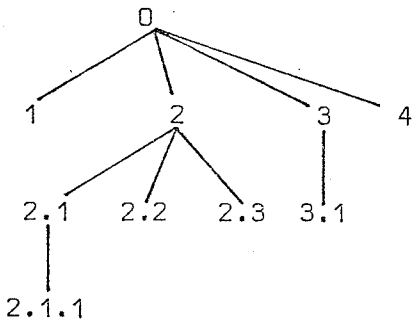
C' = {0,1,2,2.1,2.1.1,2.1.1.1,2.1.1.1.1,2.1.2,2.1.2.1,2.1.2.1.1, 2.1.2.1.2,2.1.2.2,2.1.2.4}



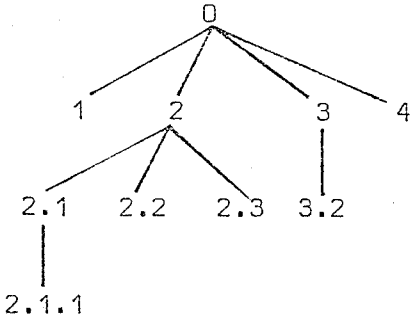
$\vdash_{(A,B,\tau)}$



Où X = 2.1 et $\mathscr{E}_X(C)$ =



$\psi_f(\mathscr{E}_X(C))$ =



3 - TRANSFORMATION ELEMENTAIRE D'ARBORESCENCE

Une transformation élémentaire d'arborescence est définie par une transformation élémentaire d'élément d'arborescence (A,B,τ).

α) Iransformation élémentaire en un point

L'arborescence $\{C\}$ est transformée en l'arborescence $\{C'\}$ par la transformation élémentaire (A,B,τ) appliquée sur le point $\{X_C\}$ si et seulement si :

$$\exists X_C \in \{X_C\},\ C \in \{C\},\ C' \in \{C'\},\ f \in \mathcal{F} : C \underset{(A,B,\tau)}{\overset{X,f}{\longmapsto}} C'$$

$$(\text{Not } \{C\} \underset{(A,B,\tau)}{\overset{\{X_C\}}{\longmapsto}} \{C'\}$$

β) Transformation élémentaire

L'arborescence $\{C\}$ est transformée en l'arborescence $\{C'\}$ par la transformation (A,B,τ) si et seulement si :

$$\exists C \in \{C\},\ C' \in \{C'\} : C \underset{(A,B,\tau)}{\longmapsto} C' \quad (\text{Not} : \{C\} \underset{(A,B,\tau)}{\longmapsto} \{C'\})$$

Ces définitions introduisent la notion de transformations élémentaires équivalentes $(A,B,\tau)_f$ rendant le diagramme suivant commutatif :

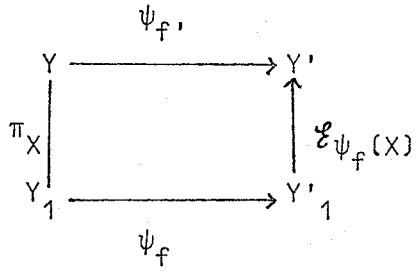
$$\exists \psi_{f'} :\quad \begin{array}{ccc} C_1 & \overset{(A,B,\tau)_f}{\longmapsto} & C'_1 \\ \psi_f \downarrow & & \downarrow \psi_{f'} \\ C & \underset{(A,B,\tau)}{\longmapsto} & C' \end{array}$$

Une arborescence $\{C\}$ est transformée en une arborescence $\{C'\}$ si et seulement si :

$$\forall\, C \in \{C\},\ C' \in \{C'\} : \exists f,\ f' \in \mathcal{F} : C \underset{(A,B,\tau)_f}{\longmapsto} \psi_f^{-1}(C')$$

4 - LEMME

Pour tout élément d'arborescence X de $\mathcal{U}$ et f de $\mathcal{F}$, il existe un élément f' de $\mathcal{F}$ tel que le diagramme suivant soit commutatif pour tout élément Y de $\mathcal{U}$.



DEMONSTRATION

f' est définie par : $f'_W(i) = f_{X.W}(i)$ $\forall$ $W \in \mathcal{U}$, $i \in N^+$.

5 - THEOREME T1

Pour toute arborescence {C} et toute transformation élémentaire (A,B,τ), la condition suivante est nécessaire et suffisante pour que cette transformation soit définie sur {C} :

$$\exists C \in \{C\} : \exists X \in C \wedge A \subset \mathcal{E}_X(C)$$

DEMONSTRATION

* *La propriété est nécessaire de par la définition d'une transformation élémentaire.*

* *La propriété est suffisante :*

Pour compléter la définition, il suffit de construire ψ_f *telle que :*

$$A' = \psi_f(A) \wedge \forall\, Y, Y' \in \psi_f(\mathcal{E}_X(C)) :$$

$$- R_{A'}(Y) \neq 0 \Longrightarrow B \wedge \{\tau_f(\mathcal{A}_{A'}(Y)).R_{A'}(Y)\} = \emptyset$$

$$- R_{A'}(Y) \neq 0 \wedge R_{A'}(Y') \neq 0 \Longrightarrow$$

$$(\tau_f(\mathcal{A}_{A'}(Y)).R_{A'}(Y) = \tau_f(\mathcal{A}_{A'}(Y')).R_{A'}(Y') \Longrightarrow Y = Y')$$

Si $A = \mathcal{E}_X(C)$, *alors* $\psi_f = \psi_I$ *car* $\forall\, Y \in \mathcal{E}_X(C),\ R_A(Y) = 0$

Sinon, soit :

- $k = k' + 1 \wedge k' = max\ \{j \mid X.j \in B\}$
- $h = h' + 1 \wedge h' = max\ \{j \mid X.j \in \mathcal{E}_X(C)\}$
- $Z \in B$ *soit* $\tau^{-1}(Z) = \{y_Z^1, \ldots, y_Z^{n_Z}\}$

Chaque ensemble $\tau^{-1}(Z)$ *ainsi défini est fini et peut être ordonné.*

Soit $\mathcal{O}_Z$ *une application de* $\tau^{-1}(Z)$ *dans* N^+ *possédant la propriété :*

$$Y,\ Y' \in \tau^{-1}(Z) \Longrightarrow \begin{cases} - \ \mathcal{O}_Z(Y) = \mathcal{O}_Z(Y') \Longrightarrow Y = Y' \quad \textit{(injective)} \\ - \ i \leqslant \mathcal{O}_Z(Y) \Longrightarrow \exists Y' \in \tau^{-1}(Z) : \mathcal{O}_Z(Y') = i \end{cases}$$

(surjective sur un segment initial de N^+.

Alors, soit f définie par :

$\forall\, Y.W : R_A(Y.W) = W,\ Y \in \tau^{-1}(Z),\ W \notin N^+ : f_{Y.W} = I$

$\forall\, Y_Z^i \in \mathcal{E}_X(C) \cap \tau^{-1}(Z),\ j \in N^+ :$

∘ *Si* $Y_Z^i.j \in \mathcal{E}_X(C) \wedge Y_Z^i.j \notin A \Longrightarrow f_{Y_Z^i}(j) = Pr(\mathcal{O}_Z(Y_Z^i))^j X \prod_{i=1}^{\mathcal{O}(Y_Z^i)} Pr(kXh)^{kXh}$

Où $Pr(i)$ *représente le ième nombre premier*

∘ *Si* $Y_Z^i.j \notin \mathcal{E}_X(C) \wedge \exists l \in N^+ : j = Pr(\mathcal{O}_Z(Y_Z^i))^l X \prod_{i=1}^{\mathcal{O}(Y_Z^i)} Pr(kXh)^{kXh} \wedge$

$Y_Z^i.l \in \mathcal{E}_X(C) \Longrightarrow f_{Y_Z^i}(j) = l$

∘ *Sinon :* $f_{Y_Z^i}(j) = j$

ψ_f *est définie pour tout élément de* $\mathcal{U}$ *et forme une fonction d'équivalence, et par définition :*

∘ $\psi_f(A) = A$

∘ *chaque* $f_{Y_Z^i}$ *réalise la permutation :*

$$\forall\, j : Y_Z^i.j \in \mathcal{E}_X(C).A \wedge Y_Z^i \in A : Pr(\mathcal{O}_Z(Y_Z^i))^j X \prod_{i=1}^{\mathcal{O}(Y_Z^i)} Pr(kXh)^{kXh} \longleftrightarrow j$$

par définition de h, l'élément ainsi défini n'appartient pas à $\mathcal{E}_X(C)$ *et par définition de k, il ne peut appartenir à B. Donc :*

$$\forall\, Y \in \psi_f(\mathcal{E}_X(C)),\ R_A(Y) \neq 0 \Longrightarrow B \cap \{\tau_f(A_A(Y)).R_A(Y)\} = \emptyset$$

Montrons la propriété :

$$\forall\, Y,\ Y' \in \psi_f(\mathcal{E}_X(C)) : R_A(Y) \neq 0 \wedge R_A(Y') \neq 0 \Longrightarrow$$

$$(\tau_f(A_A(Y)).R_A(Y) = \tau_f(A_A(Y')).R_A(Y') \Longrightarrow Y = Y') :$$

Comme $A_A(Y)$ *et* $A_A(Y')$ *ne peuvent contenir un nombre plus grand que k et que les premiers éléments de* $R_A(Y)$ *et* $R_A(Y')$ *sont des nombres supérieurs à k, nécessairement :*

$$A_A(Y) = A_A(Y') \wedge R_A(Y) = R_A(Y')$$

$R_A(Y)$ *et* $R_A(Y')$ *sont de la forme :*

$$R_A(Y) = Pr(\mathcal{O}_Z(Y_1))^{j} X \prod_{i=1}^{\mathcal{O}_Z(Y_1)} Pr(kXh)^{kXh}.\ W$$

$$R_A(Y') = Pr(\mathcal{O}_Z(Y'_1))^{j_1} X \prod_{i=1}^{\mathcal{O}_Z(Y'_1)} Pr(kXh)^{kXh}.\ W'$$

où $Z = \tau_f(A_A(Y)) = \tau_f(A_A(Y'))$ *et* $Y_1 = A_A(Y)$, $Y'_1 = A_A(Y')$

nécessairement :

$$Pr(\mathcal{O}_Z(Y_1))^{j} X \prod_{i=1}^{\mathcal{O}_Z(Y_1)} Pr(kXh)^{kXh} = Pr(\mathcal{O}_Z(Y'_1))^{j_1} X \prod_{i=1}^{\mathcal{O}_Z(Y'_1)} Pr(KXh)^{kXh}$$

et $W = W'$

l'unicité de la décomposition en facteurs premiers implique :

$$Pr(\mathcal{O}_Z(Y_1))^{j} = Pr(\mathcal{O}_Z(Y'_1))^{j_1} \Longrightarrow Y_1 = Y'_1 \wedge j = j_1 \Longrightarrow$$

$$A_A(Y).jW = A_A(Y').j.W' \Longrightarrow W = W' \wedge Y = Y'$$

6 - DEFINITION : SEPARATION

Soit (A,B,τ) une transformation élémentaire, C un élément de $\mathcal{P}(\mathcal{U})$ et f une fonction de $\mathcal{F}$. f réalise la séparation de C par rapport à (A,B,τ) si et seulement si :

$$Sep((A,B,\tau),C,f) \iff A' = \psi_f(A) \wedge \forall\, Y,Y' \in \psi_f(C) :$$

$$\circ\ R_{A'}(Y) \neq 0 \implies B \cap \{\tau_f(\mathcal{A}_{A'}(Y)).R_{A'}(Y)\} = \emptyset$$

$$\circ\ R_{A'}(Y) \neq 0 \wedge R_{A'}(Y') \neq \emptyset \implies$$

$$(\tau_f(\mathcal{A}_{A'}(Y)).R_{A'}(Y) = \tau_f(\mathcal{A}_{A'}(Y')).R_{A'}(Y')$$

$$\implies Y = Y')$$

9 - TRANSFORMATION ELEMENTAIRE SELECTIVE

Une transformation élémentaire sélective implique la notion de points sélectionnés. Soit $\{C\}$ une arborescence ; un ensemble sélectionné de $\{C\}$ est un sous-ensemble de points de $\{C\}$, $\{\{X_C\}\}$. Une transformation élémentaire sélective est un triplet $(A,B_B,,\tau)$ tel que :

- (A,B,τ) soit une transformation élémentaire
- $B' \subset B$ un ensemble sélectionné de B.

Sur une arborescence $\{C\}$, on distingue deux ensembles de points sélectionnés : l'ensemble sélectionné image et l'ensemble sélectionné source. (Noter respectivement $\{\{X_C\}\}_I$ et $\{\{X_C\}\}_S$).

Une arborescence $\{C\}$ munie de deux ensembles sélectionnés $\{\{X_C\}\}_I$ et $\{\{X_C\}\}_S$ est transformée par la transformation sélective $(A,B_B,,\tau)$ en une arborescence munie des deux ensembles sélectionnés $\{\{X'_C,\}\}_I$ et $\{\{X'_C,\}\}_S$ si et seulement si :

$$\exists C \in \{C\},\ Y \in C,\ C' \in \{C'\},\ f \in \mathcal{F} :$$

$$* \quad C \xrightarrow[(A,B,\tau)]{Y,f} C'$$

$$* \quad \forall\, Y' \in A,\ \exists Y \in \mathcal{U} : YY' \in C \wedge \{YY'_C\} \in \{\{X_C\}\}_S$$

$$* \quad \{\{X'_C\}\}_S = \{\{X'_C, \mid \exists Y' \in B' : C \xrightarrow[(A,B,\tau)]{Y,f} C' \wedge X' = YY'\}\}$$

$$* \quad \{\{X'_C,\}\}_I = \{\{Im(X,C') \mid X \in C \wedge \{X_C\} \in \{\{X_C\}\}^I\}\}$$

NOTATION :

$$\{C\},\ \{\{X_C\}\}_S,\ \{\{X_C\}\}_I \vdash_{(A,B_{B'},\tau)} \{C'\},\ \{\{X'_{C'}\}\}_S,\ \{\{X'_{C'}\}\}_I$$

* Application d'une transformation sélective à une arborescence non munie d'ensembles sélectionnés

Définition

Si $\{C\}$ est une arborescence non munie d'ensembles sélectionnés et $(A,B_{B'},\tau)$ une transformation élémentaire sélective, l'application de cette transformation est définie par :

$$\{C\} \vdash_{(A,B_{B'},\tau)} \{C'\},\ \{\{X'_{C'}\}\}_S,\ \{\{X'_{C'}\}\}_I \iff$$

$$\{C\},\ \{\{X_C \mid X \in C\}\}_S,\ \{\emptyset\}_I \vdash_{(A,B_{B'},\tau)} \{C'\},\ \{\{X'_{C'}\}\}_S,\ \{\{X'_{C'}\}\}_I$$

10 - TRANSFORMATION D'ELEMENT D'ARBORESCENCE

Une transformation d'élément d'arborescence est définie par une suite finie de transformations élémentaires.

NOTATION :

$$\{(A_i,\ B_i,\ \tau^i)\}_{i\in I} \quad \textit{Ou} \ \{A_i \xrightarrow[\tau^i]{} B_i\}_{i\in I}$$

Soit C un élément d'arborescence. Il est transformé en un élément C' par la transformation $\{A_i \xrightarrow[\tau^i]{} B_i\}_{i\in I}$ enracinée en $\{X_i\}_{i\in I}$ et séparée par $\{f_i\}_{i\in I}$ si et seulement si :

- $\forall\ i\in I\ :\ X_i \in C \wedge A_i \subset \mathscr{E}_{X_i}(C)$

- $\forall\ i,j\in I\ :\ i \neq j \Longrightarrow \pi_{X_i}(A_i) \cap \pi_{X_j}(A_j) = \emptyset$

- $\forall\ X_i\ :\ (\forall\ j \neq i,\ j\in I,\ X_i \not\leq X_j),\ \exists C_i\ :$

$$C \vdash_{(A,B_i,\tau^i)} C_i \quad \vdash^{\{X_j,f_j\}_{j\in I-\{i\}}}_{\{(A_j,B_j,\tau^j)\}_{j\in I-\{i\}}} C'$$

NOTATION :

$$C \;\underset{\{(A_i,\, B_i,\, \tau^i)\}_{i\in I}}{\overset{\{X_i,\, f_i\}_{i\in I}}{\longmapsto}}\; C'$$

REMARQUE :

Cette définition est univoque par rapport à un ensemble de points donnés et un élément d'arborescence donné de par la définition des C_i *et de la propriété.*

$\forall\, X_i, X_j : X_i \nleq X_j \wedge X_j \nleq X_i$, *le diagramme suivant est commutatif :*

$$\begin{array}{ccc} C & \underset{(A_i,\, B_i,\, \tau^i)}{\overset{X_i,\, f_i}{\longmapsto}} & C_i \\ (A_j,\, B_j,\, \tau^j) \Big\downarrow X_j,\, f_j & & (A_j,\, B_j,\, \tau^j) \Big\downarrow X_j,\, f_j \\ C_j & \underset{(A_i,\, B_i,\, \tau^i)}{\overset{X_i,\, f_i}{\longmapsto}} & C_{ij} \end{array}$$

Car nous avons :

$$X_i \nleq X_j \Longrightarrow \mathcal{E}_{X_i}(C) = \mathcal{E}_{X_i}(C_j)$$

$$\text{et } X_j \nleq X_i \Longrightarrow \mathcal{E}_{X_j}(C) = \mathcal{E}_{X_j}(C_i)$$

L'élément C est transformé en l'arborescence C' par la transformation $\{(A_i, B_i, \tau^i)\}_{i\in I}$ si et seulement si :

$$\forall\, i\in I,\ \exists f_i \in \mathcal{F},\ X_i \in C : C \;\underset{\{(A_i,\, B_i,\, \tau^i)\}}{\overset{\{X_i,\, f_i\}_{i\in I}}{\longmapsto}}\; C'$$

NOTATION :

$$C \;\underset{\{(A_i,\, B_i,\, \tau^i)\}_{i\in I}}{\longmapsto}\; C'$$

* Image d'un point de l'élément d'arborescence C par la transformation $\{(A_i,B_i,\tau^i)\}_{i\in I}$:

L'image d'un point de l'élément d'arborescence C par la transformation $\{(A_i,B_i,\tau^i)\}_{i\in I}$ est définie récursivement de la façon suivante :

$\forall\ X\in C$:

$$C \vdash\!\!\!\xrightarrow[(A_i,\ B_i,\ \tau^i_j)]{(X_i,\ f_i)} C_i \vdash\!\!\!\xrightarrow[\{(A_j,\ B_j,\ \tau^j)\}_{j\in I-\{i\}}]{\{X_j,\ f_j\}_{j\in I-\{i\}}} C'$$

$$im(X,C,C') = im(im(X,C,C_i),C_i,C')$$

11 - EXEMPLE DE TRANSFORMATION

Soit la transformation définie par :

$$\{(A_1,\ B_1,\ \tau^1),\ (A_2,\ B_2,\ \tau^2)\}$$

$A_1 = \{0,1,2,2.1,2.2,3\}$

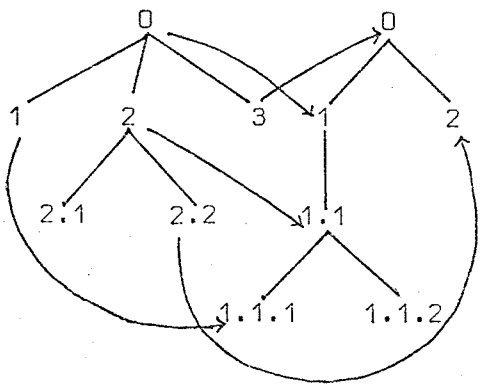
$B_1 = \{0,1,1.1,1.1.1,1.1.2,2\}$

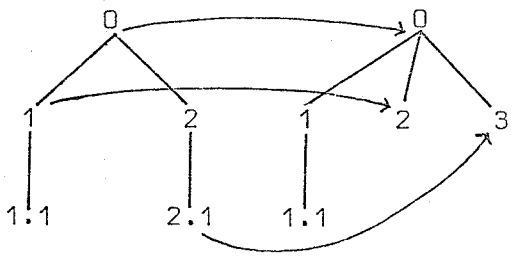
$A_2 = \{0,1,1.1,2,2.1\}$

$B_2 = \{0,1,1.1,2,3\}$

$\tau^1 = \{0 \to 1,\ 1 \to 1.1.1,\ 2 \to 1.1,\ 2.2 \to 2,\ 3 \to 0\}$

$\tau^2 = \{0 \to 0,\ 1 \to 2,\ 2.1 \to 3\}$

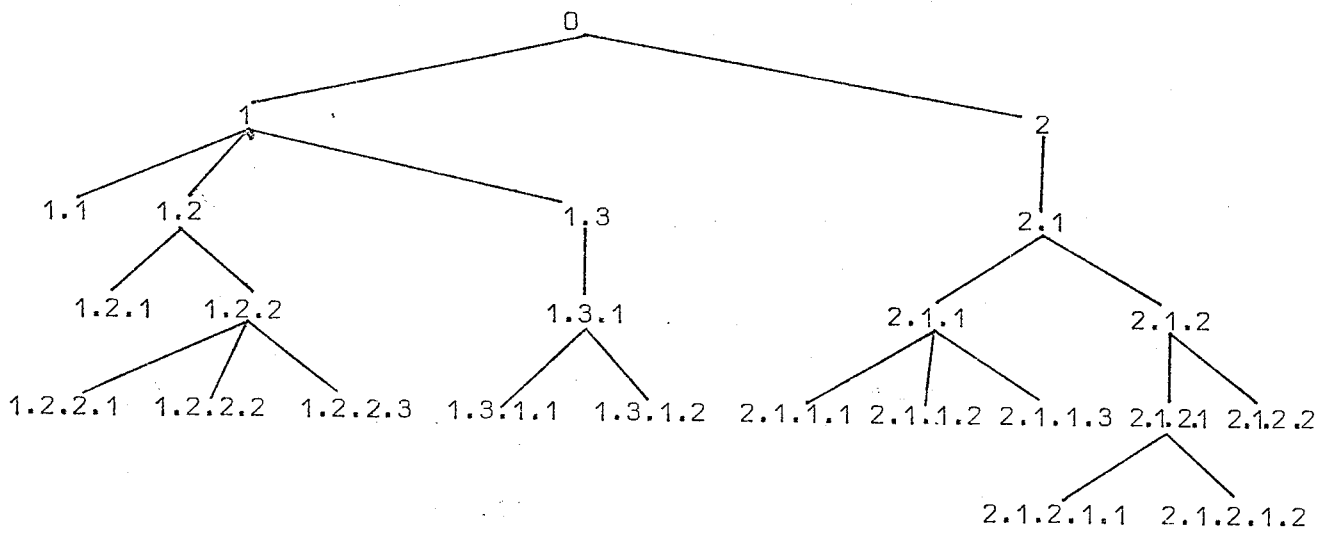




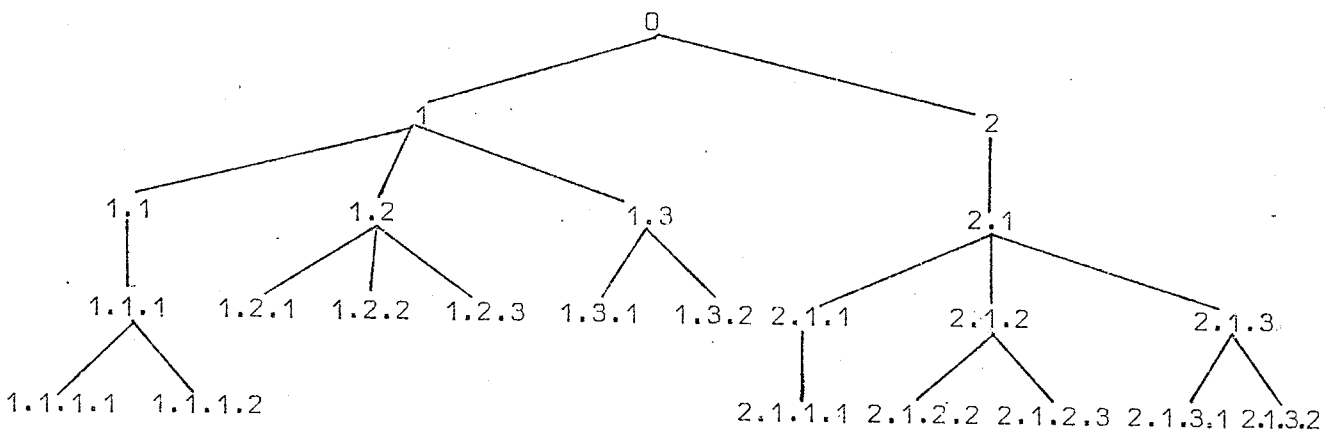
L'élément d'arborescence C est transformé en l'élément d'arborescence C' si et seulement si C et C' sont définis par :

C = {0,1,1.1,1.2,1.2.1,1.2.2,1.2.2.1,1.2.2.2,1.2.2.3,1.3,1.3.1,
1.3.1.1,1.3.1.2,2,2.1,2.1.1,2.1.1.1,2.1.1.2,2.1.1.3,2.1.2,
2.1.2.1,2.1.2.1.1,2.1.2.1.2,2.1.2.2}

C' = {0,1,1.1,1.1.1,1.1.1.1,1.1.1.2,1.2,1.2.1,1.2.2,1.2.3,1.3,
1.3.1,1.3.2,2,2.1,2.1.1,2.1.1.1,2.1.2,2.1.2.2,2.1.2.3,2.1.3,
2.1.3.1,2.1.3.2}

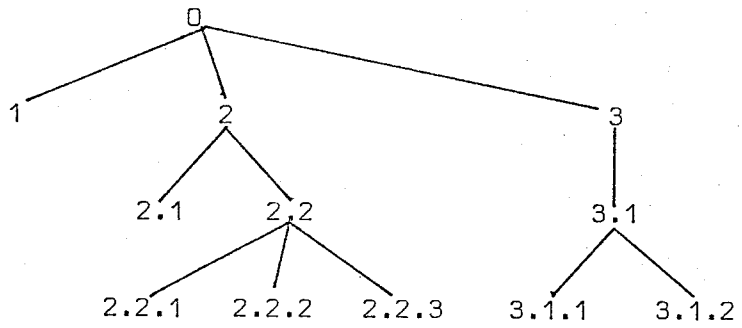


$\{(A_1, B_1, \hat{\tau}^1), (A_2, B_2, \hat{\tau}^2)\}$

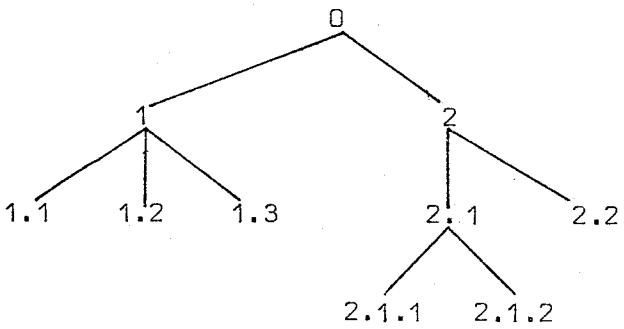


Nous avons alors :

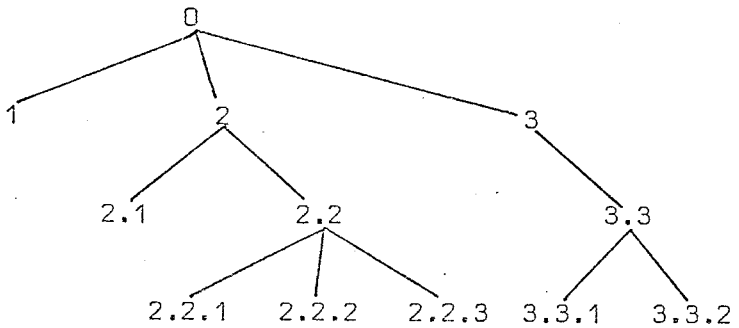
- $X_1 = 1$
- $X_2 = 2.1$
- $\mathcal{E}_{X_1}(C) = \{0,1,2,2.1,2.2,2.2.1,2.2.2,2.2.3,3,3.1,3.1.1,3.1.2\}$



- $\mathcal{E}_{X_2}(C) = \{0,1,1.1,1.2,1.3,2,2.1,2.1.1,2.1.2,2.2\}$



- $\psi_{f_2} = \psi_I$
- $\psi_{f_1} : \psi_{f_1}(\mathcal{E}_{X_1}(C)) = \{0,1,2,2.1,2.2,2.2.1,2.2.2,2.2.3,3,3.3,$
  $3.3.1,3.3.2\}$



12 - TRANSFORMATION D'ARBORESCENCE

Une transformation d'arborescence est définie par une transformation d'élément d'arborescence $\{(A_i,B_i,\tau^i)\}_{i\in I}$.

L'arborescence $\{C\}$ est transformée en l'arborescence $\{C'\}$ par la transformation $\{(A_i,B_i,\tau^i)\}_{i\in I}$ si et seulement si :

$$\exists C\in\{C\},\ C'\in\{C'\} : C \xmapsto[\{(A_i,\ B_i,\ \tau^i)\}_{i\in I}]{} C'$$

$$(\text{Not} : \{C\} \xmapsto[\{(A_i,\ B_i,\ \tau^i)\}_{i\in I}]{} \{C'\}$$

Cette définition introduit la notion de transformation équivalente $\{(A_i,\ B_i,\ \tau^i)\}^f_{i\in I}$ rendant le diagramme suivant commutatif :

$$\begin{array}{ccc} C_1 & \xmapsto[\{(A_i,\ B_i,\ \tau^i)\}^f_{i\in I}]{} & C'_1 \\ \exists\psi_f : \ \psi_f \downarrow & & \downarrow \psi_{f'} \\ C & \xmapsto[\{(A_i,\ B_i,\ \tau_i)\}_{i\in I}]{} & C' \end{array}$$

Nous avons alors la propriété :

L'arborescence $\{C\}$ est transformée en l'arborescence $\{C'\}$ par la transformation $\{(A_i,B_i,\tau^i)\}_{i\in I}$ si et seulement si :

$$\forall\ C\in\{C\},\ C'\in\{C'\},\ \exists f,\ f'\in\mathcal{F}:$$

$$C \xmapsto[\{(A_i,\ B_i,\ \tau^i)\}^f_{i\in I}]{} \psi_f^{-1}(C')$$

Une transformation d'arborescence est sélective si chaque transformation élémentaire qui la compose est sélective.

13 - THEOREME T3

Pour toute arborescence $\{C\}$ et toute transformation $\{(A_i, B_i, \tau^i)\}_{i\in I}$ la condition suivante est nécessaire et suffisante pour que la transformation soit définie sur $\{C\}$.

$$\exists C\in\{C\} : \exists\{X_i\}_{i\in I}\subset C :$$

$$* \ \forall\, i\in I : A_i \subset \mathcal{C}_{X_i}(C)$$

$$* \ \forall\, i,j\in I,\ i\neq j \Longrightarrow \pi_{X_i}(A_i) \cap \pi_{X_j}(A_j) = \emptyset$$

DEMONSTRATION

* *La condition est nécessaire de par la définition de la transformation.*

* *La condition est suffisante pour chaque transformation* (A_i, B_i, τ^i) *de par le théorème T1. De plus, chaque transformation* (A_i, B_i, τ^i) *soit n'altère pas* $\pi_{X_j}(A_j)$, *soit impose que la transformation* (A_j, B_j, τ^j) *ait été effectuée.*

B - TRANSFORMATION ORIENTEE ET PARTIELLEMENT ORIENTEE

Les arborescences orientées étant des cas particuliers des arborescences partiellement orientées, seule la définition de la transformation des arborescences partiellement orientées est donnée.

L'orientation partielle conduit à deux modes de transformations : les transformations respectant l'ordre partiel de départ et les transformations modifiant cet ordre. Le premier mode de transformation est un cas particulier du second mode, le cas où la modification conduit à l'identité.

1 - ORDRE TOTAL ENTRE LES POINTS D'UN ELEMENT D'ARBORESCENCE

Soit C un élément d'arborescence, X et Y deux éléments de C.
La relation $\underset{T}{\leqslant}$ entre X et Y est définie par :

$$X \underset{T}{\leqslant} Y \Longleftrightarrow (X = W.i.W' \wedge Y = W.j.W' \wedge i \leqslant j \wedge (W' = 0 \vee W'' = 0 \vee j \neq i)$$

Propriété

Cette relation est une relation d'ordre total sur tout sous-ensemble C de u.

2 - TRANSFORMATION ELEMENTAIRE PARTIELLEMENT ORDONNEE

a) Transformation d'élément d'arborescence

Une transformation élémentaire d'arborescence partiellement ordonnée est définie par la donnée d'une transformation élémentaire d'élément d'arborescence (A,B,τ) et d'une relation d'ordre total $\leqslant_A$ sur A. (Not $(A,B,\tau, \leqslant_A)$ ou $A \xrightarrow[\tau,\leqslant_A]{} B$).

Soit C un élément d'arborescence et C' une partie de C appelée ensemble de références. C est transformé en l'élément d'arborescence C" avec l'ensemble de référence C"' par la transformation $A \xrightarrow[\tau,\leqslant_A]{} B$ enracinée en X et séparée par f si et seulement si :

- $X \subset C$, $A \subset \mathcal{E}_X(C)$

- $f \in \mathfrak{F}$ * ψ_f est ordonnée sur $\mathcal{E}_X(C')$

  * $\forall\ Y, Y' \in \psi_f(\mathcal{E}_X(C))$

  * $R_A(Y) \neq 0 \Longrightarrow B \wedge \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\} = \emptyset$

  * $R_A(Y) \neq 0 \wedge R_A(Y') \neq \emptyset$

  $\Longrightarrow (\tau_f(\mathcal{A}_A(Y)).R_A(Y) = \tau_f(\mathcal{A}_A(Y')).R_A(Y') \Longrightarrow Y = Y')$

  * $R_A(Y) \neq 0 \wedge R_A(Y') \neq 0$

  $\Longrightarrow (\tau_f(\mathcal{A}_A(Y)) = \tau_f(\mathcal{A}_A(Y')) \Longrightarrow (\mathcal{A}_A(Y) \leqslant_A \mathcal{A}_A(Y')$

  $\Longrightarrow \tau_f(\mathcal{A}_A(Y)).R_A(Y) \leqslant_f \tau_f(\mathcal{A}_A(Y')).R_A(Y')$

- $C'' = S_C(X) \cup \pi_X(B) \cup (\bigcup_{y \in \psi_f(\mathcal{E}_X(C))} \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\}))$

- $C''' = (C' \cap S_C(X)) \cup (\bigcup_{y \in \psi_f(\mathcal{E}_X(C'))} \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\})$

NOTATION :

$$C_{C'} \xmapsto[(A,B,\tau,\leqslant_A)]{X,f} C''_{C'''}$$

L'élément C avec l'ensemble de référence C' est transformé en l'élément C" avec l'ensemble de référence C‴ par la transformation $(A,B,\tau,\leqslant_A)$ si et seulement si :

$$\exists f \in \mathfrak{F},\ \exists X \in C\ :\ C_{C'} \xmapsto[(A,B,\tau,\leqslant_A)]{X,f} C''_{C'''}$$

NOTATION :

$$C_{C'} \xmapsto[(A,B,\tau,\leqslant_A)]{} C''_{C'''}$$

b) Transformation d'arborescence partiellement orientée

L'arborescence partiellement orientée $\{C_{C'}\}_o$ est transformée en l'arborescence partiellement orientée $\{C''_{C'''}\}_o$ par la transformation $(A,B,\tau,\leqslant_A)$ appliquée sur $\{X_C\}_o$ si et seulement si :

$$\exists X_C \in \{X_C\}_o : C_{C'} \in \{C_{C'}\}_o, \exists C''_{C'''} \in \{C''_{C'''}\}_o, \exists f \in \mathcal{F} :$$

$$C_{C'} \xrightarrow[(A,B,\tau,\leqslant_A)]{X,f} C''_{C'''} \quad \text{Not} : (\{C_{C'}\}_o \xrightarrow[(A,B,\tau,\leqslant_A)]{\{X_C\}} \{C''_{C'''}\}_o)$$

L'arborescence partiellement orientée $\{C_{C'}\}_o$ est transformée en l'arborescence partiellement orientée $\{C''_{C'''}\}_o$ par la transformation $(A,B,\tau,\leqslant_A)$ si et seulement si :

$$\exists C_{C'} \in \{C_{C'}\}_o,\ C''_{C'''} \in \{C''_{C'''}\}_o : C_{C'} \xrightarrow[(A,B,\tau,\leqslant_A)]{} C''_{C'''}$$

NOTATION :

$$(\{C_{C'}\}_o \xrightarrow[(A,B,\tau,\leqslant_A)]{} \{C''_{C'''}\}_o)$$

Ces définitions introduisent la notion de transformations équivalentes $(A,B,\tau,\leqslant_A)_f$ rendant le diagramme suivant commutatif.

$$\exists \psi_{f'}^{C'''_1} :$$

$$\begin{array}{ccc} C_{1_{C'_1}} & \xrightarrow{(A,B,\tau,\leqslant_A)_f} & C''_{1_{C'''_1}} \\ \Big\downarrow {\scriptstyle \psi_f^{C'_1}} & & \Big\downarrow {\scriptstyle \psi_{f'}^{C'''_1}} \\ C_{C'} & \xrightarrow[(A,B,\tau,\leqslant_A)]{} & C''_{C'''} \end{array}$$

Une arborescence $\{C_{C'},\}_o$ est transformée en une arborescence $\{C"_{C"'},\}_o$ par la transformation $(A,B,\tau,\leqslant_A)$ si et seulement si :

$$\forall\, C\in\{C_{C'},\}_o,\ C"\in\{C"_{C"'},\}_o\ :\ \exists f,\ f'\in\mathcal{F}\ :\ C_{C'} \xrightarrow[(A,B,\tau,\leqslant_A)_f]{} \psi_{f'}^{C"'^{-1}}(C"_{C"'})$$

3 - THEOREME T'1

Pour toute arborescence partiellement orientée $\{C_{C'},\}_o$ et toute transformation élémentaire partiellement orientée $(A,B,\tau,\leqslant_A)$, la condition suivante est nécessaire et suffisante pour que la transformation soit définie sur $\{C_{C'},\}_o$.

$$\exists C_{C'},\in\{C_{C'},\}_o\ :\ \exists X\in C\ \wedge\ A\subset\mathcal{Y}_X(C)$$

DEMONSTRATION

La démonstration de ce théorème est identique à celle du théorème T1. Seule la fonction d'ordre $\mathcal{O}_Z$ est définie plus restrictivement que dans le premier cas. $\mathcal{O}_Z$ doit posséder la propriété :

$$Y,Y'\in\tau^{-1}(Z) \Longrightarrow (Y\leqslant_A Y' \Longrightarrow \mathcal{O}(Y)\leqslant\mathcal{O}(Y'))$$

4 - DEFINITION : SEPARATION PARTIELLEMENT ORIENTEE

Soit $(A,B,\tau,\leqslant_A)$ une transformation élémentaire, C un élément de $\mathcal{P}(\mathcal{U})$ et f une fonction de $\mathcal{F}$. f réalise la séparation de C par rapport à $(A,B,\tau,\leqslant_A)$ relativement à C' si et seulement si :

$$\begin{aligned}
\mathrm{Sep}((A,B,\tau,\leqslant_A),\ C_{C'},,f) \Longleftrightarrow\ & * \text{ f est ordonnée sur C'}\\
& * \forall\, Y,Y'\in\psi_f(C)\ :\\
& * R_A(Y)\neq 0 \Longrightarrow B\ \wedge\ \{\tau_f(\mathcal{A}_A(Y)).R_A(Y)\} = \emptyset\\
& * R_A(Y)\neq 0\ \wedge\ R_A(Y')\neq 0 \Longrightarrow\\
& \quad (\tau_f(\mathcal{A}_A(Y)).R_A(Y) = \tau_f(\mathcal{A}_A(Y')).R_A(Y') \Rightarrow Y=Y')\\
& * R_A(Y)\neq 0\ \wedge\ R_A(Y')\neq 0 \Longrightarrow\\
& \quad (\tau_f(\mathcal{A}_A(Y)) = \tau_f(\mathcal{A}_A(Y')) \Rightarrow (\mathcal{A}_A(Y) <_A \mathcal{A}_A(Y') \Longrightarrow\\
& \quad \tau_f(\mathcal{A}_A(Y)).R_A(Y) \leqslant \tau_f(\mathcal{A}_A(Y')).R_A(Y')))
\end{aligned}$$

6 - TRANSFORMATION D'ARBORESCENCE PARTIELLEMENT ORIENTEE

Une transformation partiellement orientée d'arborescence partiellement orientée est définie par une transformation d'élément d'arborescence $\{(A_i, B_i, \tau_i, \leqslant_{A_i})\}_{i\in I}$.

a) Transformation partiellement orientée d'élément d'arborescence partiellement orientée

Une transformation d'élément d'arborescence est définie par une suite finie de transformations élémentaires (Not : $\{(A_i, B_i, \tau_i, \leqslant_i)\}_{i\in I}$ ou $\{A_i \xrightarrow[\tau^i, \leqslant_{A_i}]{} B_i\}_{i\in I}$).

Soit C un élément d'arborescence et C' un sous-ensemble de C, l'ensemble de référence.

1 - C est transformé en l'élément d'arborescence C" avec l'ensemble de référence C"' par la transformation $\{A_i \xrightarrow[\tau^i, \leqslant_{A_i}]{} B_i\}_{i\in I}$ enracinée en $\{X_i\}_{i\in I}$ et séparée par $\{f_i\}_{i\in I}$ (Not : $C_C, \vdash\!\!\!\xrightarrow[\{(A_i, B_i, \tau^i, \leqslant_{A_i})\}_{i\in I}]{\{X_i, f_i\}_{i\in I}} C"_{C'''}$ si et seulement si :

- $\forall\, i\in I,\ X_i \in C \wedge A_i \subset \mathscr{E}_{X_i}(C)$
- $\forall\, i,j\in I,\ i\neq j \Longrightarrow \pi_{X_i}(A_i) \cap \pi_{X_j}(Aj) = \emptyset$
- $\forall\, X_i : (\forall\, j \neq i,\ j\in I,\ X_i \not\leqslant X_j),\ \exists C_i$:

$$C_C, \vdash\!\!\!\xrightarrow[(A_i, B_i, \tau^i, \leqslant_{A_i})]{X_i, f_i} C_i \quad C_{C',i} \vdash\!\!\!\xrightarrow[\{(A_j, B_j, \tau^j, \leqslant_{A_j})\}_{j\in I-i}]{\{X_j, f_j\}_{j\in I-i}} C"_{C'''}$$

2 - C est transformé en l'élément d'arborescence C" avec l'ensemble de références C"' par la transformation $\{A_i \xrightarrow[\tau^i, \leqslant_{A_i}]{} B_i\}$ si et seulement si :

$\forall\, i\in I,\ \exists X_i \in C,\ f_i \in \mathfrak{F}$:

$$C_C, \vdash\!\!\!\xrightarrow[\{(A_i, B_i, \tau^i, \leqslant_{A_i})\}_{i\in I}]{\{X_i, f_i\}_{i\in I}} C"_{C'''},$$

b) <u>Transformation partiellement ordonnée d'arborescence partiellement ordonnée</u>

L'arborescence $\{C_{C'}\}_o$ est transformée en l'arborescence $\{C''_{C''}\}_o$ par la transformation $\{(A_i, B_i, \tau^i, \leqslant_{A_i})\}_{i\in I}$ si et seulement si :

$$\exists C_{C'} \in \{C_{C'}\}_o,\ C''_{C''} \in \{C''_{C''}\}_o :$$

$$C_{C'} \xrightarrow[\{(A_i, B_i, \tau^i, \leqslant_{A_i})\}_{i\in I}]{} C''_{C''}\ \ (\text{Not} : \{C_{C'} \xrightarrow[\{(A_i, B_i, \tau_f^i, \leqslant_{A_i})\}_{i\in I}]{} C''_{C''}\})$$

7 - <u>THEOREME T'3</u>

Pour toute arborescence partiellement ordonnée $\{C_{C'}\}_o$ et toute transformation partiellement ordonnée $\{(A_i, B_i, \tau^i, \leqslant_{A_i})\}_{i\in I}$, la condition suivante est nécessaire et suffisante pour que la transformation soit définie sur $\{C_{C'}\}_o$.

$$\exists C_{C'} \in \{C_{C(}\}_o : \exists \{X_i\}_{i\in I} \subset C :$$

$$* \ \forall\ i \in I : A_i \subset \mathcal{E}_{X_i}(C)$$

$$* \ \forall\ i,j \in I,\ i \neq j \Longrightarrow \pi_{X_i}(A_i) \wedge \pi_{X_j}(A_j) = \emptyset$$

C - TRANSFORMATIONS ETIQUETEES

Ces transformations forment la plus grande part des transformations d'arborescences réalisées pratiquement. Elles ne diffèrent des transformations non étiquetées que par les restrictions d'applications dues à la présence d'étiquettes et par la modification éventuelle de ces étiquettes. Il n'est nécessaire de définir que les transformations étiquetées orientées. En effet, l'application d'une transformation orientée à une arborescence non orientée (ensemble de références vides) conduit à la transformation non ordonnée définie par les mêmes éléments. On considère les arborescences étiquetées sur le vocabulaire V.

1 - TRANSFORMATION ELEMENTAIRE ETIQUETEE

Une transformation élémentaire étiquetée est définie par un cortège de la forme : $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ où :

- $(A,B,\tau,\leqslant_A)$ forme une transformation ordonnée.
- ρ_A est une application de A dans V.
- $\{\rho_X\}_{X\in A}$ est une famille d'applications $\rho_X : V \to V$ telles que

  ρ_X ne soit pas définie si et seulement si $\tau(X)$ n'est pas définie

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''},\rho')\}_\circ$ par la transformation élémentaire $(A,B,\tau,\leqslant_A,\rho_A\{\rho_X\}_{X\in A})$ enracinée en $\{X_C\}$ si et seulement si :

$$* \quad \{C_{C'},\}_\circ \vdash \frac{\{X_C\}}{(A,B,\tau,\leqslant_A)} \{C''_{C'''}\}_\circ$$

$$* \quad C_{C'} \vdash \frac{(X,f)}{(A,B,\tau,\leqslant_A)} C''_{C'''} \;,\; (C_{C'},\rho)\in\{(C_{C'},\rho)\}_\circ \;,$$

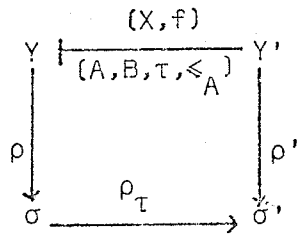
$$(C''_{C'''},\rho')\in\{(C''_{C'''},\rho')\}_\circ \;;$$

$X_C \in \{X_C\} \Longrightarrow$ Si ρ_τ est définie par :

$$\rho_\tau(W) = \rho(W) \quad \forall\ W \notin \pi_X(A)$$

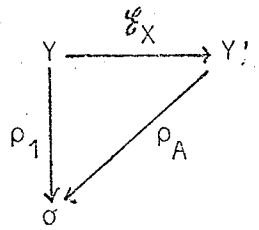
$$\rho_\tau(X.W) = (\rho_W \circ \rho)(X.W) \quad \forall\ X.W \in \pi_X(A)$$

Le diagramme suivant commute pour tout élément Y de C.



$$* \; C_{1_{C'_1}} \xrightarrow[(A,B,\tau,\leqslant_A)]{(X,f)} C''_{1_{C''_1}} \quad \wedge \quad (C_{1_{C'_1}}, \rho_1) \in \{(C_C, \rho)\}_\circ ,$$

$(C''_{C'''_1}, \rho'_1) \in \{(C''_{C'''}, \rho')\}$, alors le diagramme suivant commute pour tout élément Y de $\pi_X(A)$:



NOTATION :

$$\{(C_{C'}, \rho)\}_\circ \xrightarrow[(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})]{\{X_C\}} \{(C''_{C'''}, \rho')\}_\circ$$

Une arborescence partiellement orientée et étiquetée $\{(C_{C'}, \rho')\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''}, \rho')\}_\circ$ par la transformation $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ si et seulement si :

$\exists \{X_C\} \in \{(C_C, \rho)\}_\circ$:

$$\{(C_C, \rho)\}_\circ \xrightarrow[(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})]{\{X_C\}} \{(C''_{C'''}, \rho')\}_\circ$$

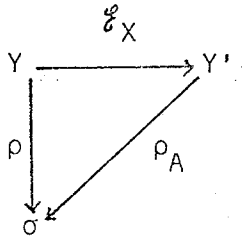
2 - THEOREME T"1

Pour toute arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_o$ et toute transformation élémentaire orientée et étiquetée $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ la condition suivante est nécessaire et suffisante pour que la transformation soit définie sur $\{(C_{C'},\rho)\}_o$.

$\exists (C_{C'},\rho)\in\{(C_{C'},\rho)\}_o$: $\exists X\in C$:

* $A\subset \mathcal{E}_X(C)$

* $\forall\ Y\in\pi_X(A)$, le diagramme suivant commute



DEMONSTRATION

- La condition est nécessaire d'après la définition de la transformation.

- D'après le théorème T'1, il existe $\{C''_{C'''}\}$ tel que :

$$\{C_{C'}\}_o \overset{X_1}{\underset{(A,B,\tau,\leqslant_A)}{\vdash}} \{C''_{C'''}\}_o$$

Soit ρ' définie par :

$$C_{C'} \overset{X_f}{\underset{(A,B,\tau,\leqslant_A)}{\vdash}} C''_{C'''} \text{ alors :}$$

$$W\notin\pi_X(A) \wedge W \overset{(X,f)}{\underset{(A,B,\tau,\leqslant_A)}{\vdash}} W' \Longrightarrow \rho'(W') = \rho(W)$$

$$W\in\pi_X(A) \wedge W \overset{(X,f)}{\underset{(A,B,\tau,\leqslant_A)}{\vdash}} W' \Longrightarrow \rho'(W') = (\rho_{\mathcal{E}_X(W)}\circ\rho)(W)$$

Soit alors $\{(C''_{C'''}, \rho')\}_o$ *l'arborescence partiellement ordonnée et étiquetée.*

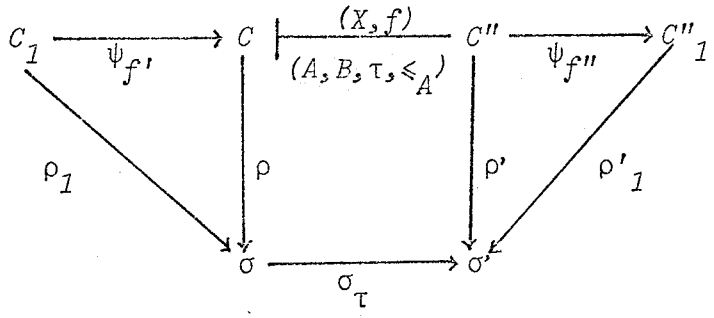
Nous avons :

$$* \quad \{C_{C'}\}_o \overset{\{X_C\}}{\underset{(A,B,\tau,\leqslant_A)}{\vdash\!\!\!\longrightarrow}} \{C''_{C'''}\}_o$$

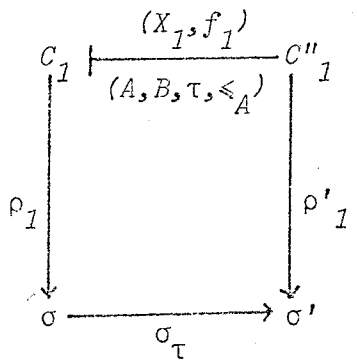
$$* \quad C_{1_{C'_1}} \overset{(X_1,f_1)}{\underset{(A,B,\tau,\leqslant_A)}{\vdash\!\!\!\longrightarrow}} C''_{1_{C'''_1}}, \quad (C_{1_{C'_1}}, \rho_1) \in \{(C_{C'}, \rho)\}_o,$$

$$(C''_{1_{C'''_1}}, C_1) \in \{(C''_{C'''}, \rho')\}_o, \; X_{1_{C_1}} \in \{X_C\} \Longrightarrow$$

$\exists \psi_{f'}, \psi_{f''}$ *: le diagramme suivant commute pour tout élément* Y *de* C_1 *:*



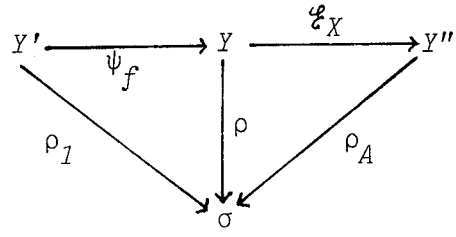
donc le diagramme suivant commute pour tout élément Y *de* C_1 *:*



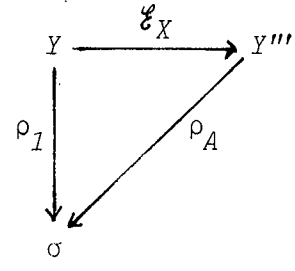
$$* \; C_{1_{C'_1}} \overset{(X,f)}{\underset{(A,B,\tau,\leqslant_A)}{\longmapsto}} C''_{1_{C'''_1}}, \; (C_{1_{C'_1}}, \rho_1) \in \{(C_{C'}, \rho)\}_o, (C''_{1_{C'''_1}}, \rho'_1)$$

$$\in \{(C''_{C'''}, \rho')\}_o \Longrightarrow \exists \psi_f' \text{ tel que le diagramme suivant commute}$$

pour tout élément Y' de $\pi_X(A)$.



donc le diagramme suivant commute pour tout élément Y de $\pi_X(A)$:



3 - TRANSFORMATION ETIQUETEE

Une transformation étiquetée est définie par une suite finie de transformations élémentaires étiquetées : $\{A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i}\}_{i \in I}$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'}, \rho)\}_o$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''}, \rho')\}_o$ par la transformation $\{(A_i, B_i, \tau_i, <_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\}_{i \in I}$ si et seulement si :

$$* \; \{C_{C'}\}_o \overset{}{\underset{\{(A_i,B_i,\tau_i,\leqslant_{A_i})\}_{i\in I}}{\longmapsto}} \{C''_{C'''}\}_o$$

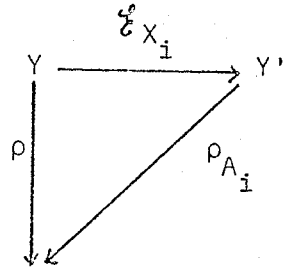
$$* \; (C_{1_{C'_1}}, \rho_1) \in \{(C_{C'}, \rho)\}_o, \; (C''_{1_{C'''_1}}, \rho'_1) \in \{(C''_{C'''}, \rho')\}_o,$$

$$\exists \{X_i\}_{i \in I} : C_{1_{C'_1}} \overset{}{\underset{\{(A_i,B_i,\tau_i,\leqslant_{A_i})\}_{i\in I}}{\longmapsto}} C''_{1_{C'''_1}}$$

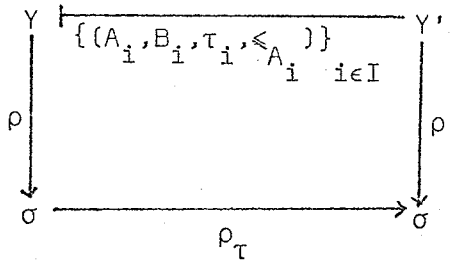
Alors :

Si $\rho_\tau = \overset{0}{_{i\in I}}\ \rho_{\tau_i}$:

α) Le diagramme suivant commute pour tout élément Y de $\pi_X(A_i)$ et pour tout i :



β) Le diagramme suivant commute pour tout élément Y de C_1 :



4 - <u>THEOREME T"</u>$_3$

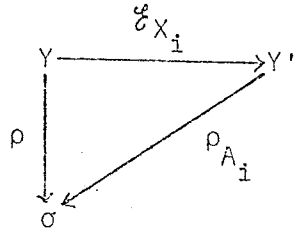
Pour toute arborescence partiellement orientée et étiquetée $\{(C_{C'}, \rho)\}_\circ$ et toute transformation ordonnée étiquetée $\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i\in A_i})_{i\in I}\}$ la condition suivante est nécessaire et suffisante pour que la transformation soit définie sur $\{(C_{C'}, \rho)\}_\circ$.

$\exists (C_{C'}, C) \in \{(C_{C'}, \rho)\}_\circ$: $\exists \{X_i\}_{i\in I} \subset C$:

* $\forall\ i\in I$: $A_i \subset \mathcal{C}_{X_i}(C)$

* $\forall\ i,j\in I$: $i\neq j \Longrightarrow \pi_{X_i}(A_i) \cap \pi_{X_j}(A_j) = \emptyset$

* $\forall\ i$, le diagramme suivant commute pour tout élément Y de $\pi_{X_i}(A_i)$:



D - TRANSFORMATIONS SPECIALES

Les transformations étiquetées définies précédemment sont des transformations très générales. Dans certaines applications, le choix d'une figure supportant la transformation doit être lié à certaines contraintes. C'est le but de ces dernières transformations. Les contraintes apportées sont toutes des contraintes relatives à l'ensemble des sous-arborescences formant le pivot de la transformation. Les premières contraintes portent sur les transformations élémentaires et sont essentiellement des contraintes de position. Ensuite, des contraintes relatives nécessitant une transformation comportant au moins deux sous-arborescences forment la deuxième classe de ces transformations.

1 - TRANSFORMATIONS ELEMENTAIRES SPECIALES

a) Transformation sommet

Une transformation élémentaire sommet et étiquetée est définie par une transformation élémentaire étiquetée $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''},\rho')\}_\circ$ par la transformation sommet $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})_S$ si et seulement si

$$\{(C_{C'},\rho)\}_\circ \overset{\{o_C\}}{\underset{(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})}{\longmapsto}} \{(C''_{C'''},\rho')\}_\circ$$

b) Transformation complète

Une transformation élémentaire complète et étiquetée est définie par une transformation élémentaire étiquetée $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''},\rho')\}_\circ$ par la transformation élémentaire complète $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})_C$ si et seulement si :

$$*\{(C_{C'},\rho)\}_\circ \;\;\underset{(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A}}{\overline{\qquad\qquad\qquad\qquad}}\;\; \{(C''_{C''},\rho')\}_\circ$$

$$* \; X_C \in \{X_C\} \wedge (C_{C'},\rho) \in \{(C_{C'},\rho)\}_\circ \implies$$

A est une sous-arborescence complète de $\mathscr{E}_X(C)$.

c) Transformation feuille

Une transformation élémentaire feuille et étiquetée est définie par une transformation élémentaire étiquetée $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C''},\rho')\}_\circ$ par la transformation élémentaire feuille $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})_f$ si et seulement si :

$$*\{(C_{C'},\ )\}_\circ \;\;\underset{(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})}{\vdash\!\overline{\qquad\qquad\qquad\qquad}}\;\; \{(C''_{C''},\rho')\}_\circ$$

$$* \; X_C \in \{X_C\} \wedge (C_{C'},\rho) \in \{(C_{C'},\rho)\}_\circ \implies$$

A est une sous-arborescence feuille de $\mathscr{E}_X(C)$.

d) Transformation élémentaire à points adjacents

Une transformation élémentaire à points adjacents est définie par un élément de la forme $(A,(B,D),\tau)$.

- (A,B,τ) est une transformation élémentaire
- D est un ensemble de couples d'éléments de B.

Un élément d'arborescence C est transformé en un élément C' par la transformation $(A,(B,D),\tau)$ enracinéé en X et séparée par f si et seulement si :

- $C \;\underset{(A,B,\tau)}{\overset{X,f}{\vdash\!\!\overline{\qquad}}}\; C'$

- $\forall\; W = (X_1,X_2) \in D,\ Adj(C,\pi_X^C(X_1)\pi_X^C(X_2))$

NOTATION :

$$C \vdash^{X,f}_{(A,(B,\mathcal{D}),\tau)} C'$$

Un élément d'arborescence C est transformé en un élément C' par la transformation (A,(B,D),τ) si et seulement si :

$$\exists X\in C,\ f\in\mathfrak{F} : C \vdash^{X,f}_{(A,(B,D),\tau)} C'$$

NOTATION :

$$C \vdash_{(A,(B,\mathcal{D}),\tau)} C'$$

Une transformation élémentaire partiellement orientée, étiquetée, à points adjacents est définie par un élément de la forme $(A,(B,D),\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ où $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ est une transformation élémentaire partiellement orientée et étiquetée et D est un ensemble de couples d'éléments de B.
Son application est déduite naturellement de la combinaison des différentes définitions précédentes.

e) Transformation élémentaire spéciale

Une transformation élémentaire spéciale est une transformation élémentaire possédant une ou plusieurs caractéristiques définies ci-dessus. Une transformation élémentaire spéciale est définie par :

$$(A,B,\tau\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})_\sigma$$

Où : - $(A,B,\tau,\leqslant_A,\rho_A,\{\rho_X\}_{X\in A})$ est une transformation élémentaire

- σ un mot sur {C,f,S,N}
- la transformation possède les propriétés
  - transformation sommet si $\sigma = \sigma_1 S\sigma_2 \quad \sigma_1,\sigma_2\in\{C,f,S,I\}^*$
  - transformation feuille si $\sigma = \sigma_1 f\sigma_2 \quad \sigma_1,\sigma_2\in\{C,f,S,I\}^*$
  - transformation complète si $\sigma = \sigma_1 C\sigma_2 \quad \sigma_1,\sigma_2\in\{C,f,S,I\}^*$

2 - TRANSFORMATIONS SPECIALES

Une transformation spéciale est définie par une suite de transformations élémentaires spéciales $\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\sigma_i\}_{i \in I}$

Une arborescence partiellement orientée et étiquetée $\{(C_{C'}, \rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''}, \rho)\}_\circ$ par la transformation spéciale $\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\sigma_i\}_{i \in I}$ si et seulement si :

$$* \{(C_{C'}, \rho)\}_\circ \vdash\!\!\!\xrightarrow[\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\}_{i \in I}]{} \{(C''_{C'''}, \rho')\}_\circ$$

$$* \{(C_{C'}, \rho)\}_\circ \vdash\!\!\!\xrightarrow[\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\}_{i \in I_1}]{\{\{X_i\}\}_{i \in I_1}} \{(C_{2_{C'_2}}, \rho_1)\}_\circ ,$$

$$\{(C_{1_{C'_1}}, \rho_1)\}_\circ \vdash\!\!\!\xrightarrow[(A_j, B_j, \tau_j, \leqslant_{A_j}, \rho_{A_j}, \{\rho_{X_j}\}_{X_j \in A_j})]{\{X_j\}} \{(C_{2_{C'_2}}, \rho_2)\}_\circ ,$$

$$\{(C_{2_{C'_2}}, \rho_2)\}_\circ \vdash\!\!\!\xrightarrow[\{(A_i, B_i, \tau_i, \leqslant_{A_i}, \rho_{A_i}, \{\rho_{X_i}\}_{X_i \in A_i})\}_{i \in I_2}]{\{\{X_i\}\}_{i \in I_2}} \{(C''_{C'''}, \rho')\}_\circ ,$$

$$I = I_1 \cup I_2 \cup \{j\},\ I_1 \cap I_2 = \emptyset,\ j\ I_1 \cup I_2$$

implique :

$$\{(C_{1_{C'_1}}, \rho_1)\}_\circ \vdash\!\!\!\xrightarrow[(A_j, B_j, \tau_j, \leqslant_{A_j}, \rho_{A_j}, \{\rho_{X_j}\}_{X_j \in A_j})\sigma_j]{\{X_j\}} \{(C_{2_{C'_2}}, \rho_2)\}_\circ .$$

Donc, une transformation spéciale peut être effectuée si chaque transformation élémentaire la composant est définie.

Propriété

Pour qu'une transformation spéciale soit définie, en plus une transformation élémentaire peut être une transformation sommet.

3 - TRANSFORMATIONS RELATIVES

Une transformation $\{(A_i,B_i,\tau_i)\}_{i\in I}$ ne peut être définie sur l'arborescence $\{C\}$ que si les arborescences $\{\{A_i\}\}_{i\in I}$ sont des sous-arborescences étrangères de $\{C\}$. Les transformations relatives imposent en plus à cet ensemble d'arborescences d'être soit indépendant, soit complet, soit d'avoir une hiérarchie bien définie.

a) Transformation indépendante

Une transformation orientée, étiquetée et indépendante $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}^I_{i\in I}$ est définie par une transformation orientée et étiquetée $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},C)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''},\rho')\}_\circ$ par la transformation indépendante $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\}^I_{i\in I}$ si et seulement si :

$$\exists\{\{X^i_C\}\}_{i\in I}\ :$$

$$* \{(C_{C'},\ )_\circ \vdash^{\{\{X^i_C\}\}_{i\in I}}_{\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}} \{(C''_{C'''},C')\}_\circ$$

$$* \{X^i_C\}_{i\in I}\in\{\{X^i_C\}\}_{i\in I} \Longrightarrow (\forall\ i,j\in I,\ i\neq j \Longrightarrow X_i\not\leqslant X_j \wedge X_j\not\leqslant X_i)$$

b) Transformation complète

Une transformation orientée étiquetée et complète $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}^C_{i\in I}$ est définie par une transformation orientée et étiquetée $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_\circ$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C'''},\rho')\}_\circ$ pour la transformation complète $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}^C_{i\in I}$ si et seulement si :

$$\exists\{\{X^i_C\}\}_{i\in I}$$

$$* \{(C_{C'},\rho)\}_\circ \vdash_{\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}}^{\{\{X^i_C\}\}_{i\in I}} \{(C''_{C'''},\rho')\}_\circ$$

$$* \{X^i_C\}_{i\in I}\in\{\{X^i_C\}\}_{i\in I} \Longrightarrow$$

$$- \forall\, i,j\in I,\ i\neq j \Longrightarrow (X_i\not\leqslant X_j \wedge X_j\not\leqslant X_i)$$

$$- \forall\, X\in C : \exists i\in I : (X\leqslant X_i) \vee \mathcal{E}_{X_i}(X)\in A_i$$

c) Transformation hiérarchisée

Soit $\{A_i\}_{i\in I}$ un ensemble d'éléments d'arborescences. Un ensemble de références hiérarchiques sur cet ensemble, $H_{\{A_i\}_{i\in I}}$, est défini par un ensemble de triplets de la forme (A_i,X_i,A_j) tel que :

$$- (A_i,X_i,A_j)\in H_{\{A_i\}_{i\in I}} \Longrightarrow$$

$$* i,j\in I$$

$$* i\neq j$$

$$* X_i\in A_i$$

$$* (A'_i,X'_i,A'_j)\neq(A_i,X_i,A_j) \wedge (A'_i,X'_i,A'_j)\in H_{\{A_i\}_{i\in I}} \Longrightarrow$$

$$(A_i = A'_i \Longrightarrow A''_j\neq A_j \wedge A'_i = A_j \Longrightarrow A'_j\neq A_i)$$

Une transformation orientée étiquetée et hiérarchisée est définie par un couple de la forme $(\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}, H_{\{A_i\}_{i\in I}})$

où : - $\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}$ est une transformation orientée et étiquetée.

- $H_{\{A_i\}_{i\in I}}$ est un ensemble de références hiérarchiques sur $\{A_i\}_{i\in I}$.

Une arborescence partiellement orientée et étiquetée $\{(C_{C'},\rho)\}_o$ est transformée en une arborescence partiellement orientée et étiquetée $\{(C''_{C''},\rho')\}_o$ pour la transformation hiérarchisée $(\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I},H_{\{A_i\}_{i\in I}})$ si et seulement si :

$$\exists\{\{X_C^i\}\}_{i\in I} :$$

$$* \{(C_{C'},\rho)\}_o \xrightarrow[\{(A_i,B_i,\tau_i,\leqslant_{A_i},\rho_{A_i},\{\rho_{X_i}\}_{X_i\in A_i})\sigma_i\}_{i\in I}]{\{\{X_C^i\}\}_{i\in I}} \{(C''_{C''},\rho')\}_o$$

$$* \{X^i\}_{i\in I}\in\{\{X_C^i\}\}_{i\in I} \Longrightarrow$$

$$- \forall\ (A_i,X'_i,A_j)\in H_{\{A_i\}_{i\in I}} : \pi_{X^i}(X'_i)\leqslant X^j$$

E - GRAMMAIRE TRANSFORMATIONNELLE

1 - DEFINITION

Une grammaire transformationnelle est définie par un ensemble ordonné fini de transformations d'arborescences.

NOTATION :

$$G = \{\{(A_i, B_i, \tau^i)\}_{i \in I_j}\}_{j \in I}$$

Une arborescence {C} est transformée en une arborescence {C'} par la grammaire G si et seulement si :

Il existe une suite finie $j_1, \ldots, j_p$ telle que :

* $\forall h : \{C_h\} \xrightarrow[\{(A_i, B_i, \tau^i)\}_{i \in I_{j_h}}]{} \{C_{h+1}\}$

* $\{C\} = \{C_1\}, \{C'\} = \{C_p\}$

* $\forall l \in I'$: la transformation $\{(A_i, B_i, \tau^i)\}_{i \in I_l}$ n'est pas définie sur $\{C_p\}$.

NOTATION :

$$\{C\} \xrightarrow[G]{*} \{C'\}$$

2 - GRAMMAIRE TRANSFORMATIONNELLE GENERATIVE

Une grammaire transformationnelle générative est définie par un couple de la forme (G,S) où :

G est une grammaire transformationnelle

S est un ensemble fini d'arborescences

L'ensemble des arborescences engendrées par une grammaire transformationnelle générative est défini par L(G,S) :

$$L(G,S) = \{\{A\} \mid \exists \{A'\} \in S : \{A'\} \xrightarrow[G]{*} \{A\}\}$$

3 - GRAMMAIRE TRANSFORMATIONNELLE DE RECONNAISSANCE

α - Prédicat de présence associé à un ensemble d'arborescences $\{\{X_1\},\ldots,\{X_n\}\}$.

Un prédicat de présence associé à un ensemble d'arborescences $\{\{X_1\},\ldots,\{X_n\}\}$ est un prédicat à une place d'argument défini sur l'ensemble des arborescences de la manière suivante :

- $\{\{X_{i_1}\},\ldots,\{X_{i_p}\}\} \subset \{\{X_1\},\ldots,\{X_n\}\} \Longrightarrow$

  $(P(\{A\}) \longleftrightarrow \{\{X_{i_1}\},\ldots,\{X_{i_p}\}\}$ est un ensemble de sous-arborescences de l'arborescence $\{A\})$ est un prédicat de présence sur l'ensemble $\{\{X_1\},\ldots,\{X_n\}\}$.

- $\{\{X_{i_1}\},\ldots,\{X_{i_p}\}\} \subset \{\{X_1\},\ldots,\{X_n\}\} \Longrightarrow$

  $(P(\{A\}) \longleftrightarrow \{\{X_{i_1}\},\ldots,\{X_{i_p}\}\}$ est un ensemble de sous-arborescences particulières de l'arborescence $\{A\})$ est un prédicat de présence sur $\{\{X_1\},\ldots,\{X_n\}\}$.

- Si P_1 et P_2 sont des prédicats de présence sur $\{\{X_1\},\ldots,\{X_n\}\}$ alors le sont aussi les prédicats suivants

  - $P_1 \wedge P_2$
  - $P_1 \vee P_2$
  - $\neg P_1$

β - Grammaire transformationnelle de reconnaissance

Une grammaire transformationnelle de reconnaissance est définie par un couple de la forme (G,P) où :

G est une grammaire transformationnelle

P est un ensemble fini de prédicats de présence définis sur un ensemble fini d'arborescences.

L'ensemble des arborescences acceptées par une grammaire transformationnelle de reconnaissance est définie par L(G,P) :

$$L(G,P) = \{\{A\} \mid \exists \{A'\} : \{A\} \xrightarrow[G]{*} \{A'\} \wedge (\exists P_i \in P : p_i(\{A'\}))\}$$

4 - RELATION ENTRE LES GRAMMAIRES TRANSFORMATIONNELLES ET LES GRAMMAIRES SYNTAGMATIQUES

Dans ce cas nous considérons des arborescences étiquetées et les règles de transformations des différentes grammaires sont ordonnées et étiquetées.

Théorème G1 :

Pour toute grammaire syntagmatique générative, il existe une grammaire transformationnelle générative équivalente.

DEMONSTRATION

Il suffit de considérer chaque chaîne de V^* comme un ensemble de feuilles d'une arborescence dont l'ancêtre est un sommet fictif ω. A chaque règle de la grammaire syntagmatique G correspond alors une transformation image évidente de la grammaire G'. Nous avons alors l'identité suivante :

$$\varphi \xrightarrow[G]{*} \psi \iff \begin{array}{c}\omega \\ | \\ \varphi\end{array} \xrightarrow[G']{*} \begin{array}{c}\omega \\ | \\ \psi\end{array}$$

Théorème G2 :

Pour toute grammaire syntagmatique générative G, il existe une grammaire transformationnelle de reconnaissance G' acceptant L(G').

DEMONSTRATION

Cette grammaire s'obtient de manière similaire à celle donnée précédemment. Nous avons les règles suivantes :

$$\varphi \longrightarrow \psi \in G \implies \begin{array}{ccc}\omega & \longrightarrow & \omega \\ | & & | \\ \varphi & & \psi\end{array} \in G'$$

Seuls les mots W engendrés par G possèdent la propriété :

$$\begin{array}{ccc}\omega & \xrightarrow{*} & \omega \\ | & & | \\ W & G' & S\end{array}$$

Donc, le seul prédicat de présence est celui qui reconnait la sous-arborescence sommet feuille et complète $\begin{array}{c}\omega \\ | \\ S\end{array}$.

Conséquence :

L'arrêt d'une grammaire transformationnelle sur une arborescence donnée est indécidable.

L'utilisation effective de grammaires transformationnelles sur ordinateur implique la nécessité d'obtenir un prédicat d'arrêt décidable.
Les différentes restrictions suivantes permettent d'obtenir des grammaires transformationnelles dont l'arrêt est décidable, c'est-à-dire que pour toute grammaire ainsi définie, le nombre de pas effectué par cette grammaire lorsqu'elle est appliquée à une arborescence quelconque est fini.

5 - GRAMMAIRE UNITAIRE

Une grammaire transformationnelle unitaire est définie par une grammaire transformationnelle G.

Une arborescence {C} est transformée en une arborescence {C'} par une grammaire transformationnelle unitaire G si et seulement si :

$\exists \{(A_i, B_i, \tau^i)\}_{i \in I_j} \in G$ tel que :

$$\{C\} \xrightarrow[\{(A_i, B_i, \tau^i)\}_{i \in I_j}]{} \{C'\}$$

L'application d'une grammaire transformationnelle unitaire ne comporte qu'un pas pour toute arborescence {C}.

6 - GRAMMAIRE EXHAUSTIVE

Une grammaire transformationnelle exhaustive est définie par une grammaire transformationnelle G dont toutes les règles sont étiquetées :

$$G = \{(t_j, \{(A_i, B_i, \tau^i)\}_{i \in I_j})\}_{j \in I}$$

Une arborescence à points marqués relativement à une grammaire exhaustive G est définie par un couple (μ, V) où :

- μ est une arborescence {C} (ou une arborescence étiquetée $\{(C, \rho)\}$)
- V est une arborescence étiquetée sur T $\{(C', \rho')\}$ tel que :

  $\{C\} = \{C'\}$ et $T = \mathcal{P}(\bigcup_{j=\in I} \{t_j\})$

Dans une arborescence à points marqués, chaque point fait référence à un ensemble de règles. Notons ρ_ϕ l'application qui à chaque point associe l'ensemble vide.

L'application d'une règle transformationnelle d'une grammaire exhaustive sur une arborescence à point marqué ne peut s'effectuer que si la racine de cette transformation n'est pas marquée pour cette règle.
De plus, une transformation complète la marque sur les différents points de l'arborescence résultante.

$$(\{C\},\{(C,\rho)\}) \vdash \frac{\{X_i\}_{i\in I_j}}{\{A_i,B_i,\tau^i\}_{i\in I_j}} (\{C'\},\{(C'',\rho')\})$$

si et seulement si :

$$- \{C\} \vdash \frac{\{X_i\}_{i\in I_j}}{\{(A_i,B_i,\tau^i)\}_{i\in I_j}} \{C'\}$$

$$- \forall\, i \in I_j : X_{i_C} \in \{X_i\}, (C,\rho) \in \{(C,\rho)\} \Longrightarrow t_j \notin \rho(X_i)$$

$$-(C'',\rho') \in \{(C'',\rho')\} \Longrightarrow$$

$$* \forall\, X \in C'' : \forall\, i \in I_j : X_i \not\lesssim X : \rho'(X) = \rho(X)$$

$$* \forall\, X \in C'' : \exists\, i \in I_j : X_i \lesssim X : \rho'(X) = \rho(I_m^{-1}(X)) \cup \{t_j\}$$

Un point marqué conserve sa marque par une transformation.
Si ce point est impliqué dans la transformation il reçoit la marque supplémentaire de cette transformation.

Une arborescence $\{C\}$ est transformée en une arborescence $\{C'\}$ par la grammaire transformationnelle exhaustive G si et seulement si :

$$(\{C\},\{(C,\rho_\phi)\}) \xrightarrow[G]{*} (\{C'\},\{(C',\rho)\})$$

Les marques effectuées sur les points ne sont utilisées qu'au cours des transitions ; nous noterons simplement :

$$\{C\} \xrightarrow[G]{*} \{C'\}$$

Propriété

De par sa définition, l'application d'une grammaire exhaustive G sur une arborescence {C} donnée ne peut exéder un nombre de pas fini.
En effet, pour une règle donnée, son application diminue le nombre de points où elle est applicable.

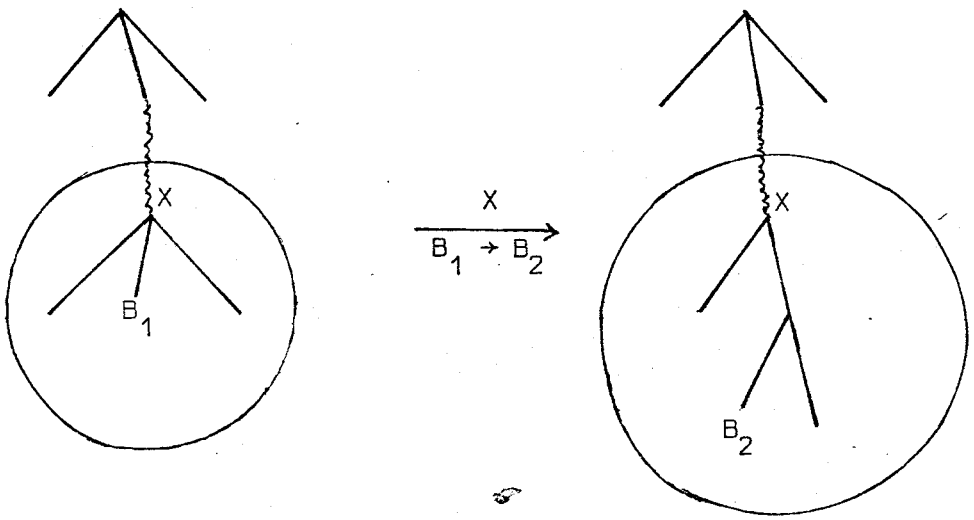
7 - GRAMMAIRE RECURSIVE

α - Transformation simple

La définition originale des transformations de ce type (Rosen [66]) porte uniquement sur des arborescences ordonnées et a pour support l'ensemble des éléments simples d'arborescences. La généralisation a une arborescence quelconque et immédiate car cette définition est considérée sur un représentant de la classe d'équivalence qui forme cette arborescence (ordonnée, non ordonnée, étiquetée, etc...).

Une transformation est définie par un couple d'éléments simples d'arborescences (Notation $B_1 \rightarrow B_2$). Un élément simple d'arborescence A est transformé en un élément simple A' par la transformation $B_1 \rightarrow B_2$ si et seulement si :

$$\exists X \in A : \; - \; \mathcal{E}_X(A) = B_1$$
$$- \; S_X(A) = S_X(A')$$
$$- \; \mathcal{E}_X(A') = B_2$$



NOTATION :

$$A \xrightarrow[B_1 \rightarrow B_2]{X} A'$$

β - Grammaire transformationnelle récursive

Une grammaire transformationnelle récursive est définie par une suite de triplets de la forme (r,σ,p) où, r est une transformation d'arborescences $\{(A_i,B_i,\tau^i)\}_{i\in I}$, σ une suite de points ordonnés $\{\sigma_i\}_{i\in I'}$, et p une application de I' dans I tel que :

$$- \forall i,j\ ,\ i>j\ ,\ p(i) = p(j) \Longrightarrow \sigma_i \lneq \sigma_j$$

Une arborescence {C} est transformée en une arborescence {C'} par une règle (r,σ,p) de la grammaire récursive G si et seulement si :

$$* \{C\} \vdash^{\{X_i\}_{i\in I}}_{r} \{C_1\}$$

$$* \forall\ i \in I' : \{C_i\} \xrightarrow[B_i \to B'_i]{X_{p(i)}\sigma_i} \{C_{i+1}\} \text{ tel que :}$$

$$B_i = \mathcal{E}_{X_{p(i)}\sigma_i}(C_i) \wedge B_i \overset{*}{\underset{G}{\Longrightarrow}} B'_i$$

$$* \{C'\} = \max_{i\in I'} \{C_i\}$$

La restriction apportée sur la suite des points implique que les transformations simples qui définissent la récurrence doivent être effectuées à partir des points les plus bas de l'arborescence transformée.

γ - Propriété

Dans le cas général, une règle récursive peut être applicable et son résultat ne pas être défini. Ceci est notamment le cas où chaque transformation appliquée produit une arborescence sur laquelle plusieurs règles sont applicables. Une condition simple de décidabilité peut être apportée sur les règles récursives.

Théorème G3

L'application d'une règle de toute grammaire récursive dont toutes les règles possèdent la propriété r nécessite un nombre fini de pas.

Une règle d'une grammaire récursive $(\{A_i,B_i,\tau^i\}_{i\in I}, \{\sigma_i\}_{i\in I},p)$ possède la propriété r si et seulement si :

$$\forall\ i \in I' : \mathrm{Card}(\xi_{\sigma_i}(B_{p(i)})) < \mathrm{Card}(A_{p(i)})$$

La démonstration de cette propriété est immédiate du fait que le nombre de points où les règles de la grammaire récursive peuvent s'appliquer diminue après chaque appel récursif.

Une grammaire récursive dont toutes les règles possèdent la propriété r est nommée r-grammaire.

Cette propriété n'implique pas qu'une grammaire récursive transforme une arborescence quelconque en un nombre fini de pas.

Une grammaire récursive peut être unitaire ou exhaustive, d'où la propriété suivante :

Théorème G4

Pour toute r-grammaire unitaire ou exhaustive G, l'ensemble L(G,P) est récursif pour tout ensemble P.

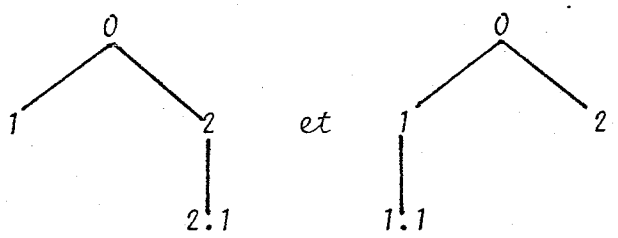
Pratiquement, on assigne à une grammaire récursive G la propriété r en interdisant à certains points de participer à toute transformation. Ceci permet d'obtenir une r-grammaire tout en ayant la possibilité d'engendrer de nouveaux points.

CHAPITRE V

TRANSDUCTEURS TRANSFORMATIONNELS

INTRODUCTION

La simulation des grammaires transformationnelles par des transducteurs composés implique une représentation linéaire des arborescences. Le langage de représentation est défini à partir des éléments simples. La représentation linéaire d'un élément simple est donnée par l'application d'une bijection de l'ensemble des éléments simples sur le langage de parenthèses défini sur le vocabulaire $V_T = \{[,]\}$. Un mot de V^*_T représente un élément simple si et seulement si ce mot est correctement parenthésé. Au moyen de cette définition, une arborescence admet plusieurs représentants linéaires, car elle admet plusieurs éléments simples. Les éléments simples d'une arborescence forment une classe d'équivalences. Cette classe d'équivalences induit une classe d'équivalence sur le langage de représentation. Ainsi, les mots [[ ] [[ ]]] et [[[ ]] [ ]] sont équivalents et représentent les éléments simples suivants :



La notion de sous-arborescence induit la notion de sous-mot. Un sous-mot d'un mot u de L est un mot W de L pouvant se "retrouver" à l'intérieur de ce mot u. La façon de retrouver ce mot ne doit pas être quelconque, mais suivre l'opération image sur les mots de L de la fonction d'effacement. La propriété pour une arborescence {A} d'admettre une sous-arborescence {B} est alors identique à la propriété d'existence de deux mots de L représentant les arborescences {A} et {B} et tel que, le mot représentant l'arborescence {B} soit un sous-mot de celui représentant l'arborescence {A}.

Les arborescences orientées n'admettent qu'un seul élément simple et ont un représentant linéaire unique.

Pour la représentation linéaire des arborescences étiquetées, le vocabulaire de représentation est formé par le vocabulaire d'étiquettes augmenté du symbole "]". La représentation linéaire d'une arborescence étiquetée s'obtient à partir de la représentation linéaire de l'arborescence en remplaçant chaque symbole "[" par le symbole d'étiquette du point correspondant.

La définition de sous-mot s'étend au langage de représentation des arborescences étiquetées et la propriété pour une arborescence donnée d'admettre une sous-arborescence est équivalente également à la présence d'un sous-mot. Dans le cas des sous-arborescences particulières, telles que les sous-arborescences sommets, feuilles, etc..., les restrictions imposées sur les éléments d'arborescences introduisent des restrictions images sur les mots de représentaioon. Ainsi, par exemple, la présence d'une arborescence sommet {B} dans l'arborescence {A} est identique à l'existence d'un sous-mot représentant l'arborescence {B}, et contenant le premier et le dernier caractère du mot représentant l'arborescence {A}(sous-mot initial).

La simulation d'une transformation par un transducteur composé sera effectuée à l'aide de trois transducteurs élémentaires. Le premier de ces transducteurs permet de reconnaître la présence d'une sous-arborescence dans l'arborescence source. Le second transducteur permet de séparer les différents éléments remarquables induits par la présence de la sous-arborescence. Le troisième transducteur termine la simulation de la transformation et permet de regrouper les différents éléments dans l'arborescence cible.

Le transducteur de reconnaissance est construit pour une arborescence donnée à partir d'un schéma de récurrence :

- La construction d'un transducteur qui reconnaît une arborescence réduite à un seul point est immédiate.

- Une arborescence contenant plus d'un point peut être considérée comme un ensemble d'arborescences rattachées à la racine. On effectue alors la construction du transducteur de reconnaissance de cette arborescence à partir des transducteurs qui reconnaissent les différentes arborescences précédemment déterminées.

Le fonctionnement de ces transducteurs s'effectue simultanément et commence après la lecture d'un point. Pour se terminer de façon positive, cette recherche doit avoir reconnu toutes les arborescences engendrant l'arborescence recherchée, c'est-à-dire, tous les transducteurs fonctionnant simultanément doivent être dans leur état final. L'état initial et l'état final de ce transducteur sont des états associés à la racine de l'arborescence recherchée comme sous-arborescence. Ainsi, chaque état du transducteur est associé à un point puisque d'après la définition, tout état provient d'un transducteur déjà construit.

Les contraintes de fonctionnement du transducteur pour un état particulier conduisent à des contraintes sur le point associé à cet état. Ainsi, la construction du transducteur reconnaissant des sous-arborescences particulières est similaire à la construction précédente ; seules s'ajoutent quelques restrictions pour rendre compte de la recherche particulière que l'on impose.
Par exemple, la reconnaissance d'une sous-arborescence sommet impose au fonctionnement des différents transducteurs de commencer seulement après la lecture de la racine, c'est-à-dire, après la lecture du premier symbole d'entrée. Le cas de la reconnaissance d'ensembles de sous-arborescences est identique au précédent puisqu'il est utilisé pour la construction du transducteur reconnaissant une arborescence. La contrainte sur la dépendance d'une arborescence par rapport à une autre, conduit à une recherche de la sous-arborescence dépendante seulement après lecture d'une feuille de la première arborescence. Afin de réaliser la séparation des différents éléments remarquables produits par la présence d'une sous-arborescence, le transducteur de reconnaissance fournit comme élément de sortie le mot miroir de l'entrée. A chaque symbole de ce mot est associé l'état courant du transducteur de reconnaissance. Ainsi, chaque symbole du mot de sortie contient le symbole d'entrée et l'ensemble des états possibles correspondants, c'est-à-dire, l'ensemble des points de la sous-arborescence à reconnaître qu'il peut éventuellement représenter. Le second transducteur lit ce mot et sépare les différents éléments, tels que le sommet et les éléments dépendant d'un même point. Cette séparation est possible du fait qu'à chaque symbole est associé l'ensemble des états possibles du transducteur de reconnaissance. Le transducteur de synthèse est le plus simple ; il effectue une écriture des différents éléments séparés dans l'arborescence résultante. Il réalise donc, avant tout, l'écriture de cette arborescence résultante. Ainsi, la simulation d'une transformation est réalisée par un transducteur composé à pile d'ordre 3.

La simulation des transformations d'arborescences étiquetées est identique à la simulation des transformations d'arborescences. Le traitement des étiquettes est séparé du traitement de la structure et est constitué par un ensemble de prédicats et de fonctions de modification.
A l'intérieur du transducteur de reconnaissance, la progression d'un état associé à un point n'est possible que si le prédicat correspondant à ce point est vérifié. Le transducteur qui effectue la synthèse de l'arborescence résultante réalise également pour chaque point de cette arborescence le calcul de la nouvelle étiquette.

La simulation d'une grammaire transformationnelle par un transducteur composé est alors immédiate puisqu'une grammaire transformationnelle est constituée d'un ensemble de règles de transformations. Les différentes restrictions sur les grammaires définies précédemment correspondent à des restrictions sur la composition des différents transducteurs qui simulent ces transformations. La réalisation d'un système à partir de transducteurs nécessite l'existence de transducteurs constructeurs. La construction d'une grammaire transformationnelle au moyen du langage des différents transducteurs ne serait pas raisonnable. De plus, la gestion de la mémoire pour ce système est facilitée par l'existence de transducteurs universels. Le langage de construction utilisé par ce transducteur est identique au langage de représentation. L'existence du transducteur constructeur est donnée par les différentes démonstrations. Ces démonstrations étant toutes des schémas de récursion, le transducteur constructeur sera évidemment un transducteur à pile. Du fait que le vocabulaire est fini, et d'autre part, que la fonction de transformation est représentée point par point, les transformations simulées par des transducteurs obtenus à partir de transducteurs constructeurs à pile ont un nombre borné de points. En pratique, le nombre de points qui interviennent dans une transformation est peu élevé (moins de 100) et conduit à un choix raisonnable de cette borne pour un ordinateur de capacité moyenne.

A - REPRESENTATION LINEAIRE DES ARBORESCENCES

1 - LANGAGE DE REPRESENTATION

Les arborescences seront représentées linéairement à partir des mots du langage L, défini sur $V_T = \{[,]\}$ par les conditions suivantes :

* $[.] \in L$

* $W_1,\ldots,W_n \in L \Longrightarrow [.W_1,\ldots,W_n.] \in L$

2 - PROPRIETE

Soient W_1 et W_2 deux mots de L. Pour tout mot W_3 et W_4 de V^*, nous avons la propriété :

$$W_1.W_3 = W_2.W_4 \Longleftrightarrow W_1 = W_2 \wedge W_3 = W_4$$

DEMONSTRATION

$\Longleftarrow$: *évident*

$\Longrightarrow$: *étant donnés deux mots de L, toute concaténation à droite par une chaîne quelconque non vide ne peut donner un mot de L.*

$W \in L,\ W.X \in L \Longrightarrow X = \varepsilon$

Alors : $W_1.W_3 = W_2.W_4 \Longrightarrow \exists X_1$ *ou* X_2, *mots de* V^* *tels que :*

$$\text{Soit } W_1 = W_2.X_1$$
$$\text{Soit } W_2 = W_1.X_2$$

W_1 *et* W_2 *étant des mots de L, nécessairement* $X_1 = X_2 = \varepsilon$

donc $W_1 = W_2 \Longrightarrow W_3 = W_4$

Corollaire

Soient deux suites de mots de L, $W_1,\ldots,W_n$ et $W'_1,\ldots,W'_m$. Nous avons la propriété :

$$W_1 \ldots W_n = W'_1 \ldots W'_m \Longleftrightarrow n = m \wedge (W_1 = W'_1) \wedge \ldots \wedge (W_n = W'_n)$$

3 - THEOREME DE REPRESENTATION

La fonction $\mathcal{L}$ définie ci-après est une bijection de l'ensemble des éléments simples sur le langage L.

$$\mathcal{L}(\{0\}) = [.]$$

$$\mathcal{L}(\{0\} \cup (\underset{i\in[I]}{\cup} \pi_i(A_i))) = [.\ \underset{i\in[I]}{\pi}\ \mathcal{L}(A_i).]$$

avec $[I] = \{1,2,\ldots,I\}$

DEMONSTRATION

- *Injection*

Démontrons la propriété suivante par récurrence sur le nombre n de points d'un élément d'arborescence A :

$$\mathcal{L}(A) = \mathcal{L}(A') \Longrightarrow A = A'$$

α) La propriété est vraie pour n = 1, car il existe un seul élément simple de cardinal 1 : {0}.

β) Supposons la propriété vraie pour tout élément simple de cardinal inférieur ou égal à n, et soit deux éléments simples A et A' de cardinal inférieur ou égal à n+1.

$$A = \{0\} \cup (\underset{i\in[I]}{\cup} \pi_i(A_i))$$

$$A' = \{0\} \cup (\underset{j\in[I]}{\cup} \pi_j(A_j))$$

$$\mathcal{L}(A) = \mathcal{L}(A') \Longleftrightarrow [.\ \underset{i\in[I]}{\pi}\ \mathcal{L}(A_i).] = [.\ \underset{j\in[I]}{\pi}\ (A'_j).]$$

$$\Longleftrightarrow \underset{i\in[I]}{\pi}\ \mathcal{L}(A_i) = \underset{j\in[I]}{\pi}\ \mathcal{L}(A'_j)$$

$$\Longleftrightarrow \underset{i\in I}{\wedge}\ \mathcal{L}(A_i) = \mathcal{L}(A'_i)$$ *d'après le corollaire précédent*

$$\Longrightarrow \underset{i\in I}{\wedge}\ A_i = A'_i$$ *d'après l'hypothèse de récurrence*

$$\Longrightarrow A = A'$$

- *Surjection*

On montre par récurrence qu'il existe une construction d'éléments simples compatibles avec la construction de L.

- $\mathcal{L}(\{O\}) = [.]$

- *Soit* $W_1,\ldots,W_n \in L$,

$$\exists A_1,\ldots,A_n : \forall_i\ W_i = \mathcal{L}(A_i) \Longrightarrow \exists A = \{O\} \cup \left(\bigcup_{i=1}^{n} \pi_i(A_i)\right) :$$

$$\mathcal{L}(A) = [.W_1,\ldots,W_n.]$$

4 - REPRESENTATION LINEAIRE

Définition

$\mathcal{L}(\gamma(A))$ est par définition :

- un représentant linéaire de l'arborescence {A}
- un représentant linéaire de l'arborescence $\{A_{A'}\}$ $\forall\ A' \subset A$
- un représentant linéaire de l'arborescence $\{A\}_\circ$

Propriété

- une arborescence non orientée ou partiellement orientée admet plusieurs représentants linéaires.
- une arborescence orientée admet un seul représentant linéaire.

5 - SOUS-MOTS D'UN MOT DANS L

- sous-mots propres

Soit $W \in L, u \in V_T^*$:

u est un sous-mot propre de W si et seulement si :

$\exists V.Z \in L \cup \{\varepsilon\} : W = V.u.Z$

V et Z sont appelées bornes du sous-mot propre u.

- sous-mots

Soit $W \in L$, $u \in V_T^*$:

u est un sous-mot de W si et seulement si :

$$* \; W = \prod_{i=1}^{2n+1} W_i$$

$$* \; u = \prod_{j=1}^{n} W_{2j}$$

$$* \; W_1 . W_{2n+1} \in L \cup \{\varepsilon\} \wedge \forall h : 1 \leq h < n : [. W_{2h+1} .] \in L$$

W_1 et W_{2n+1} sont appelées bornes du sous-mot u.

∘ sous-mot initial

Un sous-mot est dit initial si ses bornes sont toutes les deux égales au mot vide ε.

REMARQUE

Un sous-mot propre est un sous-mot.

∘ sous-mots indépendants

Soit un ensemble $u_1, \ldots, u_m$ d'éléments de V^* et W un mot de L.

L'ensemble $u_1, \ldots, u_m$ forme un ensemble de sous-mots indépendants de W, si et seulement si :

$$* \; W = \prod_{i=1}^{2n+1} W_i \cup \{\varepsilon\} \wedge \forall l : 1 \leq l < m \; [. W_{2l+1} .] \in L.$$

$$* \; \exists h, \text{ permutation } [1 : m] \longmapsto [1 : m] :$$

$$\forall k : 1 \leq k \leq m : u_{h(k)} \text{ est un sous-mot de } W_{2k}$$

$$* \; W_1 . W_{2n+1} \in L \cup \{\varepsilon\}$$

∘ sous-mots indépendants complets

Soit un ensemble $u_1, \ldots, u_m$ d'éléments de V^* et W un mot de L.

L'ensemble $u_1, \ldots, u_m$ forme un ensemble de sous-mots indépendants complets de W si et seulement si :

$$* \; W = \prod_{i=1}^{2n+1} W_i$$

$$* \; \exists h, \text{ permutation } [1 : m] \longmapsto [1 : m] :$$

$$\forall k : 1 \leq k \leq m :$$

$$- W_{2k} = \prod_{j=1}^{2n_k+1} W_{2k_j}$$

$$- u_{h(k)} = \prod_{j=1}^{n_k} W_{2k_{2j}}$$

$$- W_{2k_1} . W_{2k_{2n_k+1}} \in L \wedge \forall\, l : 1 \leqslant l < m : [. W_{2k_{2l+1}} .] \in L$$

$$- \forall\, l : 0 \leqslant l < m : W_{2k_{2l+1}} = W'_1 W'_2 W'_3 \Longrightarrow W'_1, W'_2, W'_3 \notin L$$

$$* \forall\, l : 0 \leqslant l < m : W_{2l+1} = W'_1 W'_2 W'_3 \Longrightarrow W'_1, W'_2, W'_3 \notin L$$

6 - EQUIVALENCE DANS L

α) Définition

Deux mots de L, W et W' sont dits équivalents si et seulement si :

- Soit : W = W' = [.]
- Soit : ∘ $W = [. \prod_{i=1}^{n} W_i .] \wedge \forall_i : W_i \in L$

∘ $W' = [. \prod_{i=1}^{n} W'_i .] \wedge \forall_i : W'_i \in L$

∘ ∃f, permutation : [1 : n] ↦ [1 : n] : W_i et $W'_{f(i)}$ sont équivalents.

NOTATION :

$W \sim W'$

β) Propriété : Lemme V1

1. La relation définie ci-dessous est une relation d'équivalence sur l'ensemble des mots de L.

2. Pour tout couple d'éléments simples d'arborescences A et A' :

$$A \sim A' \Longleftrightarrow \mathcal{L}(A) \sim \mathcal{L}(A')$$

DEMONSTRATION

1 - La relation définie entre deux mots de L est une relation d'équivalence ; la détermination des différentes fonctions sont déterminées de la façon suivante :

- *réflexivité* : $f = I$
- *symétrie* : $f' = f^{-1}$
- *transitivité* ; $f'' = f' \circ f$

2 - Soient deux éléments simples d'arborescences A et A'.

$W = \mathcal{L}(A)$, $W' = \mathcal{L}(A')$

- $A = A' = \{O\} \Longleftrightarrow W = W' = [.]$
- $A \neq \{O\} \Longleftrightarrow W = [. \overset{n}{\underset{i=1}{\pi}} W_i .] \wedge \forall_i,\ W_i \in L$
- $A' \neq \{O\} \Longleftrightarrow W' = [. \overset{n}{\underset{i=1}{\pi}} W'_i .] \wedge \forall_i,\ W'_i \in L$

Alors :

$A \sim A' \Longleftrightarrow \exists f' \in \mathfrak{F} : A = \psi_{f'}(A')$

$\Longleftrightarrow A = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A))) \wedge A' = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A')))$

$\wedge \forall_i : \mathcal{E}_i(A) = \mathcal{E}_i(\psi_{f'}(A'))$

$\Longleftrightarrow A = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A))) \wedge A' = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A'))) \wedge$

$\exists h$ *permutation* : $[1:n] \mapsto [1:n]$: $\mathcal{E}_i(A) \sim \mathcal{E}_{f(i)}(A')$

$\Longleftrightarrow W = [.(\overset{n}{\underset{i=1}{\pi}} W_i).] \wedge W' = [. \overset{n}{\underset{i=1}{\pi}} W'_i .] \wedge$

$(\forall_i : W_i, W'_i \in L) \wedge \exists f$ *permutation* : $[1:n] \mapsto [1:n]$:

$\forall_i : W_i \sim W'_{f(i)}$

$\Longleftrightarrow W \sim W'$

7 - CARACTERISATION LINEAIRE DES SOUS-ARBORESCENCES

* Lemme V2

L'arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si, il existe deux éléments simples, B de {B} et A de {A} tels que $\mathcal{L}(B)$ soit un sous-mot de $\mathcal{L}(A)$.

DEMONSTRATION

D'après le lemme IV-5, l'arborescence {B} est une sous-arborescence de {A} si et seulement si, il existe deux éléments simples B de {B} et A de {A} tels que : $B \subset \mathcal{E}_X(A)$, $X \in A$.

La propriété est alors évidente d'après la définition de $\mathcal{L}$.

* Lemme V3

L'arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si, pour tout élément simple B de {B} et tout élément simple A de {A}, il existe un mot W de L, tel que :

- W est un sous-mot de $\mathcal{L}(A)$
- $W \sim \mathcal{L}(B)$

DEMONSTRATION

Cette propriété est directement déduite du corollaire III-4 et du lemme V1.

Dans le cas des arborescences orientées, il existe un seul élément simple. Nous avons donc la propriété :

* Lemme V4

L'arborescence orientée $\{B\}_o$ est une sous-arborescence orientée de l'arborescence orientée $\{A\}_o$ si et seulement si, pour tout élément simple B de $\{B\}_o$ et tout élément A de $\{A\}_o$, $\mathcal{L}(B)$ est un sous-mot de $\mathcal{L}(\gamma(A))$.

DEMONSTRATION

D'après le lemme III-9, l'arborescence orientée $\{B\}_o$ est une sous-arborescence orientée de l'arborescence orientée $\{A\}_o$ si et seulement si, pour tout élément simple B de $\{B\}_o$ et tout élément A de $\{A\}_o$,

il existe un élément X de $\gamma(A)$ et une fonction f de $\mathfrak{F}$ telle que :

- ψ_f *est ordonnée sur B*
- $\psi_f(B) \subset \mathcal{E}_X(\gamma(A))$

Alors :

$$\mathcal{L}(\gamma(A)) = u.V.W \wedge \mathcal{L}(\mathcal{E}_X(\gamma(A))) = V \wedge u.W \in L \cup \{\varepsilon\}$$

$$\mathcal{L}(\mathcal{E}_X(\gamma(A))) = [. \prod_{i \in I} \mathcal{L}(\mathcal{E}_i(\mathcal{E}_X(\gamma(A)))).]$$

où $I = \{i \mid i \in N^+ \wedge i \in \mathcal{E}_X(\gamma(A))\}$

Nous montrons alors la propriété suivante par récurrence sur le nombre de points de l'arborescence $\{B\}_o$ *:*

$\psi_f(B) \subset \mathcal{E}_X(\gamma(A)) \Longleftrightarrow \mathcal{L}(B)$ *est un sous-mot initial de* $\mathcal{L}(\mathcal{E}_X(\gamma(A)))$

- *Si* $\{B\} = \{O\}$ *alors la propriété est immédiate*
- *Sinon, si la propriété est vraie pour toute arborescence ayant au plus n points :*

$$\mathcal{L}(\mathcal{E}_X(\gamma(A))) = [. \prod_{i=1}^{n} \mathcal{L}(\mathcal{E}_i(\mathcal{E}_X(\gamma(A)))).]$$

$$\mathcal{L}(B) = [. \prod_{i=1}^{k} \mathcal{L}(\mathcal{E}_i(B)).]$$

$\psi_f(B) \subset \mathcal{E}_X(\gamma(A)) \wedge \psi_f$ *ordonnée sur B*

$\Longleftrightarrow$ ψ_f *est ordonnée sur B et* $\forall j \in B : \mathcal{E}_{\psi_f(j)}(\psi_f(B)) \subset \mathcal{E}_{\psi_f(j)}(\mathcal{E}_X(\gamma(A)))$

$\exists f_O$ *permutation ordonnée sur* $N^+ \cap B : \forall j \in B,$

$\exists \psi_{f_j}$ *ordonnée sur* $\mathcal{E}_j(B)$ *telle que :* $\psi_{f_j}(\mathcal{E}_j(B)) \subset \mathcal{E}_{\psi_{f_O}(j)}(\mathcal{E}_X(\gamma(A)))$

$\Longleftrightarrow$ $\exists f_O$ *permutation ordonnée sur* $N^+ \cap B : \forall j \in B :$

$$\mathcal{L}(\mathcal{E}_{\psi_{f_0}(j)}(\mathcal{E}_X(\gamma(A)))) = \prod_{i=2}^{2n_h} W_{h_i} \quad avec\ h = \psi_{f_0}(j) \wedge$$

$$\mathcal{L}(\mathcal{E}_j(B)) = \prod_{i=2}^{n_h} W_{h_{2i}} \wedge$$

$$\forall\ l : 1<l<n_h : [.W_{h_{2l+1}}.] \in L$$

$$\Longleftrightarrow \mathcal{L}(\mathcal{E}_X(\gamma(A))) = [.\ \prod_{j\in I}\ (\prod_{i=2}^{2n_j} W_{j_i}).] \wedge$$

$$\mathcal{L}(B) = [.\ \prod_{j\in B}\ (\prod_{i=1}^{n_h} W_{h_{2i}}).] \quad avec\ h = f_0(j) \wedge$$

$$\forall\ j\in I,\ \forall\ l : 1<l<n_j : [.W_{j_{2l+1}}.] \in L$$

$$\Longleftrightarrow \mathcal{L}(B)\ est\ un\ sous\text{-}mot\ initial\ de\ \mathcal{L}(\mathcal{E}_X(\gamma(A)))$$

De plus, nous avons la propriété :

Il existe X, tel que $\mathcal{L}(B)$ est un sous-mot initial de $\mathcal{L}(\mathcal{E}_X(\gamma(A)))$ si et seulement si $\mathcal{L}(B)$ est un sous-mot de $\mathcal{L}(\gamma(A))$.

Une arborescence partiellement orientée admet plusieurs éléments simples. Néanmoins, cet ensemble d'éléments simples est plus restreint que dans le cas des arborescences non orientées. Nous obtenons seulement la propriété :

L'arborescence partiellement orientée $\{B_B,\}_\circ$ est une sous-arborescence partiellement orientée de l'arborescence partiellement orientée $\{A_A,\}_\circ$ si et seulement si, il existe deux éléments B_B, et A_A, de $\{B_B,\}_\circ$ et $\{A_A,\}_\circ$ tels que :

$\mathcal{L}(\gamma(B))$ est un sous-mot de $\mathcal{L}(\gamma(A))$.

8 - LANGAGE DE REPRESENTATION LINEAIRE DES ARBORESCENCES ETIQUETEES

Une arborescence étiquetée sur V est représentée linéairement par le langage L_V défini sur $V_T = V \cup \{]\}$ de la façon suivante :

* $\sigma.] \in L_V\ \forall\ \sigma\in V$

* $W_1,\ldots,W_n \in L_V,\ \sigma\in V \Longrightarrow \sigma.\ W_1,\ldots,W_n.] \in L_V$

Propriété

La substitution définie ci-après est telle que pour tout mot W de L_V, h(W) est un mot de L :

$$h : \begin{cases} h(\sigma) = [\ \forall\ \sigma \in V \\ h(]) =] \end{cases}$$

Alors : $h(W) \in L \iff W \in L_V$

Cette propriété implique les propriétés données pour les mots de L.

* $W_1, W_2 \in L_V$, $W_3, W_4 \in V_T^*$:

$W_1.W_3 = W_2.W_4 \iff W_1 = W_2 \wedge W_3 = W_4$

* $W_1, \ldots, W_n, W'_1, \ldots, W'_m \in L_V$

$W_1, \ldots, W_n = W'_1, \ldots, W'_m \iff n = m \wedge (W_1 = W'_1) \wedge \ldots \wedge (W_n = W'_n)$

9 - THEOREME DE REPRESENTATION

La fonction $\mathcal{L}$ définie ci-après est une bijection de l'ensemble des éléments simples étiquetés sur V sur le langage L_V :

* $\mathcal{L}((\{0\},\rho)) = \sigma.] \iff \rho(0) = \sigma$

* $\mathcal{L}((\{0\} \cup (\bigcup_{i \in [I]} \pi_i(A_i)),\rho)) = \sigma. \pi_{i \in [I]} \mathcal{L}((A_i,\rho_i)).]$

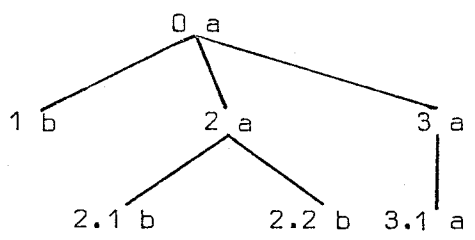
$\iff \rho(0) = \sigma \wedge \forall i \in [I]$, $W \in \mathbf{u}$: $\rho(i.W) = \rho_i(W)$

Exemple :

$A = \{0,1,2,3,2.1,2.2,3.1\}$

$\rho = \{0 \to a,\ 1 \to b,\ 2 \to a,\ 2.1 \to b,\ 2.2 \to b,\ 3 \to a,\ 3.1 \to a\}$

A =



$\mathcal{L}((A,\rho)) = a\ b\ ]\ a\ b\ ]\ b\ ]\ ]\ a\ a\ ]\ ]\ ]$

$\mathcal{L}((A',\rho')) = a\ a\ a\ ]\ ]\ b\ ]\ a\ b\ ]\ b\ ]\ ]\ ]$

et $(A,\rho) \sim (A',\rho')$

10 - REPRESENTANT LINEAIRE D'UNE ARBORESCENCE ETIQUETEE

Définition

$\mathcal{L}(A',\rho'))$ est un représentant linéaire de l'arborescence étiquetée $\{(A,\rho)\}$ si et seulement si :

$$\exists (A_1,\rho_1) \in \{(A,\rho)\} :$$

$$A' = \gamma(A_1) \wedge (A',\rho') \sim (A_1,\rho_1)$$

11 - SOUS-MOTS DANS L_V

Définition

α) Sous-mot propre

Soit $W \in L_V$, $u \in V_T^*$:

u est un sous-mot propre de W si et seulement si :

$$\exists V.Z \in L_V \cup \{\epsilon\} : W = V.u.Z$$

V et Z sont appelées bornes du sous-mot propre.

β) Sous-mot

$W \in L_V$, $u \in V_T^*$:

u est un sous-mot de W si et seulement si :

$$W = \prod_{i=1}^{2n+1} W_i \wedge u = \prod_{j=1}^{n} W_{2j} \wedge W_1.W_{2n+1} \in L_V \cup \{\epsilon\} \wedge$$

$$\forall h : 1<h<n, \exists \sigma \in V : \sigma.W_{2h+1}.] \in L_V$$

W_1 et W_{2h+1} sont appelées bornes du sous-mot propre.

12 - EQUIVALENCE DANS L_V

α) Définition

Deux mots de L_V, W et W' sont dits équivalents si et seulement si :

- Soit : $W = W' = \sigma.]$, $\sigma \in V$
- Soit : ∘ $W = \sigma. \prod_{i=1}^{n} W_i.] \wedge \forall i,\ W_i \in L_V$

∘ $W' = \sigma. \prod_{i=1}^{n} W'_i.] \wedge \forall i,\ W'_i \in L_V$

∘ ∃f, bijection : $[1 : n] \mapsto [1 : n]$:

W_i et $W_{f(i)}$ sont équivalents

NOTATION :

$W \sim W'$

β) Propriété : Lemme V

1. La relation définie ci-dessus est une relation d'équivalence sur l'ensemble des mots de L_V.

2. Pour tout couple d'éléments simples d'arborescence A et A' :

$$(A,\rho) \sim (A',\rho') \Longleftrightarrow \mathcal{L}((A,\rho)) \sim \mathcal{L}((A',\rho'))$$

DEMONSTRATION

1 - La relation définie entre des mots de L_V est une relation d'équivalence, les différentes fonctions sont déterminées de la façon suivante :

- *Réflexivité : $f = I$*
- *Symétrie : $f' = f^{-1}$*
- *Transitivité : $f'' = f' \circ f$*

2 - Soit $W = \mathcal{L}((A,\rho))$, $W' = \mathcal{L}((A',\rho'))$. Montrons l'équivalence par récurrence sur le nombre de points de l'arborescence {A}.

- $A = A' = \{O\}$:

$(A,\rho) \sim (A',\rho) \Longleftrightarrow W = W' = W' = \sigma.]$

- $A \neq \{O\} \Longrightarrow W = \sigma.\overset{n}{\underset{i=1}{\pi}} W_i.] \wedge \forall i : W_i \in L_V$

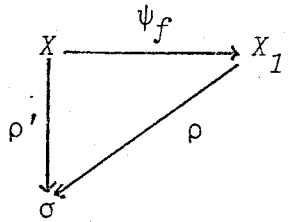
$A \neq \{O\},\ (A,\rho) \sim (A',\rho') \Longrightarrow A' \neq \{O\} \Longrightarrow$

$W' = \sigma'.\overset{m}{\underset{i=1}{\pi}} W'_i.] \wedge \forall i : W'_i \in L_V$

Alors :

$(A,\rho) \sim (A',\rho') \Longleftrightarrow \exists f' \in \mathfrak{F} : .\ A = \psi_f(A')$

. *le diagramme suivant commute pour tout élément X de A'*

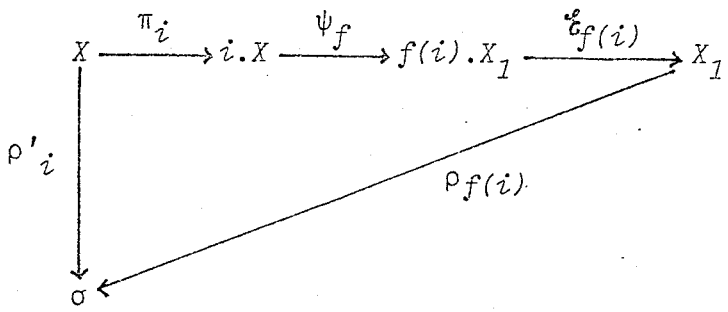


$\Longleftrightarrow A = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A))) \wedge A' = \{O\} \cup (\overset{n}{\underset{i=1}{\cup}} \pi_i(\mathcal{E}_i(A'))) \wedge$

$\forall i : \mathcal{E}_i(A) = \mathcal{E}_i(\psi_{f'}(A')) \wedge \rho(O) = \rho'(O) \wedge$

Si $\rho_i(W) = \rho(i.W)\ \forall\ W,\ \rho'_i(W) = \rho'(i.W)\ \forall\ W,$

le diagramme suivant commute pour tout i et tout élément X de $\mathcal{E}_i(A')$:



$$\Longleftrightarrow A = \{O\} \cup \left(\bigcup_{i=1}^{n} \pi_i(\mathcal{E}_i(A))\right) \wedge A' = \{O\} \cup \left(\bigcup_{i=1}^{n} \pi_i(\mathcal{E}_i(A'))\right) \wedge$$

$$\rho(O) \wedge \rho'(O) \wedge \exists f \text{ permutation } : [1 : n] \mapsto [1 : n] :$$

$$(\mathcal{E}_i(A), \rho_i) \sim (\mathcal{E}_{f(i)}(A'), \rho'_{f(i)})$$

$$\Longleftrightarrow W = \sigma.\left(\prod_{i=1}^{n} W_i\right).] \wedge W' = \sigma.\left(\prod_{i=1}^{n} W'_i\right).] \wedge$$

$$(\forall i : W_i, W'_i \in L_V) \wedge \exists f \text{ permutation } : [1 : n] \mapsto [1 : n] :$$

$$\forall i : W_i \sim W'_{f(i)}$$

$$\Longleftrightarrow W \sim W'$$

13 - CONCATENATION LINEAIRE DES SOUS-ARBORESCENCES ETIQUETEES

La concaténation linéaire des sous-arborescences étiquetées est identique à celle des sous-arborescences non étiquetées.

* Lemme V2 :

L'arborescence $\{(B,\rho')\}$ est une sous-arborescence de l'arborescence $\{(A,\rho)\}$ si et seulement si, il existe deux éléments simples (B,ρ') de $\{(B,\rho')\}$ et (A,ρ) de $\{(A,\rho)\}$ tels que $\mathcal{L}(\{(B,\rho')\})$ soit un sous-mot de $\mathcal{L}(\{(A,\rho)\})$.

* Lemme V3 :

L'arborescence $\{(B,\rho')\}$ est une sous-arborescence de l'arborescence $\{(A,\rho)\}$ si et seulement si :

pour tout élément simple (B,ρ') de $\{(B,\rho')\}$ et tout élément simple (A,ρ) de $\{(A,\rho)\}$, il existe un mot W de L_V, tel que :

- $\mathcal{L}(\{(B,\rho')\})$ est un sous-mot de W
- $\mathcal{L}(\{(A,\rho)\}) \sim W$

* Lemme V4 :

L'arborescence orientée et étiquetée $\{(B,\rho')\}_\circ$ est une sous-arborescence orientée et étiquetée de l'arborescence orientée et étiquetée $\{(A,\rho)\}_\circ$ si et seulement si pour tout élément simple (B,ρ') de $\{(B,\rho')\}_\circ$ et tout élément (A,ρ) de $\{(A,\rho)\}$: $(\gamma(A),\rho_1) \sim (A,\rho) \Longrightarrow \mathcal{L}(\{(B,\rho')\})$ est un sous-mot de $\mathcal{L}(\{(\gamma(A),\rho_1)\})$.

L'arborescence partiellement orientée et étiquetée $\{(B_B,,\rho')\}_o$ est une sous-arborescence de l'arborescence partiellement orientée et étiquetée $\{(A_A,,\rho)\}_o$ si et seulement si, il existe deux éléments $(B_B,,\rho')$ et $(A_A,,\rho)$ de $\{(B_B,,\rho')\}_o$ et $\{(A_A,,\rho)\}_o$ tels que :

$$(\gamma(B),\rho'_1) \sim (B,\rho') \wedge (\gamma(A),\rho_1) \sim (A,\rho) \Longrightarrow$$

$$\mathcal{L}((\gamma(B),\rho'_1)) \text{ est un sous-mot de } \mathcal{L}((\gamma(A),\rho)).$$

14 - CARACTERISATION DE SOUS-ARBORESCENCES PARTICULIERES

Les propriétés suivantes caractérisent des sous-arborescences particulières et sont valables pour les arborescences étiquetées, orientées, partiellement orientées, ou quelconques.

a) Sous-arborescence sommet

Une arborescence {B} est une sous-arborescence sommet de l'arborescence {A} si et seulement si, il existe deux éléments B de {B} et A de {A} tels que :

$\mathcal{L}(B)$ est un sous-mot initial de $\mathcal{L}(A)$.

b) Sous-arborescence feuille

Une arborescence {B} est une sous-arborescence feuille de l'arborescence {A} si et seulement si :

$\exists B \in \{B\}$, $A \in \{A\}$:

- $\mathcal{L}(B)$ est un sous-mot de $\mathcal{L}(A)$ et :

$$\mathcal{L}(A) = \prod_{i=1}^{2n+1} W_i \,,\quad \mathcal{L}(B) = \prod_{j=1}^{n} W_{2j}$$

$$W_1.W_{2n+1} \in L \cup \{\varepsilon\} \wedge \forall h : 1<h<n,\ [.W_{2h+i}.] \in L$$

Alors :

$$\mathcal{L}(B) = W.[.].W' \Longleftrightarrow \wedge\ W' = W^{(3)}.W^{(4)}$$

$$\exists 1 : W = W^{(1)}.W^{(2)}$$

$$\wedge\ W_{21} = W^{(2)}.[.].W^{(3)} \wedge W^{(1)} = \prod_{i=1}^{1-1} W_{2i} \wedge W^{(4)} = \prod_{i=1+1}^{n} .W_{2i}$$

c) Sous-arborescence complète

Une arborescence {B} est une sous-arborescence complète de l'arborescence {A} si et seulement si :

$\exists B \in \{B\}, A \in \{A\}$:

- $\mathcal{L}(B)$ est un sous-mot de $\mathcal{L}(A)$ et :

$$\mathcal{L}(A) = \prod_{i=1}^{2n+1} W_i, \quad \mathcal{L}(B) = \prod_{j=1}^{n} W_{2_j}$$

$$W_1 . W_{2n+1} \in L \cup \{\varepsilon\} \wedge \forall h : 1<h<n, [.W_{2h+1}.] \in L$$

Alors :

$$\forall h : 1<h<n, W_{2h+1} \neq \epsilon \Longrightarrow W_{2h} = W'_{2h}.[\wedge$$

$$W_{2h+2} =].W'_{2h+2}, W'_{2h} \wedge W'_{2h+2} \in V^*$$

Ainsi, les éléments supplémentaires peuvent être ajoutés seulement sur les feuilles de l'arborescence {B}.

d) Sous-arborescence à points adjacents

Une sous-arborescence {B} de l'arborescence {A} admet le couple de points $(\{Y_B\},\{Y'_B\})$ comme points adjacents, si et seulement si :

$\exists A \in \{A\}, B \in \{B\}$:

- $\mathcal{L}(B)$ est un sous-mot de $\mathcal{L}(A)$

$$\mathcal{L}(A) = \prod_{i=1}^{2n+1} W_i, \quad \mathcal{L}(B) = \prod_{j=1}^{n} W_{2_j}$$

$$W_1 . W_{2n+1} \in L \cup \{\epsilon\} \wedge \forall h : 1<h<n : [.W_{2n+1}.] \in L$$

Alors :

Soit $\mathcal{L}(B) = W^{(1)} . \mathcal{L}(\mathcal{E}_Y(B)) . \mathcal{L}(\mathcal{E}_{Y'}(B)) . W^{(2)} \wedge$

$$(W_{2k} = W'_{2k} . \mathcal{L}(\mathcal{E}_Y(B)) \wedge W_{2k+2} = \mathcal{L}(\mathcal{E}_{Y'}(B)) . W'_{2k+2} \Longrightarrow$$

$$W_{2k+1} = \varepsilon)$$

Soit $\mathcal{L}(B) = W^{(1)}.\mathcal{L}(\mathcal{E}_{Y'}(B)).\mathcal{L}(\mathcal{E}_{Y}(B)).W^{(2)} \wedge$

$(W_{2k} = W'_{2k}.\mathcal{L}(\mathcal{E}_{Y'}(B)) \wedge W_{2k+2} = \mathcal{L}(\mathcal{E}_{Y}(B)).W'_{2k+2} \Longrightarrow$

$W_{2k+1} = \varepsilon)$

e) Sous-arborescences étrangères et indépendantes

Les deux sous-arborescences {B} et {C} de l'arborescence {A} sont deux sous-arborescences étrangères et indépendantes, si et seulement si :

$\exists A \in \{A\}, B \in \{B\}, C \in \{C\}$:

$\mathcal{L}(A) = [.W^{(1)}.W^{(2)}.]$ et :

- Soit $\mathcal{L}(B)$ un sous-mot non initial de $[.W^{(1)}.]$ et $\mathcal{L}(C)$ un sous-mot non initial de $[.W^{(2)}.]$
- Soit $\mathcal{L}(C)$ un sous-mot non initial de $[.W^{(1)}.]$ et $\mathcal{L}(B)$ un sous-mot non initial de $[.W^{(2)}.]$

f) Sous-arborescences étrangères et dépendantes

La sous-arborescence {C} est une sous-arborescence étrangère et dépendante de la sous-arborescence {B} de l'arborescence {A} par rapport au point $\{Y_B\}$ de dépendance, si et seulement si :

$\exists B \in \{B\}, C \in \{C\}, A \in \{A\}$:

$\mathcal{L}(B)$ et $\mathcal{L}(C)$ sont des sous-mots de $\mathcal{L}(A)$ et

$$\mathcal{L}(B) = W^{(1)}.\mathcal{L}(\mathcal{E}_{Y}(B)).W^{(2)} = W^{(1)}.[.].W^{(2)}$$

$$\mathcal{L}(A) = \prod_{i=1}^{2n+1} W_i, \quad \mathcal{L}(B) = \prod_{j=1}^{n} W_{2_j}, \quad W_1.W_{2n+1} \in L \cup \{\varepsilon\},$$

$\forall h : 1<h<n : [.W_{2h+1}.] \in L$

Alors :

$$\exists k : k<n \wedge W^{(1)}.[= \prod_{i=1}^{k} W_{2_i} \wedge].W^{(2)} = \prod_{i=k+1}^{n} W_{2_i} \wedge$$

$\mathcal{L}(C)$ est un sous-mot de W_{2k+1}

B - THEOREMES DE RECONNAISSANCE

1 - THEOREME R1

Pour toute arborescence {B}, il existe un transducteur à pile déterministe T ne dépendant que de {B}, tel que les propriétés suivantes soient équivalentes pour toute arborescence {A}.:

{B} est une sous-arborescence de l'arborescence {A} $\Longleftrightarrow$

$$T(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \ \forall\ A \in \{A\},\ q_0 \in \rho_{Q_0}(T)$$

DEMONSTRATION

On démontre la propriété par récurrence sur le nombre de points de l'arborescence {B}. D'après le lemme V3, l'arborescence {B} est une sous-arborescence de l'arborescence {A} si et seulement si pour tout élément simple B de {B} et tout élément A de {A}, il existe un mot W de T tel que :

- *W est un sous-mot de $\mathcal{L}(\gamma(A))$*
- *$W \sim \mathcal{L}(B)$*

Le transducteur T sera construit à partir d'un élément simple de {B}. La construction se fera en deux phases. La première ayant pour but la construction d'un transducteur T' qui possède la propriété :

$$T'(q_0, \mathcal{L}(\gamma(A)) \neq \emptyset\ \forall\ A \in \{A\},\ q_0 \in \rho_{Q_0}(T) \Longleftrightarrow W \text{ est un}$$

sous-mot initial de $\mathcal{L}(\gamma(A)) \wedge W \sim \mathcal{L}(B)$.

La seconde phase construit le transducteur T à partir du transducteur T'.

1 - Si {B} est une arborescence contenant un seul point, alors {B} = {{0}} et si B est un élément simple de {B}, $\mathcal{L}(B) = [.]$

Le transducteur T' est donné par :

$$T' = (V,\ V_E,\ V_S,\ V_p,\ Q,\ \delta',\ \{\{q_0\}\},\ F,\ 1,\ 1)$$

$$V = V_E \cup V_S \cup V_p \cup \{\check{b}\}$$

$$V_E = \{[,]\}$$

$V_S = V_E \times Q$

$V_p = Q$

$Q = \mathcal{P}(Q') \wedge Q' = \{q_0, q_1, q_f\}$

$F = \{Q'' \mid Q'' \cap \{q_f\} \neq \emptyset\}$

$\delta : \delta'([, Q_2, Q_1) = ([, Q_2 Q_1, Q'_1, D, ([, Q_1), G) \; \forall Q_1, Q_2 \in Q$

$$Q'_1 = \begin{cases} q_1 \text{ si } q_0 \in Q_1 \\ \emptyset \text{ sinon} \end{cases}$$

$\delta'(], Q_2, Q_1) = (], \varepsilon, Q'_1, D, (], Q_1), G) \; \forall Q_1, Q_2 \in Q$

$$Si\ Q''_1 = \begin{cases} q_f \text{ si } q_1 \in Q_1 \text{ alors } Q'_1 = Q_2 \cup Q''_1 \\ \emptyset \text{ sinon} \end{cases}$$

L'état q_f apparaît dans l'état courant du transducteur si et seulement si : $\mathcal{L}(\gamma(A)) = [.W.] \wedge [.W.] \in L$

Donc, l'état q_f apparaît dans l'état courant du transducteur si et seulement si $\mathcal{L}(B)$ est un sous-mot initial de $\mathcal{L}(\gamma(A))$.

Le transducteur T est alors défini par :

$T = (V, V_E, V_S, V_p, Q, \delta, \{\{q_0\}\}, F, 1, 1)$

où :

$\delta(\sigma, Q_2, Q_1) = (\sigma', Q'_2, Q'_1, M, \gamma, M') \iff$

$\delta'(\sigma, Q_2, Q_1) = (\sigma', Q'_2, Q''_1, M, \gamma, M') \wedge$

$Q'_1 = Q''_1 \cup \{q_0\} \cup (Q_1 \cap \{q_f\})$

D'après la définition de T' et du fait que l'état q_0 soit stationnaire, l'état q_f apparaît dans l'état courant du transducteur si et seulement si :

$\mathcal{L}(\gamma(A)) = W_1.[.W_2.]\, W_3, [.W_2.] \in L$

Lorsque l'état q_f est apparu dans l'état courant du transducteur, il y est stationnaire. Donc, $T(\{q_0\}, \mathcal{L}(\gamma(A))) \neq \emptyset \Longleftrightarrow \{B\}$ est une sous-arborescence de l'arborescence $\{A\}$.

2 - Supposons la propriété vraie pour toute arborescence contenant au plus (n-1) points. Soit $\{B\}$ une arborescence contenant n points et B un élément simple de $\{B\}$. Alors :

$$\mathcal{L}(B) = [. \prod_{i=1}^{p} \mathcal{L}(\mathcal{E}_i(B)).], \quad p \leq n-1$$

Soient T'_i les transducteurs possédant la propriété du théorème par rapport à $\mathcal{L}(\mathcal{E}_i(B))$ (Sous-mot initial).

$$T'_i = (V_i, V_E, V_{S_i}, V_{p_i}, \delta'_i, \{\{q_{0_i}\}\}, F_i, 1, 1)$$

où :

$$V_i = V_E \cup V_{S_i} \cup V_{p_i} \cup \{\check{b}\}$$

$$V_E = \{[,]\}$$

$$V_{S_i} = V_E \times Q_i$$

$$V_{p_i} = Q_i$$

$$Q_i = \mathcal{P}(Q'_i)$$

$$F_i = \{Q'' \mid Q'' \cap \{q_{f_i}\} \neq \emptyset\}$$

On suppose de plus que : $\forall i,j,\ i \neq j \Longrightarrow Q_i \cap Q_j = \emptyset$

Alors, le transducteur T' est défini par :

$$T' = (V, V_E, V_S, V_p, \delta', \{\{q_0\}\}, F, 1, 1)$$

où :

$$V = V_E \cup V_S \cup V_p \cup \{\check{b}\}$$

$$V_E = \{[,]\}$$

$$V_S = V_E \times Q$$

$V_p = Q$

$$Q = \mathcal{P}(Q') \wedge Q' = \bigcup_{i=1}^{n} Q_i \cup \{q_0, q_f\} \cup \left(\bigcup_{k=1}^{p} \mathcal{P}\left(\bigcup_{i=1}^{p} \{q_{f_i}\}\right)^k\right)$$

$$F = \{Q'' \mid Q'' \cap \{q_f\} \neq \emptyset\}$$

δ' *est défini par :*

$$* \ \delta'([, Q_2, Q_1) = ([, Q_2 \ Q_1, Q'_1, D, ([, Q_1), G)$$

où :

$$Q'_0 = \begin{cases} \emptyset \text{ si } q_0 \notin Q_1 \\ \bigcup_{i=1}^{p} \{q_{0_i}\} \text{ si } q_0 \in Q_1 \end{cases}$$

$$- \ Q''_1 = \{q'_i \mid \exists Q_{i_1} \subset Q_1, Q_{i_2} \subset Q_2 : \delta'_i([, Q_{i_2}, Q_{i_1}) =$$

$$([, Q_{i_2} \ Q_{i_1}, Q'_{i_1}, D, ([, Q_{i_1}), G) \wedge q'_i \in Q'_{i_1}\}$$

$$- \ Q'_1 = Q''_1 \cup Q'_0$$

$$* \ \delta'(], Q_2, Q_1) = (], \varepsilon, Q'_1, D, (], Q_1), G)$$

où :

$$Q_2 = Q'_2 \cup \{\pi_{i=1}^{h} R_i\} \wedge \{\pi_{i=1}^{h} R_i\} \in \pi_{k=1}^{p} \left(\mathcal{P}\left(\bigcup_{i=1}^{p} \{q_{f_i}\}\right)\right)^k \cup \emptyset$$

$$\wedge \ Q'_2 \cap \left(\bigcup_{l=1}^{p} \left(\mathcal{P}\left(\bigcup_{i=1}^{p} \{q_{f_i}\}\right)\right)^k\right) = \emptyset$$

$$Q''_1 = \{q'_i \mid \exists Q_{i_1} \subset Q_1, Q_{i_2} \subset Q_2 : \delta'(], Q_{i_2}, Q_{i_1}) =$$

$$(], \varepsilon, Q'_{i_1}, D, (], Q_{i_1}), G) \wedge q'_i \in Q'_{i_1}\}$$

$$- \ R = \{q_{f_i} \mid q_{f_i} \in Q''_1\}$$

$$- Q_R = \begin{cases} \{\prod_{i=1}^{h+1} R_i\} \Longleftrightarrow R_{h+1} = R \wedge \exists\{q_1,\ldots,q_{h+1}\} : \\ \qquad (q_i \neq q_j \ \forall\ 1\leqslant i\neq j\leqslant h+1 \wedge q_i \in R_i) \\ \{\{\prod_{i=1}^{h} R_i\} \text{ sinon} \end{cases}$$

$$- Q_F = \begin{cases} \{q_f\} \text{ si } (\mathcal{P}(\bigcup_{i=1}^{p} \{q_{f_i}\}))^p \cap Q_1 \neq \emptyset \\ \emptyset \text{ sinon} \end{cases}$$

Alors :

$$Q'_1 = Q'_2 \cup (Q''_1 - R) \cup Q_R \cup Q_F$$

Nous avons les propriétés suivantes :

- D'après l'hypothèse de récurrence, l'état q_{f_i} apparaît dans l'état courant du transducteur à partir de l'état q_{0_i} et après lecture d'un mot W_i, si et seulement si W'_i est un sous-mot initial de $W_i \wedge W'_i \sim \mathcal{L}(\mathcal{G}_i(B))$.

- L'état q_{0_i} apparaît dans l'état courant du transducteur à partir de l'état $\{q_0\}$ après lecture du symbole "[". Ensuite, après toute lecture de mot de L, l'état q_{0_i} apparaît également dans l'état courant du transducteur (fonctionnement de la pile pour un mot de L).

- L'état q_{f_i} apparaît dans l'état courant du transducteur si et seulement si : - le transducteur lit un mot de la forme $[.\prod_{j=1}^{l} W_j$

- $\forall\ t,\ 1\leqslant t<l,\ W_t \in L$

- W'_i est un sous-mot initial de $W_{l-1} \wedge W'_i \sim \mathcal{L}(\mathcal{G}_i(B))$

- Lorsque l'état q_{f_i} apparaît dans l'état courant du transducteur, il apparaît dans un ensemble de la forme $\bigcup_{k=1}^{p} (\mathcal{P}(\bigcup_{i=1}^{p} \{q_{f_i}\}))^k$.

- Si un élément de la forme $\prod_{i=1}^{k} R_i$ tel que pour tout i, $R_i \in \mathcal{P}(\bigcup_{i=1}^{p} \{q_{f_i}\})$ apparaît dans l'état courant du transducteur, les ensembles

constitutifs R_i de cet élément apparaîtrons dans l'état courant du transducteur après chaque lecture supplémentaire d'un mot de L. (Fonctionnement de la pile sur un mot de L).

- L'état q_f apparaît dans l'état courant du transducteur, si et seulement si : $\mathcal{L}(\gamma(A)) = [. \prod_{i=1}^{m} W_i .] \wedge \forall i,\ W_i \in L \wedge \exists \{i_1, \ldots, i_p\} : \forall h : \exists W'_{i_h}$ *sous-mot initial de* $W_{i_h} \wedge W'_{i_h} \sim \mathcal{L}(\mathcal{C}_h(B))$.

Donc, l'état q_f apparaît dans l'état courant du transducteur, si et seulement si :

$\exists W$: - *W est un sous-mot initial de* $\mathcal{L}(\gamma(A))$
- $W \sim \mathcal{L}(B)$

Le transducteur T est alors défini par :

$$T = (V,\ V_E,\ V_S,\ V_p,\ Q,\ \delta,\ \{\{q_0\}\},\ F,\ 1,\ 1)$$

où :

$$\delta(\sigma,\ Q_2,\ Q_1) = (\sigma',\ Q'_2,\ Q'_1,\ M,\ \gamma,\ M') \Longleftrightarrow$$

$$\delta'(\sigma,\ Q_2,\ Q_1) = (\sigma',\ Q'_2,\ Q''_1,\ M,\ \gamma,\ M') \wedge Q'_1 = Q''_1 \cup \{q_0\} \cup (Q_1 \cap \{q_f\})$$

D'après la définition de T' et du fait que l'état q_0 soit stationnaire, l'état q_f apparaît dans l'état courant du transducteur si et seulement si :

$$\mathcal{L}(\gamma(A)) = W_1 . [. W_2 .] . W_3,\ [. W_2 .] \in L$$

$$\wedge \exists W' \text{ sous-mot initial de } [. W_2 .] : W' \sim \mathcal{L}(B)$$

Lorsque l'état q_f est apparu dans l'état courant du transducteur, il y est stationnaire. Donc :

$T(\{q_0\},\ \gamma(A)) \neq \emptyset \Longleftrightarrow \{B\}$ *est une sous-arborescence de* $\{A\}$.

2 - DEFINITION : ETATS ASSOCIES A UN POINT

Soit T (ou T') un transducteur de reconnaissance construit de la même manière que précédemment à partir d'une arborescence {B}.

$$T = (V, V_E, V_S, V_p, Q, \delta, \{\{q_0\}\}, F, 1, 1)$$

où :

$$F = \{Q'' \mid Q'' \cap \{q_f\} \neq \emptyset\}$$

$$Q = \mathcal{P}(Q')$$

$$Q' = \bigcup_{i=1}^{n} Q_i \cup \{q_0, q_f\} \cup \left(\bigcup_{k-1}^{p} \mathcal{P}\left(\bigcup_{i=1}^{p} \{q_{f_i}\} \right) \right)^k \vee Q' = \{q_0, q_1, q_f\}$$

L'ensemble des états associés à la racine de l'arborescence {B} sont définis de la façon suivante :

- q_0 est l'état prédécesseur
- q_f est l'état successeur
- $Q' - \{q_0, q_f\}$ sont les états internes
- Dans le cas où l'arborescence {B} contient plus d'un point,
  - $\bigcup_{i=1}^{p} \{q_{0_i}\}$ sont les états sous-prédécesseurs
  - $\bigcup_{i=1}^{p} \{q_{f_i}\}$ sont les états sous-successeurs

Cette définition est valable pour tous les points de l'arborescence {B} d'après la construction récursive de T.

3 - THEOREME R2 : SOUS-ARBORESCENCES PARTICULIERES

Soit {B} une arborescence quelconque. Il existe un transducteur à pile déterministe T ne dépendant que de {B} tel que les propriétés suivantes soient équivalentes pour toute arborescence {A}.

- T1 : l'arborescence {B} est une sous-arborescence sommet de l'arborescence {A} $\Longleftrightarrow T_1(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \ \forall A \in \{A\}, \ q_0 \in \rho_{Q_0}(T_1)$

- T2 : l'arborescence {B} est une sous-arborescence feuille de l'arborescence {A} $\Longleftrightarrow T_2(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \ \forall A \in \{A\}, q_0 \in \rho_{Q_0}(T_2)$

- T3 : l'arborescence {B} est une sous-arborescence complète de l'arborescence {A} $\Longleftrightarrow T_3(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \ \forall A \in \{A\}, q_0 \in \rho_{Q_0}(T_3)$

- T_4 : $\{B\} = \{A\} \Longleftrightarrow T_4(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \ \forall A \in \{A\}, q_0 \in \rho_{Q_0}(T_4)$

DEMONSTRATION

Soit le transducteur T associé à l'arborescence {B} et dont la construction est définie précédemment (Théorème R1) :

$$T = (V, V_E, V_S, V_p, Q, \delta, \{\{q_0\}\}, F, 1, 1)$$

$$V = V_E \cup V_S \cup V_p \cup \{\tilde{b}\}$$

$$V_E = \{[,]\}$$

$$V_S = V_E \times Q$$

$$V_p = Q$$

$$Q = \mathcal{P}(Q')$$

$$F = \{Q'' \mid Q'' \cap \{q_f\} \neq \emptyset\}$$

Toutes les restrictions définies dans ce théorème conduisent à des restrictions sur la fonction de transition δ par rapport aux états associés aux différents points concernés.

* *Restriction T1* :

Pour que l'arborescence {B} soit une sous-arborescence sommet de l'arborescence {A}, il faut et il suffit qu'il existe un sous-mot initial W de $\mathcal{L}(\gamma(A))$ tel que $W \sim \mathcal{L}(B)$.

Le transducteur T_1 est donc le transducteur T' donné dans la construction de T.

* *Restriction T2* :

L'arborescence {B} est une sous-arborescence feuille de l'arborescence {A} si et seulement si chaque feuille de {B} est une feuille de {A}. Pour qu'une feuille de l'arborescence {B} soit une feuille de l'arborescence {A}, la restriction sur la fonction de transition δ est donnée par :

- Soit q_i, q_{i+1}, q_{i+2} les états prédécesseurs internes et successeurs associés à cette feuille. Alors :

$$T_2 = (V, V_E, V_S, V_p, Q, \delta_2, \{\{q_0\}\}, F, 1, 1)$$

$$(\,], \varepsilon, Q'_1, D, (\,], Q_1), G) \in \delta_2(\,], Q_2, Q_1) \Longleftrightarrow$$

$$(\,], \varepsilon, Q'_1, D, (\,], Q_1), G) \in \delta(\,], Q_2, Q_1)$$

$$([, Q_2, Q''_1, Q'_1, D, ([, Q_1), G) \in \delta_2([, Q_2, Q_1) \Longleftrightarrow$$

$$([, Q_2, Q_1, Q'_1, D, ([, Q_1), G) \in \delta([, Q_2, Q_1) \wedge Q''_1 = Q_1 - \{q_{i+1}\}$$

L'état q_{i+2} apparaît dans l'état courant du transducteur à partir de l'état q_i, seulement après lecture d'un mot de la forme [.].

* *Restriction T3* :

L'arborescence {B} est une sous-arborescence complète de l'arborescence {A} si chaque point de l'arborescence {B} est un point "complet". La restriction sur la fonction de transition δ est donnée par :

- Soit q_i, Q_i, q_{f_i} et $\overset{P_i}{\underset{i=1}{\cup}} \{q_{f_i}\}$ les états prédécesseurs internes, successeurs et sous-successeurs associés à un point. (Un point qui est une feuille de {B} est "complet").

A l'intérieur de l'état courant du transducteur, on considère les états de la forme $(\overset{P_i}{\underset{i=1}{\cup}} \{q_{f_i}\})^k$.

Alors, $T_3 = (V, V_E, V_S, V_p, Q, \delta_3, \{\{q_0\}\}, F, 1, 1)$

Où :

$$([, Q_2, Q_1, Q'_1, D, ([, Q_1), G) \in \delta_3([, Q_2, Q_1) \Longleftrightarrow$$

$$([, Q_2, Q_1, Q'_1, D, ([, Q_1), G) \in \delta([, Q_2, Q_1)$$

$$(\Gamma,\ \varepsilon,\ Q''_1,\ D,\ (\Gamma,\ Q_1),\ G)\in\delta_3(\Gamma,\ Q_2,\ Q_1) \Longleftrightarrow$$

$$(\Gamma,\ \varepsilon,\ Q'_1,\ D,\ (\Gamma,\ Q_1),\ G)\in\delta(\Gamma,\ Q_2,\ Q_1)\ \wedge$$

$$Q''_1 = \begin{cases} Q'_1 - \bigcup_{j=1}^{P_i} \{q_{0_j}\} \text{ si } Q'_1 \cap \left(\bigcup_{k=1}^{P_i} \mathcal{P}\left(\bigcup_{i=1}^{P_i} \{q_{f_i}\}\right)^k\right) = \emptyset \ \vee \\ \qquad Q'_1 \cap \left(\bigcup_{k=1}^{P_i} \mathcal{P}\left(\bigcup_{i=1}^{P_i} \{q_{f_i}\}\right)^k\right) = Q_2 \cap \left(\bigcup_{k=1}^{P_i} \mathcal{P}\left(\bigcup_{i=1}^{P_i} \{q_{f_i}\}\right)^k\right) \\ Q'_1 \text{ sinon} \end{cases}$$

Sous un point, la recherche des descendants possibles continue si chaque lecture d'un mot de L fournit un nouvel élément.

* *Restriction T4* :
Le transducteur T_4 *comporte les trois restrictions précédentes.*

4 - THEOREME R3 : ENSEMBLE DE SOUS-ARBORESCENCES

α) Pour tout ensemble d'arborescences $\{B_1\},\ldots,\{B_n\}$, il existe un transducteur T ne dépendant que de l'ensemble $\{B_1\},\ldots,\{B_n\}$, tel que les propriétés suivantes soient équivalentes pour toute arborescence {A}.

L'ensemble d'arborescences $\{B_1\},\ldots,\{B_n\}$ est un ensemble de sous-arborescences de l'arborescence {A} $\Longleftrightarrow T(q_0,\mathcal{L}(\gamma(A))) \neq \emptyset\ \forall\ q_0\in\rho_{Q_0}(T),\ A\in\{A\}$.

β) Pour tout ensemble d'arborescences $\{B_1\},\ldots,\{B_n\}$, il existe un transducteur ne dépendant que de l'ensemble $\{B_1\},\ldots,\{B_n\}$ tel que les propriétés suivantes soient équivalentes pour toute arborescence {A} :

1 - L'ensemble d'arborescences $\{B_1\},\ldots,\{B_n\}$ est un ensemble indépendant de sous-arborescences de l'arborescence {A} $\Longleftrightarrow T^{(1)}(q_0,\mathcal{L}(\gamma(A))) \neq \emptyset$ $\forall\ A\in\{A\},\ q_0\in\rho_{Q_P}(T^{(1)})$.

2 - L'ensemble d'arborescences $\{B_2\},\ldots,\{B_n\}$ est un ensemble de sous-arborescences étrangères dépendant de la sous-arborescence $\{B_1\}$ de l'arborescence {A} $\Longleftrightarrow T^{(2)}(q_0,\mathcal{L}(\gamma(A))) \neq \emptyset\ \forall\ A\in\{A\},\ q_0\in\rho_{Q_0}(T^{(2)})$.

DEMONSTRATION

α) *Soient* T_i *les transducteurs de reconnaissances pour chaque arborescence* $\{B_i\}$ *définis dans le théorème R1.*

$$T_i = (V_i, V_E, V_{S_i}, V_{p_i}, Q_i, \delta_i, \{\{q_{0_i}\}\}, F_i, 1, 1)$$

où : $Q_i = \mathcal{P}(Q'_i)$, $F_i = \{Q'' \mid Q'' \cap \{q_{f_i}\} \neq \emptyset\}$

$$T = (V, V_E, V_S, V_p, Q, \delta, Q_0, F, 1, 1)$$

où : $V = V_E \cup V_S \cup V_p \cup \{\tilde{b}\}$

$V_S = V_E \times Q$

$V_p = Q$

$Q = \mathcal{P}(Q') \wedge Q' = \bigcup_{i=1}^{n} Q'_i$ *(On suppose que :* $\forall i,j, i\neq j \Longrightarrow Q_i \cap Q_j = \emptyset$*)*

$F = \{Q'' \mid \bigcup_{i=1}^{n} \{q_{f_i}\} \subset Q''\}$

δ : *tous les transducteurs* T_i *peuvent être simulés ensembles :*

$([, P_2, P_1, P'_1, D, ([, P_1), G) \in \delta([, P_2, P_1) \Longleftrightarrow$

$$P_1 = \bigcup_{i=1}^{n} Q_{1_i},\ P_2 = \bigcup_{i=1}^{n} Q_{2_i},\ P'_1 = \bigcup_{i=1}^{n} Q'_{1_i} \wedge$$

$$\forall i : (Q_{1_i} = P_1 \cap Q_i,\ Q_{2_i} = P_2 \cap Q_i,\ Q'_{1_i} = P'_1 \cap Q_i) \wedge$$

$([, Q_{2_i}, Q_{1_i}, Q'_{1_i}, D, ([, Q_{1_i}), G) \in \delta_i([, Q_{2_i}, Q_{1_i})$

$(], \varepsilon, P'_1, D, (], P_1), G) \in \delta(], P_2, P_1) \Longleftrightarrow$

$$P_1 = \bigcup_{i=1}^{n} Q_{1_i},\ P_2 = \bigcup_{i=1}^{n} Q_{2_i},\ P'_1 = \bigcup_{i=1}^{n} Q'_{1_i} \wedge$$

$$\forall i : (Q_{1_i} = P_1 \cap Q_i,\ Q_{2_i} = P_2 \cap Q_i,\ Q'_{1_i} = P'_1 \cap Q_i) \wedge$$

$(], \varepsilon, Q'_{1_i}, D, (], Q_{1_i}), G) \in \delta_i(], Q_{2_i}, Q_{1_i})$

β) Ensembles particuliers de sous-arborescences

1 - Recherche d'ensembles indépendants

Le transducteur reconnaissant un ensemble de sous-arborescences indépendantes est semblable au transducteur précédent. Seules diffèrent les transitions concernant les états finaux. Soient T_i, les transducteurs de reconnaissances pour chaque arborescence $\{B_i\}$:

$$T_i = (V_i, V_E, V_{S_i}, V_{p_i}, Q_i, \delta_i, \{\{q_{0_i}\}\}, F_i, 1, 1)$$

où : $Q_i = \mathcal{P}(Q'_i)$

$$F_i = \{Q'' \mid Q'' \cap \{q_{f_i}\} \neq \emptyset\}$$

Alors : $T^{(1)} = (V^{(1)}, V_E, V_S^{(1)}, V_p^{(1)}, Q^{(1)}, \delta^{(1)}, Q_0^{(1)}, F^{(1)}, 1, 1)$

$$V^{(1)} = V_E \cup V_S^{(1)} \cup Q^{(1)} \cup \{\text{Ł}\}$$

$$V_S^{(1)} = V_E \times Q^{(1)}$$

$$Q^{(1)} = \mathcal{P}(Q^{(1)}) \wedge Q^{(1)} = \bigcup_{i=1}^{n} Q'_i \cup (\mathcal{P}(\bigcup_{i=1}^{n} \{q_{f_i}\}))$$

$$F^{(1)} = \{Q'' \mid Q'' \cap \{\bigcup_{i=1}^{n} \{q_{f_i}\}\} \neq \emptyset\}$$

Par la suite, on confondra l'élément q_{f_i} avec l'ensemble $\{q_{f_i}\}$

$\delta^{(1)}$:

$$([, P_2, P_1, P'_1, D, ([, P_1), G) \in \delta^{(1)} ([, P_2, P_1) \Longleftrightarrow$$

$$P_1 = Q_{F_1} \cup (\bigcup_{i=1}^{n} Q_{1_i}),\ P_2 = Q_{F_2} \cup (\bigcup_{i=1}^{n} Q_{2_i}),\ P'_1 = \bigcup_{i=1}^{n} Q'_{1_i} \wedge$$

$$\forall i : (Q_{1_i} = P_1 \cap Q_i,\ Q_{2_i} = F_2 \cap Q_i,\ Q'_{1_i} = P'_1 \cap Q_i)$$

$$\wedge ([, Q_{2_i}, Q_{1_i}, Q'_{1_i}, D, ([, Q_{1_i}), G) \in \delta_i ([, Q_{2_i}, Q_{1_i})$$

$$(], \varepsilon, P'_1, D, (], P_1), G) \in \delta^{(1)} (], P_2, P_1) \Longleftrightarrow$$

$$P_1 = Q_{F_1} \cup (\bigcup_{i=1}^{n} Q_{1_i}),\ P_2 = Q_{F_2} \cup (\bigcup_{i=1}^{n} Q_{2_i}),\ P'_1 = Q'_{F_1} \cup Q''_{F_1} \cup (\bigcup_{i=1}^{n} Q'_{1_i}),$$

$$Q''_{F_1} = \bigcup_{i=1}^{n} Q''_{1_i} \wedge \forall i : (Q_{1_i} = (P_1 \cap Q_i) - \{q_{f_i}\}, Q_{2_i} = (P_2 \cap Q_i) -$$

$$\{q_{f_i}\}, Q'_{1_i} = (P'_1 \cap Q_i) - \{q_{f_i}\}, Q''_{1_i} = P'_1 \cap \{q_{f_i}\} \wedge$$

$$(\square, \varepsilon, Q'_{1_i} \cup Q''_{1_i}, D, \square, Q_{1_i}), G) \in \delta_i(\square, Q_{2_i}, Q_{1_i}) \wedge$$

$$Q'_{F_1} = \bigcup_{Q \in Q_{F_1} \cup Q''_{F_1}} \; \bigcup_{Q' \in Q_{F_2}} (\{Q \cup Q'\})$$

Les ensembles d'états finaux représentent les différentes arborescences pouvant être des sous-arborescences indépendantes. Après lecture d'un mot de L, les états finaux correspondant aux sous-arborescences dépendant de ce point (racine du mot lu) sont dans les états Q_{F_1} *et* Q''_{F_1}*, et les états finaux correspondant aux sous-arborescences déjà rencontrées dans un autre mot de L (et donc indépendantes des premières) sont dans l'état* Q_{F_2}*. Le calcul de* Q'_{F_1} *est directement déduit de cette propriété.*

2 - Recherche d'ensembles dépendants

La recherche de l'ensemble de sous-arborescences indépendantes doit être effectuée à partir de chaque feuille de l'arborescence $\{B_1\}$*. En supposant que cette recherche doit être effectuée à partir d'une seule feuille de* $\{B_1\}$*, par exemple* $\{X_{B_1}\}$*, le transducteur* $T^{(2)}$ *est défini de la façon suivante :*

Soit $T^{(1)}$ *le transducteur de reconnaissance pour l'ensemble des arborescences* $\{B_2\}, \ldots, \{B_n\}$*, et soit T le transducteur de reconnaissance pour l'arborescence* $\{B_1\}$*.*

$$T = (V, V_E, V_S, V_p, Q, \delta, \{\{q_0\}\}, F, 1, 1)$$

où : $Q = \mathcal{P}(Q')$

$$T^{(1)} = (V^{(1)}, V_E, V_S^{(1)}, V_p^{(1)}, Q^{(1)}, \delta^{(1)}, Q_0^{(1)}, F^{(1)}, 1, 1)$$

où : $Q^{(1)} = \mathcal{P}(Q^{(1)\prime})$

Alors : $T^{(2)}$ *est défini par :*

$$T^{(2)} = (V^{(2)}, V_E, V_S^{(2)}, V_p^{(2)}, Q^{(2)}, \delta^{(2)}, Q_0^{(2)}, F^{(2)}, 1, 1)$$

avec : $V^{(2)} = V_E \cup V_S^{(2)} \cup V_p^{(2)} \cup \{\flat\}$

$$V_S^{(2)} = V_E \times Q^{(2)}$$

$$V_p^{(2)} = Q^{(2)}$$

$$Q^{(2)} = \mathcal{P}(Q^{(2)'}) \wedge Q^{(2)'} = Q' \cup Q^{(1)'} \quad (On\ suppose\ que\ Q' \cap Q^{(1)'} = \emptyset$$

$$Q_0^{(2)} = \{\{q_0\}\}$$

$$F^{(2)} = F$$

Soient q_{X_1}, q_{X_2}, q_{X_3} *les états prédécesseurs internes et successeurs associés à la feuille X de* B_1.

$\delta^{(2)}$:

$$([, P_2, P_1, P'_1, D, ([, P_1), G) \in \delta^{(2)} ([, P_2, P_1) \iff$$

$$P_2 = Q_2 \cup R_2,\ P_1 = Q_1 \cup R_1,\ P'_1 = ((Q'_1 \cup R'_1) - \{q_{X_2}\}) \cup Q''_1 \wedge$$

$$Q_2 = P_2 \cap Q,\ Q_1 = P_1 \cap Q,\ Q'_1 = P'_1 \cap Q,\ R_2 = P_2 \cap Q^{(1)},$$

$$R_1 = P_1 \cap Q^{(1)}$$

$$R'_1 = P'_1 \cap (Q^{(1)} - Q_0^{(-)}) \wedge$$

$$Q''_1 = \begin{cases} Q_0^{(1)} & si\ Q'''_1 = \{q_{X_2}\} \\ \emptyset & si\ Q'''_1 = \emptyset \end{cases} \wedge$$

$$([, Q_2, Q_1, Q'_1 \cup Q'''_1\ D, ([, Q_1), G) \in \delta([, Q_2, Q_1) \wedge$$

$$([, R_2, R_1, R'_1 \cup Q_0^{(1)}, D, ([, R_1), G) \in \delta^{(1)} ([, R_2, R_1)$$

$$(], \varepsilon, P'_1, D, (], P_1), G) \in \delta^{(2)} (], P_2, P_1) \iff$$

$$P_2 = Q_2 \cup R_2,\ P_1 = Q_1 \cup R_1,\ P'_1 = Q'_1 \cup R'_1$$

$$Q''_1 = \begin{cases} \{q_{X_2}\} \text{ si } R_1 \in F^{(1)} \wedge q_{X_1} \in Q_2 \\ \emptyset \text{ sinon} \end{cases} \wedge$$

$$(\Gamma,\ \varepsilon,\ Q'_1,\ D,\ (\Gamma,\ Q_1 \cup Q''_1),\ G) \in \delta(\Gamma,\ Q_2,\ Q_1 \cup Q''_1) \wedge$$

$$(\Gamma,\ \varepsilon,\ R'_1,\ D,\ (\Gamma,\ R_1),\ G) \in \delta(\Gamma,\ R_2,\ R_1)$$

Le transducteur $T^{(1)}$ commence à être simulé par le transducteur $T^{(2)}$ qu'après lecture de la feuille $\{X_{B_1}\}$ de l'arborescence $\{B_1\}$, c'est-à-dire après apparition dans l'état courant de l'élément q_{X_1}.

Cette simulation continue tant que le niveau de cette feuille n'est pas complètement analysé (fonctionnement de la pile sur un mot de L pour les états initiaux de $T^{(1)}$). Cette simulation conduit à la présence de l'état q_{X_3} après lecture de la fermeture du niveau de $\{X_{B_1}\}$ si le transducteur $T^{(1)}$ a reconnu l'ensemble des sous-arborescences $\{B_2\},\ldots,\{B_n\}$.

5 - THEOREME R1' : SOUS-ARBORESCENCES ORIENTEES

Pour toutes arborescences orientées $\{B\}_o$ (rep. partiellement orientée $\{B_{B'}\}_o$), il existe un transducteur à pile déterministe T ne dépendant que de $\{B\}_o$ (rep. de $\{B_{B'}\}_o$), tel que les propriétés suivantes soient équivalentes pour toutes arborescences $\{A\}_o$.

$\{B\}_o$ est une sous-arborescence orientée de l'arborescence $\{A\}_o \iff$

$$T(q_0,\ \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall\, A \in \{A\},\ q_0 \in \rho_{Q_0}(T)$$

respectivement :

$\{B_{B'}\}_o$ possède la propriété :

$$\exists B_{B'_\epsilon} \in \{B_{B'}\} : \exists X \in \gamma(A) : B \subset \mathcal{E}_X(\gamma(A))$$

$$\iff T(q_0,\ \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall\, A \in \{A\}_o,\ \forall\, q_0 \in \rho_{Q_0}(T)$$

Ce qui est équivalent à la propriété :

$\{B_{B'}\}_o$ est une sous-arborescence partiellement orientée de l'arborescence partiellement orientée $\{A_{\pi_X(B')}\}_o$.

DEMONSTRATION

La démonstration diffère très peu de celle du théorème R1. Seule une contrainte supplémentaire correspondant à l'ordre doit être vérifiée.

Soit le transducteur T' construit dans le théorème R1 et soit un point $\{X_B\}$ *de l'arborescence* $\{B\}$.

Nous avons si $\overset{p}{\underset{i=1}{\cup}} \{q_{f_i}\}$ *sont les états sous-successeurs associés à* $\{X_B\}$ *et* $\mathcal{O}_X$ *l'ordre des descendants de* X_B :

$$\delta'(\square, Q_2, Q_1) = (\square, \varepsilon, Q'_i, D, \square, Q_1), G)$$

où :
$$Q_2 = Q'_2 \cup \{\overset{k}{\underset{i=1}{\pi}} R_i\} \wedge \{\overset{k}{\underset{i=1}{\pi}} R_i\} \in \overset{p}{\underset{k=1}{\cup}} (\mathcal{P}(\overset{p}{\underset{i=1}{\cup}} \{q_{f_i}\}))^k \cup \emptyset$$

$$\wedge Q'_2 \cap \overset{p}{\underset{k=1}{\cup}} (\mathcal{P}(\overset{p}{\underset{i=1}{\cup}} \{q_{f_i}\})) = \emptyset$$

$$Q_R = \begin{cases} R = \{q_{f_i} \mid q_{f_i} \in Q''_1\} \\ \{\overset{h+1}{\underset{i=1}{\pi}} R_i\} \Longleftrightarrow R_{h+1} = R \wedge \exists\{q_1, \ldots, q_{h+1}\} : \\ \qquad (q_i \neq q_j \ \forall\ 1 \leqslant i \neq j \leqslant h+1 \wedge q_i \in R_i) \\ \qquad \text{contrainte supplémentaire d'ordre :} \\ \qquad q_i = q_{f_{h_i}} \wedge q_j = q_{f_{h_j}} \Longrightarrow \mathcal{O}_X(q_{f_{h_i}}) < \mathcal{O}_X(q_{f_{h_j}}) \\ \{\overset{h}{\underset{i=1}{\pi}} R_i\} \text{ sinon} \end{cases}$$

$$Q_F = \begin{cases} \{q_f\} \text{ si } (\mathcal{P}(\bigcup_{i=1}^{p} \{q_{f_i}\}))^p \cap Q_1 \neq \emptyset \\ \\ \emptyset \text{ sinon} \end{cases}$$

Alors :

$$Q'_1 = Q'_2 \cup (Q''_1 - R) \cup Q_R \cup Q_F$$

L'état successeur d'un point ne peut apparaître dans l'état courant du transducteur que si toutes les sous-arborezcences dépendantes de ce point ont été reconnues avec la contrainte imposée par l'ordre.

6 - THEOREME R2'

Soit $\{B\}_o$ une arborescence orientée (respectivement partiellement orientée $\{B_B,\}_o$) quelconque, il existe un transducteur à pile déterministe T ne dépendant que de $\{B\}_o$ (respectivement $\{B_B,\}_o$) tel que les propriétés suivantes soient équivalentes pour toutes arborescences orientées $\{A\}_o$:

- T'_1 : l'arborescence orientée $\{B\}_o$ est une sous-arborescence orientée sommet de l'arborescence orientée $\{A\}_o \iff$

$$T'_1(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_1)$$

respectivement, l'arborescence partiellement orientée $\{B_B,\}_o$ possède la propriété :

$$\exists B_B, \in \{B_B,\}_o : B \subset \gamma(A)$$

$$\iff T'_1(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_1)$$

- T'_2 : l'arborescence orientée $\{B\}_o$ est une sous-arborescence feuille de l'arborescence orientée $\{A\}_o \iff$

$$T'_2(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_2)$$

respectivement, l'arborescence partiellement orientée $\{B_B,\}_o$ possède la propriété :

$\exists B_B, \in \{B_B,\}_o$: $\{B\}_o$ est une sous-arborescence feuille de l'arborescence orientée $\{A\}_o \iff T'_2(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_2)$.

- T'_3 : l'arborescence orientée $\{B\}_o$ est une sous-arborescence complète de l'arborescence orientée $\{A\}_o \Longleftrightarrow$

$$T'_3(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall\, A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_3)$$

respectivement, l'arborescence partiellement orientée $\{B_B,\}_o$ possède la propriété :

$\exists B_B, \in \{B_B,\}_o$: $\{B\}_o$ est une sous-arborescence complète de l'arborescence orientée $\{A\}_o \Longleftrightarrow T'_3(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall\, A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_3)$

- T'_4 : $\{B\}_o = \{A\}_o \Longleftrightarrow T'_4(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset \quad \forall\, A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_4)$

respectivement, l'arborescence partiellement orientée $\{B_B,\}_o$ possède la propriété :

$$\exists B_B, \in \{B_B,\}_o : \{B\}_o = \{A\}_o \Longleftrightarrow T'_4(q_0, \mathcal{L}(\gamma(A))) \neq \emptyset$$

$$\forall\, A \in \{A\}_o,\ q_0 \in \rho_{Q_0}(T'_4)$$

7 - RECONNAISSANCE DES SOUS-ARBORESCENCES ETIQUETEES

Le cas des arborescences étiquetées est identique au cas précédent. Le langage de représentation diffère seulement par la présence d'étiquettes. Les transducteurs de reconnaissance sont les mêmes que ceux donnés précédemment. Ils ont en plus un prédicat d'étiquettes correspondant à chaque point de l'arborescence $\{(B,\rho)\}$.

Si X est un point de l'élément (B,ρ), le prédicat P_X associé à ce point est tel que :

- soit $q_{X_1}, \bigcup_{i=1}^{p} \{q_{0_i}\}$ les états prédécesseurs et sous-prédécesseurs associés à ce point X :

Les états sous-prédécesseurs apparaissent dans l'état courant du transducteur à partir de l'état q_{X_1} et de la lecture d'un symbole σ de V (correspondant au symbole "[" dans le transducteur précédemment construit), si et seulement si : $P_X(\sigma)$ est vrai.

Dans le cas où on impose en plus un prédicat portant sur l'ensemble des étiquettes de V, le transducteur devient alors non déterministe et le prédicat est alors évalué lors de la reconnaissance du dernier point de l'arborescence $\{(B,\rho)\}$.

C - THEOREMES DE TRANSFORMATION

La réalisation d'une transformation par un transducteur composé s'effectue en trois temps. Après la recherche de la sous-arborescence concernée par la transformation, les différents éléments dépendant de chaque point sont séparés puis synthétisés suivant l'arborescence résultante.

Exemple :

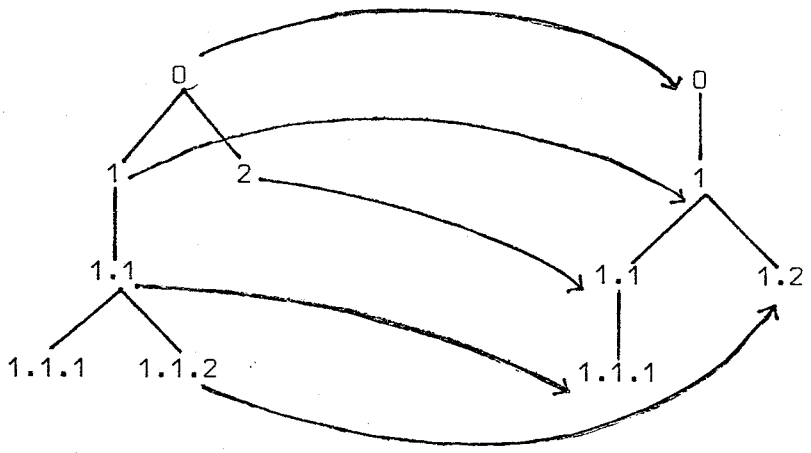
Soit la transformation (A, B, τ) définie par :

A = {0,1,1.1,1.1.1,1.1.2,2}

B = {0,1,1.1,1.1.1,1.2}

= (0 → 0, 1 → 1, 2 → 1.1, 1.1 → 1.1.1, 1.1.2 → 1.2)

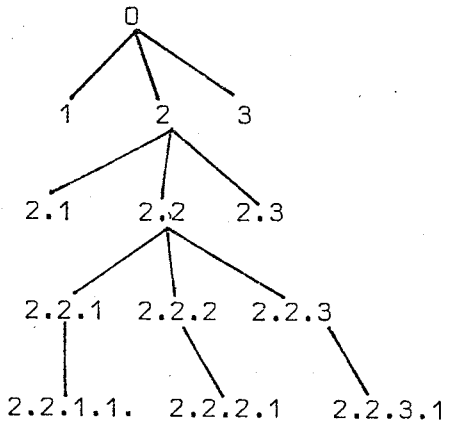
A =



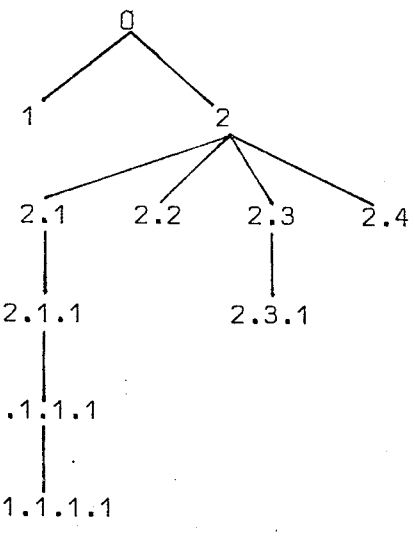
= B

Alors, soit l'arborescence C transformée en l'arborescence C' par la transformation (A,B,τ) :

C =



⊢ (A,B,τ)



C = {0,1,2,2.1,2.2,2.2.1,2.2.1.1,2.2.2,2.2.2.1,2.2.3,2.2.3.1,2.3,3}

C' = {0,1,2,2.1,2.1.1,2.1.1.1,2.1.1.1.1,2.2,2.3,2.3.1,2.4}

$\mathcal{L}$(C) = [[ ] [[ ] [[[ ]] [[ ]] [[ ]]] [ ]] [ ]]

$\mathcal{L}$(C') = [[ ] [[[[[ ]]]] [ ] [[ ]] [ ]]]

La transformation s'effectue de la façon suivante :

1 - Reconnaissance (déterministe) de la présence de l'arborescence {A} comme sous-arborescence de l'arborescence {C}.

2 - Choix d'une sous-arborescence {A} (choix déterministe si on implique une convention sur le choix de la figure, non déterministe sinon).
Le choix étant fait, on effectue un éclatement des différents éléments.

3 - Synthèse des éléments éclatés de façon à former l'arborescence C'.
Si le choix de la sous-arborescence correspond à A' = {0,2,2.2,2.2.1,2.2.3,3},
nous avons les différents éléments suivants :

[[ ] [[ ] [[[ ]] [[ ]] [[ ]]] [ ]] [ ]]

⇓

élément sommet droite *

élément dépendant de la racine de la sous-arborescence [ ] *

élément dépendant du point 2 [ ] [ ] *

élément dépendant du point 2.2 [[ ]] *

élément dépendant du point 2.2.1 [ ] *

élément dépendant du point 2.2.3 [ ] *

élément dépendant du point 3 *

élément sommet gauche *

⇓

[[[[[[ ]]]] [[ ]] [ ] [ ]] [ ]]

ce mot est équivalent au mot $\mathcal{L}$(C') :

[[ ] [[[[[ ]]]] [ ] [[ ]] [ ]]]

1 - THEOREME TR1

Pour toute transformation (A,B,τ), il existe un transducteur à pile T composé linéaire d'ordre trois, ne dépendant que de cette transformation, et possédant la propriété suivante :

$$\forall \{C\} : T(q_0, \mathcal{L}(\gamma(C))) \neq \emptyset \iff \exists \{C'\} : \{C\} \underset{(A,B,\tau)}{\longmapsto} \{C'\} \wedge$$

$$\mathcal{L}(\gamma(C')) = \pi_2(T(q_0, \mathcal{L}(\gamma(C))))$$

DEMONSTRATION

Le transducteur T simulant une transformation est un transducteur composé d'ordre 3, $C_T(1)_T(2)_T(3)$. *La puissance d'entrée des transducteurs* $T^{(1)}$ *et* $T^{(2)}$ *et la puissance de sortie des transducteurs* $T^{(1)}$ *et* $T^{(3)}$ *sont égales à un. La puissance de sortie du transducteur* $T^{(2)}$ *et la puissance d'entrée du transducteur* $T^{(3)}$ *sont fonction de l'arborescence* $\{A\}$ *et égales au nombre de points de cette arborescence plus deux.*

Le transducteur $T^{(1)}$ *est le transducteur de reconnaissance construit précédemment (théorème R1 dans le cas présent).*

$$T^{(1)} = (V^{(1)}, V_E, V_S^{(1)}, V_p^{(1)}, Q^{(1)}, \delta^{(1)}, \{\{q_0\}\}, F^{(1)}, 1, 1)$$

Où : $V^{(1)} = V_E \cup V_S^{(1)} \cup V_p^{(1)} \cup \{\bar{b}\}$

$$V_S^{(1)} = V_E \times Q^{(1)}$$

$$V_p^{(1)} = Q^{(1)}$$

$$Q^{(1)} = \mathcal{P}(Q')$$

$$F^{(1)} = \{Q'' \mid Q'' \cap \{q_f\} \neq \emptyset\}$$

et $\delta^{(1)}(\sigma, Q_2, Q_1) = (\sigma', Q'_2, Q'_1, M, \gamma, M') \iff \delta'(\sigma, Q_2, Q_1) =$

$$(\sigma', Q'_2, Q''_1, M, \gamma, M') \wedge Q'_1 = Q''_1 \cup \{q_0\} \cup$$

$$(Q_1 \cap \{q_f\})$$

Le résultat du transducteur $T^{(1)}$ est formée par l'image miroir du mot d'entrée auquel est associé pour chaque symbole l'ensemble des états de Q' présent lors de la recherche de la sous-arborescence {A}. La lecture de l'arborescence par le transducteur $T^{(2)}$ s'effectue dans le sens inverse de celle effectuée par $T^{(1)}$. Il suffit de reconnaître les symboles correspondants aux états conduisant à l'état final q_f et de séparer les différents mots "dépendants" de ces éléments.

$$T^{(2)} = (V^{(2)}, V_S^{(1)}, V_S^{(2)}, V_p^{(2)}, Q^{(2)}, \delta(2), Q_O^{(2)}, F^{(2)}, 1, n)$$

Où :

$$V^{(2)} = V_S^{(1)} \cup V_S^{(2)} \cup V_p^{(2)} \cup \{\mathfrak{b}\}$$

$$V_S^{(2)} = \{[,],*\}$$

$$V_p^{(2)} = Q^{(2)}$$

$$Q^{(2)} = F^{(1)} \cup (Q^{(1)} \cup \{q_1\}) \times \{1,\ldots,n\} \cup \{q_1\}$$

$$Q_O^{(2)} = F^{(1)}$$

$$F^{(2)} = \{q_1\}$$

n : nombre de points de l'arborescence {A} augmentés de deux.

Soit Θ une fonction d'ordre : application de A dans N^+ vérifiant les propriétés suivantes :

$$\Theta(Y) = \Theta(Y') \Longrightarrow Y = Y' \text{ (injective)}$$

$$Y \in A \wedge i \leq \Theta(Y) \Longrightarrow \exists Y' \in A : \Theta(Y') = i \text{ (surjective sur un segment initial de } N^+).$$

$\delta^{(2)}$:

Les transitions initiales ont pour but de rechercher le début du mot dont $\mathcal{L}(\Upsilon(A))$ est un sous-mot initial :

$$\delta^{(2)}((([, Q_2), Q_3, Q_1) = ((([, Q_2), \varepsilon, Q_1, D, (\Upsilon_1,\ldots,\Upsilon_n), G^n)$$

$$\forall\, Q_1 \in F^{(1)},\ Q_3 \in V_p^{(2)},\ ([, Q_2) \in V_S^{(1)},\ \Upsilon_i = \varepsilon\ \forall\ i \neq n,\ \Upsilon_n = [$$

$$\delta^{(2)}((\,],\ Q_2),\ Q_3,\ Q_1) = ((\,],\ Q_2),\ Q_3 \cdot q_1,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ Q_1 \in F^{(1)},\ Q_3 \in V_p^{(2)} \cup \emptyset,\ (\,],\ Q_2) \in V_S^{(1)},\ \gamma_i = \varepsilon\ \forall\ i \neq n,\ \gamma_n =]\ \wedge :$$

$$\delta'(\,],\ \emptyset,\ Q_2) = (\,],\ \varepsilon,\ Q'_1,\ D,\ (\,],\ Q_2),\ G) \Longrightarrow Q'_1 \notin F^{(1)}$$

Dans le cas où la lecture du mot d'entrée aboutit au début du mot reconnu, la transition à effectuer doit permettre un début de séparation des différents éléments.

$$\delta^{(2)}((\,],\ Q_2),\ Q_3,\ Q_1) = ((\,],\ Q_2),\ Q_3 \cdot q_1,\ Q'_1,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ Q_1 \in F^{(1)},\ Q_3 \in V_p^{(1)} \cup \emptyset,\ (\,],\ Q_2) \in V_S^{(1)},\ \gamma_i = \varepsilon\ \forall\ i \neq k,\ \gamma_k = *$$

et tel que :

- *k est l'ordre du point plus un dont q_f est l'état successeur associé.*
- $\delta'(\,],\ \emptyset,\ Q_2) = (\,],\ \varepsilon,\ Q''_1,\ D,\ (\,],\ Q_2),\ G) \Longrightarrow Q''_1 \in F^{(1)}$
- $Q'_1 = (Q'''_1,\ k)$ *et :*

$$Q'''_1 = \begin{cases} \text{Si le point Y dont } q_f \text{ est l'état successeur associé} \\ \text{n'est pas une feuille, alors } Q'''_1 \text{ est l'ensemble} \\ \text{des états sous-successeurs associés à ce point Y.} \\ \emptyset \text{ sinon} \end{cases}$$

On peut remarquer que lorsque le point Y n'est pas une feuille, l'ensemble Q'''_1 est contenu dans l'ensemble Q_2 et lui seul permet la réalisation de la condition sur δ'.

La progression de la séparation des différents éléments est donnée par les transitions suivantes :

$$\delta^{(2)}((\,],\ Q_2),\ Q_3,\ (Q_1,\ m)) = ((\,],\ Q_2),\ Q_3 \cdot (Q''_1, m),\ (Q'_1,\ k),\ D,$$

$$(\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ (\,],\ Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ (Q_1,\ m) \in Q^{(1)} \times \{1,\ldots,n\},\ \gamma_i = \varepsilon\ \forall\ i \neq k,$$

$$\gamma_k = *\ \text{et tel que :}$$

$- \delta'([, \emptyset, Q_2) = ([, \varepsilon, Q'_2, D, [, Q_2), G) \Longrightarrow Q_1 \cap Q'_2 \neq \emptyset$

- Soit un état q, élément de $Q_1 \cap Q'_2$. *(Pour rendre le transducteur déterministe, on peut choisir systématiquement l'état de l'intersection suivant un critère défini, comme par exemple, celui associé au point d'ordre maximal). Soit Y, le point dont q est l'état successeur associé.*

$- Q'_1 =$ { *Si Y n'est pas une feuille, alors* Q'_1 *est l'ensemble des états sous-successeurs associés à Y.*

$\emptyset$ *sinon*

$- k = \Theta(Y) + 1$

$- Q''_1 = Q_1 - \{q\}$

$\delta^{(2)}(([, Q_2), Q_3, (Q_1, m)) = (([, Q_2), \varepsilon, Q_3, D, (\gamma_1, \ldots, \gamma_n), G^n)$

$\forall\ ([, Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ Q_3 \neq q_1,\ (Q_1, m) \in Q^{(1)} \times \{1, \ldots, n\},$

$\gamma_i = \varepsilon\ \forall\ i$

Dans le cas où une transition correspondant à un point de A ne peut être effectuée, il est nécessaire de placer le mot de L correspondant sur la bande active (bande dont le numéro est dans l'état courant du transducteur). Cette écriture du mot de L s'effectue par une recherche en pile de la fin de ce mot.

$\delta^{(2)}([, Q_2), Q_3, (Q_1, m)) = ([, Q_2), Q_3.(Q_1, m), (q_1, m),$

$(\gamma_1, \ldots, \gamma_n), G^n)$

$\forall\ [, Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ (Q_1, m) \in Q^{(1)} \times \{1, \ldots, n\},\ \gamma_i = \varepsilon$

$\forall\ i \neq m,\ \gamma_m =],$

et tel que :

$\delta'([, \emptyset, Q_2) = ([, \varepsilon, Q'_2, D, [, Q_2), G) \Longrightarrow Q_1 \cap Q'_2 = \emptyset$

$\delta^{(2)}([, Q_2), Q_3, (q_1, m)) = ([, Q_2), Q_3.(q_1, m), (q_1, m), D,$

$(\gamma_1, \ldots, \gamma_n), G^n)$

$$\forall\ (]\ ,\ Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ (q_1,\ m) \in \{q_1\} X \{1,\ldots,n\},\ \gamma_i = \varepsilon\ \forall\ i \neq m,$$

$$\gamma_m =]$$

$$\delta^{(2)}((]\ ,\ Q_2),\ Q_3,\ (q_1,\ m)) = ((]\ ,\ Q_2),\ \varepsilon,\ Q_3,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ ([\ ,\ Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ (q_1,\ m) \in \{q_1\} X \{1,\ldots,n\},\ \gamma_i = \varepsilon\ \forall\ i \neq m,$$

$$\gamma_m = [$$

Lorsque le mot d'entrée a été complètement séparé, le transducteur se trouve dans l'état q_1. *Il suffit alors de placer le reste du mot d'entrée sur la première bande.*

$$\delta^{(2)}((]\ ,\ Q_2),\ q_1,\ (Q_1,\ m)) = ((]\ ,\ Q_2),\ \varepsilon,\ q_1,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ ([\ ,\ Q_2) \in V_S^{(1)},\ (Q_1,\ m) \in Q^{(1)}\ X\{1,\ldots,n\},\ \gamma_1 = *,\ \gamma_i = \varepsilon\ \forall\ i \neq 1$$

$$\delta^{(2)}((]\ ,\ Q_2),\ Q_3,\ q_1) = ((]\ ,\ Q_2),\ Q_3.q_1,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ (]\ ,\ Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ \gamma_1 =],\ \gamma_i = \varepsilon\ \forall\ i \neq 1$$

$$\delta^{(2)}((]\ ,\ Q_2),\ Q_3,\ q_1) = ((]\ ,\ Q_2),\ \varepsilon,\ Q_3,\ D,\ (\gamma_1,\ldots,\gamma_n),\ G^n)$$

$$\forall\ ([\ ,\ Q_2) \in V_S^{(1)},\ Q_3 \in V_p^{(2)},\ \gamma_1 = [,\ \gamma_i = \varepsilon\ \forall\ i \neq 1$$

A l'arrêt du transducteur $T^{(2)}$, *les différents éléments du mot d'entrée ont été séparés. Les bandes 1 et n forment le mot correspondant au sommet de la sous-arborescence et sur chaque bande de rang k+1, se trouvent les mots correspondant au descendant du point d'ordre k. Il suffit alors d'écrire l'arborescence de sortie en réalisant la fonction* τ :

$$T^{(3)} = (V^{(3)},\ V_S^{(2)},\ V_E,\ V_p^{(3)},\ Q^{(3)},\ \delta^{(3)},\ Q_0^{(3)},\ F^{(3)},\ n,\ 1)$$

Soit le transducteur de reconnaissance T'' *construit pour l'arborescence* $\{B\}$.

$$T'' = (V'', V_E, V''_S, V''_p, Q'', \delta'', \{\{q''_0\}\}, F'', 1, 1)$$

Où : $V'' = V''_E \cup V''_S \cup V''_p \cup \{\tilde{b}\}$

$V_E = \{[,]\}$

$V''_S = V_E \times Q''$

$V''_p = Q''$

$Q'' = \mathcal{P}(Q''')$

$F'' = \{Q_1 \mid Q_1 \cap \{q''_f\} \neq \emptyset\}$

et : $\delta''(\sigma, Q_2, Q_1) = (\sigma', Q'_2, Q'_1, M, \gamma, M') \Longleftrightarrow \delta'''(\sigma, Q_2, Q_1) =$

$(\sigma', Q'_2, Q''_1, M, \gamma, M') \wedge$

$Q'_1 = Q''_1 \cup \{q''_0\} \cup (Q_1 \cap \{q''_f\})$

Alors : $V^{(3)} = V_S^{(2)} \cup V_E \cup V_p^{(3)} \cup \{\tilde{b}\}$

$V_p^{(3)} = Q'''$

$Q^{(3)} = F^{(2)} \cup Q'''$

$Q_0^{(3)} = F^{(2)}$

$F^{(3)} = \{q''_f\}$

$\delta^{(3)}$:

Les premières transitions à effectuer concernent le mot représentant le sommet :

$$\delta^{(3)}((\sigma_1,\ldots,\sigma_n), \emptyset, Q_1) = ((\sigma_1,\ldots,\sigma_n), \varepsilon, Q_1, (D, N^{n+1}), \sigma_1, D)$$

$\forall\ (\sigma_1,\ldots,\sigma_n) \in (V_S^{(2)})^n,\ Q_1 \in F^{(1)}$ *et tel que* $\sigma_1 \neq *$

$$\delta^{(3)}((*, \sigma_2, \ldots, \sigma_n), \emptyset, Q_1) = ((*, \sigma_2, \ldots, \sigma_n), q_f, Q'_1, N^n, [, D)$$

$\forall (*, \sigma_2, \ldots, \sigma_n) \in (V_S^{(2)})^n$, $Q_1 \in F^{(1)}$, Q'_1 *étant l'ensemble des états sous-prédécesseurs associés à la racine de l'arborescence* $\{B\}$ *si ce point n'est pas une feuille,* $\emptyset$ *sinon.*

A l'intérieur de l'état courant du transducteur, deux sortes d'états apparaissent : les états prédécesseurs et les états successeurs.

Le transducteur construit l'arborescence $\{B\}$ *tant qu'un état sous-prédécesseur est présent dans l'état courant.*

$$\delta^{(3)}((\sigma_1, \ldots, \sigma_n), Q_2, Q_1) = ((\sigma_1, \ldots, \sigma_n), Q_2.Q''_1, Q'_1, N^n, [, D)$$

$\forall (\sigma_1, \ldots, \sigma_n) \in (V_S^{(2)})^n$, $Q_2 \in V_p^{(3)}$, $Q_1 \in Q'''$ *et tel que* Q_1 *contienne un état prédécesseur* q. *(Pour rendre* $T^{(3)}$ *déterministe, on peut choisir cet état suivant un critère défini comme dans le cas de* $T^{(2)}$*).*

Alors : $Q''_1 = (Q_1 - \{q\}) \cup \{q'\}$ *si* q' *est l'état successeur associé au point dont* q *est l'état prédécesseur associé.*

Q'_1 *est l'ensemble des états sous-prédécesseurs associés au point dont* q *est l'état prédécesseur associé si ce point n'est pas une feuille*

$Q'_1 = \emptyset$ *sinon*

L'élément de pile contient donc à chaque instant l'ensemble des états prédécesseurs non choisis et l'état successeur correspondant au point dont l'état prédécesseur a été choisi. Pour sortir un élément de pile, il faut donc "fermer" un point. Avant cette opération, il suffit de réaliser la fonction τ.

$$\delta^{(3)}((\sigma_1, \ldots, \sigma_n), Q_2, \emptyset) = ((\sigma_1, \ldots, \sigma_n), Q_2, \emptyset, (M_1, \ldots, M_n), \sigma, D)$$

$\forall (\sigma_1, \ldots, \sigma_n) \in (V_S^{(2)})^n$, $Q_2 \in V_p^{(3)}$ *et tel que :*

Q_2 *contient l'état successeur* q *dont le point associé* Y *est tel qu'il existe un point* Y' *de* A *possédant les propriétés suivantes :*

$- \tau(Y') = Y$

$- \theta(Y') = k$

$- \sigma_k \neq *$

$- \forall Y'' \in A : \tau(Y') = Y \wedge \theta(Y'') = h \Longrightarrow \sigma_h = * \vee h \geqslant k$

Alors : $\sigma = \sigma_k \wedge M_i = N \ \forall \ i \neq k, \ M_k = D$

$\delta^{(3)}((\sigma_1, \ldots, \sigma_n), Q_2, \emptyset) = ((\sigma_1, \ldots, \sigma_n), \varepsilon, Q'_2, N^n,], D)$

$\forall \ (\sigma_1, \ldots, \sigma_n) \in (V_S^{(2)})^n, \ Q_2 \in V_p^{(3)}$ *et tel que :*

Q_2 *contient l'état successeur q dont le point associé Y est tel*

que : $\forall \ Y' \in A : \tau(Y') = Y : \theta(Y') = k \Longrightarrow \sigma_k = *$

Le transducteur $T^{(3)}$ *se trouve dans l'état* q''_f *lorsque la fonction* τ *a été complètement appliquée. Il faut alors finir d'écrire le sommet de la sous-arborescence.*

$\delta^{(3)}((\sigma_1, \ldots, \sigma_n), \emptyset, q_f) = ((\sigma_1, \ldots, \sigma_n), \varepsilon, q_f, (N^{n-1}, D), \sigma_n, D)$

$\forall \ (\sigma_1, \ldots, \sigma_n) \in (V_S^{(2)})^n : \sigma_n \neq \flat$

REMARQUE

Le dernier transducteur $T^{(3)}$ *peut être un transducteur régulier car la profondeur de la pile est bornée et est indépendante de l'arborescence* {C} *d'entrée.*

2 - THEOREME TR2

Pour toute transformation $\{(A_i, B_i, \tau^i)\}_{i \in I}$, il existe un transducteur à pile T composé linéaire d'ordre trois, ne dépendant que de cette transformation et possédant les propriétés suivantes :

$$\forall \ \{C\} : T(q_0, \mathcal{L}(\gamma(C))) \neq \emptyset \Longleftrightarrow \exists \{C'\} : \{C\} \underset{\{(A_i, B_i, \tau^i)\}_{i \in I}}{\vdash} \{C'\}$$

$$\wedge \ \mathcal{L}(\gamma(C')) = \pi_2(T(q_0, \mathcal{L}(\gamma(C))))$$

DEMONSTRATION

La démonstration est identique à celle du théorème TR1. La seule contrainte est la recherche d'un ensemble de sous-arborescences étrangères $\{\{A_i\}\}_{i \in I}$. *L'existence du transducteur de reconnaissance de cet ensemble est donnée par le théorème R3.*

3 - TRANSFORMATIONS SPECIALES

Toutes les transformations spéciales nécessitent seulement des contraintes supplémentaires pour le transducteur de reconnaissance. Ces transducteurs sont donnés par le théorème R2 ; d'où le théorème :

Théorème TR3

Pour toute transformation spéciale $\{(A_i, B_i, \tau^i)\sigma_i\}_{i \in I}$, $\sigma_i \in \{c, f, s, I\}^*$, il existe un transducteur T, à pile composé linéaire et d'ordre 3, possédant la propriété :

$$\forall \{C\} : T(q_0, \mathcal{L}(\gamma(C))) \neq \emptyset \Longleftrightarrow \exists \{C'\} :$$

$$\{C\} \underset{\{(A_i, B_i, \tau^i)\}_{i \in I}}{\longmapsto} \{C'\} \wedge \mathcal{L}(\gamma(C')) = \pi_2(T(q_0, \mathcal{L}(\gamma(C))))$$

4 - TRANSFORMATIONS ETIQUETEES

Les transformations étiquetées sont réalisables par un transducteur à pile composé linéaire d'ordre 3. Le transducteur de recherche est également donné par le théorème R1 auquel est assccié en chaque point un prédicat d'étiquettes.

Théorème TR4

Pour toute transformation étiquetée $(A, B, \tau, \rho_A, \{\rho_X\}_{X \in A})$, il existe un transducteur T, à pile composé linéaire d'ordre 3, possédant la propriété :

$$\forall \{(C, \rho)\} : T(q_0, \mathcal{L}(\gamma(C), \rho')) \neq \emptyset \wedge (\gamma(C), \rho') \sim (C, \rho)$$

$$\Longleftrightarrow \exists \{(C', \rho'')\} : \{(C, \rho)\} \underset{(A, B, \tau, \rho_A, \{\rho_X\}_{X \in A})}{\longmapsto} \{(C', \rho'')\} \wedge$$

$$\mathcal{L}(C', \rho'') = \pi_2(T(q_0, \mathcal{L}(\gamma(C), \rho')))$$

5 - GRAMMAIRE TRANSFORMATIONNELLE

Une grammaire transformationnelle peut être simulée par un transducteur composé à pile. Le transducteur de transformation dans ce cas doit fournir deux états finaux différents : un état pour lequel la transformation a été effectuée et un état pour le cas où la transformation n'a pu être effectuée. Dans le second cas, le transducteur doit fournir le mot d'entrée recopié ; ceci s'effectue en ajoutant un transducteur $T^{(4)}$ au transducteur de transformation donné par le théorème TR1. Ce transducteur a pour états initiaux tous les états de $T^{(1)}$ qui ne sont pas des états initiaux de $T^{(2)}$ et $T^{(1)}$ a alors comme ensemble d'états finaux l'ensemble de tous ces états. Le transducteur $T^{(4)}$ écrit alors l'entrée de gauche à droite (mouvement de sortie "G") en supprimant l'ensemble des états associés aux éléments de V_E. Ainsi, $T^{(1)}T^{(4)}$ réalisent l'application identique dans le cas où $T^{(1)}$ ne s'arrête pas dans un état initial de $T^{(2)}$.

Théorème G1

Pour toute grammaire transformationnelle G, il existe un transducteur à pile composé T possédant les propriétés suivantes :

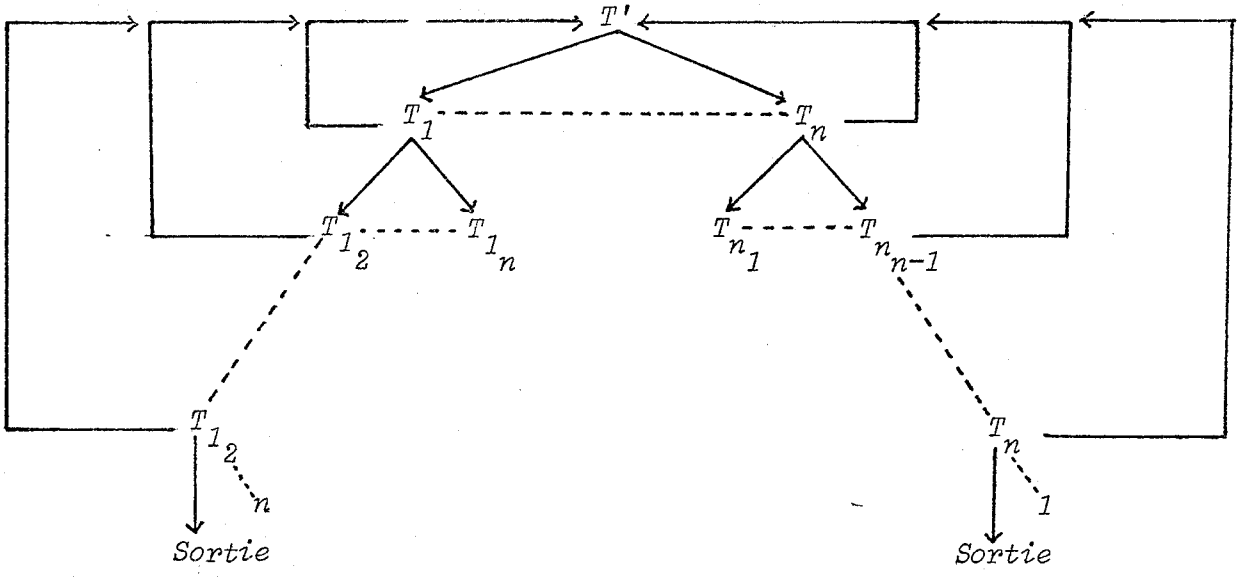
$$\forall \{C\} : T(q_0, \mathcal{L}(\gamma(C))) \neq \emptyset \Longleftrightarrow \Big[\exists \{C'\} : \{C\} \xrightarrow[G]{} \{C'\} \wedge \mathcal{L}(\gamma(C')) = \pi_2(T(q_0, \mathcal{L}(\gamma(C))))\Big]$$

$$\vee\ \pi_2(T(q_0, \mathcal{L}(\gamma(C)))) = \mathcal{L}(\gamma(C))$$

DEMONSTRATION

Le transducteur T est composé à partir du transducteur $\{T_i\}_{i\in I}$ *si chaque* T_i *réalise une transformation de la grammaire G.*

Un premier transducteur T', dont le but est de rendre la recherche de la première règle à appliquer non déterministe, donne le contrôle à chacun des T_i. *Dépendant d'un transducteur* T_i, *nous trouvons un ensemble de transducteurs* $\{T_{i_j}\}_{j\in I-\{i\}}$. *Un transducteur quelconque donne le contrôle au transducteur T'dans le cas où une transformation a été effectuée. Les derniers transducteurs donnent le contrôle à aucun autre, dans le cas où aucune transformation a été effectuée. Le transducteur est donc donné par le schéma :*



Si il existe une suite finie $j_1,\ldots,j_p$ *telle que :*

- $\forall\, h : \{C_{j_h}\} \vdash^{\{(A_i,B_i,\tau^i)\}_{i\in I_{j_h}}} \{C_{j_{h+1}}\}$

- $\{C\} = \{C_{j_1}\}$, $\{C'\} = \{C_{j_p}\}$

- $\forall\, h \in I'$ *la transformation n'est pas définie sur* $\{C_{j_p}\}$, *alors, il existe un chemin dans la composition des transducteurs tel que* $\ell(\gamma(C_{j_p}))\in\pi_2(T_W(q_{0_W}, u))$ *et où W est une permutation de I.*

6 - GRAMMAIRES TRANSFORMATIONNELLES PARTICULIERES

Les différentes grammaires transformationnelles définies dans le chapitre IV sont simulables par un transducteur à pile composé.
De plus, les différentes restrictions apportées sur l'application des règles donnent des restrictions sur la composition des transducteurs et conduisent soit à des compositions linéaires, soit à des compositions récursives.

D - CONSTRUCTIBILITE DES TRANSDUCTEURS DE TRANSFORMATIONS

La donnée d'un transducteur de transformation par ses différents états rend l'application peu commode. La donnée d'un langage souple pour la définition des transducteurs permet de construire des systèmes simulant des grammaires transformationnelles (Système C.E.T.A.).

1 - THEOREME C1

La famille des transducteurs de reconnaissance est G-constructible, relativement au langage L.

DEMONSTRATION

La démonstration du théorème R1 est un schéma de récursion sur le langage L.

2 - THEOREME C2

La famille des transducteurs de reconnaissance de sous-arborescences ayant au plus n-points est P-constructible, relativement au langage L.

DEMONSTRATION

Le schéma de récursion du théorème R1 nécessite d'empiler les différents états construits. Ainsi, le vocabulaire de pile ne peut être borné que si le nombre de points de l'arborescence à reconnaître est borné.

3 - CONSTRUCTION DES TRANSDUCTEURS DE TRANSFORMATION

Le transducteur de transformation donné par le théorème TR1 peut être construit de la même façon par un transducteur général avec le langage L. Les transducteurs $T^{(1)}$ et $T^{(2)}$ le composant sont directement déduits du transducteur de reconnaissance. Seul le transducteur $T^{(3)}$ est déduit du transducteur de reconnaissance et, en plus de la définition de la fonction τ. Il suffit alors d'associer à chaque point du langage de représentation de A un élément qui sera associé à chaque point du langage de représentation de B. Pour pouvoir définir et composer ces éléments, deux solutions sont possibles. Le premier cas correspond au théorème C3 et admet toutes les transformations. Le deuxième cas utilise un symbole par élément et donc, nécessairement un nombre de points finis.

α) Définition : langage associé (L,V) et (L,V^*) :

Soit V un vocabulaire fini tel que $V \cap \{[,]\} = \emptyset$.
Les langages associés (L,V) et (L,V^*) sont définis de la façon suivante :

- $\forall\, \sigma \in V \cup \{\varepsilon\},\ W \in V^* \quad \sigma.[.] \in (L,V) \wedge W.[.] \in (L,V^*)$

- $W_1,\ldots,W_n \in (L,V),\ W'_1,\ldots,W'_n \in (L,V^*) \Longrightarrow$

$\forall\, \sigma \in V \cup \{\varepsilon\},\ W \in V^*$: $\sigma.[.W_1,\ldots,W_n.] \in (L,V) \wedge W.[.W'_1,\ldots,W'_n.] \in (L,V^*)$.

β) Théorème C3

La famille des transducteurs de transformations est G-constructible, relativement au langage $(L,V^*).(L,V^*)$.

γ) Théorème C4

La famille des transducteurs de transformations bornées est P-constructible, relativement au langage $(L,V).(L,V)$.

REMARQUE

Une transformation (A,B,τ) est dite bornée si les éléments A et B sont bornés.

CHAPITRE VI

LE SYSTEME A.T.E.F.

INTRODUCTION

L'analyse des langues naturelles nécessite en premier lieu une analyse morphologique. Cette analyse est en général suffisamment simple pour nécessiter l'emploi d'un transducteur régulier. Un premier système destiné à l'analyse morphologique a été réalisé en 1969 (Messieurs Courtin, Rieu, Sgall (84)). Ce système a été inspiré par les algorithmes mis au point antérieurement au Centre d'Etudes pour la Traduction Automatique principalement à propos de l'analyse syntaxique. En particulier, l'utilisateur n'était pas contraint de désigner explicitement tous les états nécessaires à une grammaire morphologique. L'analyse morphologique de langues naturelles n'exige pas l'emploi de transducteur plus puissant qu'un transducteur régulier. Cependant, le résultat d'une analyse morphologique peut être amélioré (débarrassé d'une grande partie des ambiguités) par l'utilisation d'un contexte limité. Ceci nous suggère l'idée de réaliser un transducteur composé formé de deux transducteurs réguliers. Enfin, un troisième transducteur régulier peut réaliser certaines liaisons syntaxiques simples. Les trois transducteurs étant composés linéairement, nous pouvons trouver un transducteur équivalent d'ordre un. Le système A.T.E.F. correspond à un tel transducteur.
Nous avons aussi pensé qu'il était préférable d'incorporer dictionnaire et grammaire à l'intérieur du même système (comme dans le cas des systèmes Q).
Ce procédé a d'une part, l'avantage d'incorporer toutes les déclarations relatives aux différentes variables et d'autre part, de pouvoir modifier la chaîne d'entrée en cours de traitement. Cette dernière opération réduit sensiblement la taille du dictionnaire.
En outre, nous avons cherché à offrir à l'utilisateur des moyens efficaces pour accélérer le processus d'analyse. En particulier, la possibilité d'échapper à l'explosion de découpage inutile. Le vocabulaire choisi pour définir les différents éléments du système a été pris en fonction de son application à l'analyse linguistique. La grammaire est le centre du système et définit la fonction de transition du transducteur. Cette définition est largement facilitée par la possibilité de séparer en plusieurs éléments chaque transition. Le transducteur ainsi défini est non déterministe et fournit toutes les solutions possibles lors de l'analyse d'un élément. Enfin, la définition et l'utilisation de ce système sont prévues en mode conversationnel et supportées par le système CP/CMS.

A - PRINCIPE DU SYSTEME

Le système A.T.E.F. est un système simulant un transducteur régulier. La définition des différents éléments de ce transducteur est séparée et effectuée par l'intermédiaire de fichiers portant des noms empruntés à la principale application : l'analyse linguistique automatique.

Un état du transducteur est défini par un ensemble de valeurs de variables définies dans les fichiers correspondants. Les symboles du vocabulaire d'entrée sont définis dans les différents dictionnaires auxquels on associe par l'intermédiaire des formats un état du transducteur.

Une transition du transducteur est définie par la grammaire qui est composée d'un ensemble de règles. Une règle de cette grammaire permet d'effectuer une application de Q x Q dans Q si Q représente l'ensemble des états du transducteur. Afin d'obtenir un transducteur régulier composé, une règle peut elle-même faire appel à d'autres règles et peut conditionner son résultat sur l'analyse éventuelle d'un mot suivant. L'élément de sortie du transducteur est formé par une arborescence. Différentes fonctions de construction de cette arborescence sont également disponibles par l'application d'une règle. Enfin, le système étant non déterministe, il doit fournir un ensemble de solutions pour une analyse. Différentes fonctions d'accélération de l'algorithme permettent d'atteindre l'ensemble des résultats avec un minimum d'opérations.

1 - ENSEMBLE DES ETATS DU TRANSDUCTEUR

1° Ensemble des états du transducteur

Cet ensemble d'état est défini dans les fichiers "déclaration des variables". Les opérations effectuées sur ces variables permettent de faire évoluer l'état courant du transducteur. A l'intérieur de cet ensemble de variables deux types d'éléments sont définis. A chacun de ces types correspond des contraintes différentes et des fonctions d'évolution différentes.

Le premier type de variables est le type exclusif. Une variable exclusive peut prendre une seule valeur à la fois parmi l'ensemble de ces valeurs définies. L'ensemble des valeurs d'une variable exclusive est définie soit par un ensemble d'éléments, soit par un ensemble de variables exclusives.

Exemple

VAREM := (SEG(D,S,B,P))

La variable exclusive *VAREM* a pour valeur une variable exclusive *SEG* qui prend les valeurs *D* (désinence), *S*(suffixe), *B*(base), *P* (préfixe).

Une variable exclusive est dite élémentaire si elle a pour valeur un ensemble d'éléments. Une variable exclusive élémentaire peut prendre pour valeur un seul de ses éléments. Les conditions portant sur cette variable ne peuvent être que des comparaisons élémentaires (égalité ou différence). Les variables non-exclusives sont définies de la même façon que les variables exclusives. Une variable non-exclusive élémentaire peut prendre pour valeur un ensemble de ses éléments. Les conditions portant sur cette variable peuvent être des comparaisons ensemblistes (égal, contenu, inclu, différence).

Exemple

VARNM := (PREF(IN,IM,IR,IL), SDN(MENT,EUR,ESSE,ABLE,ATION,ITION)).

La variable non-exclusive *VARNM* a pour valeur deux variables non-exclusives *PREF* et *SDN* . La variable non-exclusive *PREF* (préfixe) a pour valeur les éléments *IN, IM, IR, IL*.Elle peut par exemple prendre comme valeur les ensembles suivants : {*IN* }, {*IN,IM* } .

Un cas particulier d'état des différentes variables est celui des variables non affactées. Ces variables ont alors aucune valeur. On à ce cas en précisant le nom de la variable suivie du caractère "*0*".

2° Ensemble des éléments du vocabulaire

Les éléments du vocabulaire sont définis par l'ensemble du dictionnaire. Une variable spécialisée d'unité lexicale est prédéfinie dans le système. Deux types de dictionnaire peuvent être définis suivant qu'ils font référence ou non à l'ensemble des valeurs d'unités lexicales. Un dictionnaire est composé d'une suite d'éléments. Chaque élément comprend un élément du vocabulaire d'entrée (entrée du dictionnaire), deux références de formats et éventuellement une référence d'unité lexicale.

3° Fonction de transition

La fonction de transition du transducteur est définie par la grammaire. La grammaire est une suite de règles. Chaque règle permet de faire évoluer l'état du transducteur à partir d'un élément d'entrée. A chaque élément de dictionnaire un format morphologique est associé. Ce format permet de définir l'ensemble des règles de la grammaire qui pourront être appliquées. Une règle est applicable à partir d'une entrée de dictionnaire si le nom de format morphologique associé à cette entrée figure en partie gauche de la règle. Une règle applicable est conditionnelle et est appliquée que si sa condition est vérifiée. Lorsqu'une règle est appliquée, les différentes opérations suivantes sont effectuées :

- évolution de l'état courant par l'affectation des variables
- réalisation des différentes fonctions immédiatement applicables comme les fonctions d'accélération de l'algorithme et mise en réserve des fonctions applicables à la fin du traitement comme la construction arborescente.
- transposition éventuelle de la chaîne d'entrée.

L'évaluation des conditions et l'évolution de l'état courant s'effectuent à l'aide de plusieurs ensembles de variables. On se réfère à un ensemble de variables par un code origine. Les variables portant le code origine A proviennent de l'élément de dictionnaire concerné et correspond à l'ensemble des variables formant les formats morphologique et syntaxique. Les variables portant le code origine C correspondent aux variable de l'état courant du transducteur. D'autres variables provenant d'ensembles précédemment analysés peuvent être concernées par les conditons.

4° Fonctionnement de l'analyseur

a) Analyse d'un élément invariant par exemple un article

- Variables :
  - * *morphologiques*
    - -EXC-
    - SEG := (D,S,B,P). (variable exclusive)
    - -NEX-
    - Ø
  - * *syntaxiques*
    - -EXC-
    - CAT := (VRB,NMC,NMP,PRP,AJQ,ART,ADV,PRE,CONJ).
    - DEF := (DEF,INDEF).

-NEX-
GNR := (MAS,FEM)
NBR := (SIN,PLU).

-Formats

* *morphologiques :*

*INVAR 01 ==. ** SEG : e - B .*

* *syntaxiques :*

*UN 01 ==. ** CAT -e-ART, NBR-E-MAS,*
DEF -E- INDEF.

* *dictionnaire :*

UN == INVAR (UN,ULUN).

* *grammaire :*

R001 : INVAR == VAR(c) := VAR(A)/SEG(C)-e-SEGD.

Lorsque la forme "*UN*" est lue par le transducteur, après une recherche dans le dictionnaire et la découverte de l'entrée, la règle *R001* est appliquée. La condition de cette règle demande que la variable SEG soit non affectée. Cette condition est réalisée car dans l'état initial du transducteur, toutes les variables sont non affectées. Après l'application de la règle, la forme d'entrée a été complètement segmentée (il n'y a ici qu'un seul segment) et le transducteur fournit alors comme résultat l'état courant. Ici le résultat est donné suivant le schéma.

UN → { SEG = B, CAT = ART, NBR = SING, GNR = MAS,
DEF = INDEF, UL = ULUN }

Pour ajouter l'article "*UNE*" dans le système défini ci-dessus, deux façons différentes sont possibles :

la première est celle où le dictionnaire est augmenté d'un élément :

UNE == INVAR (UNE,ULUN).

le format syntaxique "*UNE*" est défini par

*UNE 01 ==. ** CAT -E- ART, NBR -E-SING, GNR -E- FEM,*
DEF -E- INDEF.

La deuxième façon est celle où un nouveau type de dictionnaire est créé et une désinence introduite :

E == DESI (FEM).

les formats morphologique et syntaxique nouveaux sont définis par :

DESI 01 == . ** SEG -E- D .

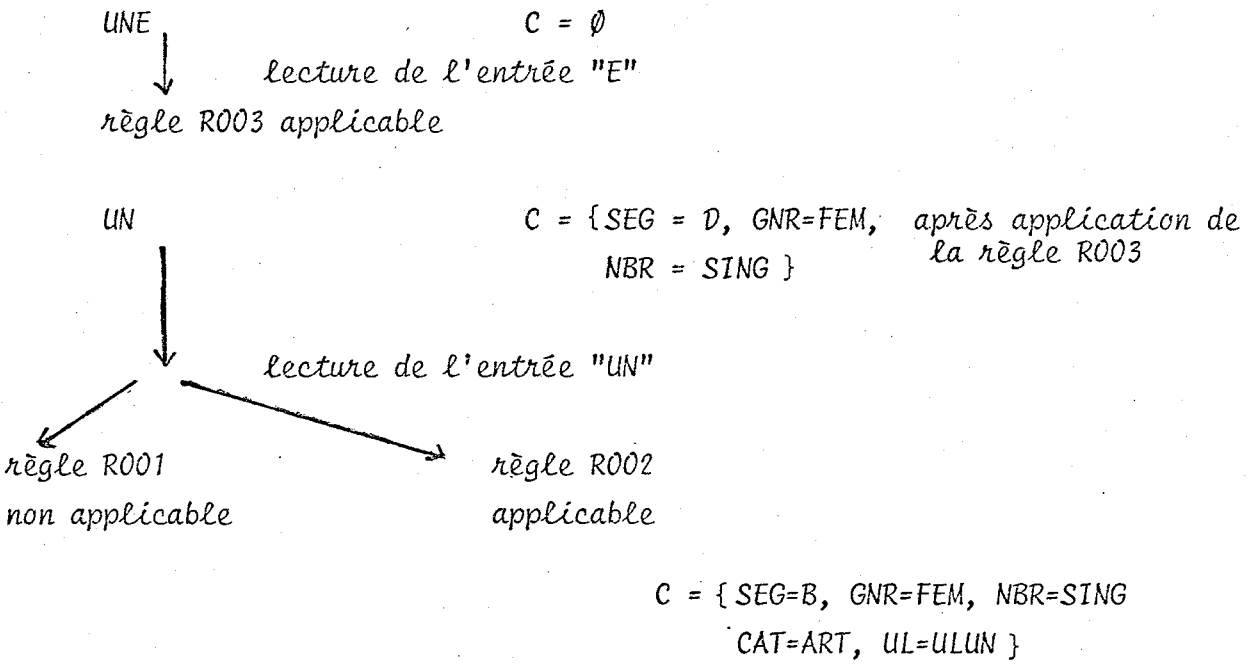
FEM 01 == . ** GNR -E- FEM , NBR -E- SING .

la grammaire contient en plus les règles suivantes :

R002 : INVAR == CAT(C) := CAT(A), DEF(C):=DEF(A), SEG(C):=SEG(A)/ SEG(C)-E-D .

R003 : DESI == VAR(C):=VAR(A) / SEG(C)-E-SEGO .

Dans ce cas l'analyse de la forme "UNE" se décompose ainsi (on suppose que l'analyse de la forme commence par la droite de la forme.(Ce sens d'analyse est un des paramètres du système).



b) Analyse d'un élément avec altération de la forme en cours d'analyse

Soit l'analyse du mot réel, ce mot conduit à l'unité lexicale réelle et peut conduire à différents mots comme : réelle, réellement, irréel, irréellement, réalité, réalisme, irréalité, irréalisme, réaliseur, réaliser, réalisation, réalisable, irréalisable, réalisateur. Afin de ne pas obtenir un dictionnaire trop important pour l'addition d'éléments nouveaux, une seule entrée sera placée dans le dictionnaire de type UL. Les autres éléments nécessaires à l'analyse de ces différentes formes seront dans les dictionnaires de type sans unité lexicale, ils serviront également à l'analyse d'autres formes.

Nous avons REEL == BAS1(SYNR,REEL).

L'analyse de "REELLE" n'est alors pas possible avec les éléments présents dans le dictionnaire. Car après la lecture de l'élément "E" le reste de la forme est "REELL" et n'est pas dans les dictionnaires. La règle R003 doit comprendre une transformation de chaîne de façon à éliminer un "L". Cette règle devient :

R003 : DESI == VAR(C):=VAR(A)/SEG(C)-E-SEGO/TCHAINE(0,'LL','L').

L'analyse de REELLE s'effectue alors de la façon suivante :

REELLE C = ∅
↓ lecture de l'entrée "E"
↓ règle R003 applicable
REELL → REEL C = SEG=D, GNR=FEM, NBR=SING
↓ lecture de l'entrée "REEL"
↓ règle R004 applicable
C = SEG=B, GNR=FEM, NBR=SING, CAT=ADJ, UL=REEL

avec R004 : BAS1 == CAT(C):=CAT(A), DEF(C):=DEF(A), SEG(C):=SEG(A) / SEG(C) -E- D .

Dans le cas où on ne veut pas mettre les préfixes "IR" dans un des dictionnaires, il est possible de considérer ce cas par l'appel de sous règle : le fichier de variables syntaxiques contient en plus la variable NEG :
NEG := (NEG).
Alors la règle R004 devient :

R004 : BAS1 == CAT(C) := CAT(A), DEF(C) := DEF(A), SEG(C):=SEG(A) / SEG(C) -E- D /// R005, R006 .

R005 : - NUL - == NEG(C):=NEG / SCHAINE(A,0,1) -E- 'IR' / TCHAINE(0,'IR''').

R006 : -NUL - == .

c) Fonction d'accélération de l'algorithme

Pour une même entrée de dictionnairelorsque plusieurs règles sont

possibles, elles sont appliquées dans l'ordre dans lequel elles se trouvent dans la grammaire. Pour arrêter l'analyse des autres règles susceptibles d'être appliquées pour ce segment, des fonctions d'accélération de l'algorithme sont disponibles. Dans l'analyse du mot "réalisme" avec les éléments déjà définis, il faut ajouter l'élément de dictionnaire "*ISME*".

ISME == FSME (SISME).

alors avec la règle

R007 : FSME == VAR(C):=VAR(A)/SEG(C)-E-SEGO/TCHAINE(1,'A','E').

l'analyse de ce mot "*REALISME*"s'effectue de la façon suivante :

REALISME
↓ *lecture de ISME*
règle R007
REEL
↓ *lecture de REEL*
1 solution
↓ *lecture de E*
règle R003

REALISM

Pour éviter la tentative d'analyse à partir de l'élément d'entrée "E", la règle *R007* peut comporter la fonction "*ARRET*".
Dans ce cas la suite de caractères *ISME* ne peut conduire à d'autres analyses éventuelles, ceci donne les éléments suivants :

R007 := FSME == VAR(C):=VAR(A), -ARRET- / SEG(C) -E- SEGO /
TCHAINE(1,'A','E').

L'analyse décrit dans la suite des règles suivantes :

REALISME
↓ *lecture de ISME*
règle R007
REEL
↓ *lecture REEL*
1 solution

d) Condition inter-forme et construction arborescente

L'analyse d'un texte s'effectue de gauche à droite. Chaque forme est analysée en tenant compte de certains résultats précédemment analysés. L'environnement de l'analyse d'une forme est donné par les éléments suivants :

- quatre formes précédentes
- forme suivante
- forme "racine" de la construction

Les actions possibles sur ces solutions sont soit des conditions portant sur les variables, soit des éliminations d'occurrence, soit la recherche de la présence de tournure, soit enfin la construction d'une arborescence. La présence de tournure est testée que dans le cas où un dictionnaire de type tournure a été déclaré. Une tournure est définie par une suite de formes. Dans le cas où sa présence a été détectée, elle peut être remplacée par une seule forme. Un exemple de tournures est donné par les suites : *"DE PLUS EN PLUS"* , *"PAR ICI"*, etc...

Le test sur le résultat des quatre formes précédentes peut permettre soit l'élimination d'une ambiguité, soit le contrôle de la construction de l'arborescence de sortie. Soit par exemple la suite suivante : *"UNE REELLE TENTATIVE ANALYTIQUE"*. Au cours de l'analyse de *"REELLE"* les éléments suivants sont définis :

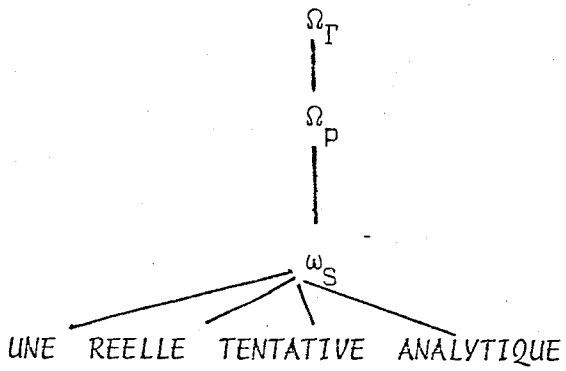
- seules les variables de la forme immédiatement précédente sont non vides et ont pour valeur le résultat de l'analyse de *"UNE"*. On réfère à ces variables par le code origine *"P1"*.

Au cours de l'analyse *"TENTATIVE"* les éléments suivants sont définis :

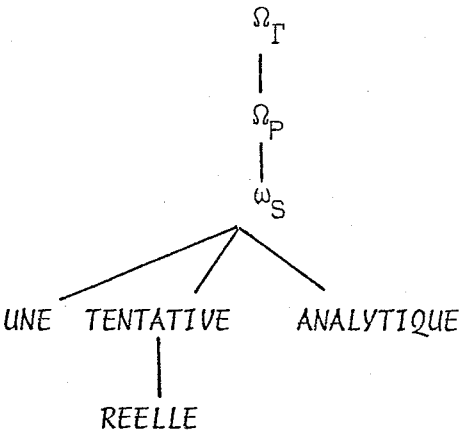
- les variables associées à l'analyse de *"UNE"* peuvent être testées. Elles ont alors comme code origine *"P2"*.

Les variables résultantes de l'analyse de *"REELLE"* peuvent être repérées par le code origine *"P1"*.

Sans fonction de construction arborescente, le résultat de l'analyse de cette suite de formes serait défini par l'arborescence suivante :



Au cours de l'analyse de "*tentative*" un début de construction peut être entrepris. L'élément *REELLE* est ici la racine de la construction précédente car il n'y a pas eu de construction lors de l'analyse de *REELLE* . Si l'analyse de *TENTATIVE* nécessite l'application d'une règle comportant la fonction "*IDN*", le résultat sera alors



Au cours de l'analyse de la forme "*ANALYTIQUE*" les variables associées au code origine "*R*" sont celles obtenues au cours de l'analyse de la forme "*TENTATIVE*".

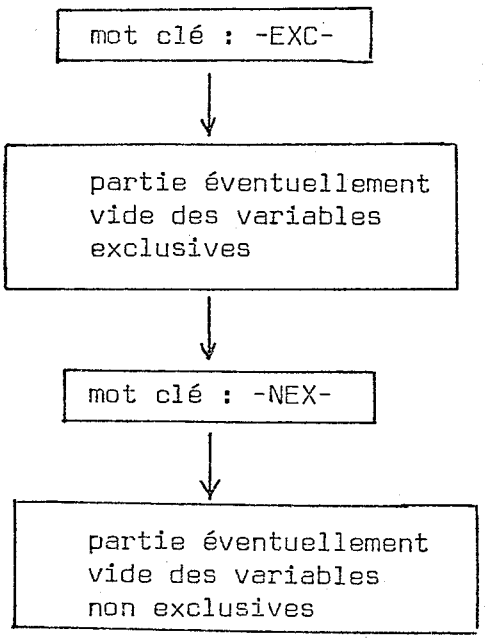
e) Autres fonctions du système

Deux autres fonctions sont disponibles dans le système. Elles ont pour but, soit de créer une nouvelle entrée de dictionnaire, soit de créer de nouvelles formes dans le texte d'entrée à partir de la forme analysée. Ces deux fonctions s'appliquent en même temps qu'une règle. La première "*-TRANS-*" complète un dictionnaire spécial dit de mots inconnus. La seconde "*SOL*" permet de traiter les mots composés en créant plusieurs ensembles de variables résultant.

B - PRESENTATION DES DIFFERENTS COMPOSANTS

1 - DECLARATION DES VARIABLES

Les déclarations de variables ont pour but la définition des différentes valeurs nécessaires au traitement. Il existe deux fichiers de déclaration de variables. Les variables morphologiques et les variables syntaxiques. Le nom de ces deux fichiers n'a aucun rapport avec leur contenu et l'utilisateur peut répartir ces variables de façon quelconque. Seul le traitement des formats diffère. Un fichier de déclaration de variables peut être résumé par le schéma suivant :



Une variable élémentaire est exclusive si on peut lui affecter qu'une seule valeur déclarée à la fois.

Le nombre total de valeurs de variables élémentaires possibles du système est : 240 valeurs non exclusives ou 2^{240} valeurs exclusives.

Une variable exclusive est soit une variable exclusive élémentaire, soit un ensemble de variable exclusives.

Une variable non exclusive est également soit une variable non exclusive élémentaire, soit un ensemble de variables non exclusives.

Un élément de déclaration de variables est composé d'un nom de variable et de la liste des valeurs associées :

NOMVAR := (VAR1(VAL11,...,VALN1),...,VARP(VAL1P,...,VALP1P)).

Ici la variable "*NOMVAR*" désigne l'ensemble des variables élémentaires "*VAR1*",...,"*VAR*". La variable élémentaire "*VAR1*" a pour valeur "*VAL11*",...,"*VALN1*".

On réfère à la valeur non affectée de la variable en concaténant à droite *"0"* au nom de cette variable.

Les noms de variables prédéfinis sont :

VAR : ensemble des variables différentes de la variable *UL*.
VARM : ensemble des variables morphologiques
VARS : ensemble des variables syntaxiques
VARNM : ensemble des variables non exclusives morphologiques
VAREM : ensemble des variables exclusives morphologiques
VARES : ensemble des variables exclusives syntaxiques
VARNS : ensemble des variables non exclusives syntaxiques
UL : variable "unité lexicale"
DICT : variable "dictionnaire"

Exemple :

Fichier *FILEMEX VARBM* Ex : Code langage *VARBM* : variable morphologique)

-EXC-

SEG := (D,S,B,P).

-NEX-

PREF := (IN, IM, IL, IR).

SYS := (SYSD(SYSD1,SYSD2), SYSV(SYSV1,SYSV2,SYSV3))

Fichier *FILEMEX VARBS* Ex : Code langage *VARBS* : variable syntaxique)

-EXC-

NEG := (NG).

-NEX-

GNR := (MAS, FEM, NEU).

2 - FORMATS

Il existe trois fichiers de formats : deux fichiers primaires et un fichier de formats composés. Seuls les fichiers primaires sont obligatoirement non vides. Les deux fichiers primaires : fichier de formats morphologiques et fichier de formats syntaxiques sont constitués de la même façon. Les formats

morphologiques ne font référence qu'aux variables morphologiques. Le fichier de formats morphologiques doit contenir le format du mot inconnu : *MØDINC*. Les formats syntaxiques font référence aux variables syntaxiques et aux variables morphologiques non exclusives.

Un format est la description d'un ensemble de valeurs de variables : il est constitué d'un nom de référence suivi d'un ensemble d'affectation de valeurs de variables séparées par des symboles ",'.

Suivant le type de variable auquel elle réfère, cette affectation peut prendre deux formes :

- si la variable est une variable exclusive alors l'affectation est une égalité à une valeur élémentaire :

VAREX -E- VAL1.

- si la variable est une variable non exclusive alors l'affectation est une égalité à une union de valeurs élémentaires :

VARNEX -E- VAL1 -U- VAL2.

L'opérateur ensembliste utilisable est seulement l'opération "*-U-*" (Union).

Les opérateurs pouvant être présents dans un format sont donc les suivants :

-E- : égalité
-U- : union

Le nom de référence est un nom de format suivi d'un numéro de carte. Lorsque la description d'un format nécessite plus d'une carte, les éléments suivants sont donnés sur une autre carte portant le même nom de format et un nouveau numéro de carte.

Exemple :

```
FORMAT1 :  01 == . ** GNR -E- MAS, CAT -E- NMC,
FORMAT1 :  02         NEG -E- NG.
FORMAT2 :  01 == . **  .
```

Le nom de format doit être inférieur à 8 caractères et le numéro de carte doit être placé en colonne 9.

Le fichier de formats composés ou formats généraux est nécessaire pour construire des formats à l'aide de formats existants.
Un format général est une suite de formats élémentaires dont la réunion donne le format désiré.
Un format général est constitué d'un nom de format, suivi de la liste des formats élémentaires qui doivent tous être des formats syntaxiques.

Exemple :

FØRMAG 01 == FØRS1, FØRS2, FØRS3.

3 - DICTIONNAIRES

Le système peut comporter au plus six dictionnaires et un dictionnaire de tournures.
Il existe deux types de dictionnaires :

- Dictionnaire dit "d'affixe" où aucune référence lexicale n'est introduite.
- Dictionnaire dit "de base" où une référence lexicale est introduite.

Un dictionnaire est une suite d'entrées. Chaque entrée étant une suite de caractères ne comportant pas de blanc et de longueur inférieure à 24 caractères. Elle est suivie de deux noms de format et éventuellement d'un nom de référence lexicale.

Exemple :

Désinence == FØRMØPH (FØRMSYN)
BASE == FØRMØPH (FØRMSYN, REFBAS)

Le dictionnaire de tournures est un dictionnaire de type base où l'entrée peut contenir des blancs à condition qu'il n'y en ait pas deux concaténés.

Exemple :

DE PLUS EN PLUS == FØRMOPH (FØRMSYN, REFPLUS)

4 - GRAMMAIRE

La grammaire consiste en une suite de règles déterminant le fonctionnement de l'automate.

Le traitement d'une règle implique :

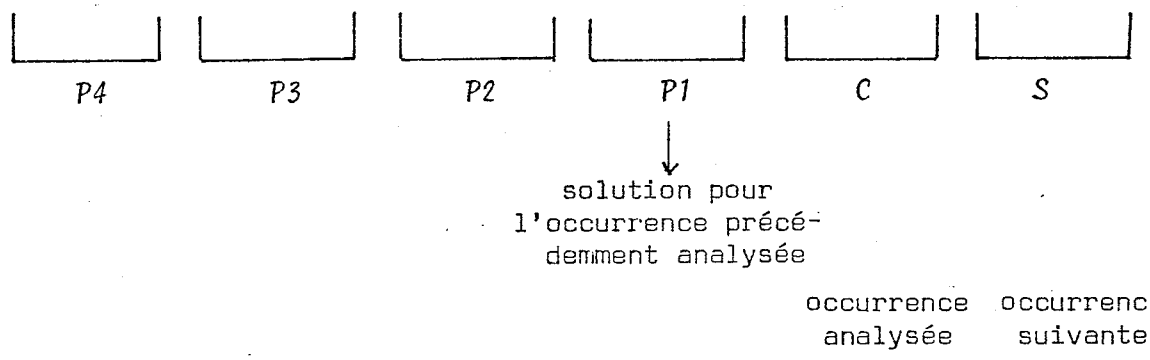
- la modification de l'état courant (C) de l'automate
- la présence d'un état de variable (A) provenant d'un dictionnaire par l'intermédiaire des formats morphologiques et syntaxiques
- l'analyse d'un découpage éventuel de la forme d'entrée

Une règle comporte deux parties :

a) la partie gauche, ensemble de formats morphologiques séparés par '-', précisant l'ensemble des éléments des différents dictionnaires qui lui feront appel.

b) La partie droite qui est divisée elle-même en cinq parties. Chaque partie est séparée par "/". Pour passer d'une partie à une autre, le nombre de caractères "/" est constant.

L'analyse d'une occurrence est placée dans l'environnement des solutions de l'analyse des formes précédentes. Le schéma suivant indique les codes origines employés.



a - La partie affectation ou modification du registre courant (C)

Chaque affectation est séparée par une ',' et permet la modification d'une variable ou d'un ensemble de variables. L'affectation d'une variable exclusive est de type direct :

Exemple :

EXCLU (C) := EXCLU1
EXCLU(C) := EXCLU(A)

L'affectation des variables non exclusives est de type ensembliste

Exemple :

NEXCLU (C) := NEXCLU (A) -U- (NEXCLU (C) -I- (NEXCLU (A) -I- -N- (NEXCLU1 -U- NEXCLU2))).

Dans la partie affectation, les opérateurs possibles sont :

:= affectation
-I- intersection
-U- union
-N- complément

Les différents codes origines des variables sont :

(C) état courant
(A) variable d'entrée
(S) variable affectée à la forme suivante
(T) variable affectée à une tournure
(R) variable affectée à la racine de l'arborescence de sortie

La partie affectation comprend en outre les différentes fonctions d'accélération de l'algorithme et de construction de l'arborescence de sortie.

Action sur la segmentation de la forme en cours

-ARRET- Annule l'analyse de tous les sous-découpages possibles de l'élément de dictionnaire activant la règle.

-*FINAL*- : Lorsqu'une solution a été trouvée à partir d'une règle contenant "-*FINAL*-, les solutions précédemment trouvées sont éliminées et l'analyse de la forme en cours est arrêtée. Seule cette solution est conservée.

-*STØP*- : Lorsqu'une solution a été trouvée à partir d'une règle contenant "-*STØP*-" l'analyse de la forme en cours s'arrête.

- *ARD*- : Annule l'analyse de tous les sous-découpages possibles de l'ensemble des éléments provenant de la même lecture des dictionnaires que le segment qui active la règle.

-*ARF*- : Annule l'analyse de tous les sous-découpage possibles de l'ensemble des éléments provenant de la même lecture du même dictionnaire que le segment qui active la règle

Action sur la suite des résultats

-*SØL*- : Crée une solution avec le masque courant (C)

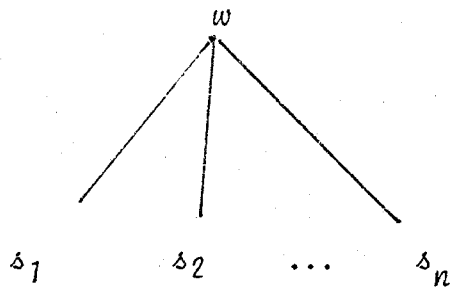
-*ELIM*(Σ) : Elimine dans la suite du résultat le masque correspond à Σ . (Σ est le code origine, les valeurs possibles sont*P4*, *P3*, *P2*, *P1*, *C* et *S*).

-*ELIT*(Σ)- : Remplace la suite des différents masques de solutions correspondants aux occurrences composant une tournure par le masque courant (C). (Σ code origine de la tournure différentes valeurs possibles sont : *P4*, *P3*, *P2*, *P1*, *C*, *S*).

-*TRANS*- : Crée une unité lexicale avec l'occurrence d'entrée (C).

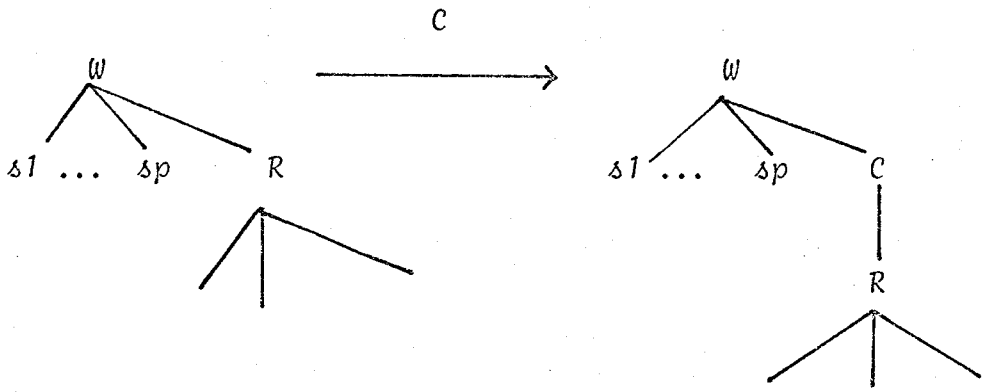
Action sur l'arborescence de sortie

Sans présence de l'une de ses fonctions, l'arborescence de sortie pour une phrase est une arborescence à un seul niveau :

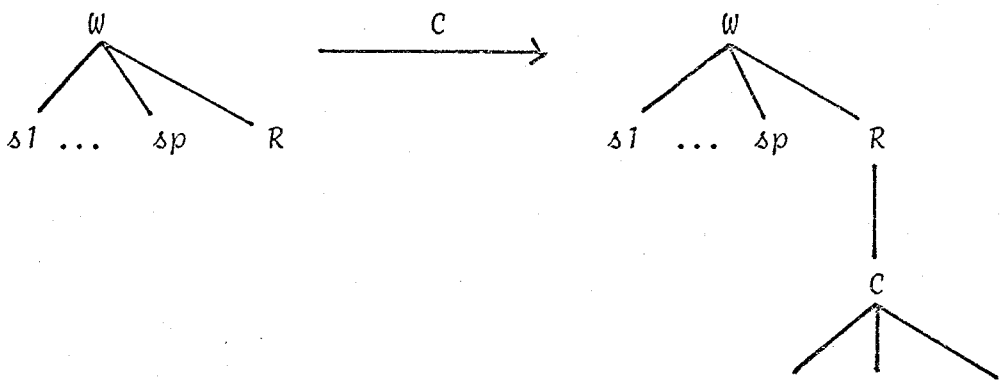


Il existe onze fonctions permettant une construction plus complexe de l'arborescence de sortie. Ces fonctions sont :

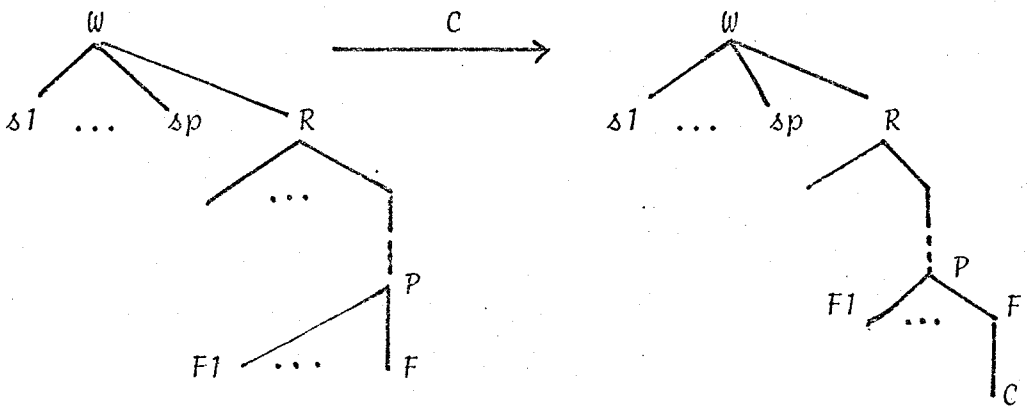
IDN :



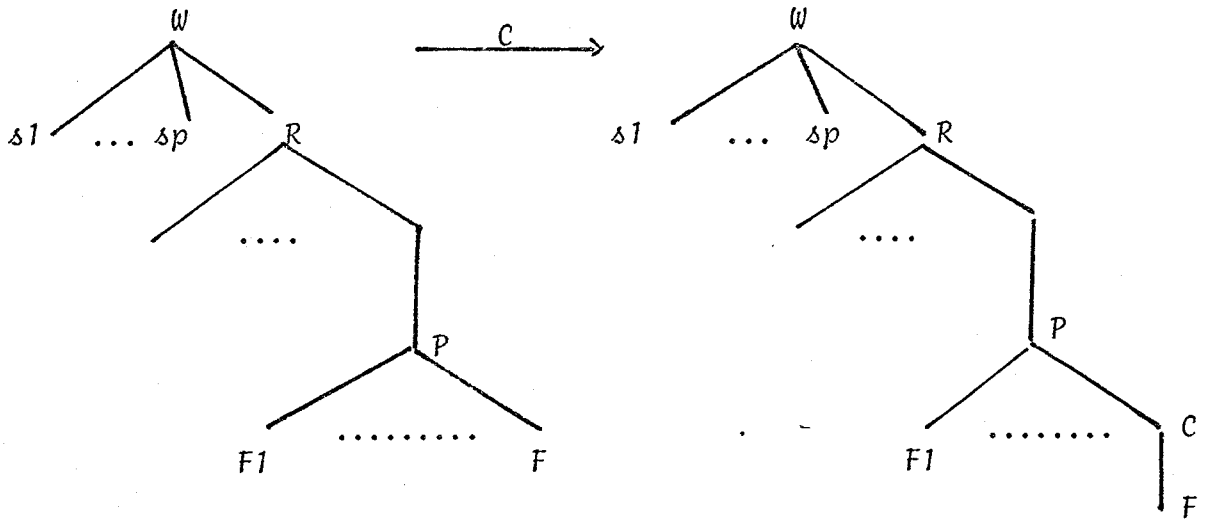
IDX :



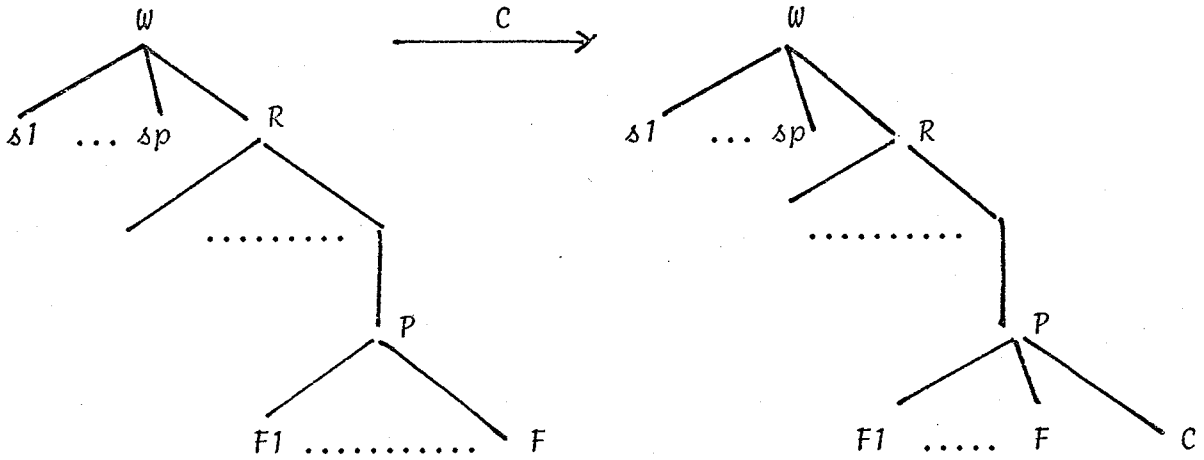
ISN :



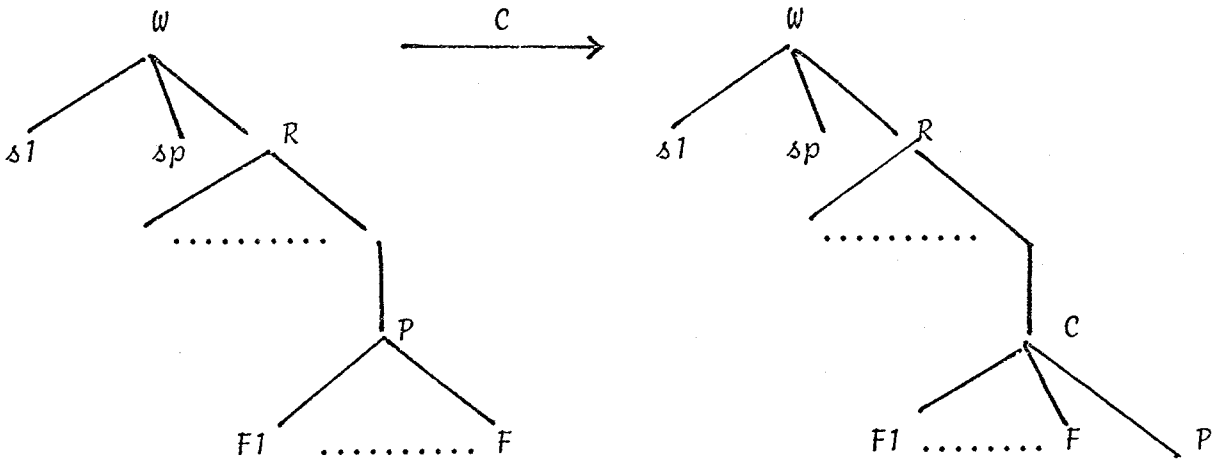
ISX :


W
C
W
s1 ... sp
R
s1 ... sp
R
....
....
P
P
F1 ..... F
F1 C
F

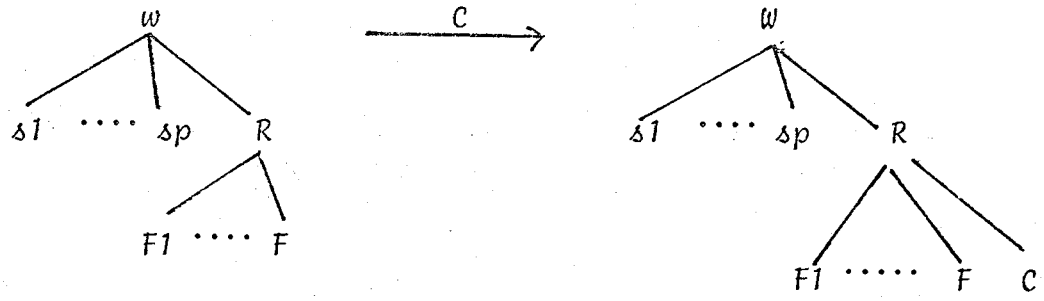
IFN :


W
C
W
s1 ... sp
R
s1 ... sp
R
.....
.....
P
P
F1 ..... F
F1 ..... F
C

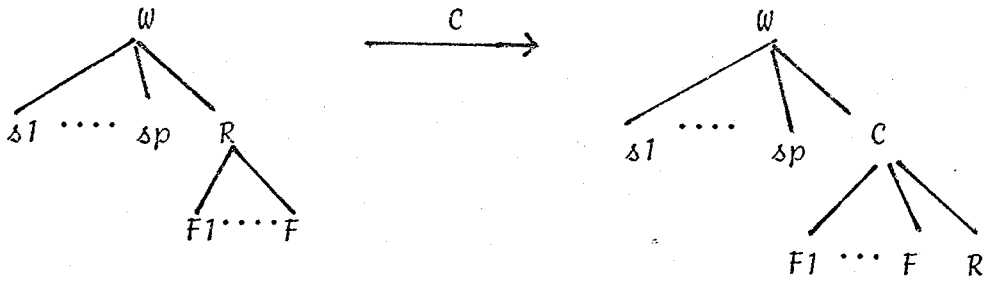
IFX :


W
C
W
s1 sp
R
s1 sp
R
.....
.....
P
C
F1 ..... F
F1 ..... F
P

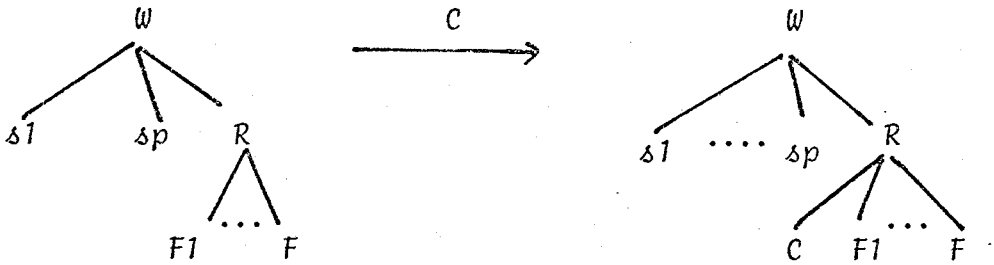
IHN :



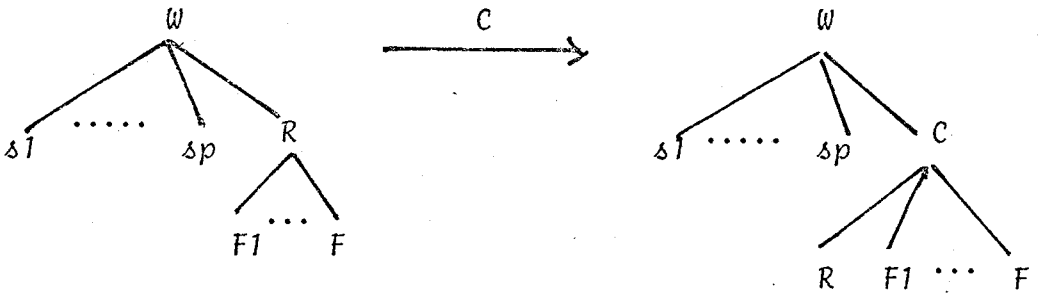
IHX :



IAN :



IAX :



DST : destruction de l'arborescence construite pour cette phrase.

b - La partie condition

La partie condition est une expression booléenne permettant de réduire les cas d'applications de la règle. Elle peut porter soit sur des variables, soit sur un des caractères de la chaîne d'entrée, soit sur la présence d'une tournure.

Les opérateurs permettant de former l'expression booléenne sont :

-*ET*- "et"
-*ØU*- "ou"
-*N*- "non"

Les éléments booléens proviennent soit de conditions sur les variables, soit de conditions sur la chaîne de caractères, soit sur la présence de tournure.

α) Présence de tournure

Le test de présence de tournure s'écrit *TØURN(ORIG)*. *"ØRIG"* étant le code origine à partir duquel on demande le test. Codes possibles : *P4, P3, P2, P1, C, S.*

β) Test sur les chaînes

Les tests permis sur les chaînes de caractères sont des tests d'égalité ou de différence. Codes opérations :

-*e*- égalité
-*NE*- différence

On réfère à une constante littérale en écrivant cette constante entre deux "'". Lorsque la constante doit contenir un caractère "'", ce caractère doit être doublé.

On réfère à une suite de caractères dans le texte d'entrée par la fonction *SCHAINE*.

Cette fonction a trois paramètres :

SCHAINE (A,B,C).

A : est le code origine, occurrence à partir de laquelle les caractères vont être issus. Codes possibles :*P4*, *P3*, *P2*, *P1*, *C*, *A*, *S*.

P4, ..., *P1*, *C*, *S* font référence à l'occurrence entière.
A fait référence au reste de l'occurrence analysée (sous-chaine de *C*).

B : est l'origine à partir de laquelle va être extraite la suite de caractères. L'origine est un nombre.
Si Σ est le sens d'analyse, le départ de l'extraction des caractères est à "Σ" du $B^{ième}$ caractère. Si B = 0 alors le départ est à "$\bar{\Sigma}$" du premier caractère (en notant par $\bar{\Sigma}$ le sens contraire).

Dans le cas où le chiffre est précédé de Ø , l'origine est prise par rapport au sens inverse de l'analyse.

Exemple :

Soit : a b c a b a
Sens d'analyse $\mathcal{D}$:

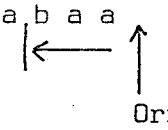
Ø0 a b c a b a
0 a b c a b a
3 a b c a b a
Ø2 a b c a b a

C : longueur ou nombre de caractères extraits à partir de l'origine. Les caractères extraits sont pris dans le sens de l'analyse ci C se compose d'un chiffre ; dans le sens contraire, si le chiffre est précédé d'un Ø.
Si la longueur est égale à '*' alors le reste de la forme est prise.

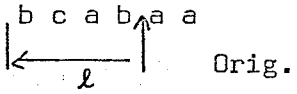
Exemple :

Soit a b c a b a a
Sens d'analyse $\mathcal{D}$:

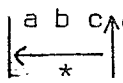
0, 3 a b c a b a a : b a a



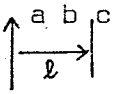
2, 4 a b c a b a a : b c a b



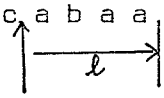
4, * a b c a b a a : a b c



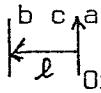
Ø0, Ø2 a b c a b a a : a b



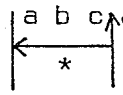
Ø3, Ø4 a b c a b a a : a b a a



Ø3, 2 a b c a b a a : b c

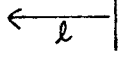


Ø3, * a b c a b a a : a b c

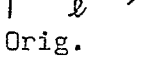


Lorsque la longueur dépasse de 1 la longueur de la chaîne d'entrée le caractère blanc est placé dans le sens défini.

Ø3, 4 a b c a b a a : ␣ a b c



Ø3, Ø5 a b c a b a a : a b a a ␣



Lorsque la longueur est trop importante, la valeur de *SCHAINE* est une suite de caractères non représentable.

γ) Test sur les variables

Les tests sur les variables se divisent en deux groupes : les tests sur les variables exclusives et les tests sur les variables non exclusives.

Les tests sur les variables exclusives ont pour opérateurs booléens l'égalité et la différence :

-E- égalité
-NE- différence

Les tests sur les variables non exclusives sont du type ensembliste.
Opérateurs :

-E-	égalité	A -E- B :	A = B
-NE-	différence	A -NE- B :	A ≠ B
-INC-	inclusion	A -INC- B :	A ⊃ B
-DANS-	inclusion	A -DANS- B :	A ⊂ B

Les opérandes des opérateurs de relation sont soit des valeurs simples (cas des variables exclusives), soit des valeurs composées (cas des variables non exclusives).

On réfère à la valeur d'une variable par son nom suivi du code origine précisant la solution où l'on doit extraire cette valeur.
Codes possibles :

Pi $1 \leqslant i \leqslant 4$	: extraction sur la solution de l'occurrence précédente en ième position
C	: extraction sur l'état courant de l'automate
A	: extraction sur le masque provenant de la référence du dictionnaire
PSi $1 \leqslant i \leqslant 9$	: extraction sur la solution créée au cours de l'analyse de cette occurrence et placée en ième position

Opérateur ensembliste pour composer des variables non exclusives

-U- union
-I- intersection
-N- complément

c) La partie transformation de la chaîne d'entrée

Cette transformation permet de modifier le reste de l'occurrence à analyser. Elle s'exprime par : *TCHAINE (Ø, C1, C2)*.

Où :

Ø : origine à partir de laquelle on teste la transformation. L'origine suit la même règle que pour *SCHAINE*.

C1 : chaîne entre deux caractères "'" qui doit être modifiée (sous réserve que cette chaîne coïncide avec celle de l'entrée). Si la chaîne n'est pas notée entre deux "'", seulement la longueur est prise en compte. La recherche de la chaîne à modifier s'effectue dans le sens de l'analyse. Dans le cas où l'origine est le sens contraire de l'analyse, la recherche de la chaîne à modifier se fera également dans le sens contraire.

C2 : chaîne entre deux "'" qui doit remplacer la chaîne correspondant à *C1*.

Lorsqu'il y a plusieurs transformations possibles, les fonctions *TCHAINE* sont séparées par des virgules et leurs effets sont cumulatifs.

d) La partie condition sur le suivant

La partie condition sur le suivant est une partie condition classique où peut figurer le code origine *(S)* et ne doit pas figurer les codes origines : *PSi* $(1 \leqslant i \leqslant 9)$ et *P4* . Elle permet de donner une solution sous réserve du résultat de l'analyse de l'occurrence suivante.

e) La partie appel de règles

Lorsque cette partie est non vide, la règle ne sera considérée comme valable que si il existe une règle appelée qui est satisfaite. Dans le cas où une règle appelée est satisfaite, les suivantes sont ignorées. Les tests de satisfaction des règles appelées se font dans l'ordre où elles sont mentionnées.

La grammaire comprend en outre obligatoirement deux règles particulières :

α) La règle du mot inconnu

Elle a pour étiquette de règle *MØTINC* : sa partie gauche comprend au moins le format *MØDINC* et sa partie droite est inconditionnelle.

β) La règle des dictionnaires initiaux

Elle est de la forme : *RDICT : (L/C)*

Où *L* est la liste des dictionnaires valable au départ de l'analyse.

N est le code des différents contrôles sur l'unité lexicale et les dictionnaires possibles.

Les différents codes pour *C* sont :

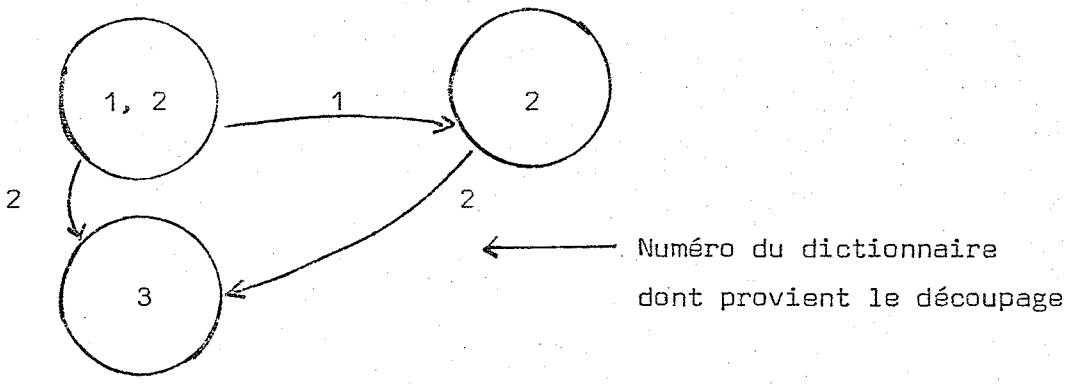
NN : enchaînement du dictionnaire et affectation de l'unité lexicale contrôlée par la grammaire

NU : enchaînement du dictionnaire contrôlé par la grammaire et affectation de l'unité lexicale contrôlée par l'algorithme

DN : enchaînement du dictionnaire contrôlé par l'algorithme et affectation de l'unité lexicale contrôlée par la grammaire

DU : enchaînement du dictionnaire et affectation de l'unité lexicale contrôlée par l'algorithme

Enchaînement du dictionnaire



Dans le cas de contrôle par l'algorithme, la grammaire peut éventuellement corriger ce contrôle en cours d'exécution.

C - TRAITEMENT INFORMATIQUE

Le système A.T.E.F. simule un transducteur non déterministe. Il est donc avant tout constitué par un automate à pile afin d'obtenir toutes les solutions possibles. A chaque choix de la fonction de transition, les différentes possibilités sont empilées et seront analysées lors d'un "back track" ultérieur. Les fonctions d'accélérations telles que -ARRET-, -FINAL-, etc... sont des fonctions agissant sur ce procédé de back-tracking. Chaque pas du transducteur nécessite une lecture de dictionnaire et le traitement d'un découpage. La lecture du dictionnaire s'effectue en deux temps. La première recherche réalise un adressage direct en fonction d'une clé et la deuxième s'effectue par dichotomie. La lecture du dictionnaire permet d'empiler tous les sous-découpages possibles qui seront analysés ultérieurement par le procédé de back-tracking. Le traitement d'un découpage comporte d'abord l'évaluation d'une condition puis, dans le cas où cette condition est satisfaisante, les opérations suivantes sont effectuées :

- mise à jour de l'état courant
- application des fonctions spéciales
- altération du reste de la forme à analyser
- recherche des règles complémentaires (sous-règles)
- indexer éventuellement le résultat sur le résultat de l'analyse de la forme suivante.

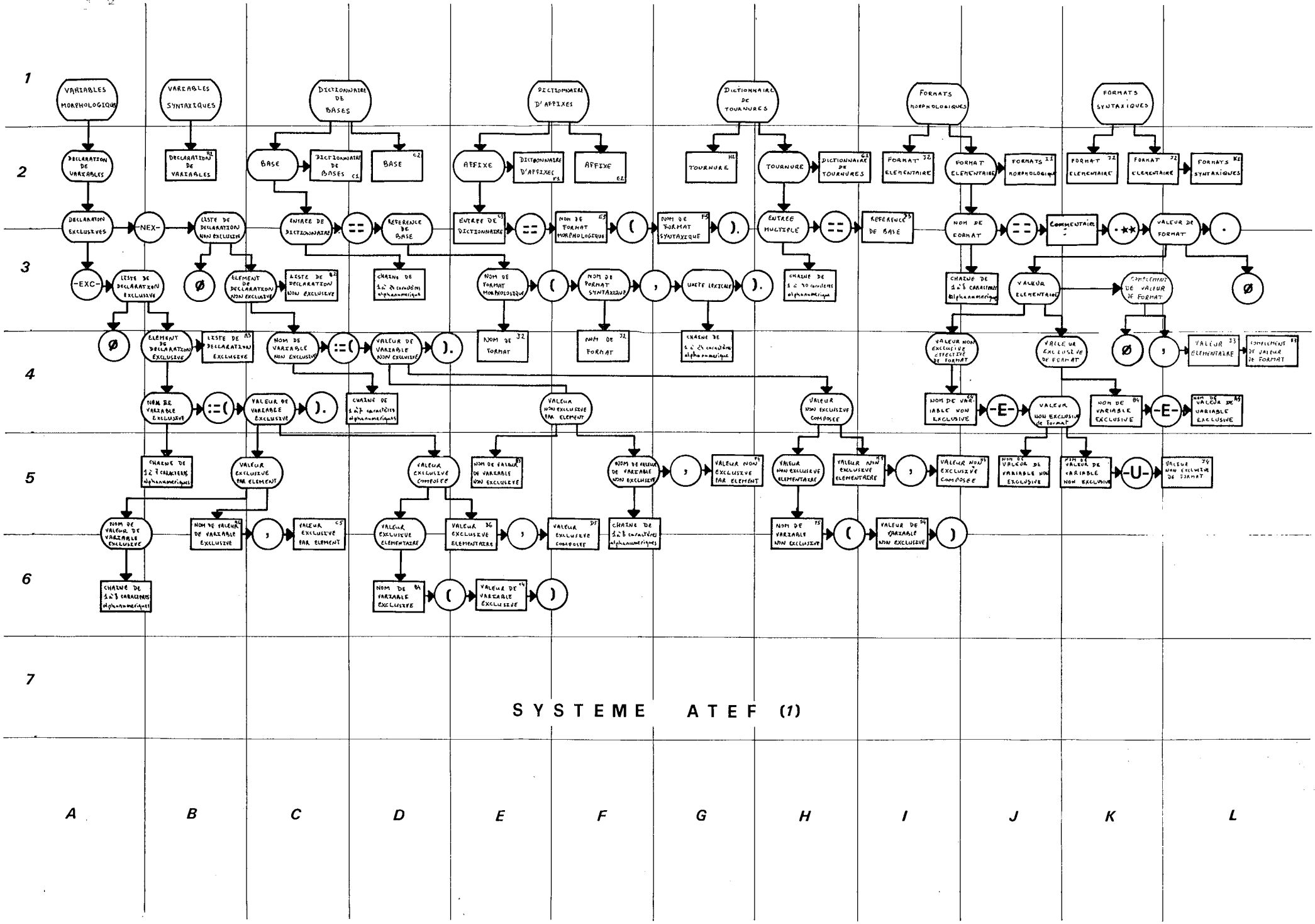
Lorsque la pile est vide, le procédé de back-tracking est déterminé et toutes les solutions ont été déterminées. Le rangement de ces solutions s'effectue suivant plusieurs arborescences de façon à obtenir l'ensemble des phrases ambiguës à ce niveau sous un même sommet appelé phrase. La détermination de la frontière d'une phrase est une fonction du système (-INIT-) et donc est laissée à l'utilisateur. De toute façon, le système arrête son analyse lorsque la zone réservée au résultat est remplie. Un superviseur peut éventuellement rappeler le système après avoir libéré cette zone. Dans ce cas l'analyse se poursuivra jusqu'à la fin du texte avec un traitement par morceau (dans le cas présent chaque morceau comprend environ 800 mots).

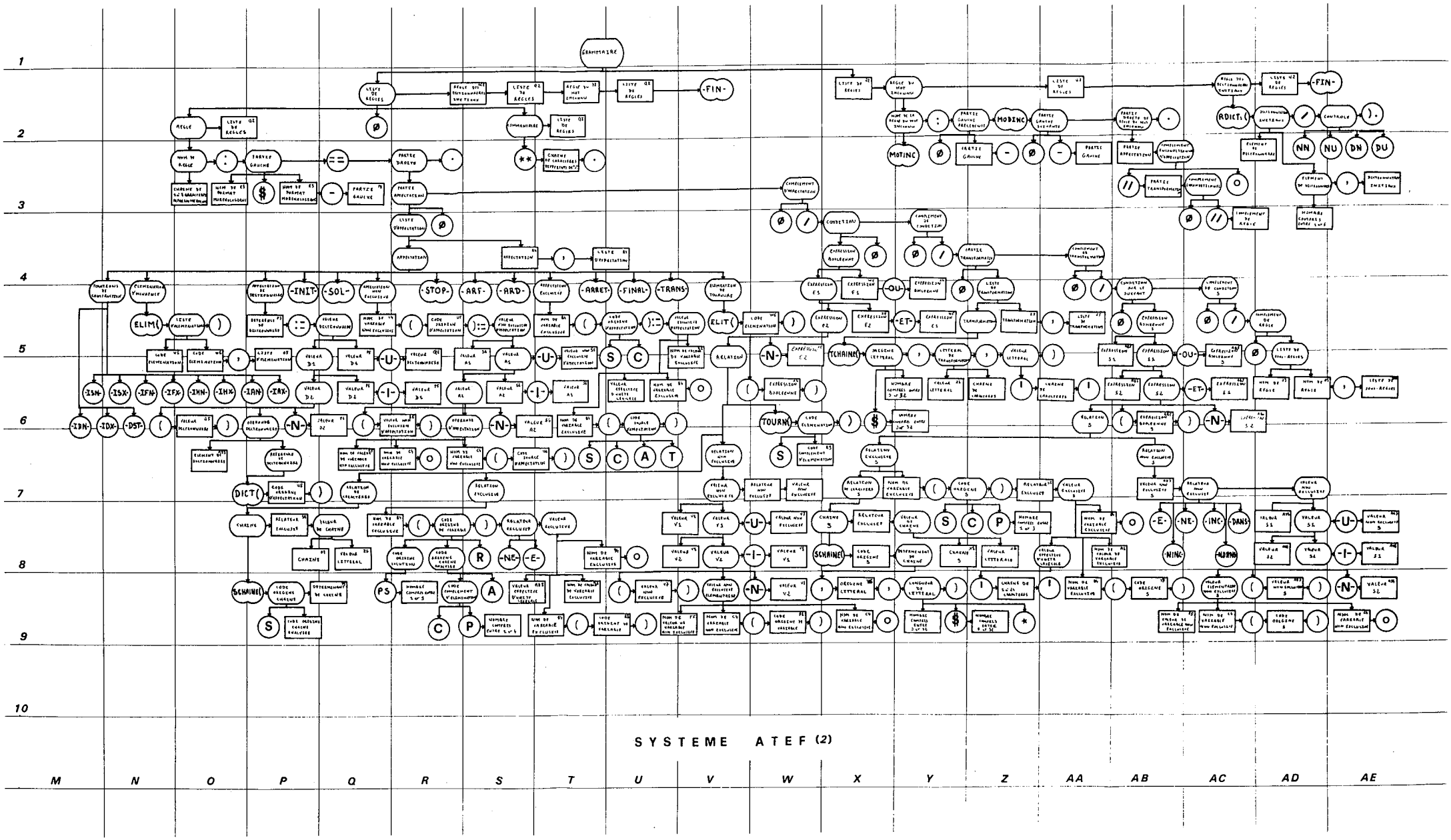
L'évolution des différentes conditions et affectations de l'état courant sont interprétées à partir d'un langage interne. Ce système comprend donc deux phases distinctes :

- La première phase analyse la syntaxe des différents composants et traduit ces données dans un langage interne.

- La deuxième phase applique le système sur un texte d'entrée.

La préparation des données ainsi que leurs modifications se font dans un mode conversationnel. L'application du système sur un texte donné est simplement commandée en mode conversationnel.


VARIABLES MORPHOLOGIQUES
VARIABLES SYNTAXIQUES
DICTIONNAIRE DE BASES
DICTIONNAIRE D'AFFIXES
DICTIONNAIRE DE TOURNURES
FORMATS MORPHOLOGIQUES
FORMATS SYNTAXIQUES
DECLARATION DE VARIABLES
DECLARATION DE VARIABLES
BASE
DICTIONNAIRE DE BASES
BASE
AFFIXE
DICTIONNAIRE D'AFFIXES
AFFIXE
TOURNURE
TOURNURE
DICTIONNAIRE DE TOURNURES
FORMAT ELEMENTAIRE
FORMAT ELEMENTAIRE
FORMATS MORPHOLOGIQUES
FORMAT ELEMENTAIRE
FORMAT ELEMENTAIRE
FORMATS SYNTAXIQUES
DECLARATION EXCLUSIVES
-NEX-
LISTE DE DECLARATION NON EXCLUSIVE
ENTREE DE DICTIONNAIRE
==
REFERENCE DE BASE
ENTREE DE DICTIONNAIRE
==
NOM DE FORMAT MORPHOLOGIQUE
(
NOM DE FORMAT SYNTAXIQUE
).
ENTREE MULTIPLE
==
REFERENCE DE BASE
NOM DE FORMAT
==
COMMENTAIRE
VALEUR DE FORMAT
-EXC-
LISTE DE DECLARATION EXCLUSIVE
Ø
ELEMENT DE DECLARATION NON EXCLUSIVE
LISTE DE DECLARATION NON EXCLUSIVE
CHAINE DE 1 à 24 caractères alphanumériques
NOM DE FORMAT MORPHOLOGIQUE
(
NOM DE FORMAT SYNTAXIQUE
,
SUITE LEXICALE
).
CHAINE DE 1 à 24 caractères alphanumériques
CHAINE DE 1 à 8 CARACTERES alphanumériques
VALEUR ELEMENTAIRE
COMPLEMENT DE VALEUR DE FORMAT
Ø
Ø
ELEMENT DE DECLARATION EXCLUSIVE
LISTE DE DECLARATION EXCLUSIVE
NOM DE VARIABLE NON EXCLUSIVE
::(
VALEUR DE VARIABLE NON EXCLUSIVE
).
NOM DE FORMAT
NOM DE FORMAT
CHAINE DE 1 à 24 caractères alphanumériques
VALEUR NON EXCLUSIVE EFFECTIVE DE FORMAT
VALEUR EXCLUSIVE DE FORMAT
Ø
,
VALEUR ELEMENTAIRE
COMPLEMENT DE VALEUR DE FORMAT
NOM DE VARIABLE EXCLUSIVE
::(
VALEUR DE VARIABLE EXCLUSIVE
).
CHAINE DE 1 à 24 caractères alphanumériques
VALEUR NON EXCLUSIVE PAR ELEMENT
VALEUR NON EXCLUSIVE COMPOSEE
NOM DE VARIABLE NON EXCLUSIVE
-E-
VALEUR NON EXCLUSIVE DE FORMAT
NOM DE VARIABLE EXCLUSIVE
-E-
NOM DE VALEUR DE VARIABLE EXCLUSIVE
CHAINE DE 1 à 7 CARACTERES alphanumériques
VALEUR EXCLUSIVE PAR ELEMENT
VALEUR EXCLUSIVE COMPOSEE
NOM DE VALEUR DE VARIABLE NON EXCLUSIVE
NOM DE VALEUR DE VARIABLE NON EXCLUSIVE
,
VALEUR NON EXCLUSIVE PAR ELEMENT
VALEUR NON EXCLUSIVE ELEMENTAIRE
VALEUR NON EXCLUSIVE ELEMENTAIRE
,
VALEUR NON EXCLUSIVE COMPOSEE
NOM DE VALEUR DE VARIABLE NON EXCLUSIVE
NOM DE VALEUR DE VARIABLE NON EXCLUSIVE
-U-
VALEUR NON EXCLUSIVE DE FORMAT
NOM DE VALEUR DE VARIABLE EXCLUSIVE
,
VALEUR EXCLUSIVE PAR ELEMENT
VALEUR EXCLUSIVE ELEMENTAIRE
VALEUR EXCLUSIVE ELEMENTAIRE
,
VALEUR EXCLUSIVE COMPOSEE
CHAINE DE 1 à 8 caractères alphanumériques
NOM DE VARIABLE NON EXCLUSIVE
(
VALEUR DE VARIABLE NON EXCLUSIVE
)
CHAINE DE 1 à 8 CARACTERES alphanumériques
NOM DE VARIABLE EXCLUSIVE
(
VALEUR DE VARIABLE EXCLUSIVE
)
SYSTEME ATEF (1)


GRAMMAIRE
LISTE DE REGLES
REGLE
LISTE DE REGLES
NOM DE REGLE
PARTIE GAUCHE
PARTIE DROITE
PARTIE APPELANTE
APPELATION
LISTE D'APPELATION
COMMENTAIRE
CHAINE DE CARACTERES
-FIN-
-INIT-
-SOL-
-STOP-
-ARF-
-ARD-
-ARRET-
-FINAL-
-TRANS-
ELIM(
DICT(
SCHAINE(
TOURN(
ELIT(
RELATION
TCHAINE
-ISN-
-ISX-
-IFN-
-IFX-
-IMN-
-IMX-
-IAN-
-IAX-
-IDN-
-IDX-
-DST-
-N-
-I-
-U-
-NE-
-E-
-INC-
-DANS-
-NN-
-NU-
-DN-
-DU-
MODINC
MOTINC
ADICT(
SYSTEME ATEF (2)

CHAPITRE VII

LE SYSTEME C.E.T.A.

INTRODUCTION

Le traitement des arborescences, de plus en plus fréquent tant en analyse des langues naturelles qu'en calcul formel, traitement des listes, etc..., nécessite la construction de grammaires transformationnelles. Le système C.E.T.A. est un système construit pour permettre la définition et la manipulation de telles grammaires. Il a pour but la définition de grammaires transformationnelles décidables définies au chapitre IV.

Un tel système peut être utilisé pour l'analyse des langues naturelles. Aussi, le vocabulaire utilisé emprunte en partie, des termes couramment employés pour ce type de traitement. Une des principales caractéristiques du traitement des langues naturelles est donnée par l'importance des éléments à traiter. Une grammaire transformationnelle d'analyse de langue naturelle ne contient, en général, que des règles relativement simples, mais leur nombre est très important. Les arborescences traitées par ces grammaires sont également très importantes. Le système C.E.T.A. a donc été conçu pour une manipulation importante de données. La possibilité dans le système C.E.T.A. de définir n'importe quelle grammaire à l'intérieur d'un ensemble de grammaires décidables permet une application de ce système à de nombreux domaines.

Le centre du système comprend deux parties. La première partie est constituée par l'ensemble des grammaires et la définition de leurs enchaînements.

La seconde partie est formée par la description des règles des différentes grammaires du système. Le système est non déterministe et fournit comme solution la première atteinte. L'algorithme de recherche de la première solution est laissée à l'utilisateur, permettant à celui-ci de prévoir le résultat du système.

ia définition et l'utilisation de ce système sont prévues en mode conversationnel et supportées par le système CP/CMS.

A - PRINCIPE DU SYSTEME

Le système C.E.T.A. est un système simulant un transducteur à pile composé qui lui même simule une grammaire transformationnelle décidable. Les différents types de grammaires décidables définies au chapitre IV sont réalisables dans le système C.E.T.A.. Les arborescences traitées par ce système sont des arborescences étiquetées, le vocabulaire d'étiquette est défini soit par les déclarations d'étiquettes, soit par les déclarations de variables du système A.T.E.F. éventuellement modifié par les déclarations d'étiquettes. Les déclarations d'étiquettes sont identiques aux déclarations de variables du système A.T.E.F.. Le système comporte un ensemble de grammaires composées linéairement. L'ensemble de ces grammaires et leur composition est défini par l'utilisateur dans la partie "-GRAM-" de l'élément d'entrée. L'ensemble des règles des différentes grammaires sont également définies dans la partie "-GRAM-" de l'élément d'entrée. L'application d'une grammaire est effectuée par un transducteur universel. Ce transducteur construit la fonction de transition correspondant à cette règle que si cette règle n'appartient pas à une autre grammaire déjà appliquée. Les règles des différentes grammaires sont mises sous forme d'un langage isomorphe au langage (L,V).

Une grammaire élémentaire est caractérisée par un ensemble de règles, chacune de ces règles pouvant être une règle simple ou récursive.
Une règle récursive est définie par un appel à une autre grammaire avec une contrainte dans l'ensemble des points transformables (grammaire sélective).
Une grammaire élémentaire peut etre de type unitaire ou exhaustif.
Une grammaire de type unitaire se caractérise par le fait qu'une seule règle de la grammaire peut être appliquée sur un point ou ses descendants.
Une grammaire de type exhaustif se caractérise par le fait qu'une règle de la grammaire ne peut s'appliquer qu'une seule fois sur un point ou ses descendants.
Une grammaire conditionnelle est composée d'une grammaire élémentaire et d'un ensemble de compositions directes conditionnelles sur différentes grammaires (aiguillage). Le résultat de cette grammaire sera pris en compte si une grammaire élémentaire dépendante non conditionnelle est appliquée sur le résultat d'une grammaire conditionnelle dépendante de celle-ci.

Une règle de grammaire définit une transformation d'arborescence et est caractérisée par les quatre éléments suivants :

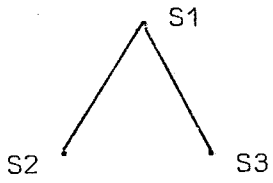
- un schéma d'arborescence, partie gauche de la règle
- une arborescence, partie droite de la règle
- une fonction de transfert
- une suite d'affectation ou modification d'étiquettes.

Enfin un ensemble de constantes d'étiquettes peut être disponible dans le fichier de formats du système.

1 - REGLES DE GRAMMAIRE

L'ensemble des règles de grammaire est défini dans la partie correspondante du fichier d'entrée. A chaque règle de grammaire est associée une étiquette ou nom de règle. La partie gauche de la règle est caractérisée par un schéma d'arborescence permettant d'appliquer cette règle sur un ensemble de cas de figures.
Exemple de schéma :

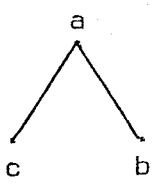
Soit le schéma *S1(S2,S3)*, il représente l'arborescence

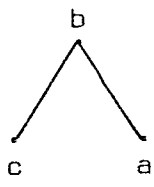


Des conditions sur les sommets permettent une première factorisation d'arborescence :
Soit le schéma :

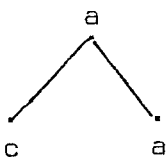
S1(S2, S3) / S1 : ETQ -E- A -OU- ETQ -E- B, S2 : ETQ -E- C /
ETQ(S3) -NINC- (ETQ(S1) -U- ETQ(S2)

Il représente les arborescences suivantes :





Il ne peut représenter l'arborescence :



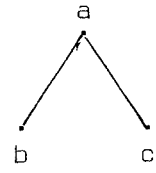
On peut remarquer que les conditions sur les points de l'arborescence sont séparées en deux groupes :

- le premier groupe de conditions concerne les conditions qui doivent être vérifiées par un sommet. Ces conditions sont dites conditions propres.

- le deuxième groupe est composé d'une condition portant sur l'ensemble des sommets du schéma ; cette condition est dite condition inter-sommets. Ces conditions sont des conditions booléennes comparant soit l'état des différentes étiquettes associées aux points du schéma, soit les structures dépendantes de ces différents points. Dans l'exemple précédent, on a supposé que l'étiquette sur chaque point contenait l'élément *ETQ* qui lui-même pouvait prendre les valeurs non exclusives a,b,c. A l'intérieur d'un même schéma, il est possible d'imposer un ordre entre les différents points ou de préciser que l'ordre des descendants d'un point est indifférent. Ainsi, en préfixant le point *S1* par l'élément "*&B*", le schéma résultant est le suivant :

S1 &B(S2,S3) / S1 : ETQ -E- A -OU- ETQ -E- B, S2 : ETQ -E- C / ETQ(S3) -NINC- (ETQ(S1) -U- ETQ(S2)).

permet de définir les arborescences précédentes mais aussi l'arborescence :



L'ordre entre les différents descendants d'un point peut être obtenu de manière plus complexe. L'ensemble des descendants d'un point est séparé en plusieurs groupe ; chaque groupe est séparé par le marquant ";". A l'intérieur d'un groupe, les éléments sont séparés par le marquant habituel ",". L'ordre peut être imposé sur les groupes et non sur les éléments des groupes, ou au contraire sur les éléments et non sur les groupes.

Exemple :

Soit le schéma *S1(S2, S3 ; S4, S5, S6 ; S7)*

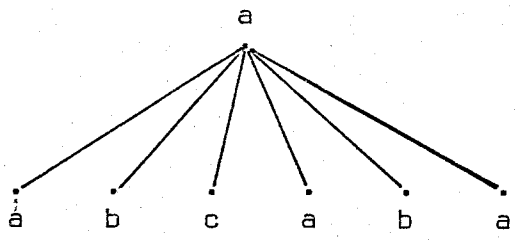
Ce schéma se compose de 7 points et l'ensemble des descendants du point *S1* est composé de 3 groupes :

{S2, S3}
{S4, S5, S6}
{S7}

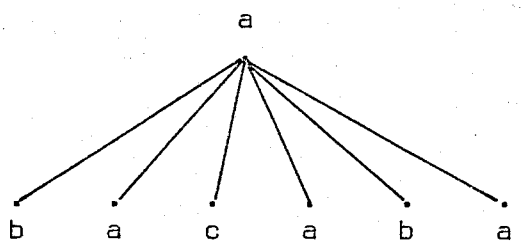
Sans éléments supplémentaires, les points référencés par *S2, S3, S4, S5, S6, S7* doivent être ordonnés dans l'arborescence image. Ainsi le schéma :

S1(S2, S3 ; S4, S5, S6 ; S7) / S2 : ETQ -E- A, S3 : ETQ -E- B

représente l'arborescence :



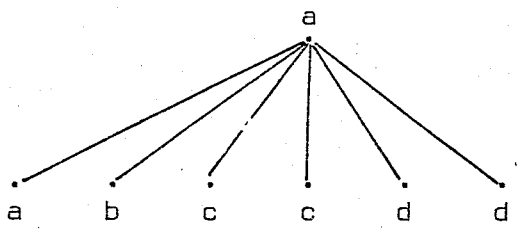
mais ne représente pas l'arborescence :



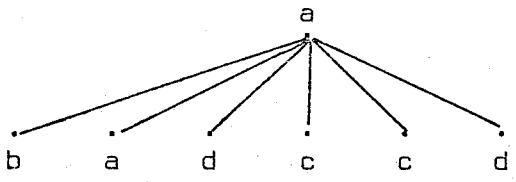
En précisant que l'ordre peut être indifférent entre les éléments des différents groupes, cette deuxième arborescence appartiendrait à l'ensemble des arborescences images d'un schéma donné. Ainsi le schéma suivant :

S1 &B (S2, S3 ; S4, S5, S6 ; S7) / S2 : ETQ -E- A, S3 : ETQ -E- B,
S4 : ETQ -E- C , S6 : ETQ -E- D

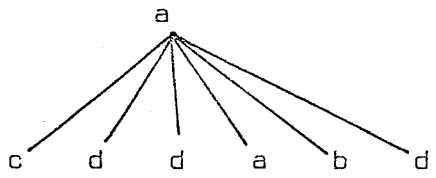
représente l'arborescence



ainsi que l'arborescence :



mais ne représente pas l'arborescence :



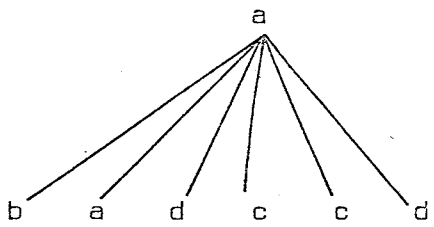
En précisant que l'ordre peut être indifférent entre les différents groupes, l'ensemble des arborescences images devient différent.

Ainsi le schéma

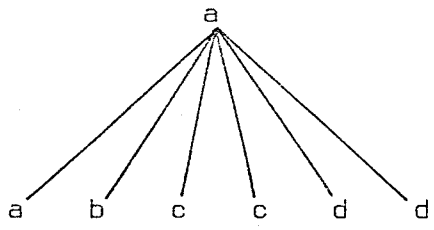
S1 &M(S2, S3 ; S4, S5, S6 ; S7) / S2 : ETQ -E- A, S3 : ETQ -E- B,

S4 : ETQ -E- C, S6 : ETQ -E- D

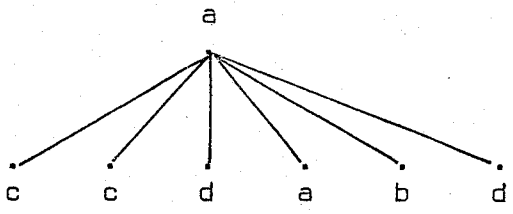
ne représente pas l'arborescence



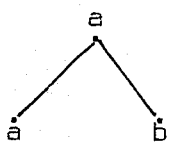
mais représente l'arborescence :



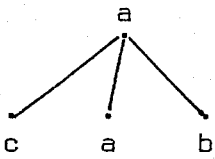
et l'arborescence :



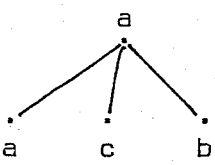
Entre deux points ordonnés d'un schéma, une condition de juxtaposition de ces points peut être imposée.
Ainsi le schéma *S1(S2,* , S3) / : ETQ -E- a, S3 : ETQ -E- b*
représente l'arborescence :



qui est une sous-arborescence de l'arborescence :



mais qui n'est pas une sous-arborescence de l'arborescence :



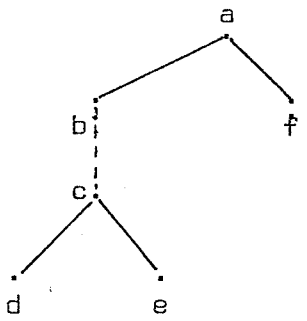
car les points vérifiant les conditions de S2 et S3 sont séparés par un 3ème point dont l'étiquette est "C" et qui ne peut donc être ni un point S1 ni un point S2.

Un schéma peut aussi représenter un ensemble d'arborescences hiérarchisées. Une arborescence dépendante est alors recherchée sous une feuille désignée de l'arborescence principale. On précise cette dépendance en plaçant l'arborescence entre deux parenthèses, la première étant précédée du marquant "?".

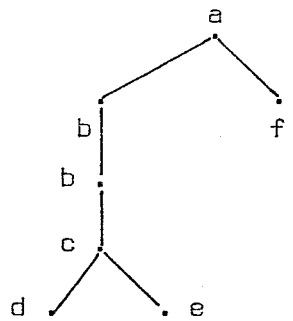
Exemple :

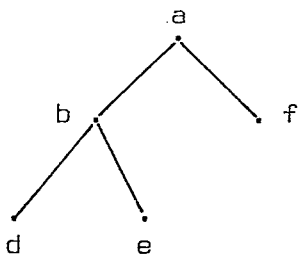
S1(S2 ?(S3(S4,S5)),S6) / S1 : ETQ -E- a, S2 : ETQ -E- b,
S3 : ETQ -E- c -OU- ETQ -E- b, S4 : ETQ -E- d, S5 : ETQ -E- e,
S6 : ETQ -E- f

Le schéma représente l'arborescence :

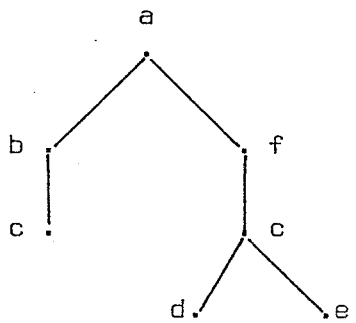


Ainsi l'ensemble des deux arborescences a(b,f) et c(d,e) sont des sous arborescences des éléments suivants :





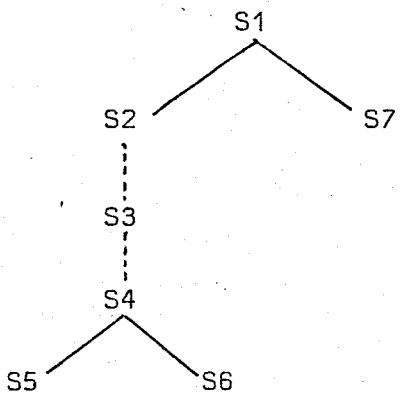
Mais cet ensemble n'est pas un ensemble de sous arborescences de l'arborescence :



L'ensemble de sous arborescence hiérarchisée ainsi défini n'est pas limité. Ainsi dans l'exemple précédent, le point correspondant à d (feuille de l'arborescence c(d,e)) peut lui-même être le début d'une recherche d'une troisième arborescence :

$$S1(S2\ ?\ (S3\ ?\ (S4(S5,S6))),\ S7)$$

représente l'ensemble d'arborescences suivant :



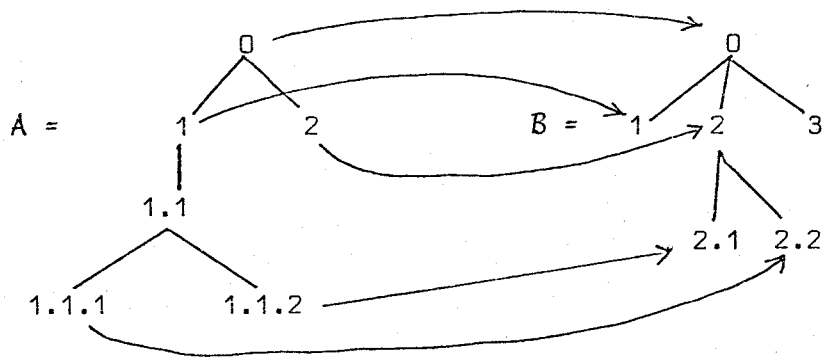
La partie droite d'une règle de grammaire est caractérisée par trois éléments : l'arborescence résultante, la fonction de transfert, les modifications d'étiquettes.

L'arborescence résultante définit la transformation de l'arborescence image reconnue. Cette arborescence ne contient pas les éléments ne représentant que des conditions sur l'arborescence image (sommets jointifs, éléments définissant l'ordre des éléments).

La fonction de transfert permet de définir complètement la transformation du point de vue de la structure.

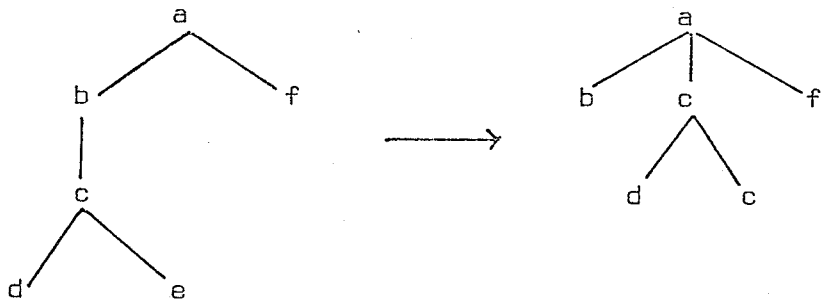
Exemple :

La transformation $(A, \mathcal{B}, \tau)$ définie par :

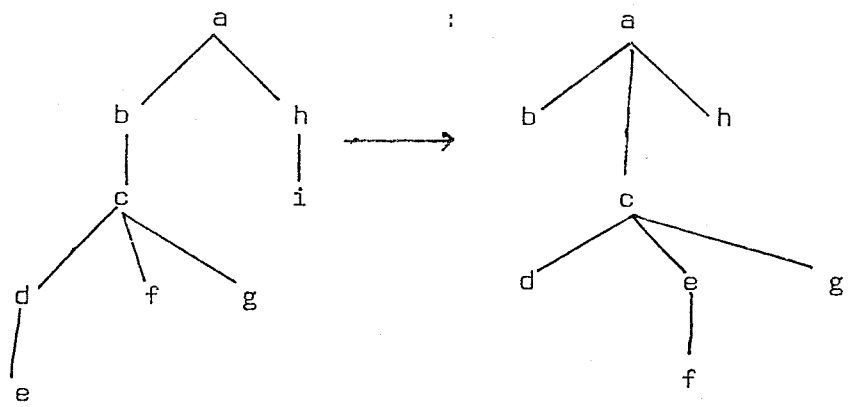


τ : $\{0 \rightarrow 0\ ,\ 1 \rightarrow 1\ ,\ 2 \rightarrow 2\ ,\ 1.1.1 \rightarrow 2.2\ ,\ 1.1.2 \rightarrow 2.1\}$

permet les transformations suivantes :



Ici les symboles sur les sommets ne sont pas concernés par les transformations.



(Le point "g" est éliminé car la fonction de transfert pour le point "c" n'est pas définie)

Cette transformation s'écrit dans le langage CETA :

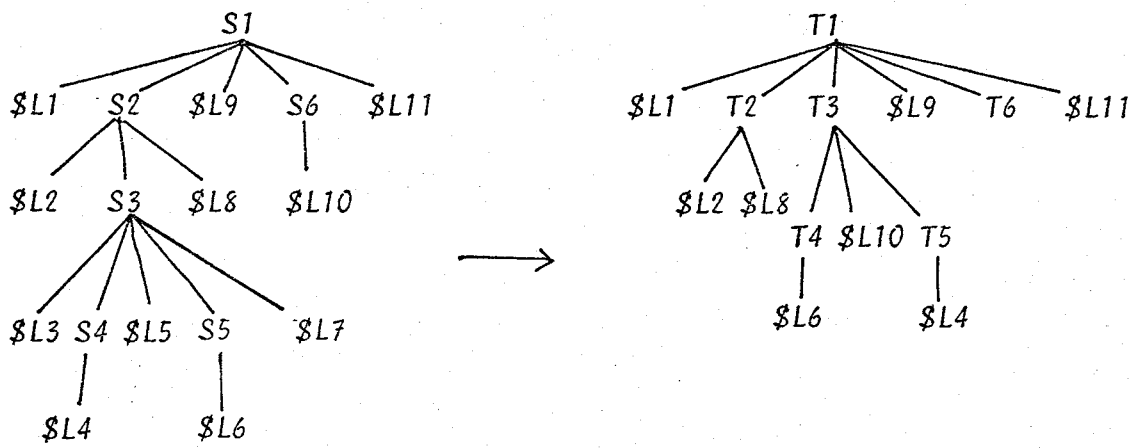
```
S1(S2(S3(S4,S5)),S6) // == T1(T2,T3(T4,T5),T6) /

T1   <--   S1;
T2   <--   S2;
T3   <--   S6;
T4   <--   S5;
T5   <--   S4;
T6   <--   * ;
*    <--   S3
```

On peut remarquer que la fonction définie ainsi est une fonction totale. Lorsque la fonction τ n'est pas définie sur un point de l'arborescence, l'écriture dans le langage CETA doit définir l'élément "*" comme image de ce point. De même lorsqu'un point de l'arborescence B n'a pas d'antécédent dans l'arborescence A, l'écriture de cette fonction donne l'élément "*" comme antécédent de ce point.

Afin de raffiner la fonction de transfert, un paramètre de "liste" peut être mentionné en partie gauche et droite d'une règle. Cet élément de liste définit alors la fonction de transfert. Un paramètre de liste est précédé du symbole "$". L'exemple suivant peut être alors écrit de la façon suivante :

```
S1($L1, S2($L2, S3($L3,S4($L4), $L5, S5($L6), $L7), $L8), $L9,
S6($L10), $L11) == T1($L1, T2($L2, $L8), T3(T4($L6), $L10, T5($L4)),
$L9, T6, $L11)
```

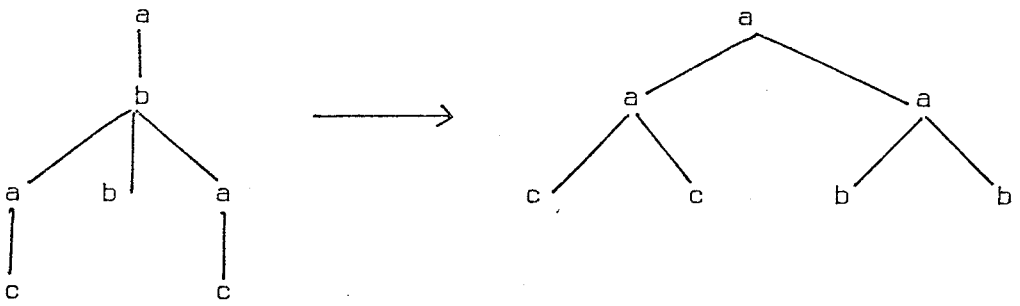


La fonction de transfert peut être définie aussi de façon implicite par un sommet portant la même étiquette en partie gauche et droite de la règle. Ainsi l'exemple précédent peut s'écrire :

```
S1(S2(S3(S4,S5)), S6) // == S1(S2, S6(S5, S4), T6) /
   *  <--  S3,
  T6  <--  * .
```

Le système manipule des arborescences étiquetées, le troisième élément de la partie droite d'une règle permet de faire évoluer les étiquettes des différents points. La modification d'une étiquette est effectuée en deux opérations. La première consiste à former un ensemble de variables qui provient soit d'une étiquette présente dans l'arborescence image, soit d'un format défini dans l'ensemble des formats du système (constante). Le deuxième élément de la définition des étiquettes est constitué par une suite de modifications ou d'affectations de l'ensemble des variables de cette étiquette. Les variables entrant dans cette affectation peuvent être prises soit parmi les étiquettes sources de la sous-arborescence image reconnue, soit parmi les variables des différents formats définis dans le système.

Exemple de règle de transformation :

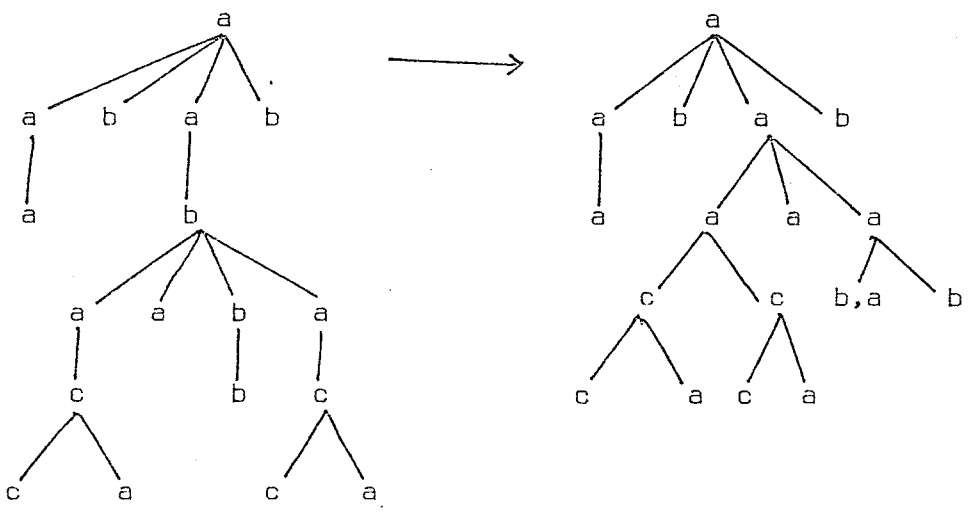


S1(S2(S3(S4), S5, S6(S7))) / S1 : ETQ -E- a, S2 : ETQ -E- b,
S5 : ETQ -E- b / ETQ(S3) -I- ETQ(S2) -NE- ETQ 0 -ET- ETQ(S4)
-E- ETQ(S7) ==
T1(T2(T3, T4), T5(T6, T7)) /

T1 <-- S1, S2 ;
T2 <-- S3;
T3 <-- S4;
T4 <-- S7;
T5 <-- S6;
*T6 <-- * ;*
*T7 <-- * ;*
** <-- S5 /*

T1 : S1 ; T2 : S3 ; T3 : S4 ; T4 : S7 ; T5 : S6 ; T6 : S2, ETQ :=
ETQ(S6) -u- ETQ(S1) ; T7 : S5 .

Cette transformation ainsi définie permet la transition suivante :



Lorsqu'un sommet porte la même étiquette en partie gauche et droite de l'arborescence, la fonction de transfert est définie implicitement. L'affectation d'étiquette est également définie pour ce point. Chacune de ces conventions est annulée dès que le nom du sommet intervient dans la définition de la fonction de transfert ou de l'affectation d'étiquette. Avec ces deux conventions, la transformation précédente s'écrit ainsi :

```
S1(S2(S3(S4), S5, S6(S7))) / S1 : ETQ -E- a, S2 : ETQ -E- b ,
    S5 : ETQ -E- b / ETQ(S3) -I- ETQ(S2) -NE- ETQ 0 -ET- ETQ(S4) -E-
    ETQ(S7) == S1(S3(S4,S7), S6(S2, S5)) /
        S1 <-- S1, S2 ;
        S6 <-- * ;
        S5 <-- * ;
        *  <-- S5 /
        S2 : S2 , ETQ := ETQ(S6) -u- ETQ (S1) .
```

Une règle comporte en outre un dernier élément permettant le choix d'une figure lorsque plusieurs figures sont possibles. Le choix de la figure s'opère en fonction des deux critères suivants :

- choix de la racine de la sous-arborescence qui peut être choisie le plus haut possible ou le plus bas possible. Lorsque la reconnaissance de plusieurs sous-arborescences est recherchée, le choix des différentes racines est indépendant des autres choix possibles.

- choix de l'ensemble des descendants d'un point qui peut être pris le plus à gauche possible ou le plus à droite possible. Le choix concernant un point se fait indépendamment des autres choix effectué sur les autres points.

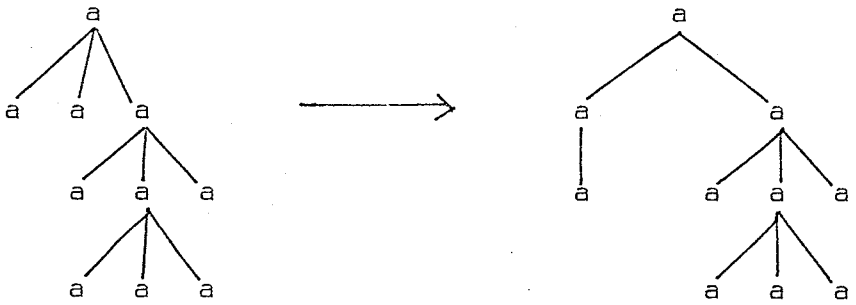
Exemple :

Soit la transformation définie par :

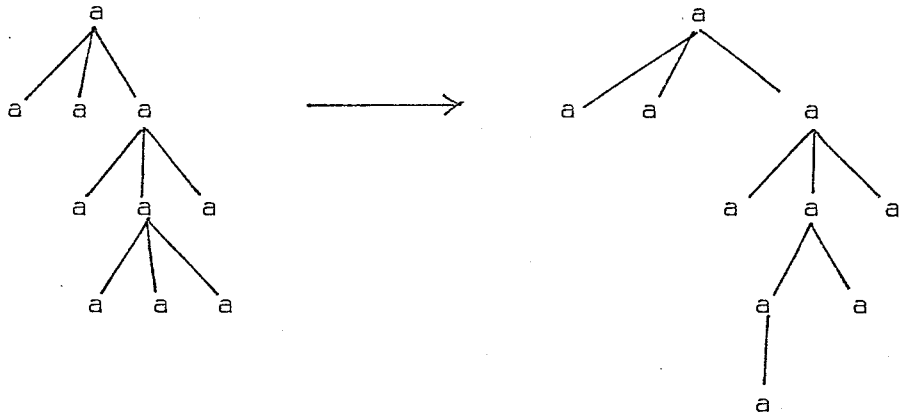
```
S1(S2, S3, S4) / S1 : ETQ -E- a, S2 : ETQ -E- a, S3 : ETQ -E- a,
    S4 : ETQ -E- a / == S1(S2(S3), S4).
```

Choix de la racine :

- haut

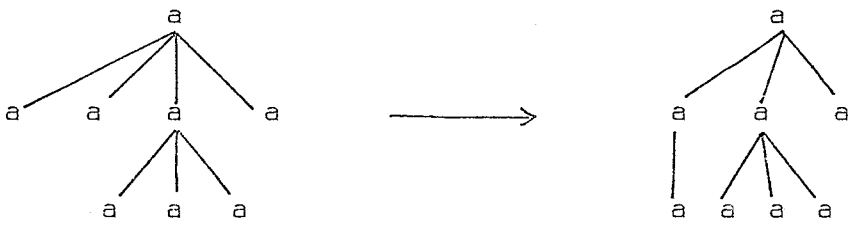


- bas

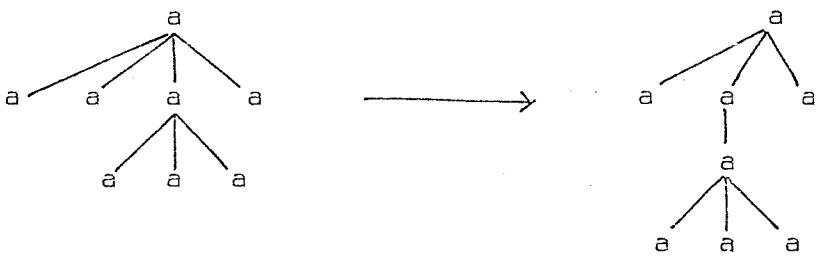


Choix sur l'ensemble des descendants de S1 :

- gauche



- droit



2 - GRAMMAIRE

a - Grammaire élémentaire

Une grammaire est constituée par une suite ordonnée de règles. Deux modes d'activation d'une grammaire sont possibles. Dans le premier mode d'activation dit mode "unitaire", l'application de la grammaire à une arborescen donnée est effectuée une seule fois sur les points de l'arborescence. Un point est la racine d'une transformation d'une règle d'une grammaire donnée si ce point est la racine de la sous-arborescence reconnue pour cette règle et si aucun point parmi ces descendants ne peut être la racine d'une transformation pour une règle de la même grammaire et de rang inférieur. Dans le cas où un point est la racine d'une transformation aucun ancêtre de ce point peut lui-même être la racine d'une autre transformation. Aussi les transformations simultanées réalisées par une grammaire sont effectuée sur un ensemble de sous-arborescences étrangères et indépendantes. Dans le mode "unitaire", le résultat de la grammair est obtenu après une application simultanée définie ci-dessus.

Dans le mode exhaustif, après une application simultanée de la grammaire, une nouvelle application est de nouveau recherchée.
Dans ce cas un point racine d'une transformation ne pourra plus être la racine de la même transformation ainsi que l'ensemble de ces descendants. Le résultat de la grammaire est donné lorsqu'aucune règle n'est applicable.

Exemple :

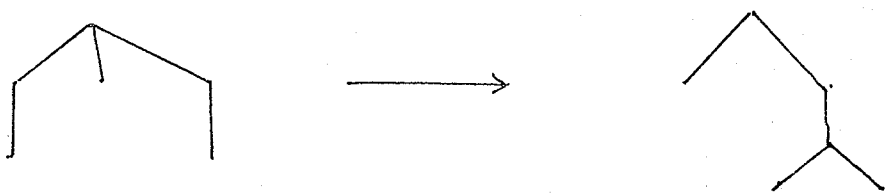
Soient les trois règles suivantes :

R1 : S1(S2(S3), S4, S5(S6)) / S1 : ETQ -E- a/ == S1(S2,S3(S4(S5,S6))).

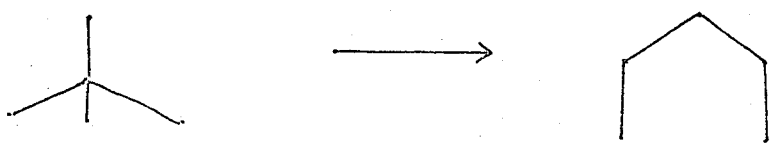
R2 : S1(S2(S3,S4,S5)) / S1 : ETQ -E- a / == S1(S2(S3), S4(S5)).

R3 : S1(S2(S3), S4(S5)) / S1 : ETQ -E- a / == S1(S2(S3,S4,S5)).

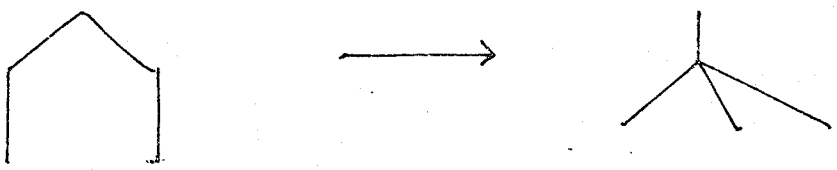
R1 :



R2 :

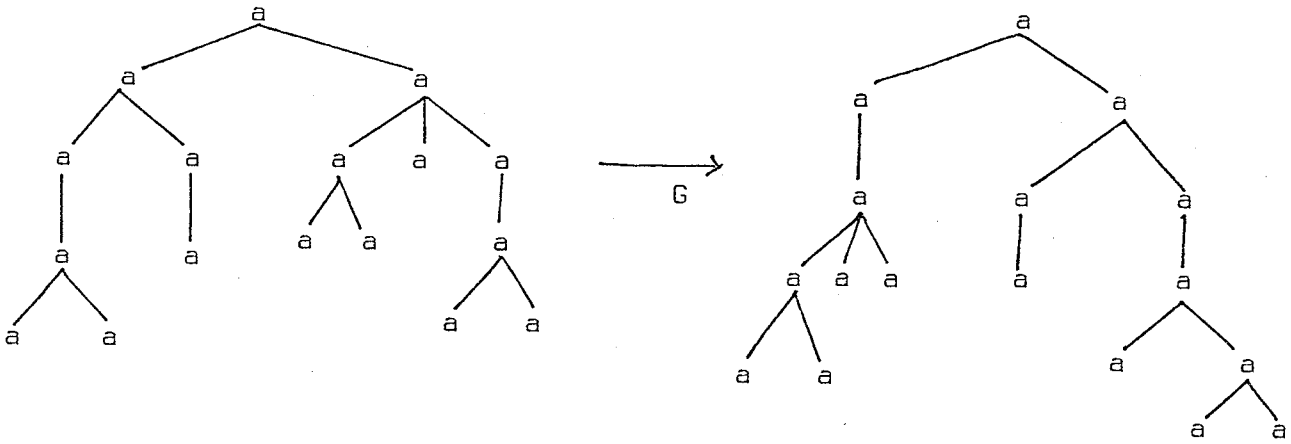


R3 :

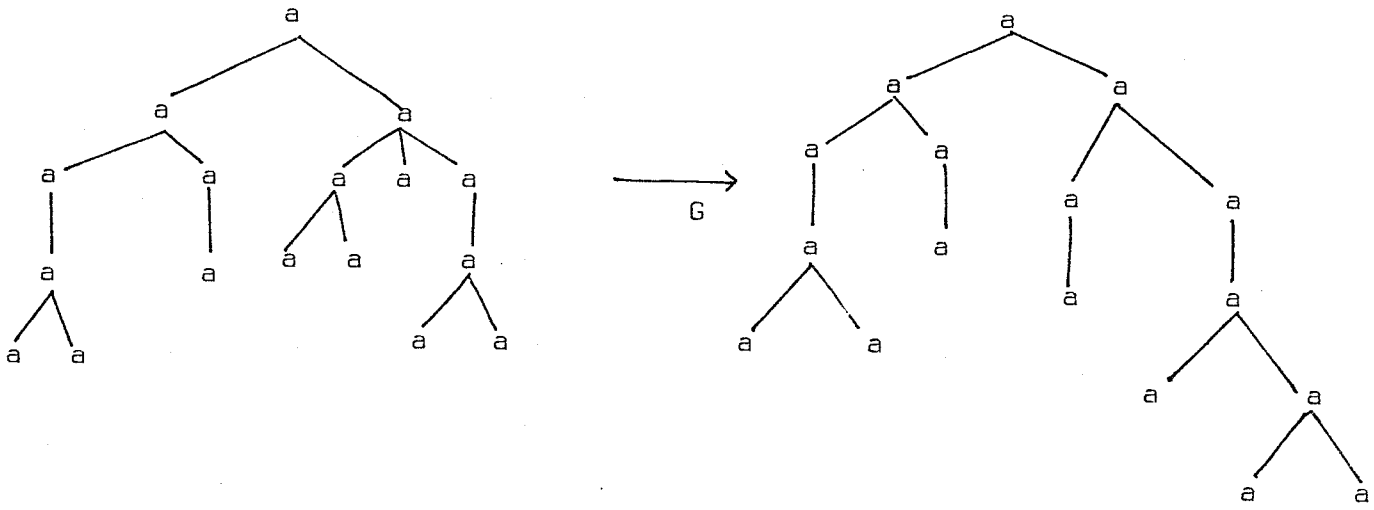


Soit alors la grammaire G composée de trois règles R1, R2, R3 prises dans cet ordre.

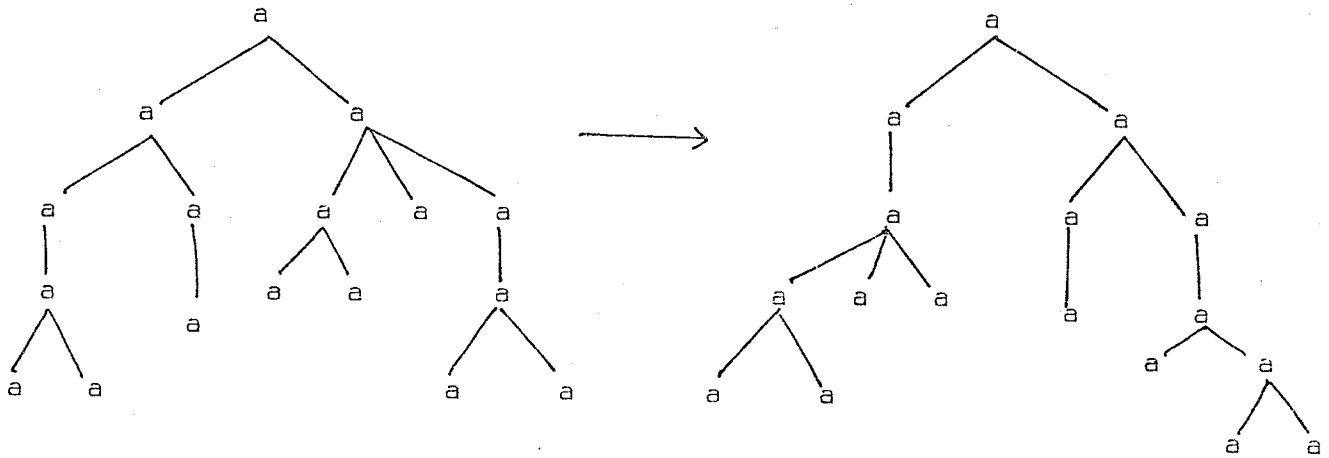
Si la grammaire est en mode unitaire :

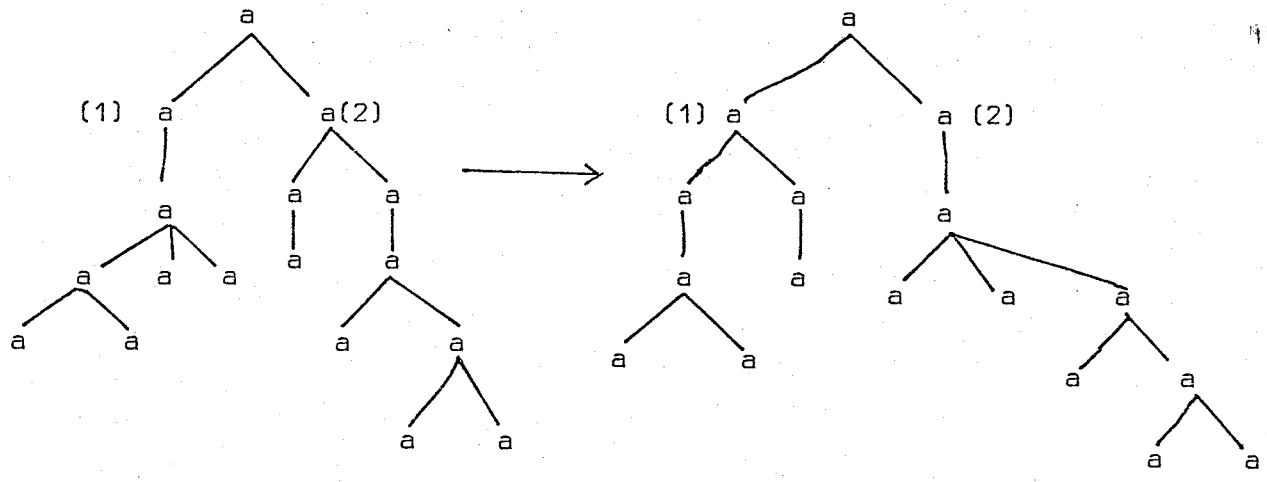


Si la grammaire est en mode exhaustif :

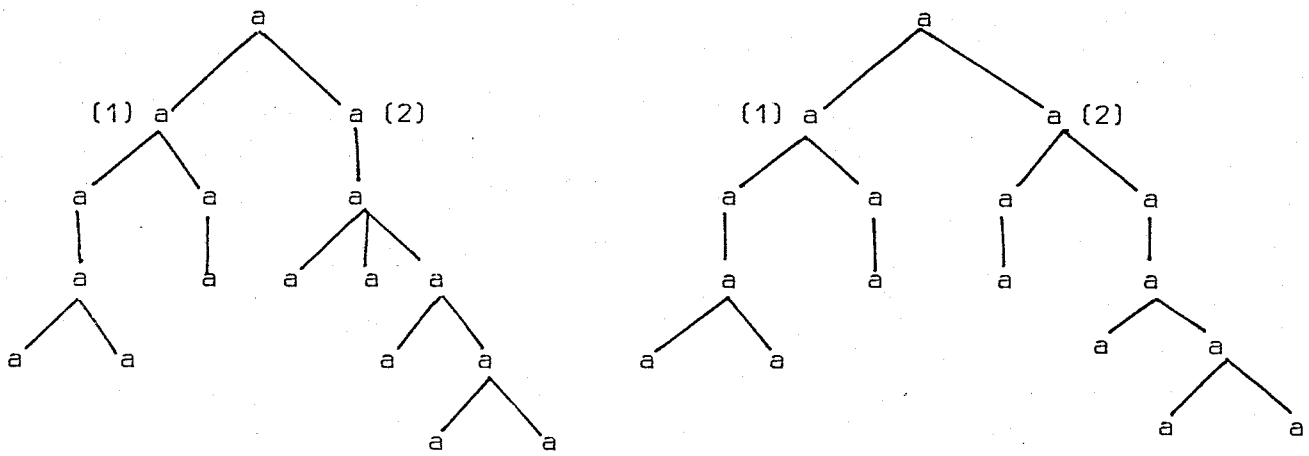


Les différentes phases sont définies de la façon suivante :

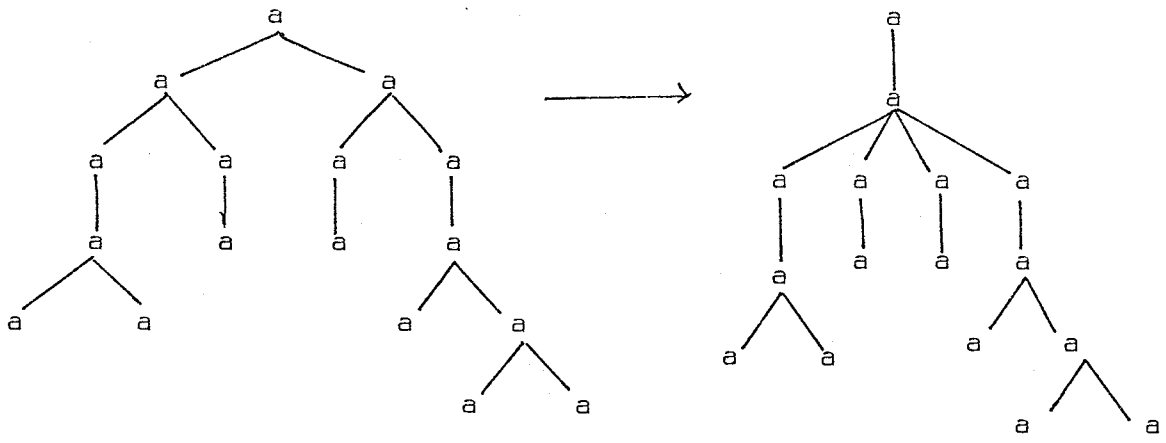


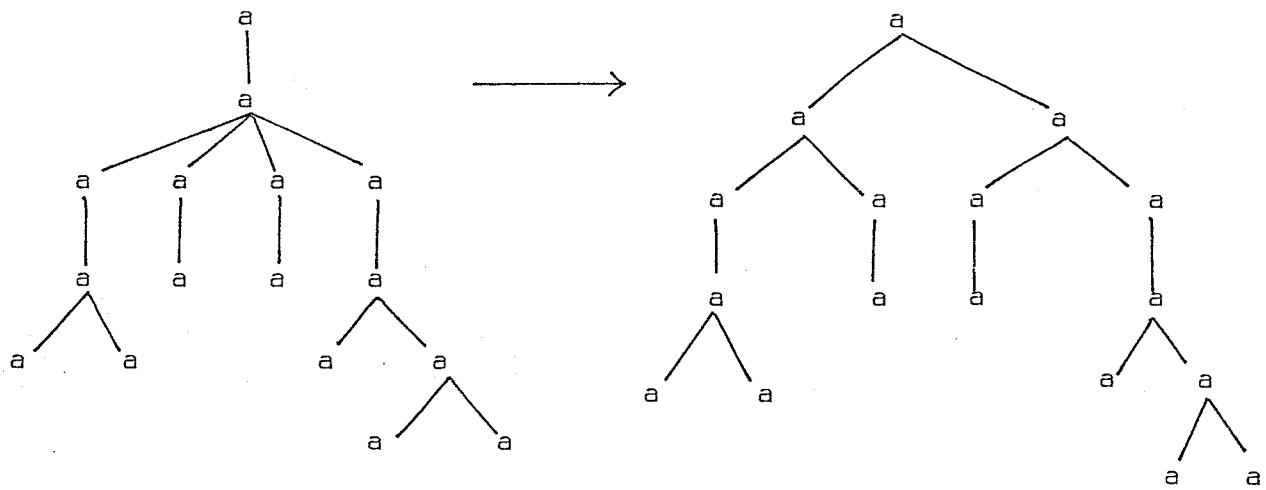


Le point (1) ou ses descendants ne peuvent être la racine de la transformation R3 et le point (2) ou ses descendants ne peuvent être la racine de la transformation R1 avant la nouvelle application de cette grammaire. Après cette 2ème application, les restrictions augmentent sur les mêmes points : le point (1) ou ses descendants ne peuvent être la racine de la transformation R3 et celle de R2, le point (2) ou ses descendants ne peuvent être la racine de la transformation R1 ou celle de R3.



L'état du point (1) n'a pas varié tandis que le point (2) est tel qu'il ne peut être (ainsi que ses descendants) la racine d'une transformation R1, R2, R3.





Une grammaire peut compoter des règles dites récursives. Une règle récursive est caractérisée par trois éléments associés :

- le premier de ces éléments est une grammaire dans laquelle seront choisies les règles participant à la récursion,

- le deuxième élément est la suite des règles de la grammaire mentionnée qui participent à la récursion,

- le troisième élément est une sous-arborescence de l'arborescence résultante de la règle récursive. Cette sous-arborescence doit avoir un nombre de points inférieur au nombre de points présents dans la partie gauche de la règle.

Le résultat de l'application d'une règle récursive est donné par l'algorithme suivant :

- Application de la règle si elle est applicable
- Sur le résultat obtenu par cette transformation, application de la grammaire précisée en conservant dans cette grammaire les règles énumérées.

Cette grammaire sera appliquée sur l'arborescence ayant pour racine le point mentionné en racine de la sous-arborescence mentionnée en troisième élément de la règle récursive. Lors de l'application de la grammaire récursive, tous les points provenant de l'application de la règle (présents en partie droite de la règle) qui ne sont pas présents dans cette sous-arborescence de récursion ne peuvent être pris comme points d'un nouveau schéma.

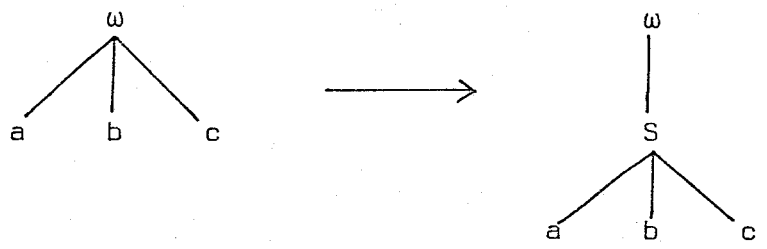
Exemple d'application :

Soit la grammaire G1 formée des deux règles R1 et R2 définies de la façon suivante :

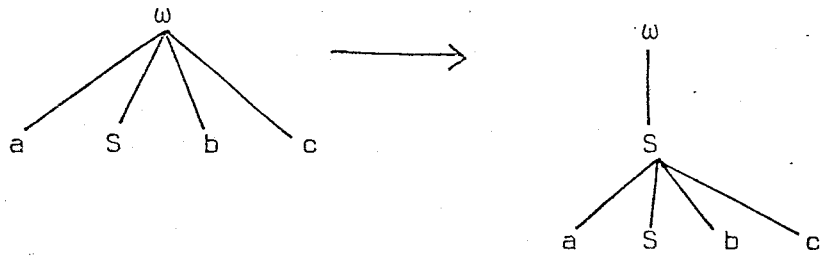
*R1 : S1($M1, S2, *, S3, $M2, S4,*) / S1 : ETQ -E-ω , S2 : ETQ -E- a, S3 : ETQ -E- b, S4 : ETQ -E- c / == S1($M1, S(S2,S3,S4), $M2) / S <-- * / S : * , ETQ := S.*

*R2 : S1($M1, S2, * , S3, * , S4, $M2, S5, *) / S1 : ETQ -E- ω, S2 : ETQ -E- a, S3 : ETQ -E- S, S4 : ETQ -E- b, S5 : ETQ -E- c/== S1($M1, S(S2, S3, S4, S5), $M2) / S <-- * /S:* , ETQ := S.*

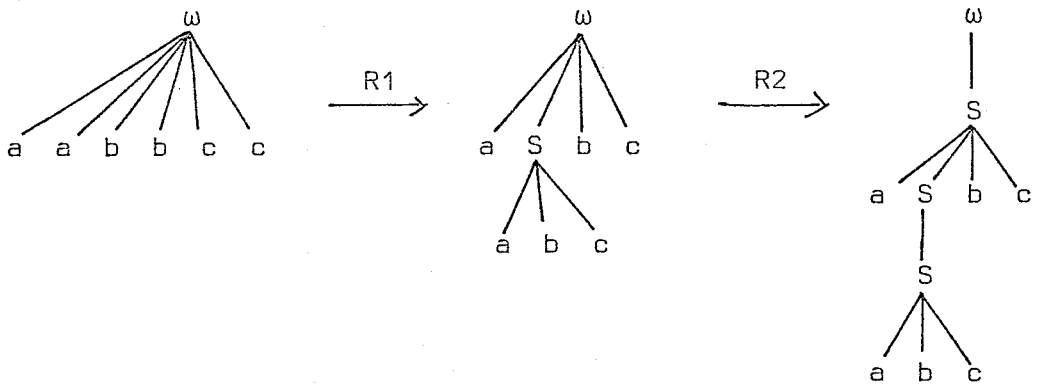
R1 :



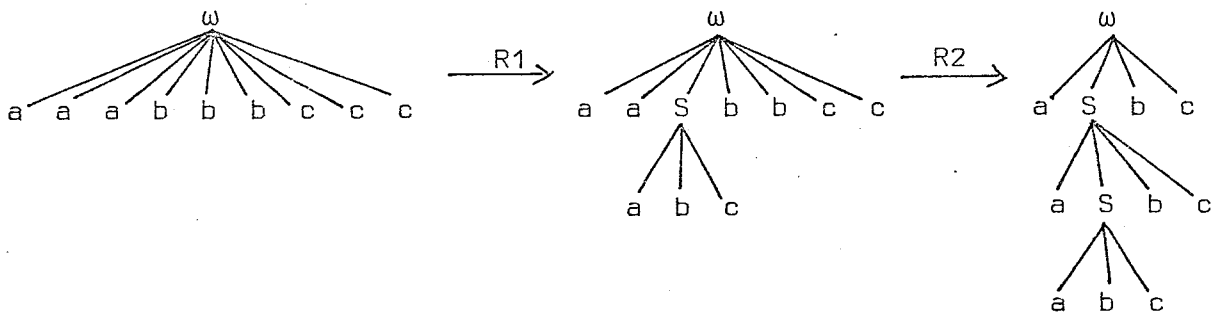
R2 :



Cette grammaire appliquée en mode exhaustif à l'arborescence ω(a,a,b,b,c,c,) donne l'arborescence ω(S(a,S(a,b,c),b,c)) :



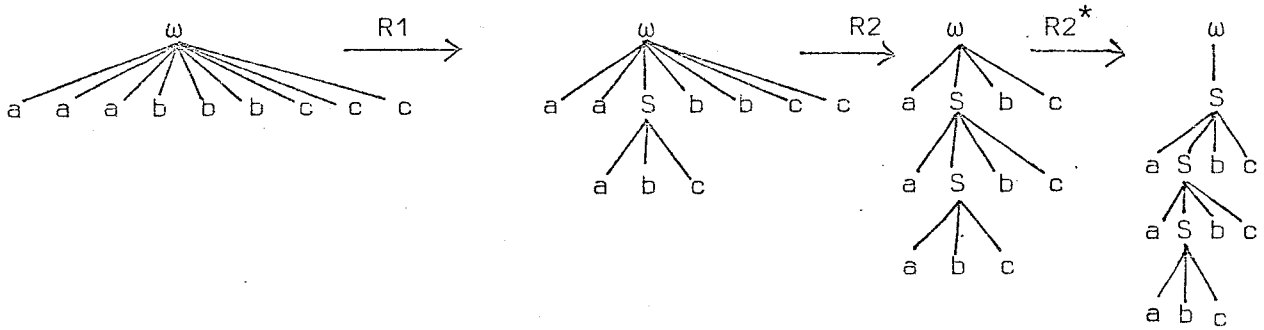
mais appliquée à l'arborescence ω (a,a,a,b,b,b,c,c,c) , elle donne l'arborescence ω (a,S(a,S(a,b,c),b,c),b,c).



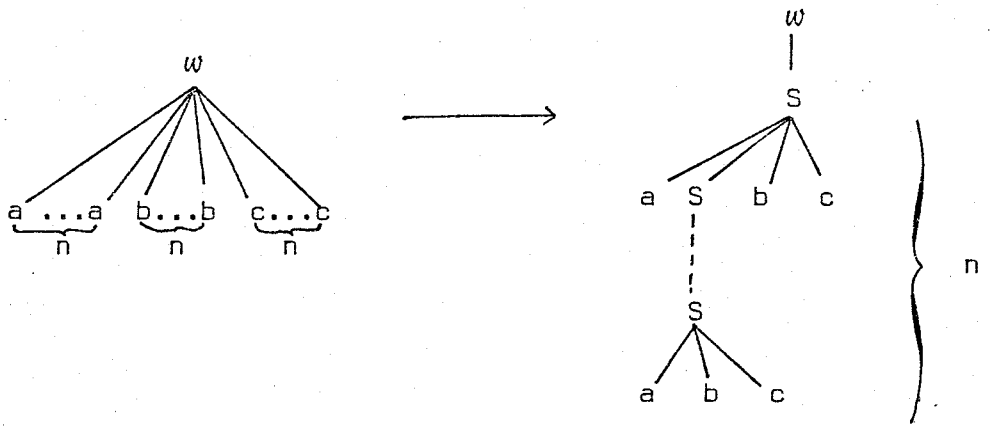
Alors que si la règle R2 est une règle récursive dont les trois éléments sont définis par :

G1 : grammaire de récursion
R2 : seule règle de G1 quiparticipe à la récursion
(S) : l'arborescence de récursion

Cette grammaire appliquée à l'arborescence ω(a,a,a,b,b,b,c,c,c) donne l'arborescence ω(S(a,S(a,S(a,b,c),b,c),b,c))



Cette grammaire ainsi définie fournit le schéma $w(S(W))$ si et seulement si l'arborescence de départ est du type $w(a^n, b^n, c^n)$.



b) Enchaînement de grammaires

Lorsqu'une grammaire a été appliquée à une arborescence donnée, deux cas peuvent se produire :

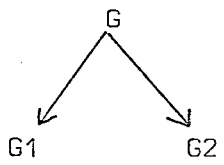
- le résultat de cette grammaire est le résultat général
- le résultat de cette grammaire est transmis (composition linéaire) à une nouvelle grammaire.

Cette dépendance d'une nouvelle grammaire s'effectue conditionnellement à la présence d'un schéma. La fin d'une grammaire est donc constituée par une suite ordonnée de schéma. Le schéma vérifié de plus petit indice détermine le résultat de la grammaire :

- soit ce schéma vérifié n'existe pas et la grammaire présente ne fournit aucun résultat,
- soit ce schéma existe et aucune grammaire n'est dépendante et le résultat de la grammaire est donné par l'arborescence transformée,
- soit ce schéma existe et indique une grammaire dépendante G. Le résultat de la grammaire générale sera alors le résultat de la grammaire G appliquée à l'arborescence transformée n. Cette grammaire G avec l'arborescence transformée donnent un résultat. Dans le cas contraire le résultat est identique au cas où ce schéma ne serait pas vérifié.

Exemple :

Avec la grammaire G1 donnée précédemment nous pouvons obtenir une grammaire conditionnelle qui accepte le seul langage $L = \{a^n b^n c^n \mid n \geq 1\}$. Il suffit de mettre un seul schéma conditionnel dépendant de cette grammaire, ce schéma étant *S1(*, S2, *) / S1 : ETQ -E- ω , S2 : ETQ -E- S /.* Alors G1 donne un résultat que si l'arborescence traitée est de la forme ω (W), W∈ L. Dans le cas où l'analyse des autres cas de mots d'entrée devrait être effectuée par une grammaire G2, il suffirait de construire l'ensemble suivant :



avec G : grammaire qui ne contient qu'une seule règle réalisant l'application identique.

Les conditions de dépendance de G1 et G2 sont toujours vérifiés alors la grammaire G appliquée à une arborescence de la forme ω(W) donne le résultat de l'analyse de G1 si W ∈ L et donne le résultat de l'analyse de G2 si W ∉ L.

c) Grammaire de condition inter-sommet

Une condition inter-sommet peut porter sur des comparaisons dynamiques de structures. On réfère à l'arborescence dépendante d'un point du schéma en plaçant ce point entre deux parenthèses précédées du symbole "*". Cette arborescence peut être modifiée avant comparaison par une grammaire du système.(appel de grammaires de condition). Les comparaisons possibles sont du type ensembliste.

Exemple :

*S1(S2, S3) // * (S2) -E- * (S3)*

Le schéma sera reconnu si l'arborescence dépendante de S2 est identique à l'arborescence dépendante de S3.

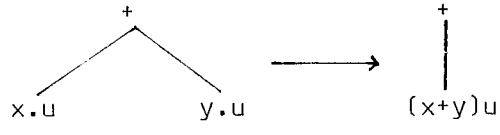
*S1A(S2, S3) // * (S2) -DANS- * GRAM(S3)*

Le schéma sera reconnu si l'arborescence dépendante de S2 est une sous-arborescence de l'arborescence résultante de l'application de la grammaire GRAM à l'arborescence dépendante de S3.

Exemple :

*R1 : PL(U1, U2) / PL : UL -E- '+' / * (U1) -E- (U2) -ET-*
PUIS(U1) -E- PUIS(U2) -ET- UL(U1) -E- UL(U2) ==
*PL(U1) / * <-- U2 / U1 : U1, COEF(U1) := COEF(U1) + COEF(U2).*

Cette règle permet un début de simplification de formule algébrique ; elle est schématisée par :



Ici, u ne représente pas une feuille, mais une arborescence complète, c'est-à-dire, une expression quelconque.

R2 : PR(COS) / PR : UL -E- '.', COS : UL -E- 'COS' -ET-
*PUIS -E- 1 / * (PR) -E- * DERIV(CØS) ==*
*SIN / * <-- PR ; SIN <-- COS / SIN : COS, UL := 'SIN'.*

Cette règle permet d'effectuer l'intégration formelle de la forme u'.COS(u) ⟶ SIN(u), u étant une expression quelconque.

B - PRESENTATION DES DIFFERENTS COMPOSANTS

1 - DECLARATIONS D'ETIQUETTES

Les déclarations d'étiquettes ont pour but de définir l'ensemble des vocabulaires sur lesquels les arborescences traitées seront étiquetées. Une étiquette est constituée par un ensemble de variables, chacune de ces variables étant définies dans les déclarations d'étiquettes. On distingue deux types de variables, les variables dites "exclusives" et les variables dites "non exclusives".
Une variable est soit une variable élémentaire, soit un ensemble de variables de même type. Une variables élémentaire exclusive ne peut prendre pour valeur qu'une seule de ses valeurs déclarées. Une variable exclusive de type particulier est donnée par les variables arithmétiques. Une variable arithmétique est une variable prenant des valeurs entières entre 0 et une borne maximum déclarée. Une variable élémentaire non exclusive peut prendre pour valeur un sous-ensemble de ses valeurs déclarées. Un élément de déclaration de variables est composé d'un nom de variable et de la liste de ces variables associées qui peuvent être soit des valeurs de varaibles, soit des variables dépendantes. Dans le cas d'une variable arithmétique, l'ensemble des valeurs se réduit à la valeur maximum prise pour cette variable.
Deux variables générales n'ont pas à être déclarées :

- La variable ayant pour valeur l'ensemble des valeurs des variables exclusives : notées VAREX.
- La variable ayant pour valeur l'ensemble des variables non exclusives : notées VARNX.

On réfère à la valeur non affectée de la variable en concaténant "0" à droite de cette variable. Dans le cas d'une variable arithmétique, la valeur non affectée est identique à la valeur nulle.

Exemple :

```
FILEMEX    VARBT
                    -EXC-
              CAT := (VRB, NMC, PRP).
              PØID := (PØICR(50), PØIDR(100)).
                    -NEX-
              ETA := (GNR(MAS, FEM, NEU), NBR(SING, PLU)).
```

2 - FORMATS D'ETIQUETTES

Un format d'étiquettes est la description d'un ensemble de variables affectées. Une variable non décrite dans le format est non affectée. Les formats

d'étiquettes permettent de définir les constantes du système. Chaque format d'étiquette est caractérisé par un nom. La description d'une variable affectée peut prendre deux aspects suivant le type de cette variable. Si la variable est exclusive, alors l'affectation est une égalité à une valeur élémentaire :

VAREX -E- VALEURI.

Si la variable est non exclusive, alors l'affectation est une égalité à une union de valeur élémentaire :

VARNEX -E- VALEURI -U- VALEURJ

Les opérateurs sont : -E- Egalité (affectation)
-U- Union

Un format d'étiquettes se présente donc par un nom de format. Un numéro de carte présent à une colonne constante permet d'ordonner la description de ce format lorsque celle-ci occupe plusieurs cartes. La description du format suit le marquant "==" et est une suite d'affectations séparées par des symboles ",".

Exemple :

```
FORMAT1  01 == CAT -E- NMC, ETA -E- MAS -U- PLUR.
FORMAT2  02 == CAT -E- PRP,
FORMAT2  02 == GNR -E- MAS, NBR -E- PLUR.
FORMAT3  01 == NBRA -E- 4.
```

3 - LE SYSTEME

La description d'un système est divisée en quatre parties distinctes. Chacune de ces parties débute par un mot clé.
Nous trouvons dans l'ordre :

- les déclarations de procédure (*-PROC-*)
- la déclaration de schémas (*-PSCHEM-*)
- la grammaire (*-GRAM-*)
- l'ensemble des règles (*-REGLES-*)

a) Déclarations de procédures

Une procédure a pour but de définir préalablement une suite d'opérations et de référencer cet ensemble par un nom. La référence définira alors par la suite l'ensemble de ces opérations. On distingue deux types de procédure, chacun de ces types comportant eux-mêmes deux genres de procédure.
Chaque procédure est précédée d'un identificateur de procédure suivi du symbole ":". Les identificateurs ont la signification suivante :

PCP : procédure de condition propre

PCIS : procédure de condition inter-sommet

PAF : procédure d'affectation

Le nom de la procédure suit la partie identification de la procédure.

Le premier type est formé par les procédures de conditions. Le premier genre de procédure de conditions est formé par les procédures de conditions propres. Ce genre est une expression booléenne sur les valeurs possibles des étiquettes sans référence de sommet.

Exemple :

GNR -E- MAS

Ce genre de procédure n'a pas de paramètres.

Suivant le type de variales auxquels ils réfèrent, les relateurs possibles sont distincts. Dans le cas des variables exclusives, les seuls relateurs possibles sont les suivants :

-E-	A -E- B	$\Longleftrightarrow A=B$
-NE-	A -NE- B	$\Longleftrightarrow A\neq B$
-INF-	A -INF- B	$\Longleftrightarrow A<B$
-NINF-	A -NINF- B	$\Longleftrightarrow A\not<B$
-SUP-	A -SUP- B	$\Longleftrightarrow A>B$
-NSUP-	A -NSUP- B	$\Longleftrightarrow A\not>B$
-INFE-	A -INFE- B	$\Longleftrightarrow A\leqslant B$
-NINFE-	A -NINFE- B	$\Longleftrightarrow A\not\leqslant B$
-SUPE-	A -SUPE- B	$\Longleftrightarrow A>B$
-NSUPE-	A -NSUPE- B	$\Longleftrightarrow A\not>B$

Dans le cas des variables non exclusives, les relateurs possibles sont :

-E-	*égalité*	A -E- B	$\Longleftrightarrow A=B$
-NE-	*différence*	A -NE- B	$\Longleftrightarrow \mathring{A}\neq B$
-DANS-	*dans*	A -DANS- B	$\Longleftrightarrow A\subset B$
-INC-	*contient*	A -INC- B	$\Longleftrightarrow A\supset B$
-NDANS-	*n'est pas dans*	A -NDANS- B	$\Longleftrightarrow A\not\subset B$
-NINC-	*ne contient pas*	A -INC- B	$\Longleftrightarrow A\not\supset B$

Les relateurs permettent de définir une valeur booléenne lorsqu'ils sont appliqués à une étiquette. On réfère à la valeur d'une variable de cette étiquette pour le nom de cette variable, ce sont des constantes de l'expression. Une contre façon de référer à des constantes est de préciser le nom d'une variable suivi du nom de format d'étiquette entre parenthèses et précédé du symbole "*".

Exemple :

*CAT(*FORMAT1)*

On réfère à la valeur non affectée par le nom de variable suivi du caractère "0". Suivant le type de variables, les éléments gauche et droit de la relation sont différents. Dans le cas des variables exclusives, chaque élément est équivalent à une valeur de la variable concernée. Les opérateurs possibles pour les variables arithmétiques sont :

+	somme
-	différence
*	produit
/	quotient

Dans le cas des variables non exclusives, chaque élément gauche et droit est équivalent à une partie de l'ensemble des valeurs de la variable concernée. Pour former cette partie, les opérateurs possibles sont les suivants :

-U-	union	A -U- B $\Leftrightarrow$ A $\cup$ B
-I-	intersection	A -I- B $\Leftrightarrow$ A $\cap$ B
-N-	complément	-N- A $\Leftrightarrow$ $\complement$A

Pour former l'expression booléenne, les connecteurs possibles sont :

-OU-	A -OU- B	$\Longleftrightarrow$ A $\vee$ B
-ET-	A -ET- B	$\Longleftrightarrow$ A $\wedge$ B
-N-	-N- A	$\Longleftrightarrow$ $\neg$A

L'expression peut également être parenthésée.

Exemple :

```
PCP :  EXP == (GNR -E- MAS -ET- NBR -NINC- (NBR -I- SING))
                                  -OU- GNR -NE- GNR (* FORMAT1).

PCP :  EXP1 == (NBRA/2) * 2 -E- NBRA -ET- GNR -E-
                                  (GNR -I- (MAS -U-FEM)).
```

Le deuxième genre de procédures de conditions est formé par les procédures de conditions inter-sommets. Ce genre de procédures a des paramètres. Chaque paramètre représente une étiquette potentielle. La procédure est une expression booléenne de conditions élémentaires. Les connecteurs possibles de cette expression booléenne sont identiques à ceux utilisables dans les procédures de conditions propres. Une condition inter-sommets élémentaire peut être du même type qu'une condition propre. Dans ce cas, chaque variable est suivie d'un code origine placé entre deux parenthèses qui précise le sommet du schéma où doit être recherché la valeur concernée de cette variable. Une condition de structure est également une condition inter-sommets élémentaire. Une condition de structure est une comparaison de deux structures de l'arborescence d'entrée. Les relateurs possibles sont :

-E-	égalité	A -E- B	l'arborescence A est identique à l'arborescence B.
-NE-	différence	A -NE- B	l'arborescence A est différente de l'arborescence B.
-DANS-	dans	A -DANS- B	l'arborescence A est une sous-arborescence de l'arborescence B.
-INC-	contient	A -INC- B	l'arborescence B est une sous-arborescence de l'arborescence B.
-NDANS-	n'est pas dans	A -NDANS-B	l'arborescence A n'est pas une sous-arborescence de l'arborescence A.
-NINC-	ne contient pas	A -NINC-B	l'arborescence B n'est pas une sous-arborescence de l'arborescence A.

On réfère à une sous-arborescence de l'arborescence d'entrée par le symbole "*" . Le code origine (point du schéma racine de l'arborescence référencée) suit le symbole "*" et est placé entre deux parenthèses.

Lorsque la structure dépendante d'un point n'est pas une arborescence, ce point appartient alors à la structure qui, par conséquent, devient une arborescence. Dans chaque structure référencée, un point du schéma et ses descendants ne peuvent appartenir à une structure dépendante d'un autre point du schéma. La structure dépendante d'un point contient donc tous les points de l'arborescence d'entrée qui dépendent de ce point sans dépendre d'un autre point du schéma dépendant de ce point de référence. Les comparaisons possibles s'effectuent entre deux arborescences dépendantes des points du schéma, modifiées éventuellement par une grammaire du système. Dans ce cas le nom de grammaire est placé entre les symboles "*" et "(". La grammaire appliquée dans ce cas comprend la grammaire référencée et les grammaires dépendantes.

Exemple :

```
PCIS : EXP(S1,S2) == (GNR(S1) -E- GNR (* FORMAT2) -ET- NBR(S1) -
       -NDANS- (NBR(S2) -I- SING)) -OU- GNR(S2) -NE- MAS.

PCIS : EXP(S1,S2,S3) == NBR(S1) -E- NBR(S2) -ET-  *(S1) -DANS-
         *(S3) -ET-  *(S1) -E- * GRAM(S2).
```

Le deuxième type de procédures est constitué par les procédures d'affectation. On distingue des procédures d'affectation propres et des procédures d'affectation inter-sommets. Chaque procédure est constituée par une liste d'affectation. Une affectation est caractérisée par un nom de variable, définissant la variable à affecter, et d'une valeur de cette variable. La valeur de cette variable répond aux mêmes critères que dans le cas des procédures de conditions.

Exemple :

```
PAF : EXAF == GNR := MAS, CAT := CAT(* FORMAT2), NBR := NBR -U-
      PLU, NBRA := 50 + NBRA/2.

PAF : EXAFI(S1,S2) == GNR := MAS, NBR := NBR(S1), CAT :=
      CAT( * FORMAT1) -U- CAT(S2), NBRA := NBRA(S1) * NBRA(S2).
```

b) Déclaration de schémas

Un schéma est défini par un ensemble d'arborescences conditionnelles. La définition d'un schéma dans la partie déclaration de schéma permet de simplifier l'écriture ultérieure de ce schéma. On réfère à un schéma déclaré par son nom associé éventuellement à des procédures paramètres. Un schéma conditionnel comprend trois éléments :

- Le premier élément permet de définir la structure du schéma. Il est constitué par un ensemble d'arborescences dépendantes d'une arborescence principale. Le point de dépendance d'une arborescence à une autre arborescence est unique. L'arborescence s'écrit comme une expression parenthésée. Chaque parenthèse ouvrante "(" est étiquetée par un nom de sommet. Celui-ci sert à repérer ce point lors de la vérification des conditions. A côté de ce nom de sommet les éléments suivants peuvent être présents :

-%X précise le choix du schéma concerné lorsque plusieurs choix sont possibles dans l'arborescence d'entrée.
Les deux choix possibles sont :

X = H : le schéma considéré est celui qui a la racine la plus proche de la racine de l'arborescence d'entrée.

X = B : le schéma considéré est celui qui a la racine la plus éloignée de la racine de l'arborescence d'entrée.

-&YZ précise l'ordre et le choix des descendants de ce point.
L'élément Y peut prendre les valeurs suivantes :

Y = O : l'ensemble des descendants de ce point seront ordonnés

Y = B : l'ensemble des groupes de descendants de ce point sont ordonnés

Y = H : l'ensemble des descendants de ce point sont ordonnés à l'intérieur de chacun des groupes.

L'ensemble des descendants d'un point sont des éléments séparés par les symboles "," et ";". Le symbole "," sépare les éléments d'un même groupe et le symbole ";" les différents groupes.

L'élément Z peut être omis ou prendre les valeurs suivantes:

Z = G : l'ensemble des points choisis comme descendants de ce point seront groupés vers la gauche

Z = D : l'ensemble des points choisis comme descendants de ce point seront groupés vers la droite.

Lorsque les deux éléments ØX et &YZ sont présents à droite d'un nom de sommet, l'ordre n'est pas indifférent ; le premier élément rencontré doit être ØX.

Après le nom de sommet et les éléments déterminants, le symbole "?" peut préciser que les descendants ne sont pas des descendants directs mais des "dépendants généralisés". Un dépendant généralisé est une arborescence dépendante de ce point pouvant être présente à n'importe quelle profondeur.

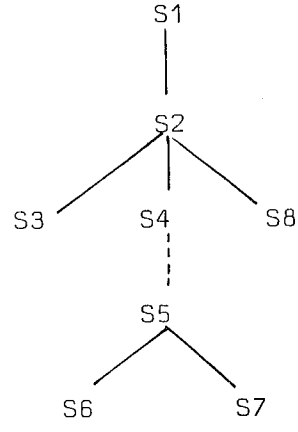
Parmi l'ensemble des descendants d'un point les symboles suivants peuvent être présents :

"*" : "anti-sommet" précise que le point doit être une feuille si ce symbole est seul. Sinon précise qu'il ne doit y avoir aucun point entre les éléments à gauche et à droite de ce symbole (éléments jointifs). La présence du symbole "*" impose l'ordre entre les deux éléments jointifs.

"ØM1" : où M1 est un nom quelconque. Cet élément précise une liste éventuellement vide. Cette liste est surtout utile dans le cas de la définition d'une transformation.

Exemple de schéma :

*S1(*,S2&H(S3;S4?(S5ØH&B(S6,*,S7)),S8(*)))*



Les différents éléments précisés dans ce schéma sont les suivants :

- le point S2 doit être le premier descendant à gauche de S1
- le point S3 doit précédé les points S4 et S8 qui eux sont dans un ordre quelconque
- le point S5 est la racine d'une arborescence dépendante de S4.

Lorsque plusieurs choix seront possibles, l'arborescence la plus près du point S4 sera prise en compte.

- les points S6 et S7 sont ordonnés et jointifs.
- le point S8 doit être une feuille.

- Le deuxième et le troisième éléments d'un schéma sont composés de conditions booléennes sur les étiquettes. Chaque partie du schéma conditionnel est séparée par le symbole "/". La deuxième partie est un ensemble de conditions booléennes propres portant sur un point. La troisième partie est une expression booléenne inter-sommet.

A la place de ces expressions, on peut faire référence à des procédures (appel de procédures). Le type de la procédure doit correspondre au type de la condition : procédure propre pour une condition propre et procédure inter-sommet pour une condition inter-sommet.

Chaque condition propre est affectée d'un nom de sommet précisant le point du schéma qui doit vérifier cette condition. Une condition est séparée d'une autre par le symbole ",". La condition inter-sommet est une condition booléenne identique à celle définie pour les procédures de condition inter-sommets. Dans le cas d'un appel de procédure inter-sommets, les paramètres d'appels de cette procédure sont les noms de sommets du schéma.

Exemple de schéma conditionnel :

SCHEM1 : S1(S2?(S3(S4,S5)))/S1:GNR -E- MAS -ET- NBR -E- PLU,

S4 : CAT -E- NMC / GNR(S3) -I- GNR(S5) -NE- GNRO -OU- CAT(S3) -E- NMC.

SCHEM 2 : S1(S2(S3,S4),S5) / S1 : $PROC1,S2 : NBR -E- SING/ $PROC(S2,S4,S5).

Où *PROC(S2,S4,S5)* est une procédure de condition inter-sommet ayant 3 paramètres et *PROC1* est une procédure de condition propre.

Lorsque le schéma comprend des paramètres, ces paramètres devront être à l'appel du schéma des noms de procédures. L'appel d'une procédure paramètre s'effectue en préfixant le nom du paramètre par le symbole "&".

Exemple :

SCHEM3(P1,P2) : S1(S2,S3(S4)) / S1 : & P1,S2:$PROC1 / & P2(S3,S4,S2)

P1 et *P2* sont des procédures paramètres, *PROC1* est une procédure de condition propre. A l'appel du schéma, *P1* devra être une procédure de condition propre et *P2* une procédure de condition inter-sommet ayant trois paramètres qui seront actualisés par S3, S4 et S2.

c) Grammaire

La grammaire est constituée par une composition de grammaires élémentaires. Chaque grammaire élémentaire est caractérisée par un nom de grammaire, une liste de nom de règles et une suite conditionnelle de grammaires dépendantes. Le nom de grammaire peut être suivi du mode d'application. Ce mode peut être soit U (mode unitaire), soit E (mode exhaustif). En l'absence d'élément, le mode U est pris par défaut. Un nom de règle peut être suivi éventuellement par des éléments de récursion. La présence des éléments de récursion indique que la règle est récursive. Les éléments de récursion sont composés d'un nom de grammaire, d'une liste de noms de règles de cette grammaire et d'une arborescence de récursion.

Une grammaire dépendante est caractérisée par un schéma ou un appel de schéma suivi d'un nom de grammaire. Les différentes grammaires dépendantes sont évaluées de la gauche vers la droite jusqu'à ce qu'une de ces grammaires produise un résultat. Une grammaire dépendante est appliquée si son schéma associé est vérifié. La présence du symbole "&NUL" à la place d'un nom de grammaire précise que la grammaire dépendante est vide. Si le schéma associé est alors vérifié, le résultat de la grammaire générale est donné par le résultat de cette grammaire élémentaire.

Exemple de grammaire :

```
GRAM(U) : R1, R2, R3, $SCHEM1 <-- GRAM1 ;
          $SCHEM2(P1, P2) <-- GRAM3 ; ω(S1) // <-- GRAM2.
          GRAM1 : RT1, RT2 ; ω // <-- &NUL.
          GRAM2 : RT1.
          GRAM3 : RG1, RG3, RG2 ; $SCHEM2 <-- &NUL ; $SCHEM1 <-- GRAM2.
```

d) Règles récursives

L'emploi de règles récursives permet d'appliquer une transformation de façon systématique sur l'arborescence d'entrée. Le résultat de l'application d'une rèlge récursive sera déterminé par l'application de la grammaire mentionnée sur la sous-arborescence ayant pour racine la racine de l'arborescence de récursion. Seules les règles énumérées composent la grammaire de récursion.
Dans le cas où l'énumération des règles de la grammaire mentionnée est remplacée par le symbole "*", toutes les règles de la grammaire mentionnées participent à la récursion, ainsi que toutes les grammaires dépendantes de celle-ci. La condition de décidabilité est la suivante :

L'arborescence de récursion doit avoir un nombre de points inférieurs au nombre de points du schéma présent en partie gauche de la règle récursive, ou, la récursion ne doit pas faire intervenir la grammaire en cours d'application (qui contient donc cette règle récursive).

Les points qui peuvent participer à la récursion sont définis par l'ensemble des points dépendant des points de l'arborescence de récursion.
Un point pouvant participer à la récursion est effectivement concerné si, il ne dépend pas d'un autre point pouvant participer à la récursion, appartenant à l'arborescence résultante de la règle, et non mentionnée dans l'arborescence de récursion.

Exemple de grammaire contenant des règles récursives :

```
GRAM(E) : R1, R2(GRAM3 / RG1, RG3 / ω(S1)); R3 ;
          $SCHEM1 <-- GRAM3 ; ω(S1) // <-- GRAM2.
          GRAM1 : RT1(GRAM, *, ω) ; ω // <-- &NUL.
          GRAM2 : RT1, RT2, RT3 ; // <-- GRAM1.
          GRAM3 : RG1, RG3(GRAM3, RG3, ω(S1)), RG2 ;
          $SCHEM2 <-- &NUL ; $SCHEM1 <-- GRAM2.
```

e) Règles

L'ensemble des règles de transformation sont placées dans une partie unique. Chaque règle comprend quatre éléments distincts :

- La partie gauche de règle est un schéma conditionnel identique aux schémas de condition.
- La partie droite est une description de la transformation à opérer sur ce schéma. Elle se compose des trois dernières parties de la règle, décrites dans l'ordre suivant :

  . l'arborescence résultante est écrite de la même façon qu'un schéma conditionnel. La présence des éléments conditionnels à l'intérieur du schéma ne peuvent être présents. Ces éléments sont :
  - l'anti sommet "*".
  - les indications d'ordre et de choix des points &X et &YZ.

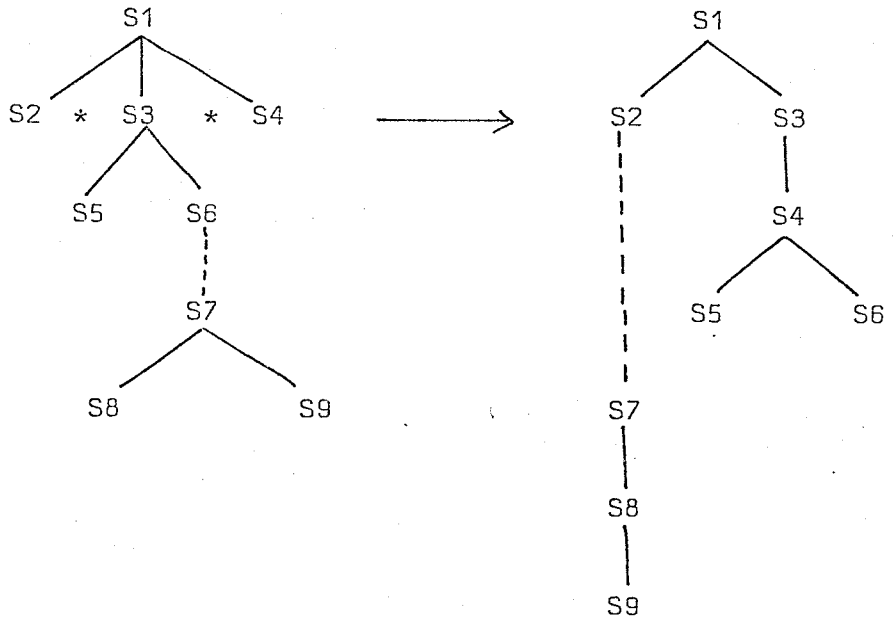
  . La fonction de transfert est décrite pour chaque point image et source Chaque élément de la fonction est donné par le point image et la liste des points sources. Dans le cas où un point n'a pas d'élément source correspondant, la fonction de transfert décrite doit lui affecter l'élément "*" comme source.
  Dans le cas où un point n'a pas d'image, la fonction de transfert doit lui affecter l'élément "*" comme image.
  Enfin lorsqu'un nom de sommet apparaît dans le schéma et dans l'arborescence résultante, il définit par défaut la fonction de transfert pour lui-même. Cette définition par défaut est annulée dès que ce nom apparaît dans une partie quelconque de la description de la fonction de transfert.

  . L'affectation des différentes étiquettes termine la description d'une règle. Chaque affectation comprend deux éléments qui sont l'origine de l'étiquette affectée à ce point et la description d'une modificatio de cette étiquette. Cette origine peut soit être un point du schéma, soit un nom de format d'étiquette. Dans le premier cas l'étiquette affectée au nouveau point est celle présente sur le point concerné du schéma. Dans le deuxième cas une nouvelle étiquette est générée et prend comme valeur la valeur du format donné.

  La description de la modification est formée d'une suite d'affectation. Chaque affectation peut être soit un appel de procédure d'affectation, soit une affectation écrite explicitement.

Les références associées aux variables dans une expression d'affectation sont des noms de sommets du schéma conditionnel, partie gauche de la règle. Lorsqu'un nom de sommet apparaît dans le schéma et dans l'arborescence résultante, il définit par défaut l'affectation d'étiquette. Cette affectation par défaut est annulée si ce nom de sommet est présent dans la liste des affectations d'étiquettes.

Exemple de règle :



*S1(S2, * , S3(S5,S6 ? (S7.B(S8, S9))), * , S4) // ==*

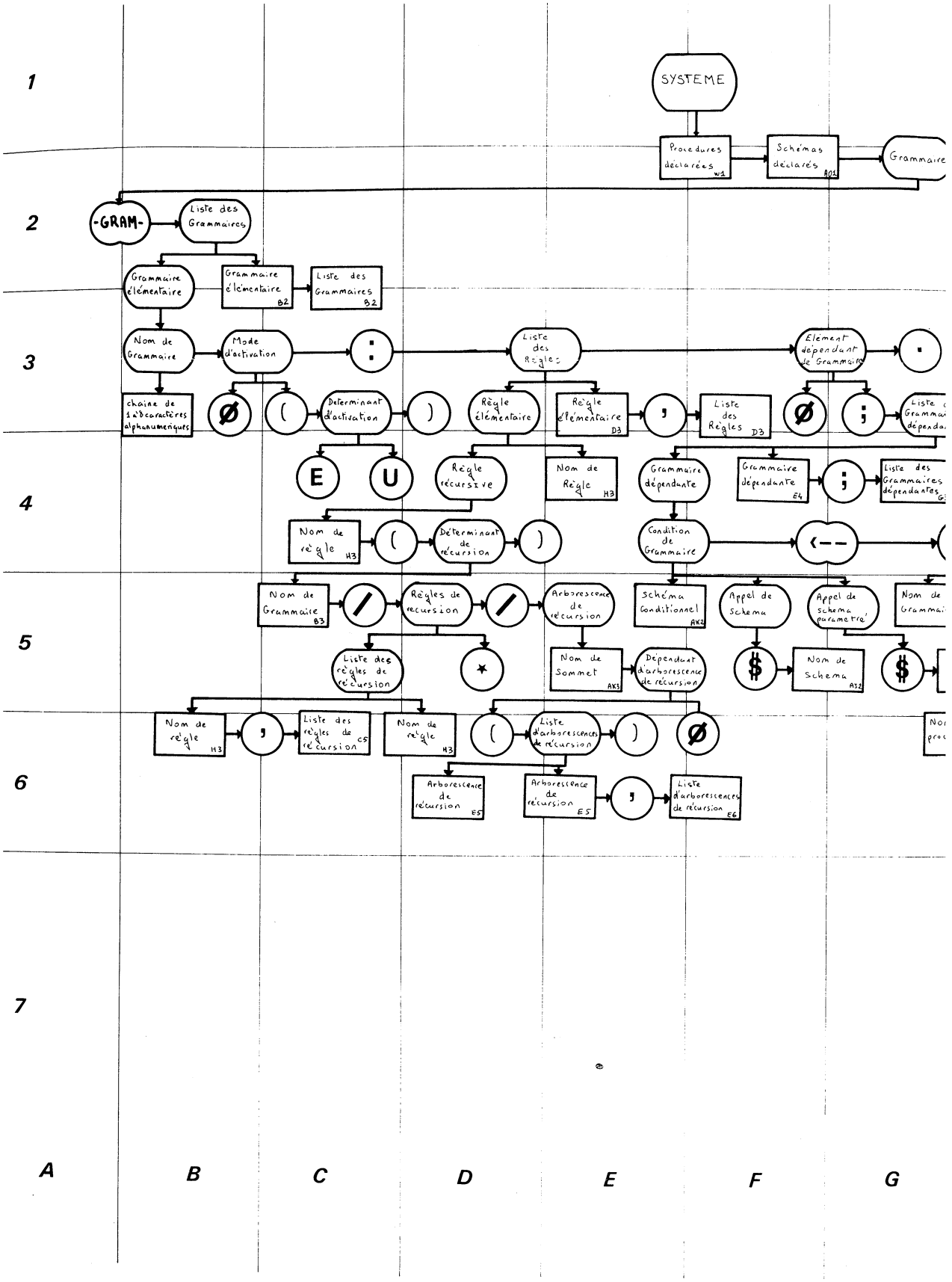
S1(S2(S6 ? (S7(S8, S9))), S3(S4(S5, S6))).

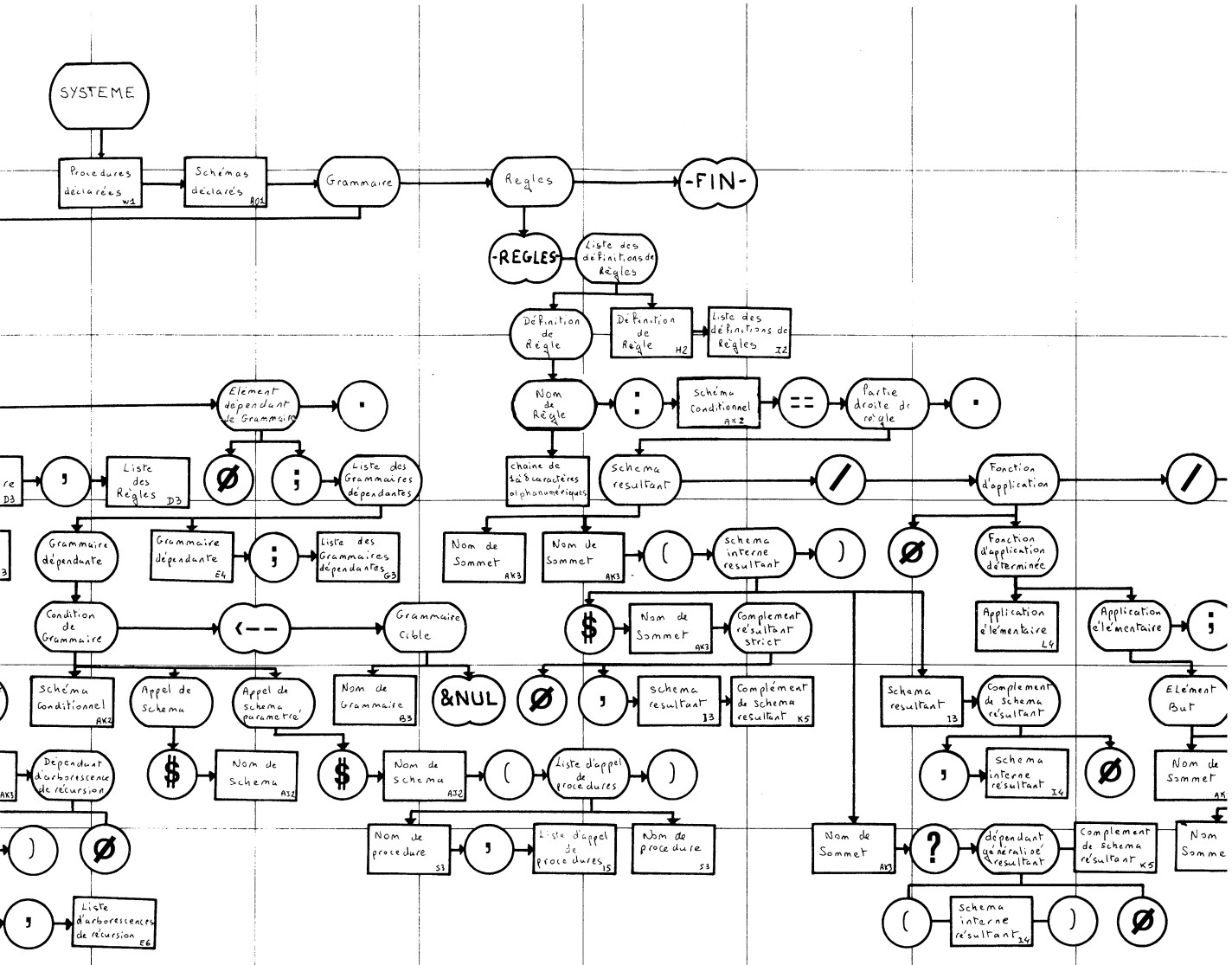
C - TRAITEMENT INFORMATIQUE

Le transducteur à simuler est un transducteur composé à pile. Sa définition est donnée de façon interne sous forme d'un langage. Le système comprend donc en premier lieu un transducteur constructeur. L'application d'une grammaire a pour effet d'activer ce transducteur qui construira le transducteur composé simulant la grammaire d'appel. Le système étant non déterministe, les données initiales seront empilées à chaque appel de grammaire. Lors de l'appel d'une grammaire et de la construction des transducteurs correspondant aux transformations employées, le système efface éventuellement les constructions antérieures. Ce cas se produit notamment lorsque les zones réservées aux tables de transition des différents transducteurs sont remplies. Le retour à la grammaire appelante nécessite alors un nouvel effacement et une nouvelle construction. De même, lorsqu'une règle de la grammaire appelée a déjà été construite, le système associe alors simplement l'état initial du transducteur correspondant au numéro d'ordre de cette règle. De même que pour le système A.T.E.F., l'évolution des différentes conditions et affectations sont interprétées à partir d'un langage interne. Ce système comprend également deux phases distinctes :

- La première phase analyse la syntaxe des différents composants et traduit ces données dans un langage interne.
- La deuxième phase applique le système sur une arborescence d'entrée.

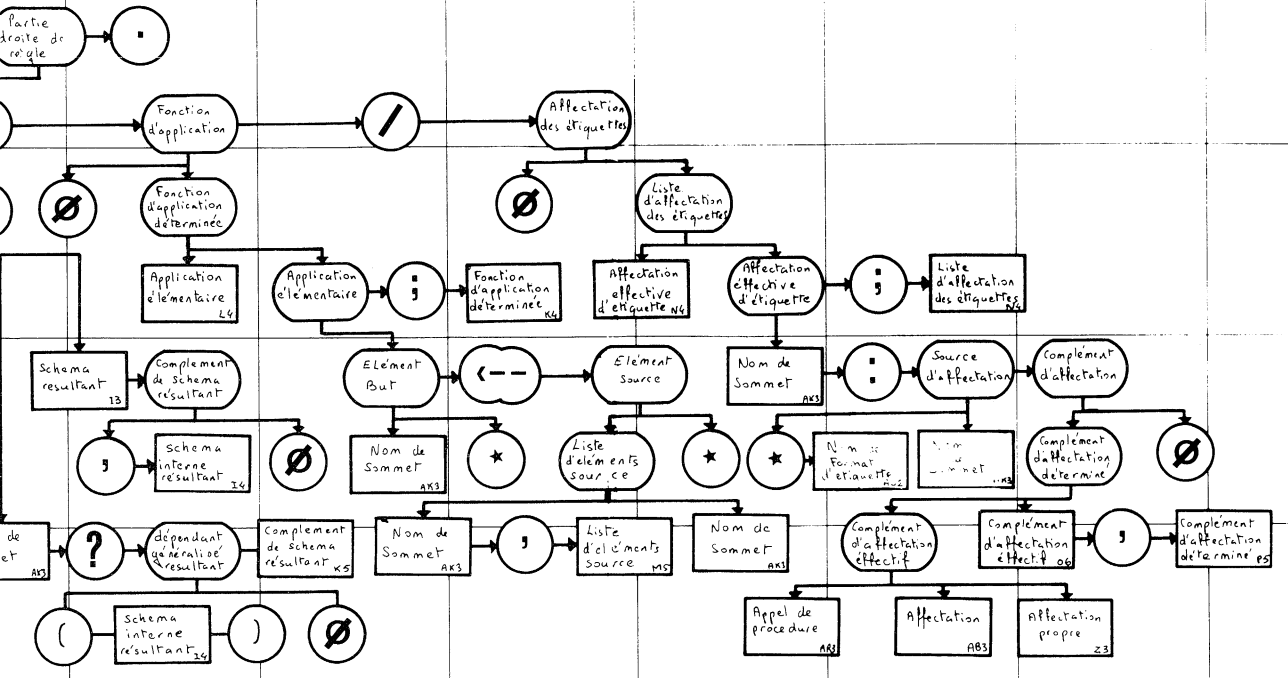
La préparation des données ainsi que leurs modifications se font en mode conversationnel.
L'application du système vers une arborescence donnée est simplement commandée en mode conversationnel.


SYSTEME
Procedures déclarées W1
Schémas déclarés A01
Grammaire
-GRAM-
Liste des Grammaires
Grammaire élémentaire
Grammaire élémentaire B2
Liste des Grammaires B2
Nom de Grammaire
Mode d'activation
:
Liste des Règles
Element dépendant de Grammaire
.
chaine de 1 à 8 caractères alphanumériques
Ø
(
Déterminant d'activation
)
Règle élémentaire
Règle élémentaire D3
,
Liste des Règles D3
Ø
;
E
U
Règle récursive
Nom de Règle H3
Grammaire dépendante
Grammaire dépendante E4
;
Nom de règle H3
(
Déterminant de récursion
)
Condition de Grammaire
Nom de Grammaire B3
/
Règles de recursion
/
Arborescence de récursion
Schéma conditionnel AK2
Appel de schema
Appel de schema parametré
Liste des règles de récursion
*
Nom de Sommet AK3
Dépendant d'arborescence de récursion
$
Nom de Schema A32
$
Nom de règle H3
,
Liste des règles de récursion C5
Nom de règle H3
(
Liste d'arborescences de récursion
)
Ø
Arborescence de récursion E5
Arborescence de récursion E5
,
Liste d'arborescences de récursion E6
1
2
3
4
5
6
7
A
B
C
D
E
F
G

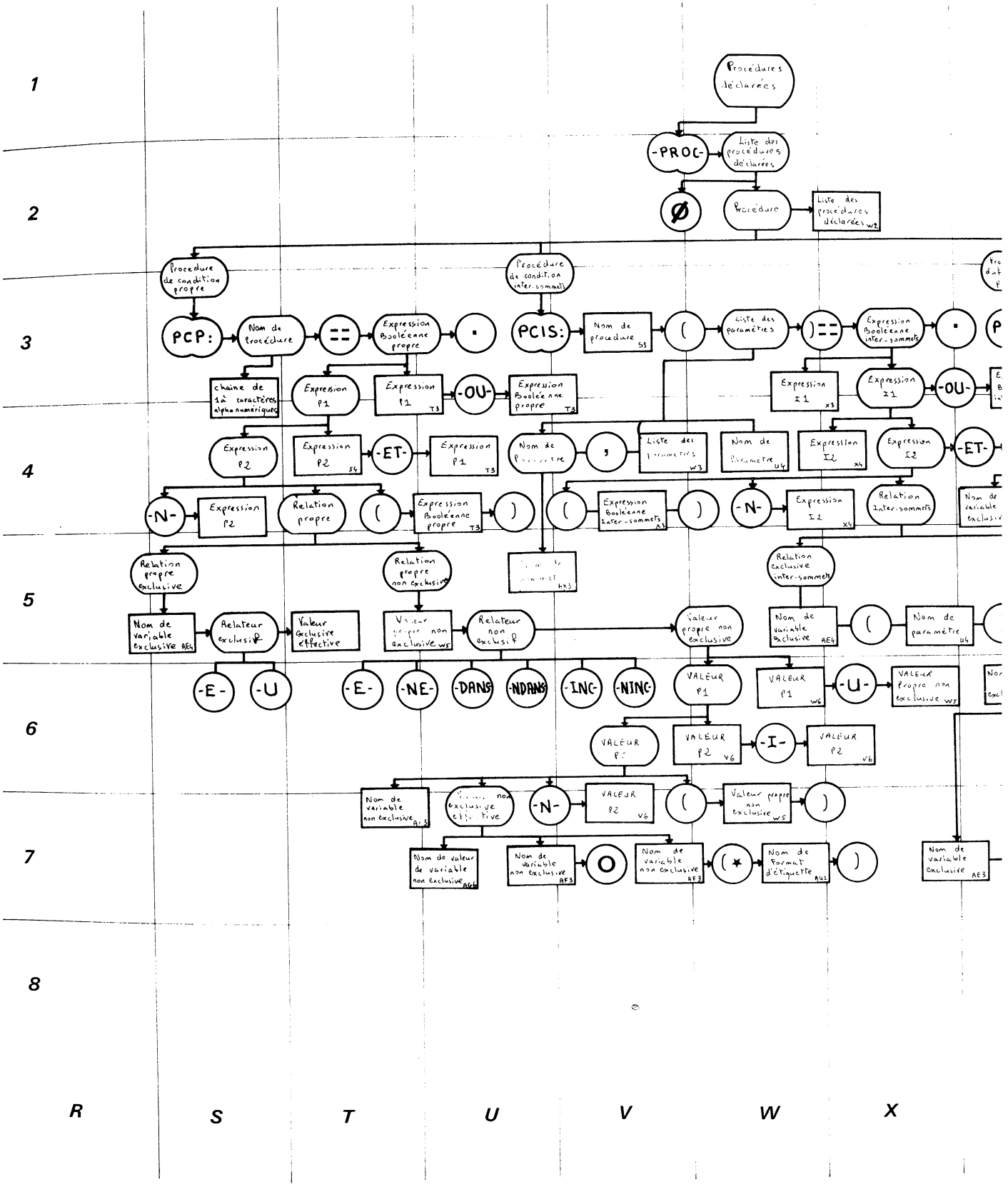


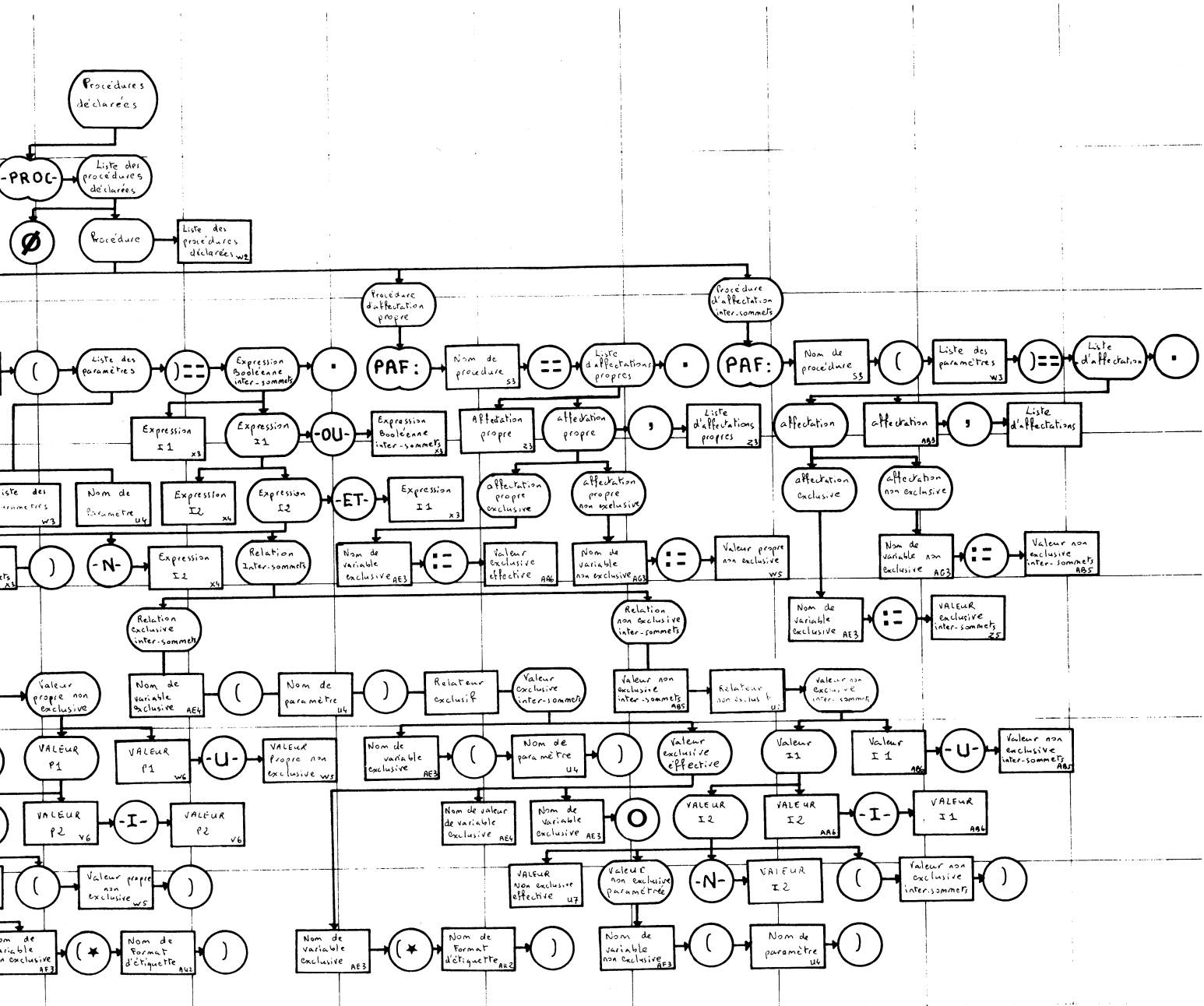
SYSTEME CETA (1)

E F G H I J K L


Partie droite de règle
Fonction d'application
/
Affectation des étiquettes
Fonction d'application déterminée
Application élémentaire L4
Application élémentaire
;
Fonction d'application déterminée K4
Liste d'affectation des étiquettes
Affectation effective d'étiquette N4
Affectation effective d'étiquette
;
Liste d'affectation des étiquettes N4
Schema resultant I3
Complement de schema résultant
Element But
<--
Element Source
Nom de Sommet AK3
:
Source d'affectation
Complément d'affectation
;
Schema interne résultant I4
Nom de Sommet AK3
*
Liste d'éléments source
*
*
Nom Format d'étiquette N2
Complément d'affectation déterminé
?
dépendant généralisé résultant
Complement de schema résultant K5
Nom de Sommet AK3
;
Liste d'éléments source M5
Nom de Sommet AK3
Complément d'affectation effectif
Complément d'affectation effectif O6
;
Complément d'affectation déterminé P5
(
Schema interne résultant I4
)
Appel de procedure AR3
Affectation AB3
Affectation propre Z3

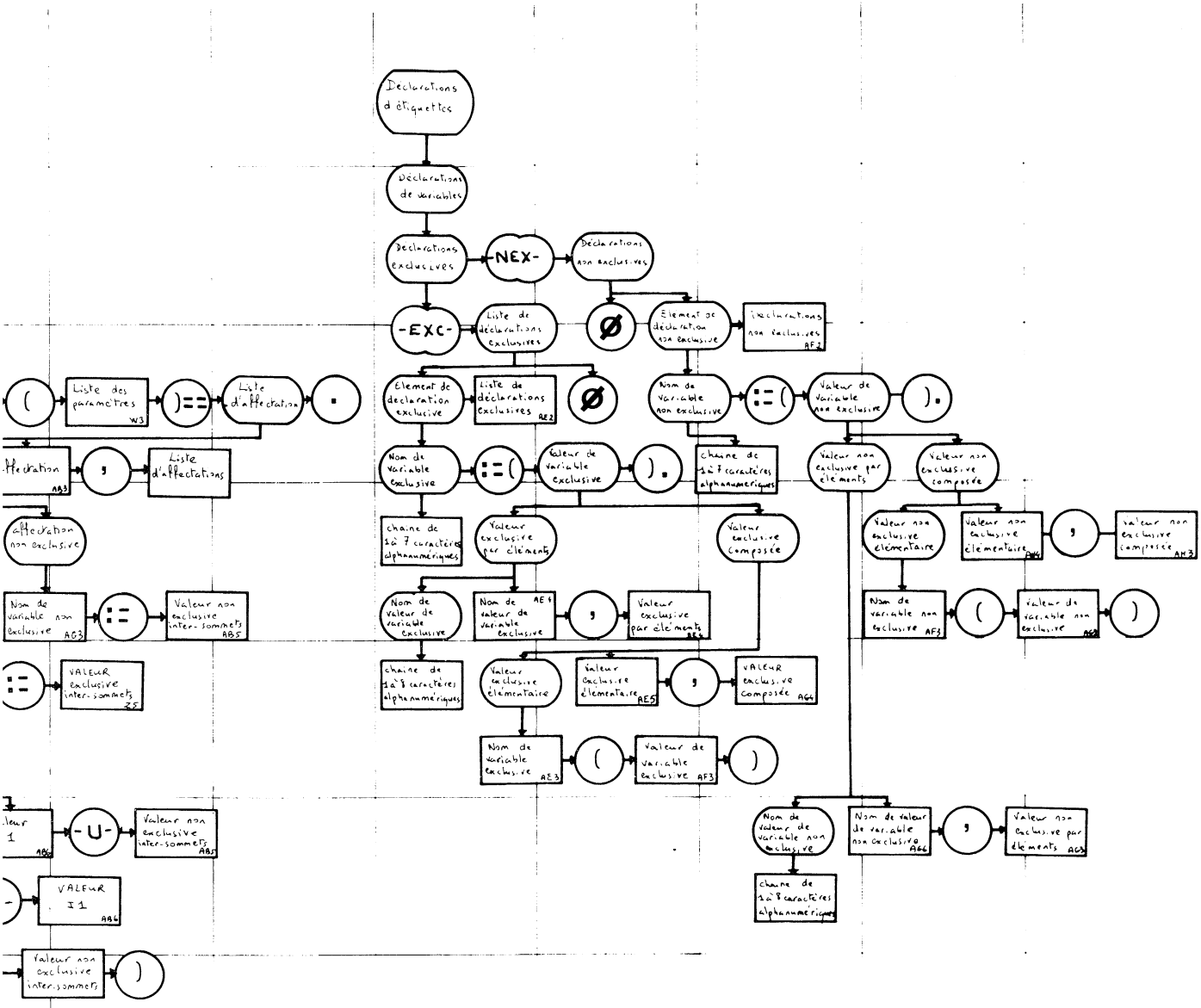
K L M N O P Q


1
2
3
4
5
6
7
8
R
S
T
U
V
W
X
Procédures déclarées
-PROC-
Liste des procédures déclarées
Ø
Procédure
Liste des procédures déclarées W2
Procédure de condition propre
Procédure de condition inter-sommets
PCP:
Nom de Procédure
==
Expression Booléenne propre
.
PCIS:
Nom de procédure S3
(
Liste des paramètres
)==
Expression Booléenne inter-sommets
chaine de 1à caractères alpha numériques
Expression P1
Expression P1 T3
-OU-
Expression Booléenne propre T3
Expression I1 X3
Expression I1
-OU-
Expression P2
Expression P2 S4
-ET-
Expression P1 T3
Nom de Procédure
,
Liste des paramètres W3
Nom de paramètre U4
Expression I2 X4
Expression I2
-ET-
-N-
Expression P2
Relation propre
(
Expression Booléenne propre T3
)
(
Expression Booléenne Inter-sommets A3
)
-N-
Expression I2 X4
Relation Inter-sommets
Relation propre exclusive
Relation propre non exclusive
Relation exclusive inter-sommets
Nom de variable exclusive AE4
Relateur exclusif
Valeur exclusive effective
Valeur propre non exclusive W5
Relateur non exclusif
Valeur propre non exclusive
Nom de variable exclusive AE4
(
Nom de paramètre U4
-E-
-U
-E-
-NE-
-DANS
-NDANS
-INC-
-NINC-
VALEUR P1
VALEUR P1 W6
-U-
VALEUR Propre non exclusive W5
VALEUR P2 V6
-I-
VALEUR P2 V6
Nom de variable non exclusive AF3
-N-
VALEUR P2 V6
(
Valeur propre non exclusive W5
)
Nom de valeur de variable non exclusive AG6
Nom de variable non exclusive AF3
O
Nom de variable non exclusive AF3
(*
Nom de Format d'étiquette AU2
)
Nom de variable exclusive AE3

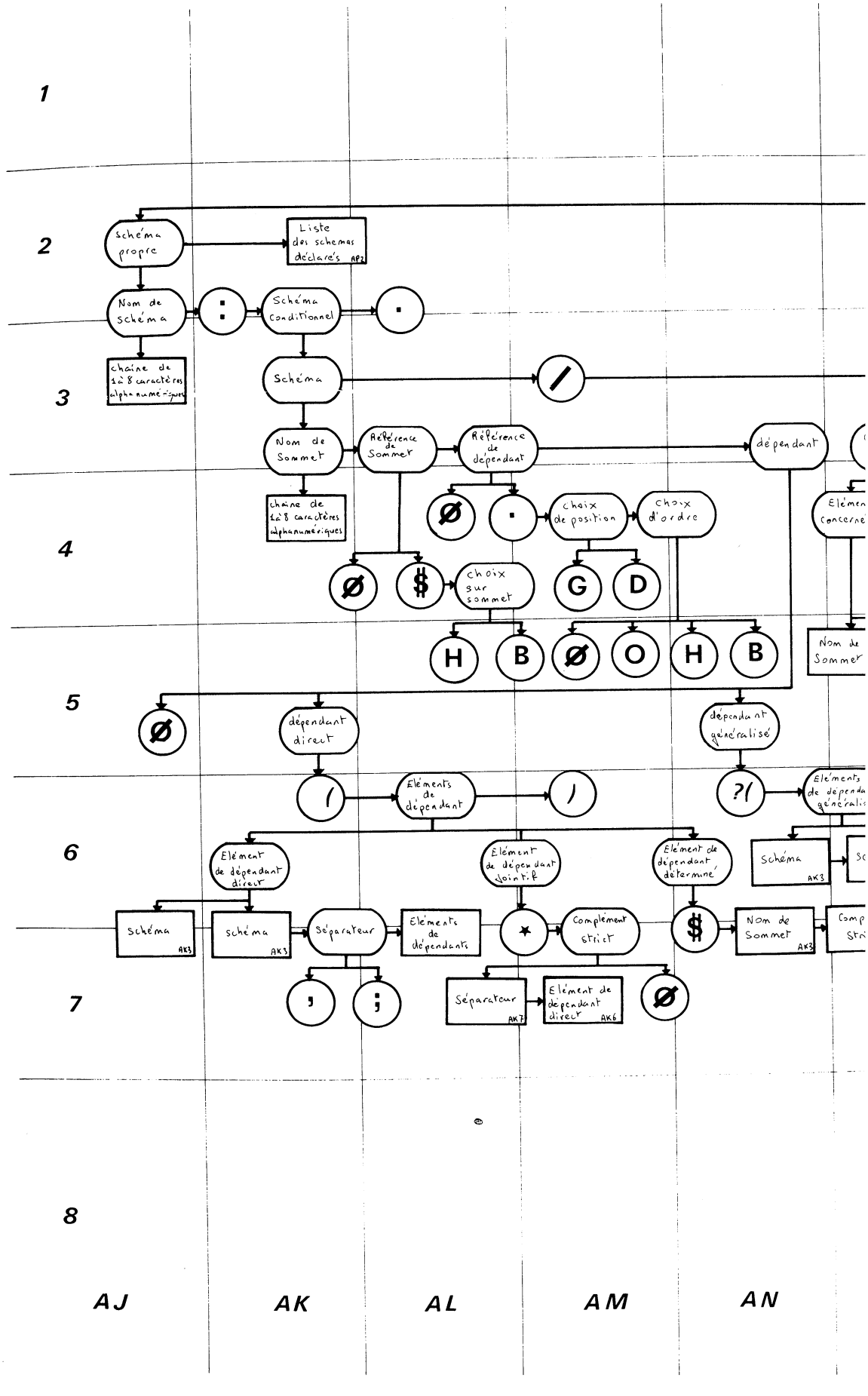


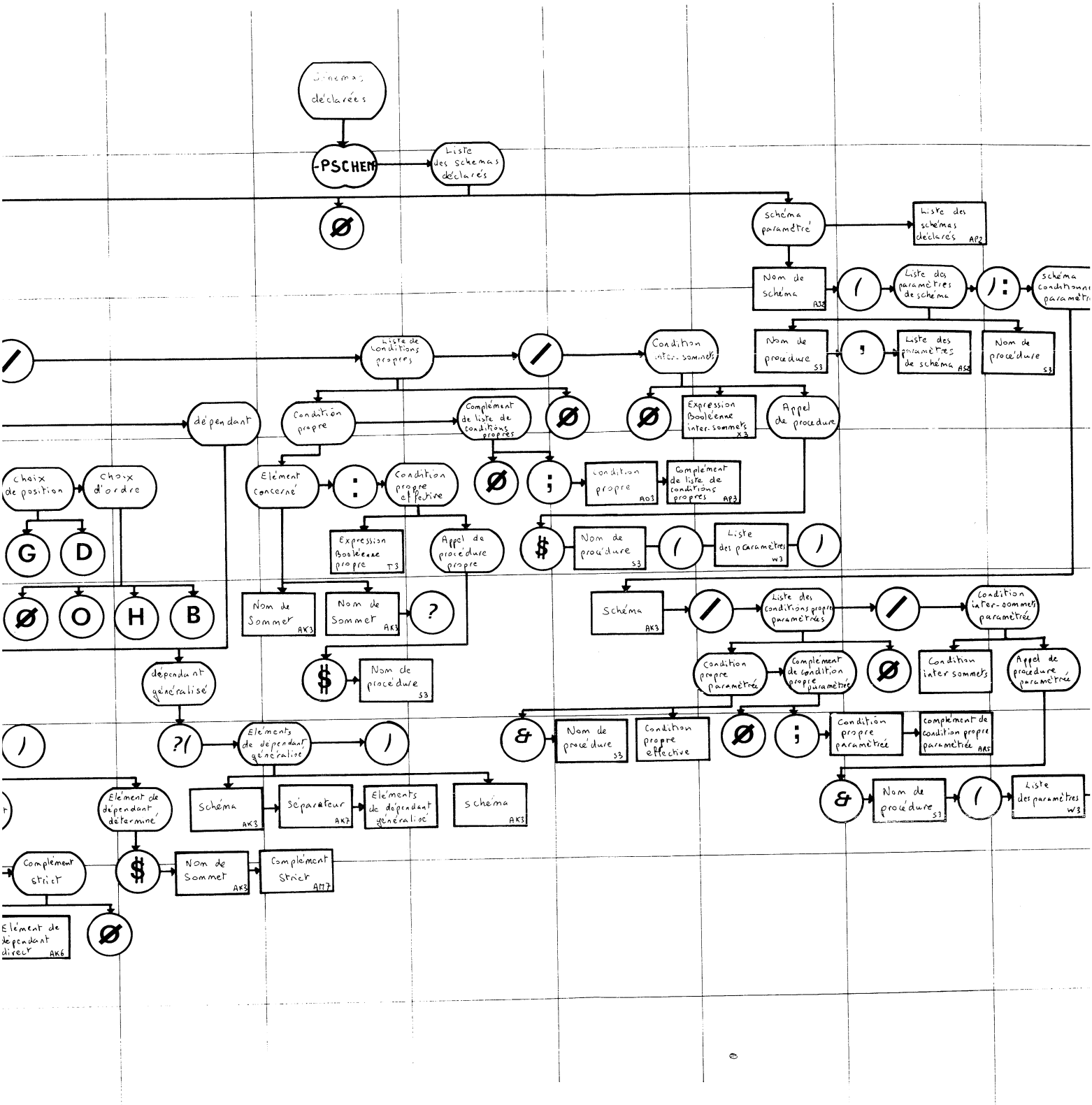
SYSTEME CETA (2)

W X Y Z AA AB AC AD



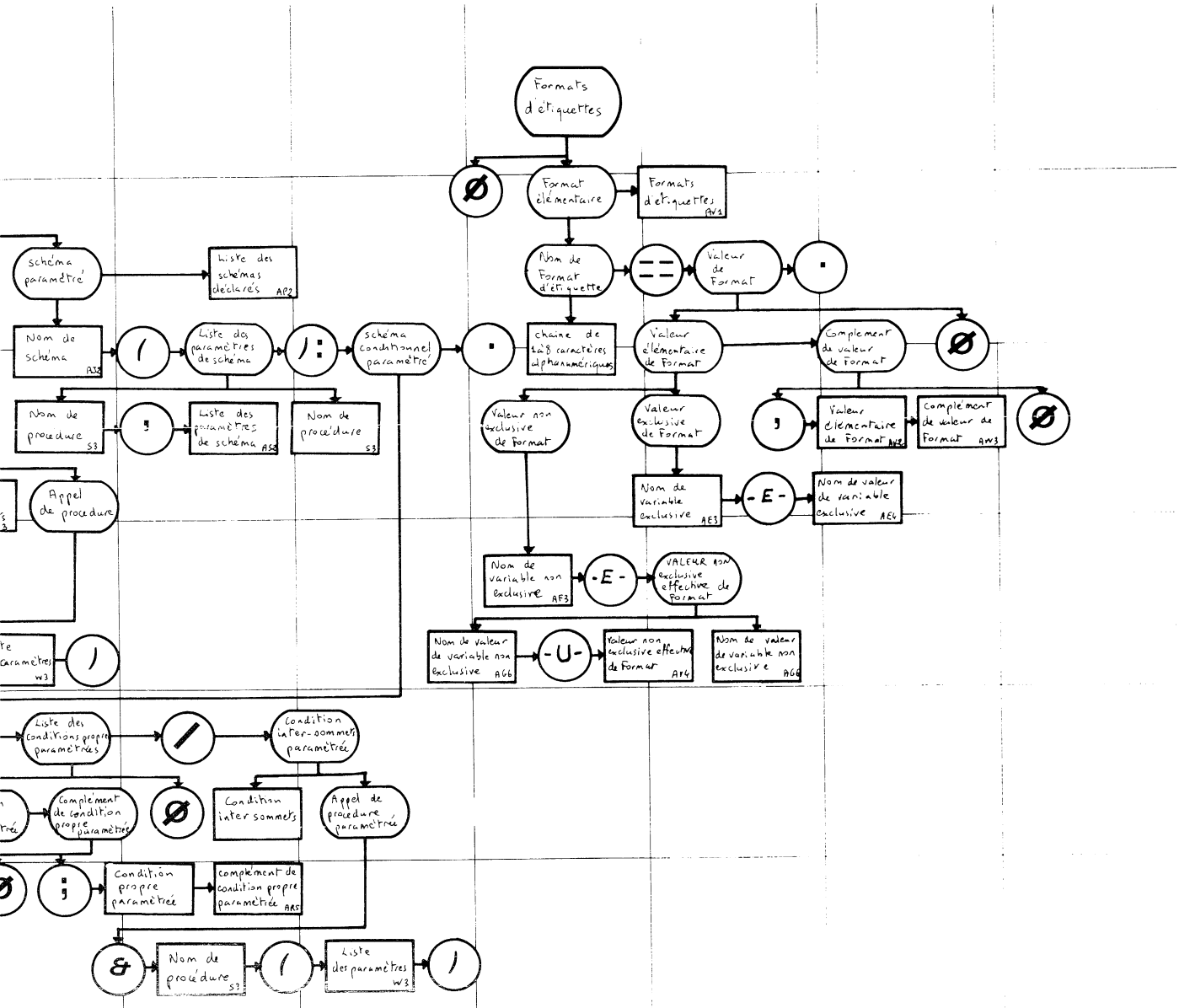
AC AD AE AF AG AH AI


1
2
3
4
5
6
7
8
AJ
AK
AL
AM
AN
schéma propre
Liste des schemas déclarés AP2
Nom de schéma
:
Schéma Conditionnel
.
chaine de 1 à 8 caractères alphanumériques
Schéma
/
Nom de Sommet
Référence de Sommet
Référence de dépendant
dépendant
chaine de 1 à 8 caractères alphanumériques
Ø
.
choix de position
choix d'ordre
$
choix sur sommet
G
D
H
B
O
dépendant direct
dépendant généralisé
(
Eléments de dépendant
)
?(
Elément de dépendant direct
Elément de dépendant joint.R
Elément de dépendant déterminé
Schéma AK3
Séparateur
Eléments de dépendants
*
complément strict
Nom de Sommet AK3
,
;
Séparateur AK7
Elément de dépendant direct AK6



SYSTEME CETA (3)

AM AN AO AP AQ AR AS AT



A (3)

AR AS AT AU AV AW AX

CHAPITRE VIII

APPLICATIONS

A - APPLICATION DU SYSTEME A.T.E.F.

Le système ATEF permet de définir un transducteur régulier composé. Le vocabualire de sortie de ce système est constitué par l'ensemble des états du transducteur défini. L'entrée de ce système est une suite de caractères. Un caractère particulier représenté par le "blanc" sépare différents groupes de caractères. Chacun de ces groupes est appelé une forme. Le système est donc orienté vers l'analyse d'une suite de formes. L'analyse d'une forme comprend la lecture des différents dictionnaires déclarés dans le système. Cette lecture réalise une comparaison entre la forme en cours d'analyse et les différentes entrées d'un dictionnaire. Cette comparaison s'effectue toujours dans le même sens qui est un paramètre du système. Le sens de l'analyse d'une forme peut s'effectuer de la gauche vers la droite, ou inversement de la droite vers la gauche. Différents niveaux d'analyse sont accessibles dans ce système. Un premier niveau d'analyse est donné par la manipulation des différents caractères d'une forme : segmentation, altération des segments, etc... Ce niveau est le plus élémentaire. Un ensemble de conditions sur l'assemblage combinatoire des différents segments de dictionnaires constitue un deuxième niveau d'analyse. Enfin, le troisième niveau d'analyse est obtenu par un ensemble de conditions portant sur un contexte inter-formes limité. Le résultat de cette analyse est constitué par une arborescence. Cette arborescence peut représenter une simple chaîne (arborescence à un niveau) ou une arborescence plus complexe résultant de constructions partielles. Ce système permet un grand nombre d'applications sur l'analyse des textes de langues naturelles. Il peut servir de base à la réalisation d'un système d'apprentissage, la grammaire étant alors orientée vers la détection des fautes.

D'autres applications de l'analyse des langues naturelles sont données par la communication homme-machine ou la traduction automatique.
Le système ATEF est alors le premier élément d'analyse. L'analyse des suite de caractères peut être employée dans d'autres domaines différents et servir de base à des systèmes plus complexes.

1 - ANALYSE DES NOMS CHIMIQUES

L'analyse donnée ici concerne une petite partie des noms des corps étudiés en chimie organique. Le nombre de ces corps utilisés croît de plus en plus et la nécessité de construire des tables de données apparaît.

La construction de telles banques de données (diverses caractéristiques des corps utilisés) nécessite une organisation par une subdivision de l'ensemble des éléments concernés. Naturellement, le nom du corps peut servir d'entrée à un tel système (une autre entrée serait donnée par la formule chimique du corps La classification des éléments s'effectue donc par le nom des différents éléments Cette classification, sans analyse préalable du nom, est arbitraire et peut par exemple être constituée par l'ordre alphabétique. L'inconvénient d'un tel système, lorsque le nombre d'éléments traités est important, réside dans le fait que deux éléments ayant des caractères chimiques semblables, peuvent être classés très différemment. Une classification par les principaux traits chimiques des éléments est donc préférable. Une telle classification nécessite une analyse qui permet de repérer les principaux traits représentés par les noms chimiques. Une telle analyse est donnée ici pour un sous-ensemble de corps. Les noms chimiques sont souvent des noms composés. Deux analyses différentes sont possibles dans le système ATEF suivant la présence ou non d'un caractère blanc entre deux mots significatifs d'un corps.

Exemple :

méthylpropane ou *méthyl propane*

Dans l'analyse présentée ici, nous supposerons qu'un mot représentant un corps ne contient pas de caractère blanc.

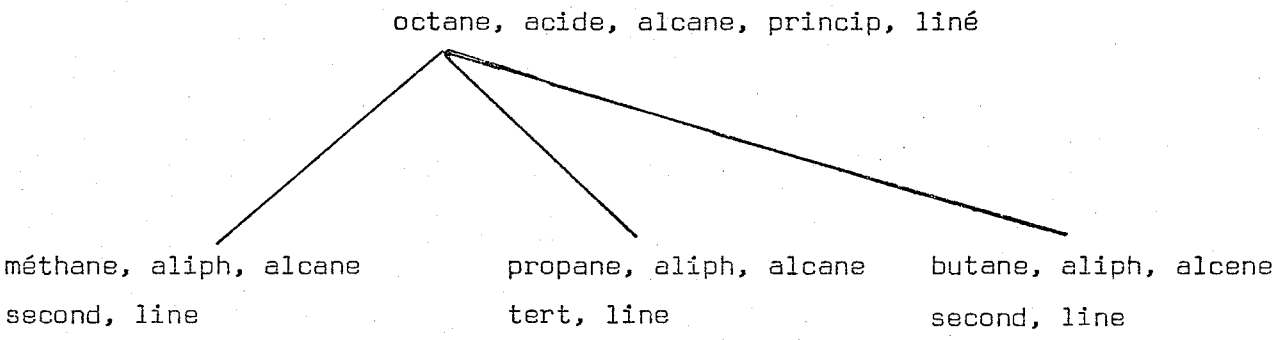
A l'intérieur d'un nom chimique on note souvent la présence de marquant de position. Ces marquants seront éliminés dans cette analyse. Le résultat de l'analyse sera une arborescence où la racine représentera le corps principal et ses descendants les différents éléments qui lui sont rattachés.

Exemple :

méthylpropobutylideneoctanoïque

(Il ne s'agit pas ici de savoir si l'existence de ce corps est possible mais d'accepter ou de refuser la formation des mots . Lorsque cette formation est possible il faut alors obtenir les différents éléments significatifs).

Le résultat de l'analyse de ce mot sera représenté par l'arborescence suivante :



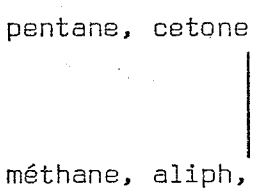
Le découpage de ce mot s'opérant ainsi :

mēth yl prop o but ylidène oct an oïque

Un autre exemple est donné par l'analyse du mot :

méthyl-4 pentène-3 one-2 (*oxyde de mésityle*)

Le nom analysé sera : *méthylpentènone*.
Le résultat sera représenté par l'arborescence :



Sur les exemples précédents, on peut remarquer que l'analyse doit fournir des valeurs propres au domaine d'analyse et non pas des éléments figés liés à l'analyse morphologique des mots. Les variables du système seront donc essentiellement constitués par des éléments ayant une signification chimique et non morphologique. Ces variables peuvent être considérées dans cette analyse comme des variables sémantiques. Elles seront déclarées dans le fichier de variables syntaxiques (ceci n'est pas obligatoire, mais permet dans ce cas de réduire le nombre de formats nécessaires).

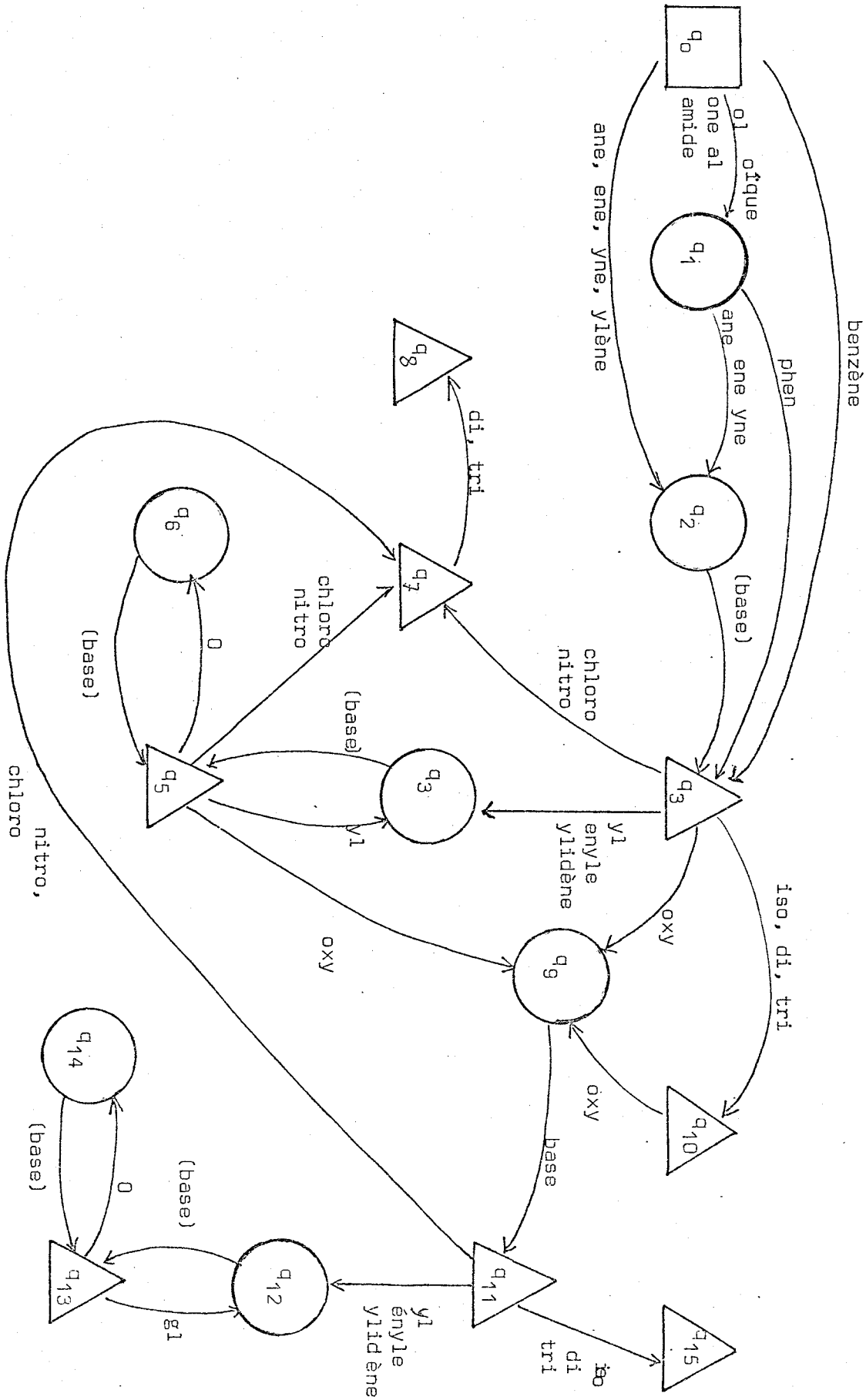
Nous trouverons ainsi comme variables les différents traits associés aux corps organiques :

- liaison
- structure
- fonction
- élément de substitution

Les variables morphologiques contiendront un seul élément déterminant la nature d'un segment : préfixe, base, suffixe.

Le schéma suivant résume les différentes transitions et les différents états du transducteur simulé.

Analyse des noms chimiques : schéma synthétique du transducteur



Les caractéristiques des différents états sont les suivantes :

q_0 SEG -E- SEGO.
q_1 SEG -E- SUFF, NIV -E- PRINCIP, FCT -NE- FCTO.
q_2 SEG -E- SUFF, NIV -E- PRINCIP, FCT -NE- FCTO.
q_3 SEG -E- BASE, NIV -E- PRINCIP.
q_4 SEG -E- DES, NIV -E- SECOND.
q_5 SEG -E- BASE, NIV -E- SECOND.
q_6 SEG -E- DES, NIV -E- TERT.
q_7 SEG -E- BASE.
q_8 SEG -E- PREF.
q_9 SEG -E- BASE, FCT -E- ETHER.
q_{10} SEG -E- PREF, NIV -E- PRINCIP.
q_{11} SEG -E- BASE, FCT -E- ETHER, NIV -E- PRINCIP.
q_{12} SEG -E- DES, FCT -E- ETHER, NIV -E- SECOND.
q_{13} SEG -E- BASE, FCT -E- ETHER, NIV -E- TERT.
q_{14} SEG -E- DES, FCT -E- ETHER, NIV -E- TERT.
q_{15} SEG -E- PREF, FCT -E- ETHER.

Les différents listings suivants représentent les fichiers externes du système pour cette analyse. Nous trouvons les éléments dans l'ordre de leur manipulation par le système :

- fichier de variables morphologiques et fichier de formats morphologiqu
- fichier de variables syntaxiques et fichier de formats syntaxiques
- fichier de grammaire
- fichier de dictionnaires
- exécution détaillée de l'analyse de deux mots
- exécution détaillée (version rapide) d'un ensemble de mots.

```
                -EXC-
 SEG:=(PREF,BASE,SUFF).  ** CARACTERISTIQUE DU SEGMENT ANALYSE.
                -NEX-
  DICT:=(1,2,3).  ** DICTIONNAIRES UTILISES POUR CETTE ANALYSE.
```



```
BASE    01==.**SEG-E-BASE.
BENZT   01==.**SEG-E-BASE.
CYCLM   01==.**SEG-E-PREF.
ELIMA   01==.**.
MODINC  01==.**.
PHEN    01==.**SEG-E-BASE.
PREFN   01==.**SEG-E-PREF.
PREOX   01==.**SEG-E-SUFF.
PRESUB  01==.**SEG-E-PREF.
SEPA    01==.**.
SFCT    01==.** SEG-E-SUFF.
SFCTD   01==.**SEG-E-SUFF.
SFCTE   01==.**SEG-E-SUFF.
SFCTS   01==.**SEG-E-SUFF.
VIDE    01==.**.
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMCH VARBS 18 MARS 1974 PAGE 1 17H 49MN 43S

```
          -EXC-
LIAISON:=(ALCANE,ALCENE,ALCYNE,PHENYL). **CARACTERISTIQUE DE LA CHAINE CARBON.
NIV:=(PRINCIP,SECOND,TERT).  **POSITION RELATIVE DE LA CHAINE.
STRUCT:=(CYCL,LINE). **FORME DE LA CHAINE CARBON.
VALENCE:=(1,2,3). **VALENCE DU RADICAL.
MULTIP:=(1,2,3,4,5,6,7,8,9). **MULTIPLICATEUR DE RADICAL.
            -NEX-
FONCTIO:=(ALIPH,ACIDE,ALCOOL,ETHER,ESTER,AMIDE,AMINE,ALDEHYDE,CETONE,AROMAT).
SUBST:=(CHLORO,NITRO,SULFO,IODO,BROMO).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMCH FTS 18 MARS 1974 PAGE 1 17H 51MN 07S

```
AL      01==.**FONCTIO-E-ALDEHYDE.
ALIPH   01==.**STRUCT-E-LINE.
AMIDE   01==.**FONCTIO-E-AMIDE.
AMINE   01==.**FONCTIO-E-AMINE.
ANE     01==.**LIAISON-E-ALCANE,FONCTIO-E-ALIPH,NIV-E-PRINCIP.
BENZ    01==.**STRUCT-E-CYCL,LIAISON-E-PHENYL,FONCTIO-E-AROMAT,NIV-E-PRINCIP.
BROMO   01==.**SUBST-E-BROMO.
CHLORO  01==.**SUBST-E-CHLORO.
CYCLO   01==.**STRUCT-E-CYCL.
ENE     01==.**LIAISON-E-ALCENE,FONCTIO-E-ALIPH,NIV-E-PRINCIP.
ENYLE   01==.**FONCTIO-E-ALIPH,LIAISON-E-ALCENE,NIV-E-SECOND.
HEXA    01==.**MULTIP-E-6.
IODO    01==.**SUBST-E-IODO.
ISO     01==.**MULTIP-E-2.
NITRO   01==.**SUBST-E-NITRO.
O       01==.**FONCTIO-E-ALIPH,LIAISON-E-ALCANE,NIV-E-TERT.
OIQUE   01==.**FONCTIO-E-ACIDE.
OL      01==.**FONCTIO-E-ALCOOL.
ONE     01==.**FONCTIO-E-CETONE.
OXY     01==.**FONCTIO-E-ETHER.
PENTA   01==.**MULTIP-E-5.
SULFO   01==.**SUBST-E-SULFO.
TETRA   01==.**MULTIP-E-4.
TRI     01==.**MULTIP-E-3.
VIDES   01==.**.
YL      01==.**FONCTIO-E-ALIPH,LIAISON-E-ALCANE,NIV-E-SECOND.
YNE     01==.**LIAISON-E-ALCYNE,FONCTIO-E-ALIPH,NIV-E-PRINCIP.
```

```
**       GRAMMAIRE D'ANALYSE DE NOMS CHIMIQUES        .
**                                                    .
**                                                    .
**  REGLES CONCERNANT LES SUFFIXES  .
     RSUF1:  SFCT   ==  VAR(C):=VAR(A),DICT(C):=2/
                        SEG(C)-E-SEG0.
     RSUF2:  SFCTS  ==   VAR(C):=VAR(A)/
                        SEG(C)-E-SEG0/
                        TCHAINE(0,'','E').
     RSUF3:  SFCT   ==   LIAISON(C):=LIAISON(A),NIV(C):=NIV(A),DICT(C):=2/
                        FONCTIO(C)-DANS-(-N-(ALIPH))-ET-SEG(C)-E-SUFF.
     RSUF4:  SFCTD-SFCTE  ==  VAR(C):=VAR(A),-ARRET-,DICT(C):=2/
                        VAR(PS1)-NE-VAR0.
**   REGLES CONCERNANT LES BASES  .
     RBAS1:  BENZT  ==   VAR(C):=VAR(A)/VAR(C)-E-VAR0///
                         RNCST,RCSTV3,RCSTV1,RCSTV2.
     RBAS2:  PHEN   ==  VARNS(C):=VARNS(A)-U-VARNS(C),VARM(C):=VARM(A)/
                        FONCTIO(C)-DANS-(-N-(ALIPH))-ET-SEG(C)-E-SUFF///
                        RNCST,RCSTV3,RCSTV1,RCSTV2.
     RBAS3:  BASE   ==  STRUCT(C):=STRUCT(A),VARM(C):=VARM(A),-ARRET-/
                        SEG(C)-NE-BASE///RNCST,RCSTV3,RCSTV1,RCSTV2.
     RBAS4:  PHEN   ==  VARNS(C):=VARNS(C)-U-VARNS(A),
                        VARES(C):=VARES(A),VARM(C):=VARM(A)/
                         VAR(PS1)-NE-VAR0-ET-SEG(C)-E-SUFF///
                         RNCST,RCSTV1,RCSTV2.
     RCOMP:  PREOX  == -SOL-,-IHN-,DICT(C):=2/
                        SCHAINE(A,0,1)-NE-' '///
                        ROXYC.
     RSEP:  SEPA  ==  VAR(C):=VAR(A),-INIT-.
     ROXYC:  VIDE  ==  VAR(C):=VAR(A).
     RCONST:  ELIMA  == .
**   REGLES CONCERNANT LES PREFIXES  .
     RPRE1:  PREFN  ==  SEG(C):=SEG(A),MULTIP(C):=MULTIP(A)/SEG(C)-E-BASE/
                        // RNCSTP,RCSTV3,RCSTV1,RCSTV2.
     RPRE2:  PRESUB  ==  SUBST(C):=SUBST(A)/
                        -N-(SUBST(A)-DANS-SUBST(C))///
                         RNCSTP,RCSTV3,RCSTV1,RCSTV2.
     RCYCL:  CYCLM  == STRUCT(C):=STRUCT(A)/SEG(C)-E-BASE-ET-STRUCT(C)-NE-CYCL//
                        /RNCSTP,RCSTV3,RCSTV1,RCSTV2.
**  REGLES DE CONSTRUCTIONS  .
     RNCST:  VIDE  ==  DICT(C):=3.
     RNCSTP:  VIDE  ==  DICT(C):=3.
     RCSTV1:  VIDE  ==  DICT(C):=1-U-2,-SOL-,-IHN-/
                          SCHAINE(A,0,1)-NE-' '-ET-NIV(C)-E-SECOND.
     RCSTV2:  VIDE  ==  DICT(C):=1-U-2,-SOL-,-IHN-/SCHAINE(A,0,1)-NE-' '-ET-
                          NIV(C)-E-TERT.
     RCSTV3:  VIDE  ==  DICT(C):=1-U-2,-SOL-,-IAX-/SCHAINE(A,0,1)-NE-' '-ET-
                          NIV(C)-E-PRINCIP.
**  REGLES OBLIGATOIREMENT PRESENTES  .
     RDICT: (1,2/NU).
     MOTINC:  MODINC  == VAR(C):=VAR(A),-TRANS-.
                                -FIN-
```

FICHIER : FILEMCH DICAFF1 18 MARS 1974 17H 58MN 12S

```
-                ==ELIMA   (VIDES   ).
AL               ==SFCTS   (AL      ).
AMIDE            ==SFCTS   (AMIDE   ).
AMINE            ==SFCTS   (AMINE   ).
ANE              ==SFCT    (ANE     ).
ENE              ==SFCT    (ENE     ).
ENYLE            ==SFCTD   (ENYLE   ).
O                ==SFCTE   (O       ).
OIQUE            ==SFCTS   (OIQUE   ).
OL               ==SFCTS   (OL      ).
ONE              ==SFCTS   (ONE     ).
YL               ==SFCTD   (YL      ).
YLENE            ==SFCT    (YNE     ).
YLIDENE          ==SFCTD   (ENYLE   ).
YNE              ==SFCT    (YNE     ).
```


FICHIER : FILEMCH DICBAS2 18 MARS 1974 17H 59MN 56S

```
.                ==SEPA    (VIDES   ,POINT                ).
BENZENE          ==BENZT   (BENZ    ,BENZENE              ).
BUT              ==BASE    (ALIPH   ,BUTANE               ).
EICOS            ==BASE    (ALIPH   ,EICOSANE             ).
ETH              ==BASE    (ALIPH   ,ETHANE               ).
HEPT             ==BASE    (ALIPH   ,HEPTANE              ).
HEPTACONT        ==BASE    (ALIPH   ,HEPTACONTANE         ).
HEPTADEC         ==BASE    (ALIPH   ,HEPTADECANE          ).
HEX              ==BASE    (ALIPH   ,HEXANE               ).
HEXADEC          ==BASE    (ALIPH   ,HEXADECANE           ).
METH             ==BASE    (ALIPH   ,METHANE              ).
NON              ==BASE    (ALIPH   ,NONANE               ).
NONADEC          ==BASE    (ALIPH   ,NONADECANE           ).
OCT              ==BASE    (ALIPH   ,OCTANE               ).
PENT             ==BASE    (ALIPH   ,PENTANE              ).
PENTADEC         ==BASE    (ALIPH   ,PENTADECANE          ).
PHEN             ==PHEN    (BENZ    ,BENZENE              ).
PROP             ==BASE    (ALIPH   ,PROPANE              ).
TRIACONT         ==BASE    (ALIPH   ,TRIACONTANE          ).
```

```
BROMO                ==PRESUB  (BROMO   ).
CHLORO               ==PRESUB  (CHLORO  ).
CYCLO                ==CYCLM   (CYCLO   ).
DI                   ==PREFN   (ISO     ).
HEXA                 ==PREFN   (HEXA    ).
IODO                 ==PRESUB  (IODO    ).
ISO                  ==PREFN   (ISO     ).
NITRO                ==PRESUB  (NITRO   ).
OXY                  ==PREOX   (OXY     ).
PENTA                ==PREFN   (PENTA   ).
SULFO                ==PRESUB  (SULFO   ).
TETRA                ==PREFN   (TETRA   ).
TRI                  ==PREFN   (TRI     ).
```

DETAIL DE L'EXECUTION , TEXTE : 3

```
** OCCURRENCE : METHYLPROPOBUTYLIDENEOCTANOIQUE
MORPHE : OIQUE                          REGLE : RSUF2    FORMAT : SFCTS    OIQUE
      ETAT COURANT : K0(UL0,SUFF,/,$,/,FONCTIO(ACIDE))
RESTE : METHYLPROPOBUTYLIDENEOCTANE
MORPHE : ANE                            REGLE : RSUF1    FORMAT : SFCT     ANE
MORPHE : ANE                            REGLE : RSUF3    FORMAT : SFCT     ANE
      ETAT COURANT : K0(UL0,SUFF,/,$,ALCANE,PRINCIP,/,FONCTIO(ACIDE))
RESTE : METHYLPROPOBUTYLIDENEOCT
MORPHE : OCT                            REGLE : RBAS3    FORMAT : BASE     ALIPH
      ETAT COURANT : K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE))
MORPHE : OCT                            REGLE : RNCST    FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE))
RESTE : METHYLPROPOBUTYLIDENE
MORPHE : OCT                            REGLE : RCSTV3   FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE))
RESTE : METHYLPROPOBUTYLIDENE
MORPHE : YLIDENE                        REGLE : RSUF4    FORMAT : SFCTD    ENYLE
      ETAT COURANT : K0(UL0,SUFF,/,$,ALCENE,SECOND,/,FONCTIO(ALIPH))
RESTE : METHYLPROPOBUT
MORPHE : BUT                            REGLE : RBAS3    FORMAT : BASE     ALIPH
      ETAT COURANT : K0(BUTANE,BASE,/,$,ALCENE,SECOND,LINE,/,FONCTIO(ALIPH))
MORPHE : BUT                            REGLE : RNCST    FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(BUTANE,BASE,/,$,ALCENE,SECOND,LINE,/,FONCTIO(ALIPH))
RESTE : METHYLPROPO
MORPHE : BUT                            REGLE : RCSTV3   FORMAT :
 ... SOUS-REGLE NON APPLICABLE
MORPHE : BUT                            REGLE : RCSTV1   FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(BUTANE,BASE,/,$,ALCENE,SECOND,LINE,/,FONCTIO(ALIPH))
RESTE : METHYLPROPO
MORPHE : O                              REGLE : RSUF4    FORMAT : SFCTE    O
      ETAT COURANT : K0(UL0,SUFF,/,$,ALCANE,TERT,/,FONCTIO(ALIPH))
RESTE : METHYLPROP
MORPHE : PROP                           REGLE : RBAS3    FORMAT : BASE     ALIPH
      ETAT COURANT : K0(PROPANE,BASE,/,$,ALCANE,TERT,LINE,/,FONCTIO(ALIPH))
MORPHE : PROP                           REGLE : RNCST    FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(PROPANE,BASE,/,$,ALCANE,TERT,LINE,/,FONCTIO(ALIPH))
RESTE : METHYL
MORPHE : PROP                           REGLE : RCSTV3   FORMAT :
 ... SOUS-REGLE NON APPLICABLE
MORPHE : PROP                           REGLE : RCSTV1   FORMAT :
 ... SOUS-REGLE NON APPLICABLE
MORPHE : PROP                           REGLE : RCSTV2   FORMAT :
```

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : CH

```
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(PROPANE,BASE,/,$,ALCANE,TERT,LINE,/,FONCTIO(ALIPH))
RESTE : METHYL
MORPHE :  YL                          REGLE : RSUF4     FORMAT : SFCTD     YL
      ETAT COURANT : K0(ULO,SUFF,/,$,ALCANE,SECOND,/,FONCTIO(ALIPH))
RESTE : METH
MORPHE :  METH                        REGLE : RBAS3     FORMAT : BASE      ALIPH
      ETAT COURANT : K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
MORPHE :  METH                        REGLE : RNCST     FORMAT :
 ... SOUS-REGLE APPLICABLE
  -->  DECOUPAGE : METH YL PROP O BUT YLIDENE OCT ANE OIQUE   K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->             K0(PROPANE,BASE,/,$,ALCANE,TERT,LINE,/,FONCTIO(ALIPH))
  -->             K0(BUTANE,BASE,/,$,ALCENE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->             K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE))

** OCCURRENCE : .
MORPHE :  .                           REGLE : RSEP      FORMAT : SEPA      VIDES
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : METHYLPENTENONE
MORPHE :  ONE                         REGLE : RSUF2     FORMAT : SFCTS     ONE
      ETAT COURANT : K0(ULO,SUFF,/,$,/,FONCTIO(CETONE))
RESTE : METHYLPENTENE
MORPHE :  ENE                         REGLE : RSUF1     FORMAT : SFCT      ENE
MORPHE :  ENE                         REGLE : RSUF3     FORMAT : SFCT      ENE
      ETAT COURANT : K0(ULO,SUFF,/,$,ALCENE,PRINCIP,/,FONCTIO(CETONE))
RESTE : METHYLPENT
MORPHE :  PENT                        REGLE : RBAS3     FORMAT : BASE      ALIPH
      ETAT COURANT : K0(PENTANE,BASE,/,$,ALCENE,PRINCIP,LINE,/,FONCTIO(CETONE))
MORPHE :  PENT                        REGLE : RNCST     FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(PENTANE,BASE,/,$,ALCENE,PRINCIP,LINE,/,FONCTIO(CETONE))
RESTE : METHYL
MORPHE :  PENT                        REGLE : RCSTV3    FORMAT :
 ... SOUS-REGLE APPLICABLE
      ETAT COURANT : K0(PENTANE,BASE,/,$,ALCENE,PRINCIP,LINE,/,FONCTIO(CETONE))
RESTE : METHYL
MORPHE :  YL                          REGLE : RSUF4     FORMAT : SFCTD     YL
      ETAT COURANT : K0(ULO,SUFF,/,$,ALCANE,SECOND,/,FONCTIO(ALIPH))
RESTE : METH
MORPHE :  METH                        REGLE : RBAS3     FORMAT : BASE      ALIPH
      ETAT COURANT : K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
MORPHE :  METH                        REGLE : RNCST     FORMAT :
 ... SOUS-REGLE APPLICABLE
  -->  DECOUPAGE : METH YL PENT ENE ONE   K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->             K0(PENTANE,BASE,/,$,ALCENE,PRINCIP,LINE,/,FONCTIO(CETONE))



-01-K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE),+ARBRE,K0(METHANE,BA
SE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)),+ARBRE,K0(PROPANE,BASE,/,$,ALCANE,TE
RT,LINE,/,FONCTIO(ALIPH)),+ARBRE,K0(BUTANE,BASE,/,$,ALCENE,SECOND,LINE,/,FONCTIO
(ALIPH)))+K0(POINT,/,$,/)-02-/

-01-K0(PENTANE,BASE,/,$,ALCENE,PRINCIP,LINE,/,FONCTIO(CETONE),+ARBRE,K0(METHANE,
BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)))-02-
```

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 4 CODE LANGUE : CH

```
** OCCURRENCE : OCTANE
  -->  DECOUPAGE : OCT ANE   K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALIPH))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : DIMETHYLPROPANAL
  -->  DECOUPAGE : DI METH YL PROP ANE AL   K0(METHANE,PREF,/,$,ALCANE,SECOND,LINE,2,/,FONCTIO(ALIPH))
  -->              K0(PROPANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALDEHYDE))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : CHLOROMETHYLPROPYLHEXANOIQUE
  -->  DECOUPAGE : CHLORO METH YL PROP YL HEX ANE OIQUE   K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH),SUBST((
))
  -->              K0(PROPANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->              K0(HEXANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : PROPOXYPROPANE
  -->  DECOUPAGE : PROP OXY PROP ANE   K0(PROPANE,BASE,/,$,LINE,/)
  -->              K0(PROPANE,SUFF,/,$,/,FONCTIO(ETHER))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : METHANAMIDE
  -->  DECOUPAGE : METH ANE AMIDE   K0(METHANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(AMIDE))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : METHYLBUTANAMIDE
  -->  DECOUPAGE : METH YL BUT ANE AMIDE   K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->              K0(BUTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(AMIDE))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)

** OCCURRENCE : METHYLCYCLOPROPANE
  -->  DECOUPAGE : METH YL CYCLO PROP ANE   K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
  -->              K0(PROPANE,BASE,/,$,ALCANE,PRINCIP,CYCL,/,FONCTIO(ALIPH))

** OCCURRENCE : .
  -->  DECOUPAGE : .   K0(POINT,/,$,/)
```

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 4 CODE LANGUE : CH

** OCCURRENCE : HEXACHLOROCYCLOHEXANE

--> DECOUPAGE : HEXA CHLORO CYCLO HEX ANE K0(HEXANE,PREF,/,$,ALCANE,PRINCIP,CYCL,6,/,FONCTIO(ALIPH),SUBST(CHLORO))

** OCCURRENCE : .
--> DECOUPAGE : . K0(POINT,/,$,/)

** OCCURRENCE : TRIPHENYLMETHANE
--> DECOUPAGE : TRI PHEN YL METH ANE K0(BENZENE,PREF,/,$,PHENYL,PRINCIP,CYCL,3,/,FONCTIO(ALIPH,AROMAT))
--> K0(METHANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALIPH))

** OCCURRENCE : .
--> DECOUPAGE : . K0(POINT,/,$,/)

** OCCURRENCE : METHYLISOPROPYLBENZENE
--> DECOUPAGE : METH YL ISO PROP YL BENZENE K0(METHANE,BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH))
--> K0(PROPANE,PREF,/,$,ALCANE,SECOND,LINE,2,/,FONCTIO(ALIPH))
--> K0(BENZENE,BASE,/,$,PHENYL,PRINCIP,CYCL,/,FONCTIO(AROMAT))

1-K0(OCTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALIPH))+K0(POINT,/,$,/)-02-
0

-01-K0(PROPANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALDEHYDE),+ARBRE,K0(METHAN
E,PREF,/,$,ALCANE,SECOND,LINE,2,/,FONCTIO(ALIPH)))+K0(POINT,/,$,/)-02-/

-01-K0(HEXANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ACIDE),+ARBRE,K0(METHANE,BA
SE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH),SUBST(CHLORO)),+ARBRE,K0(PROPANE,BASE
,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)))+K0(POINT,/,$,/)-02-/

-01-K0(PROPANE,BASE,/,$,LINE,/,+ARBRE,K0(PROPANE,SUFF,/,$,/,FONCTIO(ETHER)))+K0(
POINT,/,$,/)-02-/

-01-K0(METHANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(AMIDE))+K0(POINT,/,$,/)-02
-/

-01-K0(BUTANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(AMIDE),+ARBRE,K0(METHANE,BA
SE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)))+K0(POINT,/,$,/)-02-/

-01-K0(PROPANE,BASE,/,$,ALCANE,PRINCIP,CYCL,/,FONCTIO(ALIPH),+ARBRE,K0(METHANE,B
ASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)))+K0(POINT,/,$,/)-02-/

-01-K0(HEXANE,PREF,/,$,ALCANE,PRINCIP,CYCL,6,/,FONCTIO(ALIPH),SUBST(CHLORO))+K0(
POINT,/,$,/)-02-/

-01-K0(METHANE,BASE,/,$,ALCANE,PRINCIP,LINE,/,FONCTIO(ALIPH),+ARBRE,K0(BENZENE,P
REF,/,$,PHENYL,PRINCIP,CYCL,3,/,FONCTIO(ALIPH,AROMAT)))+K0(POINT,/,$,/)-02-/

-01-K0(BENZENE,BASE,/,$,PHENYL,PRINCIP,CYCL,/,FONCTIO(AROMAT),+ARBRE,K0(METHANE,
BASE,/,$,ALCANE,SECOND,LINE,/,FONCTIO(ALIPH)),+ARBRE,K0(PROPANE,PREF,/,$,ALCANE,
SECOND,LINE,2,/,FONCTIO(ALIPH)))-02-

-01-02-

2 - ANALYSE DE LANGUE NATURELLE

L'analyse des langues naturelles comprend plusieurs phases. Chacune de celles-ci correspond à un problème particulier et nécessite un outil logique particulier. Le système ATEF permet d'effectuer une analyse morphologique et également un début d'analyse syntaxique. L'analyse présentée ici sur le français correspond à l'analyse d'un sous-ensemble de mots essentiellement constitué de noms et d'adjectifs. Cette analyse n'a pas été conçue avec un souci d'efficacité mais plutôt dans le but de présenter les possibilités du système pour une telle analyse. Ainsi les altérations de bases effectuées ne sont pas toutes productives et ne simplifient pas énormément les dictionnaires. L'analyse des mots débute par l'examen des désinences puis des suffixes. L'analyse de ces éléments est déterminante de la catégorie d'un mot. Dans cet exemple, les variables seront toutes linguistiques, cette analyse forme un sous-ensemble de l'analyse du français réalisée par l'équipe du Groupe d'Etudes pour la Traduction Automatique.

Le transducteur défini permet l'analyse nominale et adjectivale dont le système de désinence (variable SDNA) peut être résumé par le tableau suivant :

Chaque élément du tableau contient soit le symbole "#" (qui représente la désinence nulle), soit la désinence possible.
Un exemple de mot admettant le système de désinence indiqué est donné par la liste suivante :

1 GAZ, MARRON
2 MAISON
3 CANDIDAT, NOIR
4 CHAT, CHRETIEN
5 CHAPEAU, PEAU
6 EPOUX, JOYEUX
7 SERVEUR, MENTEUR
8 ACTEUR, GENERATEUR
9 CANAL, GENERAL
10 VEUF, ACTIF
11 OURS, GRIS

N° / GNR/NBR	1	2	3	4	5	6	7	8	9	10	11
MAS/SING	#	#	#	#	#	x	eur	tern	al	f	#
FEM/SING	#	#	e	consonne double + e	#	se	euse	trice	ale	ve	e
MAS/PLUR	#	s	s	s	x	x	eurs	teurs	aux	fs	#
FEM/PLUR	#	s	es	consonne double + es	x	ses	euses	trices	ales	ves	es

Désinences nominale et adjectivale

Les différents listings suivants représentent les fichiers extrêmes du système pour cette analyse. Nous trouvons les éléments dans l'ordre de leur manipulation par le système :

- fichier de variables morphologiques et fichier de formats syntaxiques
- fichier de variables syntaxiques et fichier de formats syntaxiques
- fichier de grammaire
- fichier de dictionnaires
- exécution détaillée de l'analyse de deux mots
- exécution détaillée (version rapide) d'un ensemble de mots.

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 VARBM 18 MARS 1974 17H 09MN 34S

```
                                    -EXC-
SEG := (D,S,B,P).
                                    -NEX-
PREF  := (IN,IM,IL,IR).
    SDNA:= (1,2,3,4,5,6,7,8,9,10,11) .
SDN:= (MENT,TE1,ITE1,ETE1,EUR,ESSE,ABLE,IBLE,ATION,ITION).
PDRP := (AM,AN,VA,VAM,VN).
PDRN := (N,NAM,NAN,NVA,NVAM).
DICT := (1,2,3).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 FTM 18 MARS 1974 17H 10MN 47S

```
BA1     01== .**  SEG -E- B , SDNA -E- 1 .
BA1B    01== .**  SEG -E- B , SDNA -E- 1 .
BA10    01== .**  SEG -E- B , SDNA -E- 10 .
BA10B   01== .**  SEG -E- B , SDNA -E- 10 .
BA11    01== .**  SEG -E- B , SDNA -E- 11 .
BA11B   01== .**  SEG -E- B , SDNA -E- 11 .
BA2     01== .**  SEG -E- B , SDNA -E- 2 .
BA2B    01== .**  SEG -E- B , SDNA -E- 2 .
BA3     01== .**  SEG -E- B , SDNA -E- 3 .
BA3B    01== .**  SEG -E- B , SDNA -E- 3 .
BA4     01== .**  SEG -E- B , SDNA -E- 4 .
BA4B    01== .**  SEG -E- B , SDNA -E- 4 .
BA6     01== .**  SEG -E- B , SDNA -E- 6 .
BA6B    01== .**  SEG -E- B , SDNA -E- 6 .
BA7     01== .**  SEG -E- B , SDNA -E- 7 .
BA7B    01== .**  SEG -E- B , SDNA -E- 7 .
BA8     01== .**  SEG -E- B , SDNA -E- 8 .
BA8B    01== .**  SEG -E- B , SDNA -E- 8 .
BA9     01== .**  SEG -E- B , SDNA -E- 9 .
BA9B    01== .**  SEG -E- B , SDNA -E- 9 .
BN1     01== .** SEG -E- B , SDNA -E- 1 .
BN1B    01== .** SEG -E- B , SDNA -E- 1 .
BN10    01== .** SEG -E- B , SDNA -E- 10 .
BN10B   01== .** SEG -E- B , SDNA -E- 10 .
BN2     01== .**  SEG -E- B , SDNA -E- 2 .
BN2B    01== .**  SEG -E- B , SDNA -E- 2 .
BN3     01== .**  SEG -E- B , SDNA -E- 3 .
BN3B    01== .**  SEG -E- B , SDNA -E- 3 .
BN4     01== .**  SEG -E- B , SDNA -E- 4 .
BN4B    01== .**  SEG -E- B , SDNA -E- 4 .
BN5     01== .**  SEG -E- B , SDNA -E- 5 .
BN5B    01== .**  SEG -E- B , SDNA -E- 5 .
BN6     01== .**  SEG -E- B , SDNA -E- 6 .
BN6B    01== .**  SEG -E- B , SDNA -E- 6 .
BN7     01== .**  SEG -E- B , SDNA -E- 7 .
BN7B    01== .**  SEG -E- B , SDNA -E- 7 .
BN8     01== .**  SEG -E- B , SDNA -E- 8 .
BN8B    01== .**  SEG -E- B , SDNA -E- 8 .
BN9     01== .**  SEG -E- B , SDNA -E- 9 .
BN9B    01== .**  SEG -E- B , SDNA -E- 9 .
D0      01==. **SEG -E- D, SDNA -E- 3 -U- 4 .
D1      01== .** SEG -E- D , SDNA -E- 2 .
D3      01==. **SEG -E- D, SDNA -E- 3 -U- 4 -U- 11 .
D4      01==. **SEG -E- D, SDNA -E- 5 .
D5      01==. **SEG -E- D, SDNA -E- 6 .
D6      01==. **SEG -E- D, SDNA -E- 7.
D7      01==. **SEG -E- D, SDNA -E- 8 .
D8      01==. **SEG -E- D, SDNA -E- 9 .
D9      01==. **SEG -E- D, SDNA -E- 10 .
INVAR   01== .** SEG-E-B .
MODINC  01== .**.
PM1     01== .** SEG -E- P , PREF -E-  IN .
```

```
PM2     01== .** SEG -E- P , PREF -E-  IM .
PM3     01== .** SEG -E- P , PREF -E-  IL .
PM4     01== .** SEG -E- P , PREF -E-  IR .
PONCT   01== .** SEG-E-B .
SM1     01== .** SEG -E- S , SDN -E-  MENT .
SM10    01== .** SEG -E- S , SDN -E-  ITION .
SM2     01== .** SEG -E- S , SDN -E-  TE1 .
SM3     01== .** SEG -E- S , SDN -E-  ITE1 .
SM'     01== .** SEG -E- S , SDN -E-  ETE1 .
SM5     01== .** SEG -E- S , SDN -E-  EUR .
SM6     01== .** SEG -E- S , SDN -E-  ESSE .
SM7     01== .** SEG -E- S , SDN -E-  ABLE .
SM8     01== .** SEG -E- S , SDN -E-  IBLE .
SM9     01== .** SEG -E- S , SDN -E-  ATION .
VBM     01== .** SEG -E- B .
VBMB    01== .** SEG -E- B .
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****


```
                                                -EXC- .
 CAT := (VRB,NMC,NMP,PRP,AJQ,AJP,AJD,ART,ADV,PRE,CNJ,PCT).
DRV := (AJAV,AJNM,VBAJ,VBNM,VBAJAV).
NEG := (NG).
                                                -NEX-
GNR := (MAS,FEM) .
NBR := (SIN,PLU) .
 SEMA := (HUM,CONCR,ABSTR,ANIM,INAM,QUALI,ACTIO,OPERAT,RELAT).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 FTS 18 MARS 1974 17H 18MN 22S

```
POINT   01== .** CAT-E-PCT .
PS      01== .** NEG -E- NG .
SAR1    01== .** CAT -E- ART , NBR -E- SIN , GNR -E- MAS .
SAR2    01== .** CAT -E- ART , NBR -E- SIN , GNR -E- FEM .
SA1     01== .** CAT -E- AJQ,GNR -E- MAS -U- FEM ,NBR -E- SIN -U- PLU.
SA1C    01== .** CAT -E- AJQ,GNR -E- MAS -U- FEM ,NBR -E- SIN -U- PLU ,
SA1C    02       SEMA -E- ABSTR -U- QUALI .
SA2     01== .** CAT -E- AJQ, PDRP -E- AM .
SA22    01== .** CAT -E- AJQ, PDRP -E- AN , SDN -E- ESSE .
SA31    01== .** CAT -E- AJQ,PDRP -E- AM -U- AN, SDN -E- EUR .
SA32    01== .** CAT -E- AJQ,PDRP -E- AM -U- AN,SDN -E- ESSE .
SA33    01== .** CAT -E- AJQ,PDRP -E- AN -U- AM, SDN -E- ITE1 .
SA34    01== .** CAT -E- AJQ,PDRP -E- AM -U- AN,SDN -E- ETE1 .
SA401   01== .** CAT -E- AJQ,PDRP -E- AM,PDRN -E- N -U- NAM,PREF-E-IN.
SA531   01== .** CAT -E- AJQ, PDRP -E- AM -U- AN, PDRN -E- N -U- NAM
SA531   02         -U- NAN , SDN -E- ITE1, PREF -E- IN  , SEMA -E- ABSTR .
SA534   01== .** CAT -E- AJQ, PDRP -E- AM -U- AN, PDRN -E-
SA534   02           N -U- NAM -U- NAN , SDN -E- ITE1 , PREF -E- IR .
SN1     01== .**  GNR -E- MAS ,  NBR -E- SIN -U- PLU ,  CAT -E- NMC .
SN1A    01== .**  GNR -E- MAS ,  NBR -E- SIN -U- PLU ,  CAT -E- NMC ,
SN1A    02        SEMA -E- ABSTR .
SN2     01== .**  GNR -E- FEM ,  NBR -E- SIN -U- PLU ,  CAT -E- NMC .
SN3A    01== .**  GNR -E- MAS ,  CAT -E- NMC , SEMA -E- ANIM -U- CONCR .
SN3S    01== .** GNR -E- MAS , CAT -E- NMC , SEMA -E- CONCR -U- INAM .
SN4     01== .**  GNR -E- FEM ,  CAT -E- NMC .
SN5A    01== .** CAT -E- NMC , SEMA -E- ANIM -U- CONCR  .
SN5H    01== .** CAT -E- NMC , SEMA -E- HUM -U- CONCR  .
SPR1    01== .** CAT -E- PRP , NBR -E- SIN .
SS1     01== .**  DRV -E- AJAV ,  CAT -E- ADV .
SS2     01== .**  DRV -E- AJNM ,  CAT -E- NMC , GNR -E- FEM ,  SDNA -E- 2 .
SS3     01== .**  DRV -E- VBAJ ,  CAT -E- AJQ , GNR -E- MAS -U- FEM ,SDNA -E-2.
SS4     01== .** DRV -E- VBNM, CAT -E- NMC, GNR -E- FEM, SDNA -E- 2.
VBS1    01== .** CAT -E- VRB .
VBS21   01== .** CAT -E- VRB, PDRP -E- VA , SDN -E- ABLE .
VBS26   01== .** CAT -E- VRB,PDRP -E- VA-U-VAM,SDN -E- ABLE .
VBS30   01== .** CAT -E- VRB,PDRP -E- VN,SDN -E- ATION .
VBS41   01== .** CAT -E- VRB,PDRP -E- VA,PDRN -E- NVA,SDN -E- ABLE,
VBS41   02         PREF -E- IN .
VBS42   01== .** CAT -E- VRB,PDRP -E- VA,PDRN -E- NVA,SDN -E- IBLE,
VBS42   02      PREF -E- IN .
VBS51   01== .** CAT -E- VRB,PDRP -E- VA,PDRN -E- NVA -U- NVAM,
VBS51   02      SDN -E- ABLE , PREF -E- IN .
VBS61   01== .** CAT -E- VRB,PDRP -E- VA -U- VN,PDRN -E- NVA,
VBS61   02      SDN -E- ABLE -U- ATION,PREF -E- IR .
VS1     01== .**  GNR -E- MAS -U- FEM ,  NBR -E- PLU .
VS2     01== .**  GNR -E- MAS ,  NBR -E- PLU .
VS3     01== .**  GNR -E- FEM ,  NBR -E- SIN .
VS4     01== .**  GNR -E- FEM ,  NBR -E- PLU .
VS5     01== .**  GNR -E- MAS ,  NBR -E- SIN -U- PLU .
VS6     01== .**  GNR -E- MAS ,  NBR -E- SIN .
VS7     01== .**  GNR -E- MAS -U- FEM ,  NBR -E- SIN .
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****


```
MOTINC    :      MODINC   ==  CAT(C) := PRP .
RDICT  : (1,2 / NU ) .
RA1    :  BA2    ==  VAR(C) := VAR (A), DICT(C) := 3 ,GNR(C) := MAS -U- FEM,
        NBR(C) := SIN  , -ARRET-   /  SEG(C) -E- SEG0.
RA3   :     BA3 - BA4    ==
      VAR(C) := VAR(A) , GNR (C) := MAS , NBR(C) := SIN, DICT(C) := 3,
       -ARRET-  /  SEG(C) -E- SEG0.
RA4   :    BA11  ==
     VAR(C) := VAR(A), GNR(C) := MAS, NBR(C) := SIN -U- PLU ,
      DICT(C)  := 3  , -ARRET-    / SEG(C) -E- SEG0 .
RA5  :  BA3 - BA4 - BA6 - BA7 - BA8 - BA9 - BA10 - BA11  ==
      VAREM(C) := VAREM(A), VARNM(C) := VARNM(A) -U- VARNM(C) ,
      DICT(C) := 3  , -ARRET-  /
       SDNA(C) -I- SDNA(A) -NE- SDNA0  -ET- DRV(C) -NE- DRV0
     -ET-  (PDRP(A) -INC- AM -OU- PDRP(A) -INC- AN
       -OU-  ( PDRN(A) -INC- NAM -OU- PDRN (A) -INC- NAN )
              -ET-  SCHAINE (A,0,1) -NE- ' ' ) .
RA6    :       BA2    == VAREM(C) := VAREM(A), VARNM (C)  := VARNM (A) -U-
                  VARNM(C) ,  DICT(C) := 3 , -ARRET-  /
    ( PDRP(A)  -INC- AM -OU- PDRP(A) -INC- AN  -OU- ( PDRN(A) -INC- NAM
 -OU- PDRN(A) -INC- NAN ) -ET- SCHAINE(A,0,1) -NE- ' ' ) -ET- DRV(C) -NE- DRV0 .
RA7   : BA2 - BA3 - BA4 - BA6 - BA7 - BA8 - BA9 - BA10 - BA11  ==
VARM(C) := VARM(A) , VARES(C) := VARES(A) ,
      DICT(C) := 3 , -ARRET- /  SDNA(C) -I- SDNA (A) -NE- SDNA0  -ET-
         SEG(C) -E- D -ET- DRV(C) -E- DRV0 .
RA1B    :  BA2B    ==  VAR(C) := VAR (A), DICT(C) := 3 ,GNR(C) := MAS -U- FEM,
       NBR(C) := SIN        /  SEG(C) -E- SEG0.
RA3B   :     BA3B - BA4B    ==
      VAR(C) := VAR(A) , GNR (C) := MAS , NBR(C) := SIN, DICT(C) := 3 /
             SEG(C) -E- SEG0.
RA4B   :    BA11B  ==
     VAR(C) := VAR(A), GNR(C) := MAS, NBR(C) := SIN -U- PLU ,
      DICT(C)  := 3        / SEG(C) -E- SEG0 .
RA5B  :  BA3B - BA4B - BA6B - BA7B - BA8B - BA9B - BA10B - BA11B ==
      VAREM(C) := VAREM(A), VARNM(C) := VARNM(A) -U- VARNM(C) ,
      DICT(C) := 3       /
       SDNA(C) -I- SDNA(A) -NE- SDNA0  -ET- DRV(C) -NE- DRV0
     -ET-  (PDRP(A) -INC- AM -OU- PDRP(A) -INC- AN
       -OU-  ( PDRN(A) -INC- NAM -OU- PDRN (A) -INC- NAN )
              -ET-  SCHAINE (A,0,1) -NE- ' ' ) .
RA6B    :       BA2B   == VAREM(C) := VAREM(A), VARNM (C)  := VARNM (A) -U-
                       VARNM(C) ,  DICT(C) := 3    /
    ( PDRP(A)  -INC- AM -OU- PDRP(A) -INC- AN  -OU- ( PDRN(A) -INC- NAM
 -OU- PDRN(A) -INC- NAN ) -ET- SCHAINE(A,0,1) -NE- ' ' ) -ET- DRV(C) -NE- DRV0 .
RA7B   :  BA2B - BA3B - BA4B - BA6B - BA7B -  BA8B -  BA9B - BA10B -  BA11B ==
VARM(C) := VARM(A) , VARES(C) := VARES(A) ,
      DICT(C) := 3    /  SDNA(C) -I- SDNA (A) -NE- SDNA0  -ET-
         SEG(C) -E- D -ET- DRV(C) -E- DRV0 .
RS1     :          SM1  ==
       VAR(C) := VAR(A)  / SEG(C) -E- SEG0  /
      TCHAINE(0,'EM','ENTE') , TCHAINE(0,'I','IE') ,
        TCHAINE(0,'E1','E') .
```

```
 RS2 :   SM2 - SM3 - SM4 - SM5 - SM6     ==
      VAR(C) := VAR(A) , NBR(C) := SIN   /
       SEG(C) -E- SEG0   /  TCHAINE(0,'','E')  .
RS21  :      SM3              ==
         VAR(C) := VAR(A) , NBR(C) := SIN   /  SEG(C) -E- SEG0  /
        TCHAINE(0,'AL','ELLE')   .
RS22  :          SM2 - SM3 - SM4 - SM5 -  SM6       ==
        VAR(C) := VAR(A) , NBR(C) := PLU  /
        SEG(C) -E- D -ET- SDNA(C) -E- 2 /
        TCHAINE (0,'','E') .
RS23   :       SM3     ==
        VAR(C) := VAR(A) ,NBR(C) := PLU   /
        SEG(C) -E- D -ET- SDNA(C) -E-2  /
        TCHAINE(0,'AL','ELLE')   .
RS3   :          SM7 - SM8 - SM9 - SM10  ==
        VAR(C) := VAR(A) , NBR(C) := SIN  / SEG(C) -E- SEG0  .
RS31 :       SM7 - SM8 - SM9 - SM10    ==
         VAR(C) := VAR(A) , NBR(C) := PLU /
        SEG(C) -E- D -ET- SDNA(C) -E-2 .
RS4   4:        SM7 - SM8       ==
       SDN(C) := SDN(A) , DRV(C) := VBAJAV  / DRV(C) -E- AJAV  .
RP1  :   PM1 - PM2 - PM3 - PM4   ==
        VAREM (C) := VAREM(A) ,  NEG(C)  := NG   /
        SEG(C) -E- B   -ET-    PREF(C) -E- PREF(A)
        -ET-     (     DRV(C) -E- DRV0  -ET-  PDRN(C) -INC- N
           -OU-   DRV(C) -E- AJAV  -ET- PDRN (C) -INC- NAM
       -OU-   DRV(C) -E- AJNM  -ET- PDRN(C) -INC- NAN
      -OU-   DRV(C) -E- VBAJ  -ET- PDRN(C) -INC- NVA
       -OU-   DRV(C) -E- VBAJAV -ET- PDRN(C) -INC- NVAM ) .
RN1 :  BN1 - BA1 == VAR(C) := VAR(A) , DICT(C) := 3 , -ARRET- / SEG(C) -E-SEG0 .
RN2   :     BN2 - BN5   ==  -ARRET- ,
        VAR(C) := VAR(A) , NBR(C) := SIN , DICT(C) := 3 / SEG(C) -E- SEG0 .
RN3   :     BN2 - BN5    ==  -ARRET- ,
  VARM(C) := VARM(A) , VARES(C) := VARES(A), GNR(C) := GNR(A) , DICT(C) := 3 /
        SDNA(C) -I- SDNA(A) -NE- SDNA0  -ET-  SEG(C) -E- D .
RN4   :   BN3 - BN4      ==  -ARRET- ,
        VAR(C) := VAR(A) , GNR(C) := MAS , NBR(C) := SIN , DICT(C) := 3  /
SEG(C) -E- SEG0 .
RN5   :     BN3 - BN4 - BN6 - BN7 - BN8 - BN9 - BN10 ==  -ARRET- ,
        VARM(C) := VARM(A) ,VARES(C) := VARES(A) ,
        VARNS (C) := VARNS(A) -U- VARNS(C)  , DICT(C) := 3  /
        SDNA(C) -I- SDNA(A)  -NE-  SDNA0  -ET- SEG(C) -E-  D  .
RN1B  : BN1B - BA1B   ==  VAR(C) := VAR(A) , DICT(C) := 3   /
                SEG(C) -E- SEG0 .
RN2B  :   BN2B - BN5B   ==
        VAR(C) := VAR(A) , NBR(C) := SIN , DICT(C) := 3 / SEG(C) -E- SEG0 .
RN3B  :   BN2B - BN5B   ==
  VARM(C) := VARM(A) , VARES(C) := VARES(A), GNR(C) := GNR(A) , DICT(C) := 3 /
        SDNA(C) -I- SDNA(A) -NE- SDNA0  -ET-  SEG(C) -E- D .
 RN4B  :   BN3B - BN4B  ==
        VAR(C) := VAR(A) , GNR(C) := MAS , NBR(C) := SIN , DICT(C) := 3  /
SEG(C) -E- SEG0 .
```

```
RN5B  :  BN3B - BN4B - BN6B - BN7B - BN8B - BN9B - BN10B ==
        VARM(C) := VARM(A) ,VARES(C) := VARES(A) ,
        VARNS (C) := VARNS(A) -U- VARNS(C)  , DICT(C) := 3  /
        SDNA(C) -I- SDNA(A)  -NE-  SDNA0  -ET- SEG(C) -E-  D  .
RD1   :  D0 - D1 - D4 - D5 - D6 - D7 - D8 - D9  ==
     VAR(C) := VAR(A) / SEG(C) -E- SEG0 .
RD2   :   D3          ==
             VAR(C) := VAR(A)    /  SEG(C) -E- SEG0 /
     TCHAINE(0,'TT','T') , TCHAINE(0,'NN','N') , TCHAINE(0,'LL','L') ,
       TCHAINE(0,'SS','S')  .
RD3   :     D5 - D6 - D7 - D8 - D9 ==
      VAREM(C) := VAREM(A) , VARNM(C) := VARNM(C) -U- VARNM(A) /
SEG(C) -E- S -ET-  (DRV (C) -E- AJAV -OU- DRV(C) -E- AJNM )
 -ET- GNR(A) -E- FEM
        -ET- NBR(A) -E- SIN .
RD31  :       D3  ==
     VAREM(C) := VAREM(A) ,VARNM(C) := VARNM(A) -U- VARNM(C)  /
     GNR(A) -E- FEM -ET- NBR(A) -E-SIN  -ET-
     SEG(C) -E- S  -ET- (DRV(C) -E- AJAV -OU- DRV(C) -E- AJNM)  /
     TCHAINE(0,'LL','L') , TCHAINE(0,'NN','N') , TCHAINE(0,'SS','S') ,
     TCHAINE(0,'TT','T')  .
RV1    :   VBM    ==  VARM(C) := VARM(A) , DICT(C) := 3 , -ARRET-  /
       SEG(C)  -E-  S     -ET-
       ( PDRP(A) -INC- VA  -ET- DRV(C) -E- VBAJ
        -OU-  PDRP(A) -INC- VAM  -ET- DRV(C) -E- VBAJAV
       -OU-   PDRP(A) -INC- VN  -ET- DRV(C) -E- VBNM
       -OU- PDRN(A) -INC- NVA  -ET- DRV(C) -E- VBAJ  -ET- SCHAINE
                          (A,0,1)  -NE- ' '
       -OU- PDRN(A) -INC- NVAM -ET- DRV(C) -E- VBAJAV  -ET-
         SCHAINE (A,0 ,1) -NE-  ' '  )  .
RV1B  :  VBMB  == VARM(C) := VARM(A) , DICT(C) := 3  /
       SEG(C)  -E-  S     -ET-
       ( PDRP(A) -INC- VA  -ET- DRV(C) -E- VBAJ
        -OU-  PDRP(A) -INC- VAM  -ET- DRV(C) -E- VBAJAV
       -OU-   PDRP(A) -INC- VN  -ET- DRV(C) -E- VBNM
       -OU- PDRN(A) -INC- NVA  -ET- DRV(C) -E- VBAJ  -ET- SCHAINE
                          (A,0,1)  -NE- ' '
       -OU- PDRN(A) -INC- NVAM -ET- DRV(C) -E- VBAJAV  -ET-
         SCHAINE (A,0 ,1) -NE-  ' '  )  .
RARTP:  INVAR  ==  VAR(C):=VAR(A) , -FINAL- / VAR(C)-E- VAR0 -ET-
                     SCHAINE(A,0,1) -E- ' '.
RPHR:  PONCT  ==  VAR(C):=VAR(A) ,-INIT- , -FINAL- / VAR(C) -E- VAR0 -ET-
                    SCHAINE(A,0,1) -E- ' '.
- FIN-
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 DICAFF1 18 MARS 1974 17H 38MN 05S

```
ABLE                ==SM7     (SS3).
AL                  ==D8      (VS6).
ALE                 ==D8      (VS3).
ALES                ==D8      (VS4).
ATION               ==SM9     (SS4).
AUX                 ==D8      (VS2).
E                   ==D3      (VS3).
ES                  ==D3      (VS4).
ESSE                ==SM6     (SS2).
ETE1                ==SM4     (SS2).
EUR                 ==D6      (VS6).
EUR                 ==SM5     (SS2).
EURS                ==D6      (VS2).
EUSE                ==D6      (VS3).
EUSES               ==D6      (VS3).
F                   ==D9      (VS6).
FS                  ==D9      (VS2).
IBLE                ==SM8     (SS3).
ITE1                ==SM3     (SS2).
ITION               ==SM10    (SS4     ).
MENT                ==SM1     (SS1).
S                   ==D0      (VS2).
S                   ==D1      (VS1).
SE                  ==D5      (VS3).
SES                 ==D5      (VS4).
TEUR                ==D7      (VS6).
TEURS               ==D7      (VS2).
TE1                 ==SM2     (SS2).
TRICE               ==D7      (VS3).
TRICES              ==D7      (VS4).
VE                  ==D9      (VS3).
VES                 ==D9      (VS4).
X                   ==D4      (VS1).
X                   ==D5      (VS5).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 DICBAS2 — 18 MARS 1974 — PAGE 1 — 17H 40MN 55S

```
.                ==PONCT   (POINT   ,POINT              ).
AC               ==BN8     (SN5H    ,ACTEUR             ).
ACTI             ==BA10    (SA531   ,ACTIF              ).
AIM              ==VBM     (VBS26   ,AIMER              ).
CALCUL           ==VBM     (VBS41   ,CALCULER           ).
CANDIDAT         ==BN3     (SN5H     ,CANDIDAT           ).
CHAT             ==BN4     (SN5A     ,CHAT               ).
CHEV             ==BN9     (SN3A     ,CHEVAL             ).
CRAINTI          ==BA10    (SA33    ,CRAINTIF            ).
CRE1             ==VBM     (VBS30   ,CRE1ER              ).
D'               ==INVAR   (SPR1    ,DE                 ).
DE               ==INVAR   (SPR1    ,DE                 ).
DIVIS            ==VBM     (VBS42   ,DIVISER             ).
E1POU            ==BN6     (SN5H    ,E1POUX              ).
FEU              ==BN5     (SN3A    ,FEU                 ).
GAZ              ==BN1     (SN1     ,GAZ                 ).
GE1NE1R          ==BA9     (SA33    ,GE1NE1RAL           ).
GE1NE1RA         ==BA8     (SA1     ,GE1NE1RATEUR        ).
GRIS             ==BA11    (SA1C     ,GRIS               ).
GROS             ==BA11    (SA31    ,GROS                ).
HABITUEL         ==BA4     (SA401   ,HABITUEL            ).
JEUNE            ==BA2     (SA22    ,JEUNE               ).
JOYEU            ==BA6     (SA2     ,JOYEUX              ).
MAISON           ==BN2     (SN4     ,MAISON              ).
MUR              ==BN2     (SN3S     ,MUR                ).
ORANGE           ==BA1     (SA1C    ,ORANGE              ).
ORANGE           ==BN2     (SN4     ,ORANGE              ).
PEAU             ==BN5     (SN4     ,PEAU                ).
PERDRIX          ==BN1     (SN2     ,PERDRIX             ).
RAM              ==VBM     (VBS1    ,RAMER               ).
RARE             ==BA2     (SA34    ,RARE                ).
REPOS            ==BN1     (SN1A    ,REPOS               ).
RE1CENT          ==BA3     (SA2     ,RE1CENT             ).
RE1EL            ==BA4     (SA534   ,RE1EL               ).
RE1PAR           ==VBM     (VBS61   ,RE1PARER            ).
ROND             ==BA3     (SA31    ,ROND                ).
SERV             ==BN7     (SN5H    ,SERVEUR             ).
SOUHAIT          ==VBM     (VBS21   ,SOUHAITER           ).
TERMIN           ==VBM     (VBS51   ,TERMINER            ).
UN               ==INVAR   (SAR1    ,UN                  ).
UNE              ==INVAR   (SAR2    ,UN                  ).
UTILE            ==BA2     (SA531   ,UTILE               ).
VEU              ==BN10    (SN5H    ,VEUF                ).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***
****

FICHIER : FILEMFR1 DICAFF3 — 18 MARS 1974 — PAGE 1 — 17H 45MN 55S

```
IL               ==PM3     (PS      ).
IM               ==PM2     (PS      ).
IN               ==PM1     (PS      ).
IR               ==PM4     (PS      ).
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 1 CODE LANGUE : FR1

```
** OCCURRENCE : IRRE1ELLEMENT
MORPHE :  MENT                          REGLE : RS1      FORMAT : SM1      SS1
      ETAT COURANT : K0(UL0,S,/,SDN(MENT),$,ADV,AJAV,/)
RESTE : IRRE1ELLE
MORPHE :  E                             REGLE : RD2      FORMAT : D3       VS3
MORPHE :  E                             REGLE : RD31     FORMAT : D3       VS3
      ETAT COURANT : K0(UL0,D,/,SDNA(3,4,11),SDN(MENT),$,ADV,AJAV,/)
RESTE : IRRE1EL
MORPHE :  RE1EL                         REGLE : RA3      FORMAT : BA4      SA534
MORPHE :  RE1EL                         REGLE : RA5      FORMAT : BA4      SA534
      ETAT COURANT : K0(RE1EL,B,/,PREF(IR),SDNA(3,4,11),SDN(MENT,ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN),$,ADV,AJAV,/)
RESTE : IR
MORPHE :  IR                            REGLE : RP1      FORMAT : PM4      PS
  -->  DECOUPAGE : IR RE1EL E MENT   K0(RE1EL,P,/,PREF(IR),SDNA(3,4,11),SDN(MENT,ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN),$,ADV,AJAV,
NG,/

MORPHE :  RE1EL                         REGLE : RA7      FORMAT : BA4      SA534

** OCCURRENCE : RE1ALITE1
MORPHE :  ITE1                          REGLE : RS2      FORMAT : SM3      SS2
      ETAT COURANT : K0(UL0,S,/,SDNA(2),SDN(ITE1),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))
RESTE : RE1ALE
MORPHE :  ALE                           REGLE : RD1      FORMAT : D8       VS3
MORPHE :  ALE                           REGLE : RD3      FORMAT : D8       VS3
      ETAT COURANT : K0(UL0,D,/,SDNA(2,9),SDN(ITE1),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))
RESTE : RE1
MORPHE :  E                             REGLE : RD2      FORMAT : D3       VS3
MORPHE :  E                             REGLE : RD31     FORMAT : D3       VS3
      ETAT COURANT : K0(UL0,D,/,SDNA(2,3,4,11),SDN(ITE1),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))
RESTE : RE1AL
MORPHE :  AL                            REGLE : RD1      FORMAT : D8       VS6
MORPHE :  AL                            REGLE : RD3      FORMAT : D8       VS6
MORPHE :  ITE1                          REGLE : RS21     FORMAT : SM3      SS2
      ETAT COURANT : K0(UL0,S,/,SDNA(2),SDN(ITE1),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))
RESTE : RE1ELLE
MORPHE :  E                             REGLE : RD2      FORMAT : D3       VS3
MORPHE :  E                             REGLE : RD31     FORMAT : D3       VS3
      ETAT COURANT : K0(UL0,D,/,SDNA(2,3,4,11),SDN(ITE1),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))
RESTE : RE1EL
MORPHE :  RE1EL                         REGLE : RA3      FORMAT : BA4      SA534
MORPHE :  RE1EL                         REGLE : RA5      FORMAT : BA4      SA534
  -->  DECOUPAGE : RE1EL E ITE1   K0(RE1EL,B,/,PREF(IR),SDNA(2,3,4,11),SDN(ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN),$,NMC,AJNM,/,GNR(
FEM)
NBR(SIN))
MORPHE :  RE1EL                         REGLE : RA7      FORMAT : BA4      SA534

** OCCURRENCE : CHEVAUX
MORPHE :  AUX                           REGLE : RD1      FORMAT : D8       VS2
      ETAT COURANT : K0(UL0,D,/,SDNA(9),$,/,GNR(MAS),NBR(PLU))
```

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 1 CODE LANGUE : FR1

```
RESTE : CHEV
MORPHE :  CHEV                          REGLE : RN5      FORMAT : BN9      SN3A
  -->  DECOUPAGE : CHEV AUX   K0(CHEVAL,B,/,SDNA(9),$,NMC,/,GNR(MAS),NBR(PLU),SEMA(CONCR,ANIM))

** OCCURRENCE : ORANGE
MORPHE :  ORANGE                        REGLE : RN2      FORMAT : BN2      SN4
  -->  DECOUPAGE : ORANGE   K0(ORANGE,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))
MORPHE :  ORANGE                        REGLE : RN3      FORMAT : BN2      SN4
MORPHE :  ORANGE                        REGLE : RN1      FORMAT : BA1      SA1C
  -->  DECOUPAGE : ORANGE   K0(ORANGE,B,/,SDNA(1),$,AJQ,/,GNR(MAS,FEM),NBR(SIN,PLU),SEMA(ABSTR,QUALI))



-01-K0(RE1EL,P,/,PREF(IR),SDNA(3,4,11),SDN(MENT,ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN
),$,ADV,AJAV,NG,/)+K0(RE1EL,B,/,PREF(IR),SDNA(2,3,4,11),SDN(ITE1),PDRP(AM,AN),PD
RN(N,NAM,NAN),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))+K0(CHEVAL,B,/,SDNA(9),$,NMC,/,GNR(
MAS),NBR(PLU),SEMA(CONCR,ANIM))+K0(ORANGE,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))
-02-

-01-K0(RE1EL,P,/,PREF(IR),SDNA(3,4,11),SDN(MENT,ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN
),$,ADV,AJAV,NG,/)+K0(RE1EL,B,/,PREF(IR),SDNA(2,3,4,11),SDN(ITE1),PDRP(AM,AN),PD
RN(N,NAM,NAN),$,NMC,AJNM,/,GNR(FEM),NBR(SIN))+K0(CHEVAL,B,/,SDNA(9),$,NMC,/,GNR(
MAS),NBR(PLU),SEMA(CONCR,ANIM))+K0(ORANGE,B,/,SDNA(1),$,AJQ,/,GNR(MAS,FEM),NBR(S
IN,PLU),SEMA(ABSTR,QUALI))-02-

-01-02-
```

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : FR1

```
** OCCURRENCE : UN
  -->  DECOUPAGE : UN    K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))

** OCCURRENCE : JEUNE
  -->  DECOUPAGE : JEUNE    K0(JEUNE,B,/,SDNA(2),SDN(ESSE),PDRP(AN),$,AJQ,/,GNR(MAS,FEM),NBR(SIN))

** OCCURRENCE : CHEVAL
  -->  DECOUPAGE : CHEV AL    K0(CHEVAL,B,/,SDNA(9),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM))

** OCCURRENCE : CRAINTIF
  -->  DECOUPAGE : CRAINTI F    K0(CRAINTIF,B,/,SDNA(10),SDN(ITE1),PDRP(AM,AN),$,AJQ,/,GNR(MAS),NBR(SIN))

** OCCURRENCE : .
  -->  DECOUPAGE : .    K0(POINT,B,/,$,PCT,/)

** OCCURRENCE : UN
  -->  DECOUPAGE : UN    K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))

** OCCURRENCE : MUR
  -->  DECOUPAGE : MUR    K0(MUR,B,/,SDNA(2),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,INAM))

** OCCURRENCE : D'
  -->  DECOUPAGE : D'    K0(DE,B,/,$,PRP,/,NBR(SIN))

** OCCURRENCE : UNE
  -->  DECOUPAGE : UNE    K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : RE1CENTE
  -->  DECOUPAGE : RE1CENT E    K0(RE1CENT,B,/,SDNA(3),PDRP(AM),$,AJQ,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : MAISON
  -->  DECOUPAGE : MAISON    K0(MAISON,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : DE
  -->  DECOUPAGE : DE    K0(DE,B,/,$,PRP,/,NBR(SIN))

** OCCURRENCE : GRENOBLE
MOT INCONNU
  -->  DECOUPAGE :GRENOBLE    K0(GRENOBLE,/,$,PRP,/)

** OCCURRENCE : .
  -->  DECOUPAGE : .    K0(POINT,B,/,$,PCT,/)

** OCCURRENCE : UN
  -->  DECOUPAGE : UN    K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))

** OCCURRENCE : CHAT
```

SYSTEME A.T.E.F.

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : FR1

--> DECOUPAGE : CHAT K0(CHAT,B,/,SDNA(4),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM))

** OCCURRENCE : GRIS
--> DECOUPAGE : GRIS K0(GRIS,B,/,SDNA(11),$,AJQ,/,GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR,QUALI))

** OCCURRENCE : D'
--> DECOUPAGE : D' K0(DE,B,/,$,PRP,/,NBR(SIN))

** OCCURRENCE : UNE
--> DECOUPAGE : UNE K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : IRRE1ELLE
--> DECOUPAGE : IR RE1EL E K0(RE1EL,P,/,PREF(IR),SDNA(4),SDN(ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN),$,AJQ,NG,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : MAISON
--> DECOUPAGE : MAISON K0(MAISON,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : DE
--> DECOUPAGE : DE K0(DE,B,/,$,PRP,/,NBR(SIN))

** OCCURRENCE : REPOS
--> DECOUPAGE : REPOS K0(REPOS,B,/,SDNA(1),$,NMC,/,GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR))

** OCCURRENCE : D'
--> DECOUPAGE : D' K0(DE,B,/,$,PRP,/,NBR(SIN))

** OCCURRENCE : UNE
--> DECOUPAGE : UNE K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))

** OCCURRENCE : ACTRICE
--> DECOUPAGE : AC TRICE K0(ACTEUR,B,/,SDNA(8),$,NMC,/,GNR(FEM),NBR(SIN),SEMA(HUM,CONCR))

** OCCURRENCE : .
--> DECOUPAGE : . K0(POINT,B,/,$,PCT,/)

-01-K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))+K0(JEUNE,B,/,SDNA(2),SDN(ESSE),PDRP(AN),$,AJQ,/,GNR(MAS,FEM),NBR(SIN))+K0(CHEVAL,B,/,SDNA(9),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM))+K0(CRAINTIF,B,/,SDNA(10),SDN(ITE1),PDRP(AM,AN),$,AJQ,/,GNR(MAS),NBR(SIN))+K0(POINT,B,/,$,PCT,/)-02-/

-01-K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))+K0(MUR,B,/,SDNA(2),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,INAM))+K0(DE,B,/,$,PRP,/,NBR(SIN))+K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))+K0(RE1CENT,B,/,SDNA(3),PDRP(AM),$,AJQ,/,GNR(FEM),NBR(SIN))+K0(MAISON,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))+K0(DE,B,/,$,PRP,/,NBR(SIN))+K0(GRENOBLE,/,$,PRP,/)+K0(POINT,B,/,$,PCT,/)-02-/

-01-K0(UN,B,/,$,ART,/,GNR(MAS),NBR(SIN))+K0(CHAT,B,/,SDNA(4),$,NMC,/,GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM))+K0(GRIS,B,/,SDNA(11),$,AJQ,/,GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR,QUALI))+K0(DE,B,/,$,PRP,/,NBR(SIN))+K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))+K0(RE1EL,P,/,PREF(IR),SDNA(4),SDN(ITE1),PDRP(AM,AN),PDRN(N,NAM,NAN),$,AJQ,NG,/,GNR(FEM),NBR(SIN))+K0(MAISON,B,/,SDNA(2),$,NMC,/,GNR(FEM),NBR(SIN))+K0(DE,B,/,$,PRP,/,NBR(SIN))+K0(REPOS,B,/,SDNA(1),$,NMC,/,GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR))+K0(DE,B,/,$,PRP,/,NBR(SIN))+K0(UN,B,/,$,ART,/,GNR(FEM),NBR(SIN))+K0(ACTEUR,B,/,SDNA(8),$,NMC,/,GNR(FEM),NBR(SIN),SEMA(HUM,CONCR))+K0(POINT,B,/,$,PCT,/)-02-

B - APPLICATIONS DU SYSTEME C. E. T. A.

Le système C.E.T.A. permet de définir une grammaire transformationnelle. La plupart des traitements informatiques non numériques utilisent des grammaires transformationnelles. Le langage LISP est certainement le plus employé pour la réalisation de systèmes utilisant des grammaires transformationnelles.
Ce langage est un langage de programmation et nécessite la définition d'un algorithme. Le système C.E.T.A. contient lui-même un algorithme autonome et l'utilisateur ne peut qu'influencer cet algorithme. Il permet la définition de grammaires transformationnelles telles que celles employées pour le traitement de langues naturelles ou le traitement formel des expressions algébriques. L'entrée et la sortie de ce système sont constituées par des arborescences. Une entrée privilégiée de ce système est donnée par la sortie du système A.T.E.F., l'arborescence de sortie de ce système étant directement donnée sous une forme convenable. Les deux analyses présentées ici utilisent la sortie du système A.T.E.F.

1 - ANALYSE D'UN LANGAGE FORMEL

La première analyse présentée a pour but de montrer un aperçu des possibilités du système tout en étant relativement simple. Cette première grammaire reconnaît le langage formé par la réunion des trois langages suivants :

$$L_1 = \{a^n\ b^n\ c^n \mid n \geq 1\}$$

$$L_2 = \{w\ c\ w \mid w \in \{a,b\}^*\}$$

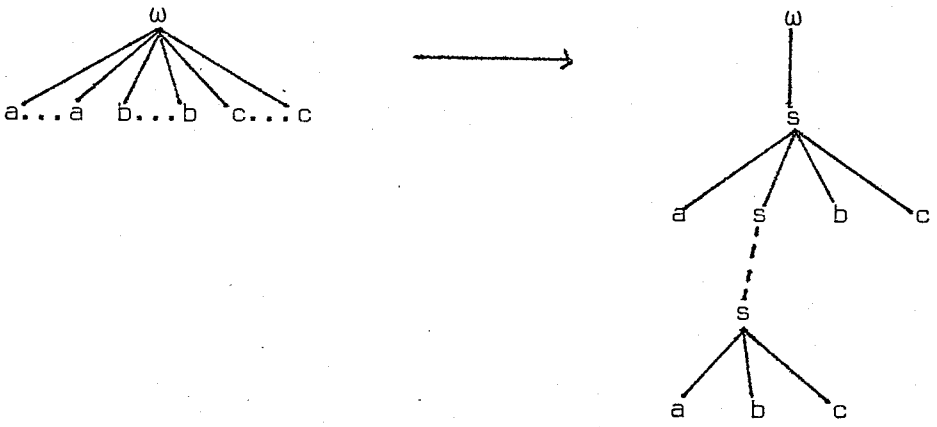
$$L_3 = \{a^n\ b^n \mid n \geq 1\}$$

La grammaire C.E.T.A. construite sera composé de quatre grammaires élémentaires :

- la grammaire GRAMA reconnaît le langage L_1
- la grammaire GRAMB reconnaît le langage L_2
- la grammaire GRAMC reconnaît le langage L_3
- la grammaire GRAMI initialise la recherche et oriente le système vers une des trois grammaires précédentes.

* Analyse du langage L_1 :

Tout mot de L_1 sera sous forme d'une arborescence à un niveau à l'entrée du système. Seuls les mots de L_1, L_2 ou L_3 conduirons à une arborescence de plus de un niveau en sortie du système. Pour les mots du langage L_1, l'arborescence résultante sera de la forme suivante :



Seuls les mots de L_1 conduirons à cette structure. Il est a remarqué que le langage des feuilles de cette structure est le langage $L'_1 = \{a(bc)^n \mid n \geq 1\}$ Les mots de ce langage n'appartenant pas à L_1, L_2 ou L_3 ne seront pas reconnus par les grammaires définies (texte 3).

Ainsi, la reconnaissance du langage $\{a^n b^n c^n \mid n \geq 1\}$ ne s'effectue pas suivant une analyse habituelle, mais avec des transformations d'arborescences dont les mots des feuilles n'ont plus aucun rapport avec l'arborescence initiale. La seule contrainte imposée est que seuls les mots du langage à reconnaître conduisent à la forme prédéterminée.

Les deux analyses données ici concernent les mots suivants :

$a^3 b^3 c^3$ pour le texte n° 2

$a^3(bc)^3$ pour le texte n° 3

```
                              -PROC-
     PCP: A ==  ETQ -E- A.
     PCP: B ==  ETQ -E- B.
     PCP: C ==  ETQ -E- C.
     PCP: S ==  ETQ -E- S.
     PAF: FS ==  ETQ := S.
                              -PSCHEM-
        FIN: OM(*,S,*) / S: $S /.
                              -GRAM-
       GRAMI(U): RA01 , RB01 , RC01 ;
                    S(A,B,C)/S:$S;A:$A;B:$B;C:$C/<-- GRAMA;
                    S(C)/S:$S;C:$C/<-- GRAMB ;
                    S(A,B)/S:$S;A:$A;B:$B/<-- GRAMC .
          GRAMA(U): RA02(GRAMA/RA02/OM(S1));
                          OM(*,S,*)/S:$S/<--&NUL.
          GRAMB(U): RB02(GRAMB/RB02/OM(S1));
                          OM(*,S,*)/S:$S/<--&NUL.
          GRAMC(U): RC02(GRAMC/RC02/OM(S1)); $FIN<--&NUL .
                              -REGLES-
     RA01: OM(A,*,B,C,*)/A:$A;B:$B;C:$C/ ==
               OM(S(A,B,C))/S<--A,B,C;A<--*;B<--*;C<--*/S:*FT1,$FS.
     RA02: OM(A,*,S,*,B,C,*)/A:$A;S:$S;B:$B;C:$C/==
               OM(S1(A,S,B,C))/S1<--A,B,C;A<--*;B<--*;C<--*/S1:*FT1,$FS.
     RB01: OM(C)/C:$C/ == OM(S(C))/S<--C;C<--*/S:*FT1,$FS.
     RB02: OM(*,X1,$L1,S,*,X2,$L2)/X1:ETQ-NE-C;S:$S/ETQ(X1)-E-ETQ(X2)==
            OM($L1,S1(X1,S,X2),$L2)/S1<--X1,X2;X2<--*;X1<--*/S1:*FT1,$FS.
     RC01:  OM(A,*,B)/A:$A;B:$B/== OM(S(A,B))/S<--A,B;B<--*;A<--*/S:*FT1,$FS .
     RC02:  OM(A,*,S,*,B)/A:$A;B:$B;S:$S/==OM(S1(A;S;B))/S1<--A,B;B<--*;A<--*/
                         S1:*FT1,$FS.
                              -FIN-
```

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME C.E.T.A.

DETAIL DE L'EXECUTION , TEXTE : 2 CODE LANGUE : GD

GRAMMAIRE : GRAM1 (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6.A , .7.B , .8.B , .9.B , .10.C , .11.C , .12.C)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 A: UL(A),ETQ(A).

SOMMET 7 B: UL(B),ETQ(B).

SOMMET 8 B: UL(B),ETQ(B).

SOMMET 9 B: UL(B),ETQ(B).

SOMMET 10 C: UL(C),ETQ(C).

SOMMET 11 C: UL(C),ETQ(C).

SOMMET 12 C: UL(C),ETQ(C).

REGLES : RA01 *.3. (.6.A,.7.B,.12.C)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.A , .5.A , .6. (.7.A , .8.B , .9.C) , .10.B , .11.B , .12.C , .13.C)))

GRAMMAIRE : GRAMA (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6. (.7.A , .8.B , .9.C) , .10.B , .11.B , .12.C , .13.C)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).
SOMMET 6 : UL(ULO),ETQ(S).
SOMMET 7 A: UL(A),ETQ(A).
SOMMET 8 B: UL(B),ETQ(B).
SOMMET 9 C: UL(C),ETQ(C).
SOMMET 10 B: UL(B),ETQ(B).
SOMMET 11 B: UL(B),ETQ(B).
SOMMET 12 C: UL(C),ETQ(C).
SOMMET 13 C: UL(C),ETQ(C).

REGLES : RA02 *.3. (.5.A,.6. ,.10.B,.13.C)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.A , .5. (.6.A , .7. (.8.A , .9.B , .10.C) , .11.B , .12.C) , .13.B , .14.C)))

APPEL RECURSIF , REGLE : RA02

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.A , .3. (.4.A , .5. (.6.A , .7.B , .8.C) , .9.B , .10.C) , .11.B , .12.C)
SOMMET 1 : UL(ULSOL).
SOMMET 2 A: UL(A),ETQ(A).
SOMMET 3 : UL(ULO),ETQ(S).
SOMMET 4 A: UL(A),ETQ(A).
SOMMET 5 : UL(ULO),ETQ(S).
SOMMET 6 A: UL(A),ETQ(A).
SOMMET 7 B: UL(B),ETQ(B).
SOMMET 8 C: UL(C),ETQ(C).
SOMMET 9 B: UL(B),ETQ(B).
SOMMET 10 C: UL(C),ETQ(C).

SOMMET 11 B: UL(B),ETQ(B).

SOMMET 12 C: UL(C),ETQ(C).

REGLES : RA02 *.1. (.2.A,.3. ,.11.B,.12.C)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3.A , .4. (.5.A , .6. (.7.A , .8.B , .9.C) , .10.B , .11.C) , .12.B , .13.C))

APPEL RECURSIF , REGLE : RA02

SOUS-ARBORESCENCE D'ENTREE

.1. (.2. (.3.A , .4. (.5.A , .6. (.7.A , .8.B , .9.C) , .10.B , .11.C) , .12.B , .13.C))

SOMMET 1 : UL(ULSOL).

SOMMET 2 : UL(UL0),ETQ(S).

SOMMET 3 A: UL(A),ETQ(A).

SOMMET 4 : UL(UL0),ETQ(S).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 : UL(UL0),ETQ(S).

SOMMET 7 A: UL(A),ETQ(A).

SOMMET 8 B: UL(B),ETQ(B).

SOMMET 9 C: UL(C),ETQ(C).

SOMMET 10 B: UL(B),ETQ(B).

SOMMET 11 C: UL(C),ETQ(C).

SOMMET 12 B: UL(B),ETQ(B).

SOMMET 13 C: UL(C),ETQ(C).

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3.A , .4. (.5.A , .6. (.7.A , .8.B , .9.C) , .10.B , .11.C) , .12.B , .13.C))

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4. (.5.A , .6. (.7.A , .8. (.9.A , .10.B , .11.C) , .12.B , .13.C) , .14.B , .15.C))))

DETAIL DE L'EXECUTION , TEXTE : 2 SYSTEME C.E.T.A. CODE LANGUE : GD

RESULTAT :

.1. (.2. (.3. (.4. (.5.A , .6. (.7.A , .8. (.9.A , .10.B , .11.C) , .12.B , .13.C) , .14.B , .15.C))))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 : UL(UL0),ETQ(S).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 : UL(UL0),ETQ(S).

SOMMET 7 A: UL(A),ETQ(A).

SOMMET 8 : UL(UL0),ETQ(S).

SOMMET 9 A: UL(A),ETQ(A).

SOMMET 10 B: UL(B),ETQ(B).

SOMMET 11 C: UL(C),ETQ(C).

SOMMET 12 B: UL(B),ETQ(B).

SOMMET 13 C: UL(C),ETQ(C).

SOMMET 14 B: UL(B),ETQ(B).

SOMMET 15 C: UL(C),ETQ(C).

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME C.E.T.A.

RESULTAT DE L'EXECUTION , TEXTE :2

CODE LANGUE GD

.1. (.2. (.3. (.4. (.5.A , .6. (.7.A , .8. (.9.A , .10.B , .11.C) , .12.B , .13.C) , .14.B , .15.C))))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 : UL(UL0),ETQ(S).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 : UL(UL0),ETQ(S).

SOMMET 7 A: UL(A),ETQ(A).

SOMMET 8 : UL(UL0),ETQ(S).

SOMMET 9 A: UL(A),ETQ(A).

SOMMET 10 B: UL(B),ETQ(B).

SOMMET 11 C: UL(C),ETQ(C).

SOMMET 12 B: UL(B),ETQ(B).

SOMMET 13 C: UL(C),ETQ(C).

SOMMET 14 B: UL(B),ETQ(B).

SOMMET 15 C: UL(C),ETQ(C).

DETAIL DE L'EXECUTION , TEXTE : 3 CODE LANGUE : GD

GRAMMAIRE : GRAMI (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6.A , .7.B , .8.C , .9.B , .10.C , .11.B , .12.C)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 A: UL(A),ETQ(A).

SOMMET 7 B: UL(B),ETQ(B).

SOMMET 8 C: UL(C),ETQ(C).

SOMMET 9 B: UL(B),ETQ(B).

SOMMET 10 C: UL(C),ETQ(C).

SOMMET 11 B: UL(B),ETQ(B).

SOMMET 12 C: UL(C),ETQ(C).

REGLES : RA01 *.3. (.6.A,.7.B,.12.C)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.A , .5.A , .6. (.7.A , .8.B , .9.C) , .10.C , .11.B , .12.C , .13.B)))

GRAMMAIRE : GRAMA (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6. (.7.A , .8.B , .9.C) , .10.C , .11.B , .12.C , .13.B)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).
SOMMET 6 : UL(UL0),ETQ(S).
SOMMET 7 A: UL(A),ETQ(A).
SOMMET 8 B: UL(B),ETQ(B).
SOMMET 9 C: UL(C),ETQ(C).
SOMMET 10 C: UL(C),ETQ(C).
SOMMET 11 B: UL(B),ETQ(B).
SOMMET 12 C: UL(C),ETQ(C).
SOMMET 13 B: UL(B),ETQ(B).

RESULTAT :

.1. (.2. (.3. (.4.A , .5.A , .6.A , .7.B , .8.C , .9.B , .10.C , .11.B , .12.C)))
SOMMET 1 : UL(ULTXT).
SOMMET 2 : UL(ULFRA).
SOMMET 3 : UL(ULSOL).
SOMMET 4 A: UL(A),ETQ(A).
SOMMET 5 A: UL(A),ETQ(A).
SOMMET 6 A: UL(A),ETQ(A).
SOMMET 7 B: UL(B),ETQ(B).
SOMMET 8 C: UL(C),ETQ(C).
SOMMET 9 B: UL(B),ETQ(B).
SOMMET 10 C: UL(C),ETQ(C).
SOMMET 11 B: UL(B),ETQ(B).
SOMMET 12 C: UL(C),ETQ(C).

*** GROUPE D'ETUDE POUR LA TRADUCTION AUTOMATIQUE ***

SYSTEME C.E.T.A.

RESULTAT DE L'EXECUTION , TEXTE :3

CODE LANGUE GD

```
.1.  (.2.  (.3.  (.4.A , .5.A , .6.A , .7.B , .8.C , .9.B , .10.C , .11.B , .12.C)))
     SOMMET 1    : UL(ULTXT).
     SOMMET 2    : UL(ULFRA).
     SOMMET 3    : UL(ULSOL).
     SOMMET 4   A: UL(A),ETQ(A).
     SOMMET 5   A: UL(A),ETQ(A).
     SOMMET 6   A: UL(A),ETQ(A).
     SOMMET 7   B: UL(B),ETQ(B).
     SOMMET 8   C: UL(C),ETQ(C).
     SOMMET 9   B: UL(B),ETQ(B).
     SOMMET 10  C: UL(C),ETQ(C).
     SOMMET 11  B: UL(B),ETQ(B).
     SOMMET 12  C: UL(C),ETQ(C).
```

DETAIL DE L'EXECUTION , TEXTE : 4 CODE LANGUE : GD

GRAMMAIRE : GRAM1 (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6.B , .7.C , .8.A , .9.A , .10.B)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 B: UL(B),ETQ(B).

SOMMET 7 C: UL(C),ETQ(C).

SOMMET 8 A: UL(A),ETQ(A).

SOMMET 9 A: UL(A),ETQ(A).

SOMMET 10 B: UL(B),ETQ(B).

REGLES : RB01 *.3. (.7.C)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.A , .5.A , .6.B , .7. (.8.C) , .9.A , .10.A , .11.B)))

GRAMMAIRE : GRAMB (U)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.A , .5.A , .6.B , .7. (.8.C) , .9.A , .10.A , .11.B)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 A: UL(A),ETQ(A).

SOMMET 6 B: UL(B),ETQ(B).

SOMMET 7 : UL(UL0),ETQ(S).
SOMMET 8 C: UL(C),ETQ(C).
SOMMET 9 A: UL(A),ETQ(A).
SOMMET 10 A: UL(A),ETQ(A).
SOMMET 11 B: UL(B),ETQ(B).

REGLES : RB02 *.3. (.4.A,.7. ,.9.A)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.A , .5.B , .6. (.7.A , .8. (.9.C) , .10.A) , .11.A , .12.B)))

APPEL RECURSIF , REGLE : RB02

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.A , .3.B , .4. (.5.A , .6. (.7.C) , .8.A) , .9.A , .10.B)
SOMMET 1 : UL(ULSOL).
SOMMET 2 A: UL(A),ETQ(A).
SOMMET 3 B: UL(B),ETQ(B).
SOMMET 4 : UL(UL0),ETQ(S).
SOMMET 5 A: UL(A),ETQ(A).
SOMMET 6 : UL(UL0),ETQ(S).
SOMMET 7 C: UL(C),ETQ(C).
SOMMET 8 A: UL(A),ETQ(A).
SOMMET 9 A: UL(A),ETQ(A).
SOMMET 10 B: UL(B),ETQ(B).

REGLES : RB02 *.1. (.2.A,.4. ,.9.A)*

ARBORESCENCE TRANSFORMEE

.1. (.2.B , .3. (.4.A , .5. (.6.A , .7. (.8.C) , .9.A) , .10.A) , .11.B)

SYSTEME C.E.T.A.

DETAIL DE L'EXECUTION , TEXTE : 4

CODE LANGUE : GD

APPEL RECURSIF , REGLE : RB02

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.B , .3. (.4.A , .5. (.6.A , .7. (.8.C) , .9.A) , .10.A) , .11.B)

SOMMET 1 : UL(ULSOL).

SOMMET 2 B: UL(B),ETQ(B).

SOMMET 3 : UL(UL0),ETQ(S).

SOMMET 4 A: UL(A),ETQ(A).

SOMMET 5 : UL(UL0),ETQ(S).

SOMMET 6 A: UL(A),ETQ(A).

SOMMET 7 : UL(UL0),ETQ(S).

SOMMET 8 C: UL(C),ETQ(C).

SOMMET 9 A: UL(A),ETQ(A).

SOMMET 10 A: UL(A),ETQ(A).

SOMMET 11 B: UL(B),ETQ(B).

REGLES : RB02 *.1. (.2.B,.3. ,.11.B)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3.B , .4. (.5.A , .6. (.7.A , .8. (.9.C) , .10.A) , .11.A) , .12.B))

APPEL RECURSIF , REGLE : RB02

SOUS-ARBORESCENCE D'ENTREE

.1. (.2. (.3.B , .4. (.5.A , .6. (.7.A , .8. (.9.C) , .10.A) , .11.A) , .12.B))

SOMMET 1 : UL(ULSOL).

SOMMET 2 : UL(UL0),ETQ(S).

SOMMET 3 B: UL(B),ETQ(B).

SOMMET 4 : UL(UL0),ETQ(S).

SOMMET 5 A: UL(A),ETQ(A).

```
            SOMMET 6   : UL(UL0),ETQ(S).
            SOMMET 7  A: UL(A),ETQ(A).
            SOMMET 8   : UL(UL0),ETQ(S).
            SOMMET 9  C: UL(C),ETQ(C).
            SOMMET 10  A: UL(A),ETQ(A).
            SOMMET 11  A: UL(A),ETQ(A).
            SOMMET 12  B: UL(B),ETQ(B).

      FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE
         .1.  (.2.  (.3.B , .4.  (.5.A , .6.  (.7.A , .8.  (.9.C) , .10.A) , .11.A) , .12.B))
      FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE
         .1.  (.2.  (.3.B , .4.  (.5.A , .6.  (.7.A , .8.  (.9.C) , .10.A) , .11.A) , .12.B))
      FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE
         .1.  (.2.  (.3.  (.4.  (.5.B , .6.  (.7.A , .8.  (.9.A , .10.  (.11.C) , .12.A) , .13.A) , .14.B))))

RESULTAT :
         .1.  (.2.  (.3.  (.4.  (.5.B , .6.  (.7.A , .8.  (.9.A , .10.  (.11.C) , .12.A) , .13.A) , .14.B))))
            SOMMET 1   : UL(ULTXT).
            SOMMET 2   : UL(ULFRA).
            SOMMET 3   : UL(ULSOL).
            SOMMET 4   : UL(UL0),ETQ(S).
            SOMMET 5  B: UL(B),ETQ(B).
            SOMMET 6   : UL(UL0),ETQ(S).
            SOMMET 7  A: UL(A),ETQ(A).
            SOMMET 8   : UL(UL0),ETQ(S).
            SOMMET 9  A: UL(A),ETQ(A).
            SOMMET 10   : UL(UL0),ETQ(S).
            SOMMET 11  C: UL(C),ETQ(C).
            SOMMET 12  A: UL(A),ETQ(A).
```

SOMMET 13 A: UL(A),ETQ(A).

SOMMET 14 B: UL(B),ETQ(B).

RESULTAT DE L'EXECUTION , TEXTE :4 CODE LANGUE GD

.1. (.2. (.3. (.4. (.5.B , .6. (.7.A , .8. (.9.A , .10. (.11.C) , .12.A) , .13.A) , .14.B))))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 : UL(UL0),ETQ(S).

SOMMET 5 B: UL(B),ETQ(B).

SOMMET 6 : UL(UL0),ETQ(S).

SOMMET 7 A: UL(A),ETQ(A).

SOMMET 8 : UL(UL0),ETQ(S).

SOMMET 9 A: UL(A),ETQ(A).

SOMMET 10 : UL(UL0),ETQ(S).

SOMMET 11 C: UL(C),ETQ(C).

SOMMET 12 A: UL(A),ETQ(A).

SOMMET 13 A: UL(A),ETQ(A).

SOMMET 14 B: UL(B),ETQ(B).

2 - ANALYSE DE LANGUE NATURELLE

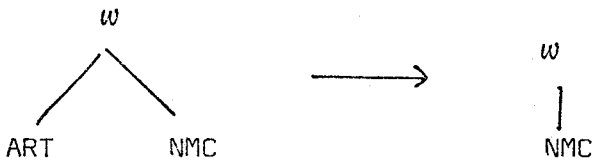
Afin de donner un aperçu du traitement des langues naturelles par un système transformationnel, nous donnons ici une ébauche de l'analyse du groupe nominal.

Cette grammaire permet les différentes opérations suivantes :

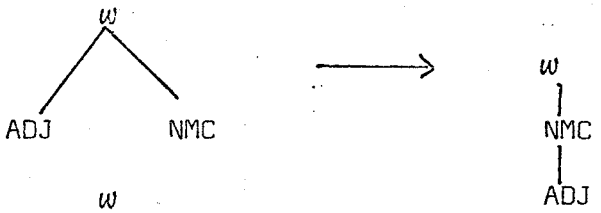
- élimination des articles (Règle R1)
- regroupement des différents adjectifs :
  - préposés (Règle R2)
  - postposés (Règle R3)
- regroupement des différents substantifs liés par une préposition : Règles R4, R5 et R6. Ces différentes règles permettent de montrer comment sont privilégiées certaines transformations par rapport à d'autres. Les conditions données ici pour ordonner ces transformations ne sont pas significatives du point de vue linguistique et sont seulement données à titre d'exemple.

Ces différentes transformations peuvent être résumées par les schémas suivants :

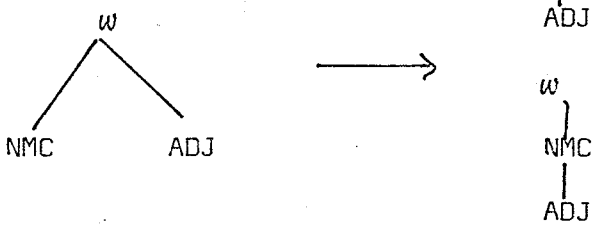
R1 :



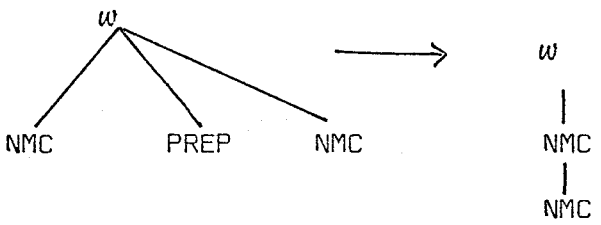
R2 :



R3 :

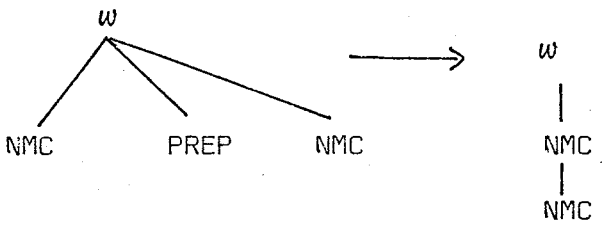


R4 :



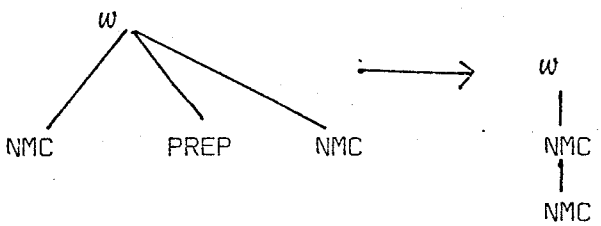
qualificateur concret

R5 :



qualificateur abstrait

R6 :



possessif

```
                        -PROC-
    PCP:  ART  ==  CAT -E- ART .
    PCP:  NMC  ==  CAT -E- NMC .
    PCIS:  ACGN(S1,S2)  ==  GNR(S1) -I- GNR(S2) -NE- GNR0 -ET-
                            NBR(S1) -I- NBR(S2) -NE- NBR0 .
    PCP:  ADJ  ==  CAT -E- AJQ .
    PAF:  AFGN(S1,S2)  ==  GNR := GNR(S1) -I- GNR(S2) ,
                           NBR := NBR(S1) -I- NBR(S2) .
    PCP:  NMCC  == CAT -E- NMC -ET- SEMA -INC- CONCR .
    PCP:  NMCA  == CAT -E- NMC -ET- SEMA -INC- ABSTR .
    PCP:  PREP  ==  CAT -E- PRP .
    PCP:  NMCP  ==  CAT -E- NMP .
                            -PSCHEM-
                            -GRAM-
          GRANM(E):  R2 (GRANM / R2 / OM(NMC) ),
                     R3 (GRANM / R3 / OM(NMC) ) ,
                     R1 (GRANM / R1 / OM ) ,
                     R5 (GRANM / R5 / OM(NMC1) ) ,
                     R6 (GRANM / R6 / OM(NMC1) ) ,
                     R4 (GRANM / R4 / OM(NMC1) ) ;
                          <-- &NUL .
                           -REGLES-
       R1:  OM(ART,*,NMC)  / ART: $ART ; NMC: $NMC / $ACGN(ART,NMC) ==
                 OM( NMC ) / * <-- ART / NMC: NMC , DEF := DEF(ART) .
       R2:  OM(ADJ,*,NMC) /ADJ: $ADJ ; NMC: $NMC / $ACGN(ADJ,NMC) ==
                 OM(NMC(ADJ)) // ADJ: ADJ , ETQ:= QUAL2 , $AFGN(ADJ , NMC)
                 ; NMC: NMC , $AFGN (ADJ , NMC ).
        R3:  OM( NMC,*,ADJ) / ADJ: $ADJ ; NMC: $NMC / $ACGN(ADJ,NMC) ==
                  OM(NMC(ADJ)) // ADJ:ADJ ,ETQ:= QUAL2 , $AFGN(ADJ,NMC) ;
                  NMC: NMC , $AFGN(ADJ,NMC).
       R4:  OM (NMC1,*,PREP,*,NMC2) / NMC1: $NMC ; NMC2: $NMC ;  PREP:
         $PREP / == OM(NMC1(NMC2)) / * <-- PREP / NMC2: NMC2 , ETQ:= POSS.
       R5:  OM(NMC1,*,PREP,*,NMC2) / NMC1:$NMCC ; NMC2:$NMCA;PREP: $PREP /
            == OM(NMC1(NMC2)) / * <-- PREP / NMC2: NMC2 , ETQ:= POSS .
       R6: OM (NMC1,*,PREP,*,NMC2) / NMC1:$NMC;NMC2:$NMCP ; PREP:$PREP /
            == OM(NMC1(NMC2)) / * <-- PREP / NMC2: NMC2 , ETQ:= QUAL1 .
                           -FIN-
```

DETAIL DE L'EXECUTION , TEXTE : 5 CODE LANGUE : FR1

GRAMMAIRE : GRANM (E)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.UN , .5.JEUNE , .6.CHEVAL , .7.CRAINTIF)))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 UN: UL(UN),SEG(B),CAT(ART),GNR(MAS),NBR(SIN).

SOMMET 5 JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS,FEM),NBR(SIN).

SOMMET 6 CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).

SOMMET 7 CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN).

REGLES : R2 *.3. (.5.JEUNE,.6.CHEVAL)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.UN , .5.CHEVAL (.6.JEUNE) , .7.CRAINTIF)))

APPEL RECURSIF , REGLE : R2

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.UN , .3.CHEVAL (.4.JEUNE) , .5.CRAINTIF)

SOMMET 1 : UL(ULSOL).

SOMMET 2 UN: UL(UN),SEG(B),CAT(ART),GNR(MAS),NBR(SIN).

SOMMET 3 CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).

SOMMET 4 JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

SOMMET 5 CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN).

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.UN , .5.CHEVAL (.6.JEUNE) , .7.CRAINTIF)))

REGLES : R3 *.3. (.5.CHEVAL,.7.CRAINTIF)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.UN , .5.CHEVAL (.6.JEUNE , .7.CRAINTIF))))

APPEL RECURSIF , REGLE : R3

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.UN , .3.CHEVAL (.4.JEUNE , .5.CRAINTIF))

SOMMET 1 : UL(ULSOL).

SOMMET 2 UN: UL(UN),SEG(B),CAT(ART),GNR(MAS),NBR(SIN).

SOMMET 3 CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).

SOMMET 4 JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

SOMMET 5 CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.UN , .5.CHEVAL (.6.JEUNE , .7.CRAINTIF))))

REGLES : R1 *.3. (.4.UN,.5.CHEVAL)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.CHEVAL (.5.JEUNE , .6.CRAINTIF))))

APPEL RECURSIF , REGLE : R1

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.CHEVAL (.3.JEUNE , .4.CRAINTIF))

SOMMET 1 : UL(ULSOL).

SOMMET 2 CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).

SOMMET 3 JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

SOMMET 4 CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.CHEVAL (.5.JEUNE , .6.CRAINTIF))))

RESULTAT :

.1. (.2. (.3. (.4.CHEVAL (.5.JEUNE , .6.CRAINTIF))))

SYSTEME C.E.T.A.

DETAIL DE L'EXECUTION , TEXTE : 5 CODE LANGUE : FR1

```
SOMMET 1   : UL(ULTXT).
SOMMET 2   : UL(ULFRA).
SOMMET 3   : UL(ULSOL).
SOMMET 4  CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).
SOMMET 5  JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).
SOMMET 6  CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUA
    L2).
```

RESULTAT DE L'EXECUTION , TEXTE :5 CODE LANGUE FR1

.1. (.2. (.3. (.4.CHEVAL (.5.JEUNE , .6.CRAINTIF))))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 CHEVAL: UL(CHEVAL),SEG(B),SDNA(9),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(CONCR,ANIM).

SOMMET 5 JEUNE: UL(JEUNE),SEG(B),SDNA(2),SDN(ESSE),PDRP(AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

SOMMET 6 CRAINTIF: UL(CRAINTIF),SEG(B),SDNA(10),SDN(ITE1),PDRP(AM,AN),CAT(AJQ),GNR(MAS),NBR(SIN),ETQ(QUAL2).

DETAIL DE L'EXECUTION , TEXTE : 7 CODE LANGUE : FR1

GRAMMAIRE : GRANM (E)

ARBORESCENCE D'ENTREE

.1. (.2. (.3. (.4.UN , .5.ACTEUR , .6.D' , .7.UNE , .8.MAISON , .9.DE , .10.REPOS , .11..)) , .12. (.13. (.14.UNE , .15.MAISON , .16.DE , .17.REPOS , .18.D' , .19.UN , .20.ACTEUR)))

REGLES : R1 *.3. (.7.UNE,.8.MAISON)*
R1 *.13. (.19.UN,.20.ACTEUR)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.UN , .5.ACTEUR , .6.D' , .7.MAISON , .8.DE , .9.REPOS , .10..)) , .11. (.12. (.13.UNE , .14.MAISON , .15.DE , .16.REPOS , .17.D' , .18.ACTEUR)))

APPEL RECURSIF , REGLE : R1

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.UNE , .3.MAISON , .4.DE , .5.REPOS , .6.D' , .7.ACTEUR)

REGLES : R1 *.1. (.2.UNE,.3.MAISON)*

ARBORESCENCE TRANSFORMEE

.1. (.2.MAISON , .3.DE , .4.REPOS , .5.D' , .6.ACTEUR)

APPEL RECURSIF , REGLE : R1

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.MAISON , .3.DE , .4.REPOS , .5.D' , .6.ACTEUR)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2.MAISON , .3.DE , .4.REPOS , .5.D' , .6.ACTEUR)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.UN , .5.ACTEUR , .6.D' , .7.MAISON , .8.DE , .9.REPOS , .10..)) , .11. (.12. (.13.MAISON , .14.DE , .15.REPOS , .16.D' , .17.ACTEUR)))

APPEL RECURSIF , REGLE : R1

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.UN , .3.ACTEUR , .4.D' , .5.MAISON , .6.DE , .7.REPOS , .8..)

REGLES : R1 *.1. (.2.UN,.3.ACTEUR)*

SYSTEME C.E.T.A.

DETAIL DE L'EXECUTION , TEXTE : 7 CODE LANGUE : FR1

```
        REGLES : R4       *.3. (.4.ACTEUR,.5.D',.6.MAISON)*
                 R4       *.10. (.11.MAISON,.13.D',.14.ACTEUR)*

    ARBORESCENCE TRANSFORMEE

          .1.  (.2.  (.3.  (.4.ACTEUR (.5.MAISON (.6.REPOS)) , .7..)) , .8.  (.9.  (.10.MAISON (.11.REPOS , .12.ACTEUR
         ))))

       APPEL RECURSIF , REGLE : R4

    SOUS-ARBORESCENCE D'ENTREE

          .1.  (.2.MAISON (.3.REPOS , .4.ACTEUR))

       FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

          .1.  (.2.  (.3.  (.4.ACTEUR (.5.MAISON (.6.REPOS)) , .7..)) , .8.  (.9.  (.10.MAISON (.11.REPOS , .12.ACTEUR
         ))))

       APPEL RECURSIF , REGLE : R4

    SOUS-ARBORESCENCE D'ENTREE

          .1.  (.2.ACTEUR (.3.MAISON (.4.REPOS)) , .5..)

       FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

          .1.  (.2.  (.3.  (.4.ACTEUR (.5.MAISON (.6.REPOS)) , .7..)) , .8.  (.9.  (.10.MAISON (.11.REPOS , .12.ACTEUR
         ))))

RESULTAT :

          .1.  (.2.  (.3.  (.4.ACTEUR (.5.MAISON (.6.REPOS)) , .7..)) , .8.  (.9.  (.10.MAISON (.11.REPOS , .12.ACTEUR
         ))))
```

ARBORESCENCE TRANSFORMEE

.1. (.2.ACTEUR , .3.D' , .4.MAISON , .5.DE , .6.REPOS , .7..)

APPEL RECURSIF , REGLE : R1

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.ACTEUR , .3.D' , .4.MAISON , .5.DE , .6.REPOS , .7..)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2.ACTEUR , .3.D' , .4.MAISON , .5.DE , .6.REPOS , .7..)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.ACTEUR , .5.D' , .6.MAISON , .7.DE , .8.REPOS , .9..)) , .10. (.11. (.12.MAISON , .13. DE , .14.REPOS , .15.D' , .16.ACTEUR)))

REGLES : R5 *.3. (.6.MAISON,.7.DE,.8.REPOS)*
R5 *.11. (.12.MAISON,.13.DE,.14.REPOS)*

ARBORESCENCE TRANSFORMEE

.1. (.2. (.3. (.4.ACTEUR , .5.D' , .6.MAISON (.7.REPOS) , .8..)) , .9. (.10. (.11.MAISON (.12.REPOS) , . 13.D' , .14.ACTEUR)))

APPEL RECURSIF , REGLE : R5

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.MAISON (.3.REPOS) , .4.D' , .5.ACTEUR)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.ACTEUR , .5.D' , .6.MAISON (.7.REPOS) , .8..)) , .9. (.10. (.11.MAISON (.12.REPOS) , . 13.D' , .14.ACTEUR)))

APPEL RECURSIF , REGLE : R5

SOUS-ARBORESCENCE D'ENTREE

.1. (.2.ACTEUR , .3.D' , .4.MAISON (.5.REPOS) , .6..)

FIN APPEL RECURSIF: ARBORESCENCE RESULTANTE

.1. (.2. (.3. (.4.ACTEUR , .5.D' , .6.MAISON (.7.REPOS) , .8..)) , .9. (.10. (.11.MAISON (.12.REPOS) , . 13.D' , .14.ACTEUR)))

RESULTAT DE L'EXECUTION , TEXTE :7 CODE LANGUE FR1

.1. (.2. (.3. (.4.ACTEUR (.5.MAISON (.6.REPOS)) , .7..)) , .8. (.9. (.10.MAISON (.11.REPOS , .12.ACTEUR
))))

SOMMET 1 : UL(ULTXT).

SOMMET 2 : UL(ULFRA).

SOMMET 3 : UL(ULSOL).

SOMMET 4 ACTEUR: UL(ACTEUR),SEG(B),SDNA(8),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(HUM,CONCR).

SOMMET 5 MAISON: UL(MAISON),SEG(B),SDNA(2),CAT(NMC),GNR(FEM),NBR(SIN),SEMA(CONCR),ETQ(POSS).

SOMMET 6 REPOS: UL(REPOS),SEG(B),SDNA(1),CAT(NMC),GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR),ETQ(POSS).

SOMMET 7 .: UL(POINT),SEG(B),CAT(PCT).

SOMMET 8 : UL(ULFRA).

SOMMET 9 : UL(ULSOL).

SOMMET 10 MAISON: UL(MAISON),SEG(B),SDNA(2),CAT(NMC),GNR(FEM),NBR(SIN),SEMA(CONCR).

SOMMET 11 REPOS: UL(REPOS),SEG(B),SDNA(1),CAT(NMC),GNR(MAS),NBR(SIN,PLU),SEMA(ABSTR),ETQ(POSS).

SOMMET 12 ACTEUR: UL(ACTEUR),SEG(B),SDNA(8),CAT(NMC),GNR(MAS),NBR(SIN),SEMA(HUM,CONCR),ETQ(POSS).

CONCLUSION

A l'issue de cette étude sur les transducteurs et les arborescences qui a conduit à la réalisation de systèmes généraux d'analyse et de transductions, il convient d'une part, de préciser la situation de ce travail au sein des recherches en informatique et d'autre part, d'examiner l'extension des applications.

La théorie des transducteurs a été constamment notre source de définition des algorithmes. Pour parvenir au but poursuivi, il a fallu étudier de nouvelles compositions de transducteurs afin de réaliser des systèmes de la puissance désirée (puissance comprise entre les fonctions récursives primitives et les fonctions totales récursives) à partir de transducteurs simples.

Il a paru intéressant d'étudier la transformation d'arborescences en termes de transformations de chaînes. Ceci nous a permis de dégager des algorithmes de type nouveau par rapport aux nombreuses études qui avaient été menées jusqu'ici sur les grammaires transformationnelles. Cette approche, évidemment, ouvre une voie tout à fait différente de celles pratiquées pour la définition et la programmation de tels algorithmes. Dans l'approche habituelle, le même langage sert à définir l'algorithme et à s'assurer que la programmation en est l'image exacte. Toutefois, il semble que la voie que nous avons suivie s'est avérée très efficace pour cerner des classes bien délimitées d'algorithmes (algorithmes compris entre deux niveaux de complexités assez voisins). En contre partie, la transcription de ces algorithmes dans un langage de programmation peut présenter quelques difficultés. En fait, dans les systèmes que nous avons développés, en particulier le système C.E.T.A. qui contient les algorithmes les plus complexes, l'assurance d'une programmation correcte peut s'établir à partir de l'étude des structures de données.

En ce qui concerne les applications, les algorithmes des modèles utilisés dans le traitement automatique des langues naturelles (en particulier, en traduction automatique) ont fourni la motivation initiale. Dans la mesure où les systèmes A.T.E.F. et C.E.T.A. sont utilisés par différentes équipes linguistiques nous nous proposons d'en corriger les faiblesses et d'en accroître la commodité. Ceci a d'ailleurs été réalisé depuis deux ans pour le système A.T.E.F. et commence pour le système C.E.T.A.

Les deux systèmes présentés ici n'ont pas cette seule application et sont indépendants l'un de l'autre. Le système A.T.E.F. peut réaliser par exemple une fonction complexe de H-code. Le système C.E.T.A. permet de simuler une grammaire transformationnelle. L'emploi de variables arithmétiques dans ce système permet de construire des grammaires dont l'application s'optimise au fur et à mesure de l'analyse. La conception de transducteurs universels permettra un développement de ce type de système vers la simulation de grammaires auto-transformables, ce mode de fonctionnement permettant la création dynamique de nouvelles règles de transformations.

BIBLIOGRAPHIE

(1) AHO A.V. and ULLMAN J.D.
Automaton analogs of syntax directed translation schemata
Proc. 9th IEEE symp. on switching and automata theory
October 1968

(2) AHO A.V.
Indexed grammars, and extension of context free grammars
Journal of the ACM : 1968 : 15 647-671

(3) AHO A.V. and ULLMAN J.D.
Syntax directed translations and the push down assembler
Journal computer system sciences 1969 : 3 37-56

(4) AHO A.V. and HOPCROFT J.E., ULLMAN J.D.
A general theory of translation
Mathematical systems theory 1969 : 3 193-221

(5) AHO A.V. and ULLMAN J.D.
Translation on context free grammars
Information and control : 1971 : 19 439-475

(6) AHO A.V. and ULLMAN J.D.
Characterizations and extensions of push down translations
Math. Systems theory 1971 : 5 172-192

(7) AHO A.V. and ULLMAN J.D.
The theory of parsing, translation and compiling
Volume 1 : Parsing Prentice Hall 1973

(8) AHO A.V. and ULLMAN J.D.
The theory of parsing, translation and copiling
Volume 2 : Compiling Prentice Hall 1973

(9) ARBIB M.A.
Theory of abstract automata
Prentice Hall 1969

(10) ARBIB M.A.
A simple self-reproducing universal automata
Information and Control 1966 : 9 177 - 189

(11) BARBAULT M.C. et DECLES J.P.
Etude structurelle et catégorielle du système de transition avec application à la science du calcul et à la linguistique mathématique
Thèse Paris 1970

(12) BARBAULT M.C. et DESCLES J.P.
Transformations formelles et théories linguistiques
Paris Dunod 1972

(13) BERGE C.
Théorie des graphes et ses applications
Dunod 1963

(14) BRAINERD W.S.
The minimalization of tree automata
Information and Control 1968 : 13 484-491

(15) BRAINERD W.S.
Tree generating regular systems
Information and Control 1969 : 14 217-231

(16) BRAINERD W.S.
Semi-thue systems and representations of trees
Conference record of the 1969, 10th American Symposium on switching and automata theory 240-244

(17) CHAUCHE J.
Transduction d'arborescences
Thèse 3ème cycle Grenoble 1971

(18) CHAUCHE J.
Transducteurs composés
Document G.E.T.A. G -2800-A - Grenoble

(19) CHAUCHE J.
Arborescences et transformations
Document G.E.T.A. G-2700-A - Grenoble

(20) CHAUCHE J.

Reconnaissance de sous-arborescences

Revue française d'automatique, informatique, recherche opérationnelle 1971 R3 89-95

(21) CHAUCHE J. - GUILLAUME P. - QUEZEL-AMBRUNAZ M.

Le système A.T.E.F.

Document G.E.T.A. G-2600-A - Grenoble

(22) CHAUCHE J.

Le système C.E.T.A.

Séminaire de programmation - Grenoble 1974

(23) COLMERAUER A.

Système Q

Université de Montréal 1971

(24) COLMERAUER A. - KANOUI H. - PASERO R. - ROUSSEL P.

Un système de communication homme-machine en français

Université d'Aix-Marseille - Octobre 1972

(25) DAVIS M.

Computability and unsolvability

Mc Graw Hill 1958

(26) EARLY J.

An efficient context free parsing algorithm

Communications of the ACM

(27) EILENBERG S. and WRIGHT J.B.

Automata in general algebra

Information and Control 1967 : 11 217-231

(28) FISCHER M.J.

Grammars with macro like productions

Proc. 9th IEEE Symp. on switching and automata theory october 1968

(29) FISCHER P.C.
Multitape and infinite state automata : a survey
Communications of the ACM 1965 : 8 799-805

(30) FISCHER P.C.
Turing machines with restricted memory access
Information and Control 1966 : 9 364-379

(31) FISCHER P.C.
On formalism for Turing machines
Journal of the ACM 1965 : 12 570-580

(32) GINSBURG S.
The mathematical theory of context free languages
Mc Graw Hill 1966

(33) GINSBURG S. and ROSE G.F.
Preservation of languages by transducteurs
Information and control 1966 : 9 153-176

(34) GINSBURG S. and PARTEE B.
A mathematical model of transformational grammars
Information and Control 1969 : 15 297-334

(35) GINSBURG S. - GREIBACH S. - HOPCROFT J.
Studies in abstracts families of languages
American mathematical society 1969

(36) GINSBURG S. and HOPCROFT J.
Image of A.F.L. under certain families of homomorphismes
Mathematical systems theory 1971 : 5 217-227

(37) GALDKY A.V. and MEL'CUK I.A.
Tree grammars (Δ-*grammars*)
International conference on computational linguistics
Stockholm Sweden 1969 : AL 3.2.

(38) GLADKY A.V. and MEL'CUK I.A.
Elements de linguistique mathématique
Dunod 1972

(39) GROSS et LENTIN
Notions sur les grammaires formelles
Gauthier-Villars 1967

(40) HARRISON M.A.
Introduction to switching and automata theory
Mc Graw Hill 1965

(41) HARRISON M.A. and SCHKOLNICK M.
A grammatical characterization of one way non-deterministic stack languages
Journal of the ACM 1971 : 18 148-172

(42) HARRISON M.A. and HAVEL I.M.
Strict deterministic grammars
Journal of computer system sciences 1973 : 7 237-277

(43) HERWOGKRENN KLAUS MÜLLNER
Bibliographie zur transformations-grammatik
Carl Winter Universitatsverlag Heidelberg 1968

(44) HENNIE F.C.
One tape, off line Turing machine computation
Information and Control 1965 : 8 553-578

(45) HERMES H.
Enumerability, decidability, computability
Springer Verlag 1965

(46) HOPCROFT J.E. and ULLMAN J.D.
Formal languages and their relation to automata
Addison Wesley 1969

(47) IBARRA O.H.
Characterization of transductions defined by abstract families of transducers
Mathematical systems theory 1971 : 5 271-281

(48) KAIN R.Y.
Automata theory : machines and languages
Mc Graw Hill 1972

(49) KNUTH D.E.
A characterization of parenthesis languages
Information and control 1967 : 11 269-289

(50) KNUTH D.E.
The art of computer programming
Vol. 1 Fundamental algorithms - Addison Wesley 1968

(51) KNUTH D.E.
The art of computer programming
Vol. 3 Sorting and searching - Addison Wesley 1973

(52) KUNTZMANN J.
Théorie des graphes et des réseaux
Dunod 1972

(53) LEVY L.S. et JOSHI A.K.
Some results in tree automata
Mathematical systems theory 1972 : 6 335-342

(54) LEWIS P.M. and STEARN R.E.
Syntax directed transduction
Journal of the ACM 1968 : 15 465-488

(55) MAL'CEV A.I.
Algorithms and recursives functions
Wolters Noordhoff 1970

(56) Mc NAUGHTON R.
Parenthesis grammars
Journal of the ACM 1967 : 14 490-500

(57) Mc NAUGHTON R. and PAPERT S.
Context free automata
The MIT Press 1971

(58) MENDELSON E.

Introduction to mathematical logic
Van Nostrand Reinhold 1972

(59) MEYER A.R. and FISCHER M.J.

Economy of description by automata, grammars and formal systems
Proc. 11th IEEE Symp. on switching and automata theory 1971

(60) MINSKY M.

Computation : Finite and infinite machines
Prentice Hall 1967

(61) PAIR C. et QUERE

Définition et études des bilangages réguliers
Information and Control 1968 : 13 563-593

(62) PARICKH R.J.

On context free languages
Journal of the ACM 1966 : 13 570-581

(63) PETRONE L.

On context free languages
Journal of the ACM 1966 : 13 570-581

(64) QUERE

Etude des ramifications et des bilangages
Thèse Nancy 1969

(65) ROGERS H.J.

The theory of recursive functions and effective computability
Mc Graw Hill 1967

(66) ROSEN B.K.

Tree manipulating systems and church-roser theorems
Journal of the ACM 1973 : 20 160-187

(67) ROUNDS W. C.

Context free grammars on trees

A. C.M. Symp. on Theory of computing (1969) 143-148

(68) ROUNDS W.C.

Mapping and grammars on trees

Mathematical systems theory 1970 : 4 257-287

(69) SALOMAA A.

A theory of automata

Pergamon Press 1969

(70) SALOMAA A.

A formal languages

Academic Press 1973

(71) SCHUTZENBERGER M.P.

On context free languages and push down automata

Information and control 1963 : 6 246-264

(72) THATCHER J.W.

Characterizing derivation tree of context free grammars through a generalization of finite automata theory

Journal computer system sciences 1967 : 317-322

(73) THATCHER J.W. and WRIGHT J.B.

Generalized finite automata theory with an application to a decision problem of second order logic

Mathematical system theory 1968 : 2 57-81

(74) THATCHER J.W.

Transformations and translations from the point of view of generalized finite automata theory

ACM Symposium on theory of computing 1969 : 129-142

(75) THATCHER J.W.

Generalized sequential machine maps

Journal computer systems sciences 1970 : 4 339-367

(76) VAUQUOIS B.
Calculabilité des langages
Cours - Grenoble 1969

(77) VEILLON G. - VEYRUNES J. - VAUQUOIS B.
Un métalangage de grammaires transformationnelles
Document C.E.T.A. - Janvier 1967

(78) VEILLON G.
Modèles et algorithmes pour la traduction automatique
Thèse - Grenoble 1970

(79) WINOGRAD
Procedures as a representation for data in a computer program for understanding natural language
M.I.T. : artificial intelligence laboratory
Cambridge - Mass. 1971

(80) WOODS W.A.
Augmented transition networks of natural language analysis
Rep. C.1. 1969 Computation laboratory - Harvard University
Cambridge - Mass.

(81) WOODS W.A.
Transition network grammars for natural language analysis
Communications of the ACM 1970 : 13 591-606

(82) YOSHI A J. - KOSARAJU S.R. - YAMADA H.M.
String adjunct grammars
Information and control 1972 : 21 93-116

(83) YOSHI A K. - KOSARAJU S.R. - YAMADA H.M.
String adjunct grammars
Information and control 1972 : 21 235-260

(84) J. COURTIN - J.L. RIEU - P. SGALL
Un métalangage pour l'analyse morphologique
Document C.E.T.A. - G.2500-A - Septembre 1969

Dernière page d'une thèse

VU

Grenoble, le

Le Président de la thèse

VU, et permis d'imprimer,

Grenoble, le

Le Président de l'Institut
National Polytechnique

Le Président de l'Université
Scientifique et Médicale