MATHÉMATIQUES ET INFORMATIQUE

Les mathématiques de la linguistique computationnelle

Premier volet : la théorie des langages[1]

Christian Retoré[2]

1947, un grand millésime : à Jean-Yves Girard, Gérard Huet et Alain Lecomte pour leur soixante ans

Cet article présente la linguistique computationnelle en mettant l'accent sur les modèles mathématiques et informatiques utilisés ou initiés par la linguistique. Nous avons organisé ces modèles en deux familles, non disjointes : la théorie des langages et la logique. Ce premier volet présente le domaine et son histoire, avant de s'intéresser à la théorie des langages formels. Cette dernière permet de décrire la structure linéaire des mots et la structure arborescente des phrases. Le second volet (à paraître dans la version électronique de la *Gazette* d'avril) portera sur l'utilisation de la logique et de la théorie des types qui permet aussi d'analyser la syntaxe des phrases mais peut en outre calculer le sens qu'elles véhiculent.

1. Situation épistémologique de la linguistique computationnelle

1.1. Origines

Les deux volets de cet article correspondent à deux traditions dans la formalisation de phénomènes linguistiques, l'une étant plus développée et suivie que l'autre.

La tradition la plus développée est sans doute la tradition hellénique qui lie logique et grammaire et a ainsi conduit, via la logique à une mathématisation du sens des énoncés et du discours. Cette approche a initialement défini conjointement les concepts logiques de prédicat et d'individu et les notions grammaticales de nom et de verbe. Elle sera en particulier approfondie durant la période scholastique et celle de Port-Royal. Elle s'intéresse principalement à la sémantique du langage naturel mais inclut néanmoins des préoccupations syntaxiques. Cette tradition permet aussi d'établir une filiation entre les catégories artistotelliciennes et celles d'Husserl.

[1] Je remercie de leurs relectures Jean-Xavier Rampon (LINA, université de Nantes) ainsi que Lionel Clément (LaBRI & INRIA, université de Bordeaux), Myriam Quatrini (IML, université d'Aix-Marseille) et Gilles Zémor (IMB, université de Bordeaux).

[2] LaBRI (CNRS et université de Bordeaux) & INRIA Bordeaux Sud-Ouest

Cette notion de catégorie et la vision fonctionnelle applicative de Frege ont conduit Lesniewski et Ajdukiewicz aux grammaires catégorielles, modèle mathématique de la syntaxe et de la sémantique du langage naturel qui sera au cœur du second volet de notre présentation de la linguistique computationnelle. [7, 4, 5, 16, 34, 17]

La seconde tradition, celle qui fait l'objet de ce premier volet de notre présentation, concerne l'étude des mécanismes calculatoires à l'œuvre dans notre utilisation du langage, que ce soit pour analyser et comprendre ou pour s'exprimer et se faire comprendre. L'étude générale des processus calculatoires a aussi intégré la logique mathématique depuis le début du vingtième siècle et se situe bien évidemment au cœur de l'informatique. Mais c'est bien avant qu'ont été découvertes les structures calculables de la langue, celles qui permettent de décrire par un nombre fini de règles de grammaire l'infinité potentielle des phrases correctes d'une langue. Une telle vision de la langue est déjà présente chez Pāṇini au 5^{e} siècle avant J.C. c'est-à-dire à peu près en même temps que l'apparition déjà mentionnée de la grammaire et de la logique dans la Grèce antique. On notera néanmoins que les voies ouvertes par ce pionnier n'ont pas été explorées avant les débuts de l'informatique, et en particulier les systèmes de Thue et de Post. Ce sont eux qui ont conduit le jeune Noam Chomsky à inventer la notion de grammaire formelle et à en étudier les propriétés mathématiques en particulier avec Marcel-Paul Schützenberger. [18, 52]

La description de la langue comme processus calculatoire ou logique dont nous parlons ici est une modélisation mathématique des processus mentaux liés au langage. Il s'agit donc de sciences cognitives, mais aussi d'informatique au sens courant puisqu'on peut, en s'en inspirant, écrire des programmes opérant sur des données langagières. De fait, le langage naturel est l'un des domaines d'application les plus anciens de l'informatique, apparu à peu près lors de la seconde guerre mondiale : c'est à ce moment qu'apparaissent les premières tentatives de traduction automatique, motivées par le conflit international en cours puis par la guerre froide qui en résulta. C'est ainsi que, curieusement, les connexions entre langue, logique et calcul se sont concrétisées.

Un tel domaine permet des applications dans les deux sens de termes, celui des mathématiciens et celui des informaticiens. D'une part, on peut parler d'application à la linguistique, puisque des modèles mathématiques sont utilisés par une autre science plus empirique et on rejoint ce que les mathématiciens appellent applications des mathématiques, les plus traditionnelles étant les applications à la physique, à l'économie et à la biologie. D'autre part, la linguistique computationnelle permet aussi des applications au sens où les informaticiens entendent ce mot, celui de des réalisation logicielles : il existe effectivement des logiciels qui mettent en œuvre les modèles pour analyser automatiquement la langue écrite ou parlée, ou, en sens inverse, produire automatiquement paroles ou écrits. Une question intéressante est de savoir si les bonnes modélisations de la faculté de langage de l'être humain conduisent à de bons logiciels ou s'il vaut mieux s'écarter des aspects cognitifs pour développer de bons logiciels. Quoi qu'il en soit, dans notre présentation, les réalisations logicielles possibles ou existantes ne retiendront notre attention que pour signaler l'utilisation des modèles mathématiques intéressants par eux-mêmes ou pour la connaissance qu'ils apportent sur notre utilisation de la langue

1.2. Un domaine à la croisée de disciplines mieux établies

Dans l'histoire récente, ce domaine entre mathématiques, linguistique et informatique a reçu diverses dénominations révèlent le point de vue de ceux qui s'y rattachent comme le montre un article assez récent [37] .

– Machine Translation C'est ainsi que sont appelé les premiers travaux, très appliqués, réalisés avec des techniques de la famille des automates d'états finis. De tels logiciels aux possibilités limitées survivent dans les outils gratuits de traduction automatique à destination du grand public. On remarquera que leur qualité n'atteint pas celle de logiciels grand public d'autres domaines (lecteurs vidéo, tableurs, ...), ce qui est inévitable vu l'ambition de l'objectif à atteindre.

– Computational linguistics Les relatifs échecs des débuts de la traduction automatique ont conduit Bar-Hillel, dont nous reparlerons au sujet des grammaires catégorielles dans le second volet de notre présentation, à un rapport [6] qui suggérait de se restreindre à des objectifs plus modestes que la traduction automatique et d'utiliser des techniques plus performantes et sophistiquées, tant du côté mathématique (logique, algèbre, probabilités) que du côté linguistique (grammaire générative, grammaire de dépendances). C'est dans cette perspective que s'inscrit notre article comme le montre son titre.

– Automatic Language Processing - Natural Language Processing Cette expression, apparue à la fin des années soixante désigne la partie applicative du précédent domaine. Initialement dédiée aux techniques d'analyse syntaxique (*parsing*) elle s'est ensuite étendue à l'utilisation pour cette tâche et pour d'autres de méthodes statistiques qui ont connu un vif essor dans les années quatre-vingt dix.

– Natural Language Understanding - Cognitive sciences Dans les années soixante dix, avec l'apparition de l'intelligence artificielle, la recherche s'est tournée vers la compréhension automatique du langage naturel, en parallèle avec la modélisation du raisonnement. À l'heure où il est de bon ton de critiquer l'intelligence artificielle, remarquons que nombre de formalismes et mêmes de notions utilisés aujourd'hui en linguistique computationnelle sont issus de l'intelligence artificielle et de son langage fétiche, Prolog : les grammaires d'unification (qui sont en fait à l'origine de Prolog), l'analyse syntaxique vue comme une déduction (*parsing as deduction*), la sémantique logique... De nos jours, cette approche intitulée « intelligence artificielle » ou « compréhension automatique du langage naturel » se retrouve dans les laboratoires de sciences cognitives et d'interface homme machine (et bien sûr d'intelligence artificielle).

La dénomination francophone de ce sujet entre linguistique, mathématiques, logique et informatique oscille entre diverses formulations. Dans les années soixante, l'expression *traduction automatique*, inspirée de *machine translation* contentait les acteurs du domaine mais par la suite l'expression *computational linguistics* n'a jamais trouvé une traduction qui s'impose : linguistique mathématique, linguistique informatique, informatique linguistique, linguistique computationnelle — on introduisit même l'expression « linguistique quantitative » pour rassembler linguistes et mathématiciens sans effrayer les premiers. Seule la traduction de *Natural Language Processing* en *Traitement Automatique des Langues* et en sa variante *Traitement*

Automatique du Langage Naturel se sont imposées, mais cela confirme notre impression d'une communauté plutôt orientée vers les aspects pratiques — à l'exception de la sémantique formelle, bien représentée, ce qui laisse un grand fossé entre le pragmatisme affiché et l'étude théorique de la pragmatique !

En France, et plus généralement dans la francophonie, ce sujet est bien représenté, principalement grâce à l'association ATALA (Association pour le Traitement Automatique des Langues) qui depuis la fin des années cinquante fédère ce domaine mieux que les organismes de recherche. Elle tient un colloque annuel « Traitement Automatique du Langage Naturel » et édite depuis sa création une revue « Traduction Automatique - Informations » devenue aujourd'hui « Traitement Automatique des Langues ».

Malheureusement, malgré la qualité de l'école française de théorie des langages formels fondé par l'éclectique Marcel-Paul Schützenberger, co-auteur avec le linguiste Noam Chosmky des premiers travaux du domaine, on ne peut pas dire que l'interaction entre, d'une part, les mathématiciens ou informaticiens et, d'autre part, les linguistes ait été très productive à la différence des États-Unis, de l'Allemagne ou des Pays-Bas. Il faut toutefois mentionner certaines exceptions comme les premiers travaux du linguiste Maurice Gross avec le mathématicien André Lentin sur les grammaires formelles pour la linguistique ou ceux plus autodidactes d'Alain Colmerauer sur les grammaires de métamorphoses développées pour le langage naturel avant que Prolog n'en surgisse. On mentionnera aussi l'attrait que la formalisation et la mathématisation ont exercé sur certains linguistes, comme Lucien Tesnières, Bernard Pottier, Jean-Claude Milner ou Antoine Culioli.

Il faut assurément mentionner trois sociétés savantes, *the Association for Computational Linguistics* (1962), les plus récentes, *Mathematics of Language* et *Foundation for Logic, Language and Information* (européenne) ; dans ce domaine mentionnons en particulier les travaux d'Eward Keenan, d'Aravind Joshi, de Bill Rounds de Stuart Shieber qui démontrent l'intérêt de la collaboration entre linguistique, informatique, logique et mathématiques.

1.3. Domaines de la linguistique et niveau d'analyse de la langue

L'objectif de notre domaine, la modélisation du langage naturel, peut paraître démesuré si on considère les phénomènes linguistiques comme un tout inorganisé. Fort heureusement, des siècles de grammaire, de grammaire comparée et de linguistique ont défini à l'intérieur de ce vaste champ d'étude qu'est la langue différents domaines, qui, certes, interagissent entre eux, mais peuvent aussi être étudiés indépendamment ; cela divise notre travail de formalisation en un ensemble d'objectifs plus restreints et plus raisonnables, suivant la tradition inaugurée par le structuralisme Saussurien [53]. Avant de décrire ces domaines de la linguistique, mentionnons une notation fort pratique des linguistes, qui consiste à placer une étoile « $*$ » devant les expressions incorrectes, et un ou plusieurs points d'interrogation « $?$ » devant une expression dont la correction pose question et rien devant une expression correcte — l'adjectif correct employé ici ne renvoie pas à un bon usage tel que la grammaire scolaire en formule mais à ce que disent effectivement les locuteurs.

– La phonétique est l'étude des sons en tant que phénomène acoustique. Elle concerne aussi bien leur production par le système phonatoire que par leur réception par le système auditif.

– La phonologie est l'étude des sons des langues comme système discret. Ce système est acquis par l'enfant dès son sixième mois. La discrétisation, le partage en zones de l'espace des possibilités, qui définit les sons d'une langue, expliquent que Bali et Paris soient indistincts pour un japonais.

– La morphologie est l'étude de la structure des mots. Ceux-ci sont composés de morphèmes, les plus petites unités de sens, d'où le nom de ce domaine. Les opérations à l'œuvre sont en général régulières au sens technique de ce mot, défini au paragraphe 3.1 — bien que certaines langues comme le navajo utilisent des constructions non contextuelles à l'intérieur même des mots ! La morphologie se divise en deux sous-domaines, qui utilisent les mêmes modèles, du moins tant qu'on laisse le sens de coté.

– La morphologie flexionnelle étudie les phénomènes de conjugaison, de déclinaison et d'adaptation au genre, au nombre, etc. qui généralement ne changent pas la catégorie grammaticale du mot (même si un participe passé, en français, est peu différent d'un adjectif).

(1) arriver → arriv[er][ons]
(2) cheval → chevaux

– La morphologie dérivationnelle étudie les règles de formation des mots par combinaison de morphèmes pleins, affixes, préfixes, infixes, suffixes. Ces opérations peuvent changer la catégorie grammaticale. Leur applicabilité suit des règles parfois non évidentes mais sémantiquement motivées.

(1) noble → noblesse
    petit → petitesse
(2) maison → maisonnette
    camion → camionnette
    carpe * → carpette
(3) désirer → désirable → indésirable
    manger → mangeable → immangeable

– La syntaxe régule l'agencement des mots dans la phrase. Il s'agit non seulement de reconnaître les suites de mots qui sont des phrases mais aussi de leur assigner une structure, généralement une structure d'arbre, qui permet ensuite d'interpréter la phrase analysée. Dans les exemples qui suivent, nous représenterons cette structure par des crochets, dont le premier porte éventuellement en indice la catégorie de l'expression entre ce premier crochet ouvrant et le crochet fermant qui lui correspond. On notera qu'une phrase structurée peut être incorrecte parce que la structure est incorrecte alors que la succession de mots est possible. Ces structures peuvent aussi être des graphes, qu'il faut souvent voir comme des arbres augmentés de quelques arcs ou arêtes. Nous préciserons cet aspect de la langue au paragraphe 3.1.

(1) Je les fais ouvrir
(2) *Je fais les ouvrir
(3) Je sais les ouvrir
(4) * Je les sais ouvrir

(5) * [[Emilie [mange des]] huîtres]
(6) Emilie [mange [des huîtres]]

– La sémantique étudie le sens des mots, des phrases en dehors du contexte où elles sont utilisées.

– la sémantique lexicale porte sur le sens des mots et les relations qui les unissent

(1) livre, imprimer (objet concret), lire (contenu abstrait)

– la sémantique formelle peut être vue comme une partie de la philosophie du langage, domaine fort ancien intimement lié à la logique. Dans sa longue histoire citons au moins deux auteurs dont les travaux restent d'actualité : Gottlob Frege au début du vingtième siècle et, même si le spectre de ses travaux est plus restreint, Richard Montague dans les années soixante-dix. La sémantique formelle fait souvent appel à deux postulats indépendants bien que souvent confondus :

– la sémantique est vériconditionnelle, c'est-à-dire qu'elle identifie le sens d'un énoncé avec ses conditions de vérité, ce qui est souvent réalisé par l'association de formules logiques ensuite interprétées dans des modèles. Ce domaine n'est pas propre à la linguistique et son étude formelle rejoint souvent l'étude des modèles pour des logiques plus riches que celle des mathématiques usuelles, comme on en trouve dans la logique mathématique ou en informatique théorique (parallélisme, vérification). Nous en reparlerons dans le second volet de notre présentation.

– la sémantique est compositionnelle et son objet est de déterminer les règles qui permettent le calcul du sens d'un constituant à partir du sens de ses parties et de sa structure syntaxique. En général, elle utilise le λ-calcul pour gérer la composition et les substitutions. Nous en reparlerons aussi dans le second volet de notre présentation.

– La pragmatique et l'énonciation explorent l'utilisation de la langue pour communiquer dans un contexte donné, par exemple à qui renvoie les indexicaux : premières et deuxièmes personnes (je, nous, vous), ici, maintenant, démonstratifs,... et plus généralement étudie le discours.

(1) Allons plutôt dans ce restaurant.

– La prosodie est l'étude de la structure du phrasé : pauses, intonation. Elle est un ingrédient souvent négligé qui participe pourtant à l'interprétation de la phrase. L'oral évite par la prosodie bien des ambiguïtés. Dans l'exemple suivant, il s'agit de l'ambiguïté entre « parler... sur » et « formation initiale... sur » :

(1) « Je serai très heureux de venir parler au LaBRI, laboratoire auquel je dois ma formation initiale en informatique, par exemple sur la lambda-DRT. »

(2) « Je serai très heureux de venir parler au LaBRI — laboratoire auquel je dois ma formation initiale en informatique — par exemple sur la lambda-DRT. »

Nous reviendrons par la suite sur les aspects formels, mais disons d'ores et déjà que les modèles informatiques et mathématiques utilisés en linguistique sont :

– PROBABILITÉS ET STATISTIQUES dont nous parlerons peu car les méthodes utilisées ne sont pas spécifiques aux structures linguistiques manipulées [38]
– GRAMMAIRES FORMELLES auxquelles est consacré ce premier volet
– LOGIQUE MATHÉMATIQUE dont traitera le second volet de notre présentation.

1.4. Exemples de traitements automatiques des langues

Pour les raisons que nous expliquerons en conclusion des deux volets, il n'est malheureusement pas clair que les modèles utilisées par les réalisations logicielles qui fonctionnent soient intéressants d'un point de vue mathématique, et, réciproquement, il n'est pas sûr non plus que les modèles mathématiquement intéressants donnent lieu à des réalisations logicielles performantes. Souvent, des modèles simples avec des techniques rudimentaires donnent de bon résultats pour peu qu'on dispose des ressources nécessaires, comme des corpus annotés, et de suffisamment d'huile de coude de programmeur. Mentionnons tout de même les applications dont le lecteur peut avoir fait l'expérience ou connaître l'existence, des plus légères aux plus consommatrices de mathématiques, d'après [32, 3].

– LA RECHERCHE D'INFORMATION ET LA CLASSIFICATION AUTOMATIQUE utilisent en général des probabilités élémentaires, et éventuellement des ensembles de règles.
– L'ALIGNEMENT AUTOMATIQUE DE TEXTES BILINGUES consiste à mettre en rapport phrase à phrase ou constituant par constituant un texte et sa traduction. Ceci ne requiert qu'un simple étiquetage syntaxique, mais est un ingrédient important de la traduction automatique par des méthodes statistiques.
– L'INDEXATION AUTOMATIQUE il fonctionne par extraction de groupe nominaux et peut se contenter d'une analyse syntaxique partielle.
– LA SIMPLIFICATION DE TEXTE procède généralement par suppression de propositions subordonnées, de compléments, etc. ; c'est une étape qui ne demande qu'une analyse syntaxique partielle et qui est utile au résumé de texte.
– L'EXTRACTION DE CONNAISSANCES LINGUISTIQUES À PARTIR DE TEXTES permet entre autres d'étudier les collocations (grosso modo les expressions semi-figées), les restrictions de sélection et d'automatiser la construction de grammaires à partir de corpus ; elle nécessite au moins un étiquetage syntaxique et utilise en général une analyse syntaxique partielle.
– LA CORRECTION ORTHOGRAPHIQUE se contente d'une analyse syntaxique partielle tandis que pour être pleinement correcte, elle devrait utiliser une analyse complète (antécédents de pronoms, dépendances non bornées, etc.) : « Quel**S** sont le**S** livre**S** que mon fils croit que sa sœur a lu**S** . »
– L'INTERROGATION DE BASES DE DONNÉES EN LANGAGE NATUREL nécessite une analyse syntaxique complète de la question afin d'obtenir une représentation du sens comme une formule logique avec souvent de nombreux quantificateurs.
– LA GÉNÉRATION AUTOMATIQUE a non seulement besoin de la structure syntaxique complète mais aussi de la structure discursive qui est en général aussi décrite par une grammaire.

2. Machines à états finis et structure linéaire

2.1. Monoïdes, semi-groupes, grammaires : définitions élémentaires

Dans un souci de simplicité, nous évoquerons surtout les langages linéaires, les suites de mots ou de morphèmes, bien que la linguistique utilise aussi des arbres, en particulier pour décrire la syntaxe de la phrase.

Un *monoïde* est un ensemble M muni d'une loi de composition associative $\cdot$ qui possède un élément neutre ε. S'il n'y a pas d'élément neutre on parle alors de *semi-groupe*.

Étant donné un ensemble E on note E^* *les suites finies d'éléments de* E, dont la suite vide notée ε et E^+ les suites finies non vides d'éléments de E. Formellement, une suite finie de longueur n est une application de $[1, n]$ dans E mais on se contente souvent de noter $a_1 \cdots a_n$ une suite de n éléments de E.

Une opération naturelle sur E^* est la *concaténation* : opération associative qui à une suite finie $a_1 \cdots a_n$ et une suite $b_1 \cdots b_p$ d'éléments de E associe la suite $a_1 \cdots a_n b_1 \cdots b_p$

Un cas particulier d'usage courant est le *monoïde libre* sur un ensemble Σ, appelé alphabet ou lexique et dont les éléments sont appelés terminaux (on dit aussi lettres, ou mots, mais dans un contexte linguistique, ces deux derniers termes portent à confusion : une phrase est une suite de mots, un mot est une suite de lettres). Ce monoïde est défini sur Σ^* avec pour loi de composition la concaténation (qui est bien associative) et pour élément neutre la suite vide ε (qui est bien élément neutre).

Un langage sur Σ est tout simplement une partie, finie ou infinie, de Σ^*.

Une grammaire est un procédé mécanique qui définit un langage comme toutes les expressions que l'on obtient à partir d'une ou plusieurs expression initiale par libre application d'un ensemble fini de règles : il s'agit donc d'une description finie d'un ensemble infini d'expressions. On en détaillera le fonctionnement ci-après.

2.2. Automates

Un automate est une machine virtuelle comme l'est une machine de Turing, mais aux pouvoirs bien plus limités. Il peut être représenté par un graphe étiqueté dont les sommets sont appelés *états* et les arcs *transitions* ; ces dernières sont étiquetées par des terminaux d'un alphabet Σ. L'un des états, appelé S, est dit initial et un ou plusieurs états sont dits finaux. L'automate produit une suite finie de terminaux de Σ, aussi appelée expression, ainsi :

– On commence dans l'état S avec l'expression vide comme expression courante.

– Si on est dans l'état X avec l'expression courante w et qu'il y a une transition de X à Y étiquetée par le terminal x de Σ alors on peut passer dans l'état Y avec wx comme expression courante.

– On ne peut s'arrêter que dans l'un des état finaux. Le résultat est l'expression courante.

Sans que cela change les langages que l'on peut décrire par un automate, on peut autoriser des transitions vides : c'est une transition étiquetée ε d'un état X vers un état Y, elle permet de passer de X à Y sans produire de nouveau symbole.

La même machine peut être utilisée pour reconnaître les suites de terminaux de Σ (l'expression courante est la partie de l'expression qui est déjà reconnue).

Un automate peut par exemple servir à produire toutes les suites de lettres ou de sons possibles en français, mais aussi les noms de nombre (voir la figure 2.2) ou de dates, voire même à décrire les constructions syntaxiques simples. [50] Les automates sont monnaie courante dans l'informatique en général et, ici comme ailleurs des questions essentielles sont la minimisation (construire un automate qui fasse le même travail avec le moins possible d'états) et la déterminisation d'un automate — un automate est dit déterministe lorsqu'il y a, pour chaque état q et pour chaque terminal x de l'alphabet au plus une transition étiquetée x au départ de q : en lisant une expression on n'a jamais aucun choix sur l'état suivant. Les automates sont connus pour engendrer une classe de langages close par

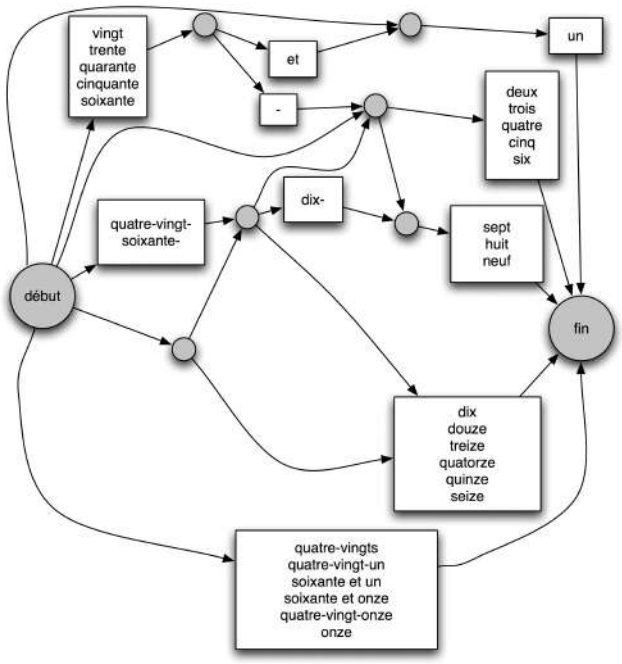


FIG. 1. Un automate qui produit ou reconnaît le nom des nombres entre un et cent.

intersection, réunion, complémentation. La classe de langages engendrés est aussi stable lorsqu'on change légèrement la définition de l'automate : un ou plusieurs états initiaux, un ou plusieurs états finaux, avec transitions vides ou sans, dont les arcs sont étiquetés par des suites finies de terminaux de Σ^* ou par un terminal exactement.

2.3. Analyse et transformation du mot : transducteurs

Ces machines, les automates, admettent une extension, les transducteurs, aussi appelés fonctions automates ou automates entrée-sortie, qui permettent notamment de modéliser la morphologie [50]. Les transitions d'un transducteur sont étiquetées par des couples de suites de terminaux, donc par des éléments de $\Sigma_1^* \times \Sigma_2^*$ dont la signification est la suivante : on lit une suite de symboles de Σ_1 et simultanément on produit une expression de Σ_2. Si les conditions suivantes sont réunies :

- l'expression produite jusqu'ici est m
- ce qu'il reste à lire de l'expression est $w' = \alpha w''$
- l'état courant est q
- il existe une transition t étiquetée α/β conduisant à l'état q'

alors on peut effectuer la transition t et après l'avoir faite, la situation est la suivante :

- l'état courant est q'
- l'expression produite est $m\beta$
- ce qu'il reste à lire de l'expression est $w' = w''$

Comme pour les automates, on entre par l'état initial avec le mot à lire (qui va être consommé petit à petit) et on doit terminer sur un état final. On produit un ensemble de couples d'expressions, c'est-à-dire une partie de $\Sigma_1^* \times \Sigma_2^*$,

Ces machines appelées transducteurs sont parfaitement adaptées à décrire la segmentation (qui détecte les séparations entre mots), les règles phonologiques (le son « s » devient « z » entre deux voyelles) et les opérations morphologiques (constitution de mots par concaténation de morphèmes, préfixes radicaux et suffixes). On peut ainsi assez simplement décrire la formation des adverbes en « -ment » du français ou la conjugaison des verbes.

D'un point de vue mathématique, les transducteurs posent le même genre de questions que les automates, plus certaines qui leurs sont propres. Par exemple un automate est toujours équivalent à un automate sans transition vide et dont les transitions sont étiquetées par exactement un terminal. Pour un transducteur, on peut se limiter à des transitions de la forme x/y avec x et y valant soit un terminal soit ε et n'étant pas tous les deux ε — mais l'impossibilité en général de n'avoir que des transitions de la forme x/y avec x et y deux terminaux, empêche la classe des relations produites par les transducteurs d'être close par intersection. Lorsqu'on peut se limiter à des transition a/b avec $a \in \Sigma_1$ et $b \in \Sigma_2$ on parle de transducteur lettre-à-lettre, lesquels constituent une classe close par intersection.

La minimisation des transducteurs est possible mais plus complexe que celle des automates : leur déterminisation n'est pas toujours possible et, lorsqu'elle l'est, elle se fait par un algorithme relativement astucieux.

Grosso modo, il existe deux manières de fabriquer un transducteur à partir d'autres transducteurs. On peut les composer « en cascade » ce qui consiste à utiliser en entrée du deuxième transducteur le mot produit par le premier transducteur : cette opération calcule effectivement la composition des relations associées aux transducteurs. On peut aussi, s'ils sont lettre-à-lettre en faire l'intersection, ce qu'on appelle abusivement composition en parallèle. Par exemple, que donnent ces deux méthodes si on souhaite définir un transducteur qui fasse précéder un mot

« m » du préfixe « multi » avec trait d'union si « m » commence par une voyelle et sans si « m » commence par une consonne ?

On peut commencer par insérer le préfixe avec un trait d'union, puis l'effacer lorsqu'il est suivi d'une voyelle, et composer « en cascade » les deux transducteurs : c'est la première méthode.

On peut aussi avec l'aide d'un symbole auxilaire ∅ qui permet de n'avoir que des transducteurs lettre-à-lettre, fabriquer des transducteurs lettre-à-lettre qui construisent les relations suivantes, où c désigne un mot commençant par une consonne et v un mot commençant par une voyelle :

– R ou tout mot m est en relation avec *multi*∅m et avec *multi-m*,

– R_v où un mot c commençant par une consonne est en relation avec *multi-c* et avec *multi*∅c tandis qu'un mot v commençant par une voyelle n'est en relation qu'avec *multi-v*, et

– R_c où un mot v commençant par une voyelle est en relation avec *multi-v* et avec *multi*∅v tandis qu'un mot c commençant par une consonne n'est en relation qu'avec *multic*.

La relation $R \cap R_v \cap R_c$ est presque la bonne, il suffit d'un post-traitement qui efface les symboles auxiliaires ∅ fait par composition en cascade avec un transducteur non lettre-à-lettre. Cette dernière méthode qui peut sembler plus compliquée permet de procéder en demandant à ce que des conditions soient simultanément satisfaites, et d'obtenir le résultat par combinaison de ces conditions formulées de manière indépendante : pour la suite de morphèmes *co in culp ation*, de telles règles sur les traits d'union avec voyelles et consonnes donneront *co-inculpation*.

Nous donnons en figure 2 la règle d'accord de l'adverbe de degré *tout*.

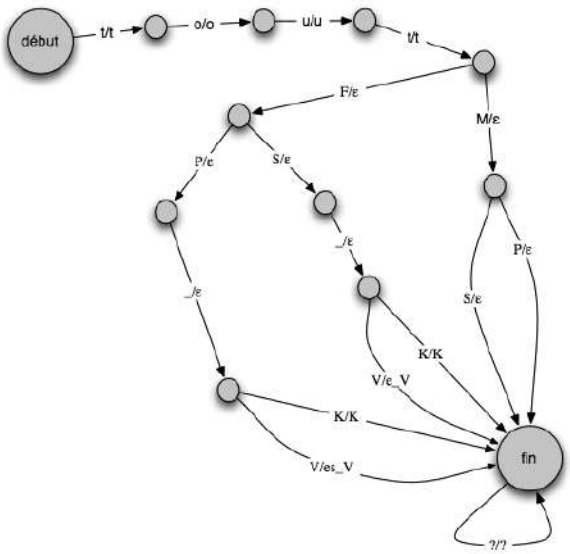


FIG. 2. Un transducteur réalisant l'accord de l'adverbe *tout* exprimant le degré complet.

Le transducteur ci-dessus représente la règle d'accord de *tout* utilisé comme adverbe. F et P désignent respectivement les traits *féminin* et *masculin*, S et P désignent respectivement les traits *singulier* et *pluriel*. Une transition K/K est une abréviation pour les les transitions de mêmes états initial et final b/b, c/c ... (pour toute consonne, y compris le *h* aspiré), de même V/es_V est une abréviations pour les transitions a/es_a, e/es_e (pour toute voyelle, y compris le *h* non aspiré). La transition $?/?$ récrit n'importe quel caractère en lui même. Le transducteur réalise entre autres *toutFS_entière* → *toute entière*.

La règle dit que *tout* est invariable, sauf si l'adjectif qui le suit est au féminin et commence par une voyelle : *les fenêtres tout entières ouvertes* et *les fenêtres toutes grandes ouvertes* — voir [28].

2.4. Un pas vers la syntaxe et un zeste de probabilités : modèles de Markov cachés pour l'étiquetage syntaxique

Les Modèles de Markov cachés, *Hidden Markov Models*, désormais HMM, sont des sorte d'automates probabilistes dont l'algorithmique est très efficace. Introduits à la fin des années soixante [8] — on pourra consulter l'excellente synthèse [15] — ils sont fort utiles à la prédiction de ce qui suit dans une chaîne, pour peu qu'il y ait une régularité dans les chaînes considérées : ce genre de modèle n'est en rien particulier à l'étude de la langue : de telles études permettent de prévoir ce qui suit dans une chaîne de signaux, de sons, d'images, de mesures, de tests de circuits, etc. Les HMM ont ainsi été utilisés pour la reconnaissance de la parole, de l'écriture, le diagnostic de systèmes électroniques, l'analyse du génome etc. Les HMM sont une méthode très efficace pour l'étiquetage grammatical d'une phrase dans une langue à ordre des mots très strict, et donc un premier pas vers l'analyse syntaxique profonde ou superficielle de la phrase. [32, 38]

Rapidement, un HMM est une sorte d'automate comme celui représenté en figure 3 :

– Les états correspondent aux catégories grammaticales.

– Chaque transition est munie d'une probabilité de sorte que la somme des probabilités des transitions issues d'un état donné est 1.

– De chaque état ou catégorie grammaticale est émis un mot avec une certaine probabilité et la somme des probabilités d'émission des mots d'une catégorie donnée est 1.

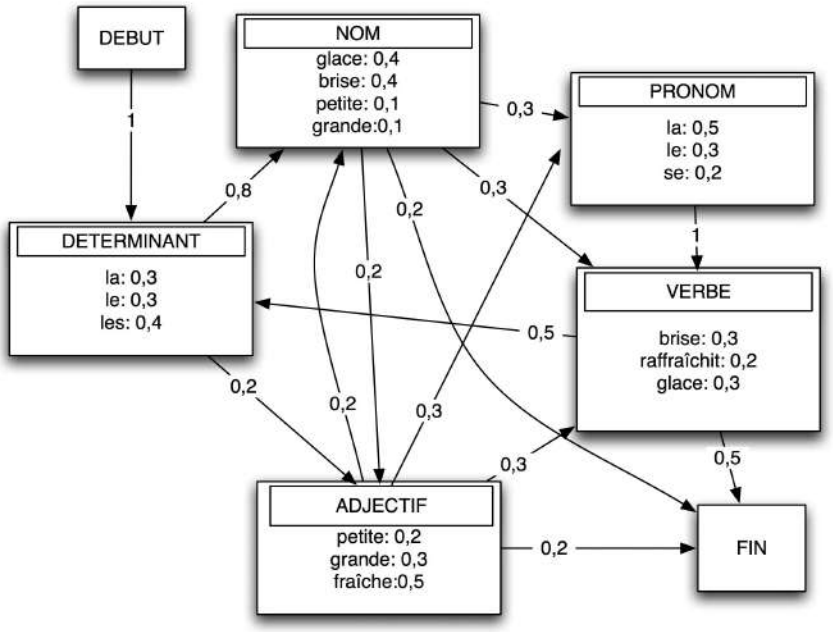
Les hypothèses faites par un tel modèle sont des hypothèses de mémoire bornée et d'indépendance aux niveaux du mot et de la suite de catégories. Au niveau de la suite des catégories, la catégorie d'un mot ne dépend que de celle du mot précédent (ou d'un nombre fixé de mots précédents). Au niveau du mot, la probabilité d'avoir un mot en connaissant sa catégorie grammaticale ne dépend ni de ce qui précède, ni de ce qui suit.

La méthode d'étiquetage fonctionne schématiquement ainsi : étant donnée une phrase, on détermine le chemin de Viterbi, c'est-à-dire le chemin le plus probable qui produise cette séquence de mots. Cette suite d'états est la suite des catégories attribuées à chacun des mots de la phrase. Elle se calcule de manière très efficace, par un algorithme de type programmation dynamique, fonctionnant en au plus $O(TN^2)$ étapes si la phrase comporte T mots. Dans l'exemple bien connu d'André

Martinet reproduit ci-dessous, on trouvera vraisemblablement deux chemins, les deux premiers, franchement plus probables que les autres dont le troisième :

(1) « La(det) petite(n) brise(v) la(det) glace(n) »
(2) « La(det) petite(adj) brise(n) la(pro) glace(v) »
(3) « La(det) petite(n) brise(n) la(det) glace(v) »

Mieux encore, le HMM est améliorable en remplaçant les probabilités par les fréquences pour des exemples obtenus par le HMM ou donnés manuellement (et dont on pense qu'ils sont correctement étiquetés). La méthode de Viterbi consiste à utiliser les chemins de Viterbi obtenus sur des phrases par le HMM ou par une étude directe, et à définir la probabilité d'émettre le mot m dans l'état s comme le nombre de fois où on a émis s depuis n divisé par le nombre de fois où l'on était dans s dans ces chemins, et similairement, à définir la probabilité de passer de l'état s à l'état s' comme le nombre de fois où on est passé de l'état s à l'état s' divisé par le nombre de fois, où, le long de ces chemins, on était dans l'état s. Cette méthode peut être itérée jusqu'à ce qu'elle se stabilise, ce qui arrive lorsqu'un extremum local est atteint. On peut même affiner cette méthode, en prenant en compte tous les chemins possibles et non pas les seuls chemins de Viterbi ; on fait alors une



$p($« La(det) petite(n) brise(v) la(det) glace(n) »$) = 3,45\ 10^{-5}$
$p($« La(det) petite(adj) brise(n) la(pro) glace(v) »$) = 3,6\ 10^{-5}$
$p($« La(det) petite(n) brise(n) la(det) glace(v) »$) = 1,72\ 10^{-5}$

FIG. 3. Un modèle de Markov caché

moyenne pondérée par les probabilités de ces divers chemins possibles ; c'est la méthode de Baum-Welsh, qui en général ne se stabilise pas mais se rapproche d'un extremum local.

Maintenant, au delà de l'élégance de ce modèle et de l'efficacité des algorithmes (le chemin de Viterbi se calcule en N^2T étapes pour une séquence de T mots si le HMM a N états), on peut se demander quel est l'apport pour la linguistique computationnelle d'une telle méthode. Pour le Traitement Automatique des Langues, l'intérêt est clair : c'est un traitement préalable et l'information supplémentaire ainsi acquise augmente l'efficacité algorithmique de l'analyse grammaticale. Encore faut-il disposer, pour la langue étudiée, d'une famille pertinente de catégories grammaticales et que celles-ci satisfassent des régularités d'ordonnancement : les langues à ordre des mots assez libre ne peuvent bénéficier de ces méthodes. Du point de vue de la connaissance linguistique, l'apport de ces méthodes est minime : les seules propriétés ainsi obtenues sont du genre « la est pronom dans 10% des cas », « un verbe est suivi d'un article dans 40% des cas », etc. À y réfléchir davantage, la faiblesse de ce modèle, comme celle de beaucoup de méthodes probabilistes, résulte de son incapacité à prendre en compte la structure de la phrase, qui est tout de même un ingrédient linguistique central : les règles de succession portent non sur les mots mais sur des groupes de mots structurés, les constituants aussi appelés syntagmes.

Les améliorations de ce modèle n'ont rien d'évident. Certes, il est clairement exclu de prendre en compte les dépendances non bornées, mais on pourrait espérer prendre en compte les quelques mots précédents. Cela n'est également pas possible, car certaines données deviennent trop rares. À la différence des séquences d'ADN construites avec quatre symboles, un corpus contient de l'ordre de 2.10^4 mots différents (en prenant en compte les formes fléchies et les noms propres). La probabilité qu'un triplet de catégories donné se réalise sous la forme d'un triplet de mots devient nulle pour nombre de triplets de mots, car il y a trop de triplets pour qu'il soient tous présents dans le corpus d'apprentissage. Avec les 2.10^4 mots du corpus, les triplets possibles de mots sont au nombre de 8.10^{12} tandis qu'un corpus de 10^6 mots contient aussi $(10^6 - 2)$ triplets de mots : il est donc quasi certain que des triplets de mots pourtant possibles ne figurent jamais dans le corpus, et encore moins dans le corpus d'apprentissage qui aura été annoté à la main.

Ce paragraphe sur les modèles de Markov cachés permet de conclure que pour une langue à ordre des mots stricts, ce genre de méthode constitue un prétraitement intéressant, avec une algorithmique élégante, mais que cela manque de contenu linguistique tandis que le sujet semble mathématiquement clos.

3. Grammaire générative et théorie des langages formels

Nous présentons ici le fleuron de la connexion entre linguistique, mathématiques et informatique : la théorie des langages formels [19, 52]. Cette théorie a connu un vif succès y compris hors de la linguistique dont elle est issue : en informatique pour la compilation ou le parallélisme, en biologie pour l'étude des séquences d'ADN et en mathématiques dans l'étude des groupes, notamment profinis. Bien qu'elle ne soit pas apparue *ex nihilo*, la notion de grammaire formelle ou de langage formel peut être attribuée à Noam Chomsky [20] et les premières propriétés mathématiques de ces objets résulte de sa collaboration avec le scientifique éclectique Marcel-Paul

Schützenberger [18]. Selon Chomsky, la première notion de grammaire formelle et plus précisément hors-contexte est due au grammairien sanskritiste Pāṇini, vers le 5^{e} s. av. J.C. et on peut les rattacher aux systèmes de Post et aux premiers modèles mathématiques du calcul, historiquement plus proches de lui. Ensuite, Noam Chomsky et son école n'ont plus cherché à formaliser eux-mêmes les modèles linguistiques qu'ils développaient, se focalisant sur la description du langage en tant qu'organe biologique soumis à l'évolution [24]. Sur ce sujet on pourra consulter le livre de Jean-Yves Pollock [48] ainsi que l'excellent ouvrage de vulgarisation de Stephen Pinker sur l'instinct de langage [44]. Des chercheurs compétents aussi bien en linguistique théorique qu'en informatique fondamentale, je pense en particulier à Edward Stabler, se sont chargés de formaliser les modèles successifs proposés par Noam Chomsky et son école [56, 55] et notamment le programme minimaliste [24].

Une première considération amène Noam Chomsky à remettre en cause le modèle comportementaliste (*behaviorist*) du langage et de son apprentissage, en vogue dans les années cinquante : une langue ne saurait se réduire à l'ensemble des phrases dites par ses locuteurs jusqu'à ce jour. En effet, on peut toujours produire des phrases nouvelles, qui sont identifiées comme des phrases par les locuteurs. Il suffit de considérer une phrase la plus longue possible dans cette vision finie de la langue et de la faire précéder de « il croit que » pour obtenir une nouvelle phrase, que les locuteurs reconnaîtrons comme correcte. Ainsi c'est la compétence linguistique du locuteur qui est devenu l'objet d'étude et non plus la régularité des distributions dans un corpus comme dans l'ère comportementaliste. .

Noam Chomsky définit plutôt la langue comme un ensemble de règles inconscientes, connu sous le nom de Langage Interne d'un individu X — bien évidemment il y a un rapport entre le langage interne de X et celui de Y s'ils font partie d'une même communauté et se comprennent. Une preuve de l'existence de telles règles et de leur acquisition est la surgénéralisation dont font preuve les enfants. Ceux-ci commencent par dire, comme ils l'entendent autour d'eux, « vous faites » puis ils disent « vous faisez » avant de revenir à la forme « vous faites ». La seule manière d'expliquer la forme non entendue « vous faisez » est de dire qu'une règle de conjugaison a été acquise et qu'elle est abusivement utilisée avant d'être bloquée par une exception. Noam Chomsky va donc proposer une description relativement uniforme de la grammaire vue comme un ensemble de règles.

3.1. Une hiérarchie de grammaires formelles

On se donne deux alphabets disjoints : N (dont les éléments sont appelés non terminaux, qui seront pour nous des catégories grammaticales) et Σ (les terminaux, qui, pour la syntaxe, seront les mots des phrases), un symbole S de N (symbole de départ, phrase) et des règles de production en nombre fini qui sont des éléments de $[(N \cup \Sigma)^* \times N \times (N \cup \Sigma)^*] \times (N \cup \Sigma)^*$ que l'on note souvent ainsi :

$$\alpha \ \rightarrow \ \beta \text{ avec } \begin{cases} \alpha \in (N \cup \Sigma)^* \times N \times (N \cup \Sigma)^* \\ \beta \in (N \cup \Sigma)^* \end{cases}$$

On dit alors que la suite α de terminaux et non terminaux, comportant au moins un non-terminal, se réécrit immédiatement en la suite β de terminaux et non-terminaux. Étant donnée une suite finie γ de $(N \cup \Sigma)^*$ on dit qu'elle se réécrit immédiatement en γ' s'il existe une règle de production $\alpha \ \rightarrow \ \beta$ et deux suites

finies γ_1 et γ_2 telles que $\gamma = \gamma_1\alpha\gamma_2$ et $\gamma' = \gamma_1\beta\gamma_2$. La réécriture entre chaînes est définie comme la clôture transitive $\xrightarrow{*}$ de la réécriture immédiate.

On a alors une définition inductive d'un langage : c'est l'ensemble des suites de *terminaux* que l'on obtient par réécriture à partir du symbole de départ S : $L(G) = \{\alpha \in \Sigma_1^* | S \xrightarrow{*} \alpha\}$.

Sans restriction aucune sur les règles de production, un tel système engendre toutes les parties (ou langages) récursivement énumérables de Σ^*. On peut distinguer diverses familles de langages (de parties de Σ^*) produites par divers types de grammaires formelles :

– Les grammaires contextuelles (*context-sensitive*) Dans leurs règles de production, le membre gauche a un nombre de symboles terminaux ou non terminaux plus petit que celui du membre droit, et par conséquent lors d'une réécriture la longueur de la suite produite augmente. De ce fait, l'appartenance d'une suite $m_1 \cdots m_p$ au langage engendré est décidable : il suffit d'avoir énuméré toutes les suites de longueur inférieure à p pour dire si $m_1 \cdots m_p$ fait partie ou non du langage engendré. Un joli résultat justifie le qualificatif « contextuelles » :

Théorème 3.1. *Tout langage produit par une grammaire contextuelle peut être produit par une grammaire de cette même classe dont toutes les règles sont de la forme $U_1XU_2 \to U_1WU_2$ avec $X \in N$ et $U_1, U_2, W \in (N \cup \Sigma)^*$ et $W \neq \varepsilon$ (X se réécrit en W dans le contexte U_1 à gauche, U_2 à droite).*[36, 29]

– Les grammaires non contextuelles aussi appelées hors-contexte ou algébriques (*context-free*) La partie gauche de leurs règles de production, α, est réduite à un non terminal. On en trouvera un exemple en figure 4. Comme on peut sans modifier le langage engendré se ramener à des grammaires non contextuelles sans productions vide sauf peut-être $S \to \varepsilon$, les grammaires algébriques sont un cas particulier de grammaires contextuelles. L'appartenance est non seulement décidable, mais elle l'est en $O(n^3)$ où n est le nombre de terminaux.

Théorème 3.2 (Forme Normale de Chomsky)**.** *Étant donnée une grammaire non contextuelle il en existe une engendrant le même langage dont les règles de production sont de la forme*

$$X \to Y\ Z \text{ ou } X \to a$$

avec $X, Y, Z \in N$ et $a \in T$. [21, 29]

Théorème 3.3 (Forme Normale de Greibach)**.** *Étant donnée une grammaire non contextuelle il en existe engendrant le même langage dont les règles de production sont de la forme*

$$X \to aX_1 \dots X_p$$

avec $X_1, \dots X_p \in N$ et $a \in T$ et on peut même se ramener à des règles dans lesquelles $p \in \{0, 1, 2\}$.[27, 29]

– Les grammaires régulières aussi appelées rationnelles (*regular*), qui correspondent aux automates de la partie 2 ont des règles de production de la forme $X \to wY$ ou $X \to w$ avec $X, Y \in N$ et $w \in T^*$. les grammaires régulières sont donc en particulier des grammaires non contextuelles.

Le type d'un langage est le type le plus spécifique d'une grammaire engendrant ce langage — car rien empêche d'écrire une grammaire très compliquée (par exemple de type le plus général) engendrant un langage qui peut aussi être produit avec une grammaire très simple (par exemple régulière).

3.2. Compétence et performance

Une distinction fondamentale introduite par Noam Chomsky est celle entre *compétence* et *performance*. En effet, même si nous possédons les règles de notre langue, ce n'est pas pour cela que nous les utilisons autant qu'il est mathématiquement possible. En particulier nous faisons un usage modéré de la récursivité. Cela ne veut pas dire qu'il faille décrire la langue exhaustivement, parce que les phrases ont moins de deux cents mots et qu'il y a moins de cinq cent mille mots. Cela serait aussi stupide que de décrire un ordinateur par un automate sous prétexte que sa mémoire est finie : ce n'est pas ainsi que ça fonctionne. La grammaire décrit donc les règles c'est-à-dire la compétence tandis

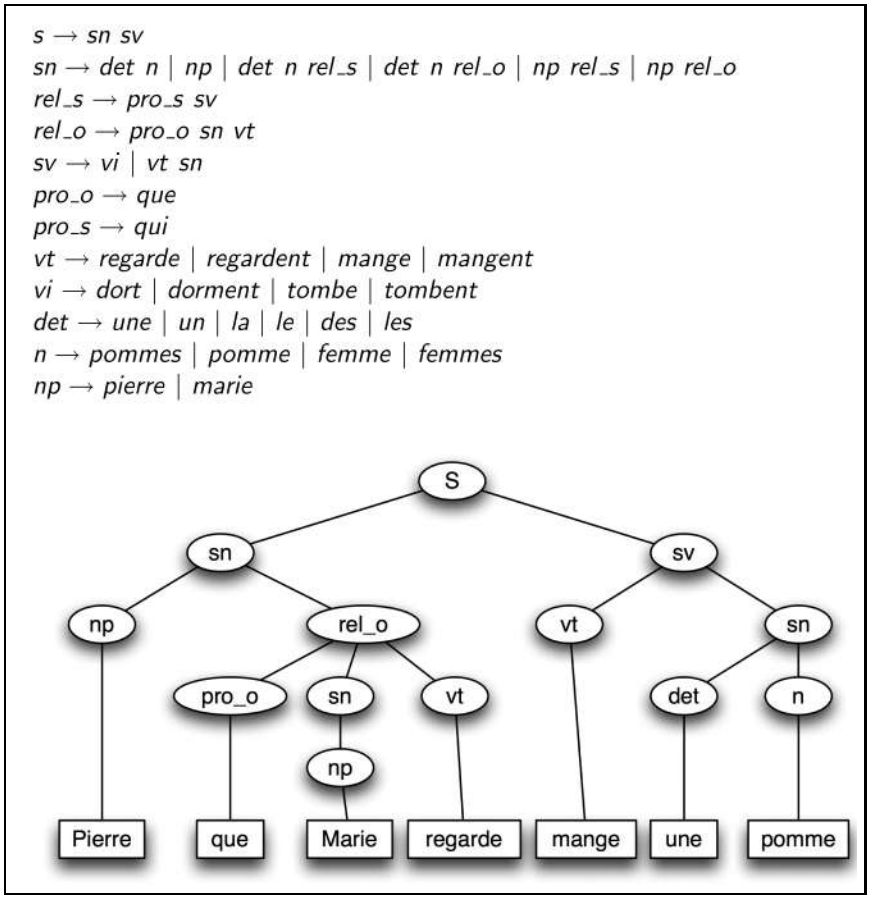


FIG. 4. Un exemple de grammaire non contextuelle et une dérivation

que la performance sera décrite comme une limitation sur l'usage ces règles, et

notamment sur le nombre d'éléments en trop ou manquants que notre mémoire à court terme peut gérer. À titre d'exemple, voici des phrases que notre compétence accepte, mais qui mettent à mal notre performance en raison du trop grand nombre de verbes attendus :

(1) Le loup a dévoré la chèvre.

(2) La chèvre que le loup a dévoré avait mangé le chou.

(3) ? Le chou que la chèvre que le loup a dévoré avait mangé appartenait au passeur.

(4) ?? Le passeur auquel le chou que la chèvre que le loup a dévoré avait mangé appartenait possède plusieurs bateaux.

(5) ??? Les bateaux que le passeur auquel le chou que la chèvre que le loup a dévoré avait mangé appartenait possède sont des barges.

Rapidement les travaux ont plutôt porté sur la compétence que sur la performance, et une question naturelle est la suivante : où se situent les grammaires des langues naturelles dans la hiérarchie abstraite décrite ci dessus, c'est-à-dire quelle est l'allure des règles exprimant la compétence des locuteurs ?

Deux principes empiriques et convaincants guident cette recherche :

(1) Le jugement de grammaticalité et l'analyse, voire la compréhension d'une phrase se fait, comme toute tâche cognitive largement automatisée, « en un temps raisonnable » ce que l'informaticien aura tendance à traduire par « en un temps polynomial en fonction du nombre de mots de la phrase ».

(2) La grammaire d'une langue donnée doit être apprenable à partir d'exemples de phrases de cette langue (et éventuellement d'une grammaire universelle innée factorisant les propriétés communes aux langues humaines). En effet, on sait que pour l'essentiel un enfant acquiert la grammaire de sa langue avant trois ans, avec relativement peu d'exemples au regard de la complexité de la grammaire particulière apprise. On sait aussi que, n'en déplaise aux parents, les exemples négatifs ne servent à rien. Un exemple négatif est une phrase dont on signale à l'enfant qu'elle est incorrecte, en la laissant sans réponse, en la signalant comme incorrecte (*On ne dit pas « Il la te donne. »*) ou en la corrigeant (*On dit « Il te la donne. »*). Quelle que soit l'option choisie, on ne lui fera pas acquérir plus vite l'ordre correct des pronoms du français ! Sur ce sujet on consultera avec profit les articles de Stephen Pinker et son ouvrage de vulgarisation sur l'instinct de langage [45, 46, 44]. Cette condition d'apprenabilité à partir d'exemples positifs seulement, clamée haut et fort par la grammaire générative est souvent laissée de coté dans la conception de modèles formels. En effet, il faut pour cela avoir des exemples structurés, et on ne sait pas trop quelle structure est connue ou perçue de l'enfant : l'intonation délimite certains constituants, le sens des mots sert aussi à structurer la phrase (et on sait qu'il doit être connu préalablement), mais comment ? Dans le second volet nous présenterons certains algorithmes d'apprentissage d'une grammaire formelle à partir d'exemples positifs qui convergent vers la grammaire à acquérir. [33, 9].

Revenons maintenant à la place des langues humaines dans la hiérarchie des grammaires formelles, en laissant de côté la condition d'apprenabilité, trop difficile à prendre en compte. D'après les exemples de relatives objets imbriquées (exemples 1 à 5 où notre performance était mise à l'épreuve, on voit que la langue contient des structures comme

(1) $\text{Sujet}_1\ \text{Sujet}_2\ \text{Sujet}_3\ \cdots\ \cdots\ \text{Verbe}_3\ \text{Verbe}_2\ \text{Verbe}_1$

éventuellement agrémentées de mots entre, ce qui montre que la classe des langages réguliers ne suffit pas à décrire la syntaxe des langues humaines. En effet, en supposant même que l'on ait qu'un seul verbe, *regarde*, et qu'un seul groupe nominal, *Marie*, on aura des phrases du genre

$$Marie\ (que\ Marie)^n\ regarde^n\ regarde\ Marie$$

ce qui est un exemple de structure que les langages réguliers ne peuvent décrire.

Les grammaires hors-contextes ne suffisent pas non plus. Par exemple, les complétives du néerlandais produisent des structures :

(2) Proposition Principale $\text{Sujet}_1\ \text{Sujet}_2\ \text{Sujet}_3 \cdots \text{Verbe}_1\ \text{Verbe}_2\ \text{Verbe}_3 \cdots$

(3) *Ik denk dat ik*$_1$ *haar*$_2$ *de nijlpaarden*$_2$ *zag*$_1$ *voeren*$_2$
Je pense que je elle/la les hippopotames voir nourrir

Je pense que je l'ai vu nourrir les hippopotames

(4) *Ik denk dat ik*$_2$ *Henk*$_2$ *haar*$_3$ *de nijlpaarden*$_3$ *zag*$_1$ *helpen*$_2$
Je pense que je Henk elle/la les hippopotames voir aider
voeren$_3$
nourrir

Je pense que j'ai vu Henk l'aider à nourrir les hippopotames

Ces exemples étudiés dans [14] montre que les arbres attendus sur les complétives du néerlandais ci-dessus ne peuvent pas être produits par une grammaire non contextuelle. Un résultat plus fort de Stuart Shieber montre que le suisse-allemand, indépendamment des arbres associés aux phrases, n'est pas un langage algébrique aussi à cause de la construction des complétives. [54].

C'est pour cela que les linguistes s'accordent à dire que la classe de langages formels nécessaire à la description de la syntaxe des langues humaines est, comme on le voit en figure 5, un peu plus que celle des langages hors contexte, qui peuvent néanmoins fournir une assez bonne approximation. On connaît diverses descriptions de familles de langages analysables en temps polynomial et qui couvrent les phénomènes syntaxiques échappant aux grammaires hors-contexte. Sans doute à cause de sa difficulté, le critère d'apprenabilité a disparu des modèles formels en syntaxe : on ne sait pas si ces formalismes sont ou non apprenables à partir d'exemples positifs et encore moins comment ils les sont à de rares exceptions près qui seront développées dans le seconde volet.

3.3. Apprenabilité et grammaire universelle

Au vu de la complexité de la grammaire à acquérir et du relativement faible nombre d'exemples entendus par le jeune apprenant, l'apprentissage semble impossible, et pourtant tout enfant y réussit sans problème ! Cela a conduit Noam Chomsky à postuler l'existence d'une grammaire universelle et innée. Celle-ci ne doit pas être vue comme régissant la structure de toutes les phrases de toutes les langues mais plutôt comme une matrice de grammaire, une grammaire paramétrable. Elle est présentée comme un organe issu de l'évolution dont serait doté l'être humain à sa naissance. Ainsi la classe des grammaires humaines se trouve-t-elle considérablement restreinte, et surtout, son apprentissage devient un processus calculatoire raisonnable puisqu'il ne s'agit que de déterminer la valeur des paramètres. Pour prendre un exemple simpliste, si l'enfant sait que le verbe

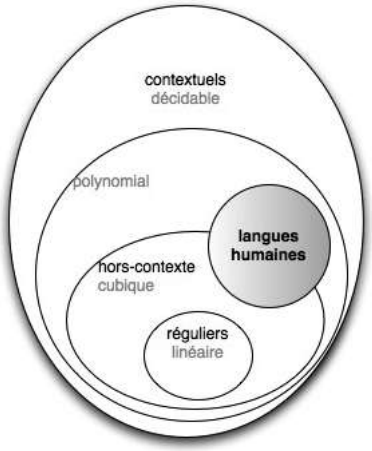


FIG. 5. Place des langages formels correspondant aux langues humaines dans la hiérarchie de Chomsky. En gris, la complexité de l'analyse en fonction du nombre de mots dans la phrase analysée.

conduit a un sujet *maman* et un objet *voiture* à partir du seul exemple *maman conduit voiture* il peut en déduire que l'ordre de sa langue est *sujet verbe objet*.

La grammaire générative s'est donc attachée à l'étude de nombreuses langues afin de dégager des principes universels qui seraient communs à toutes les langues. Les quelques principes qui ont été dégagés ne sont hélas pas particulièrement faciles à exprimer dans le cadre des grammaires formelles présentées ci-dessus. Le plus souvent ils concernent le lien entre la structure syntaxique de la phrase et sa structure logique. Afin d'illustrer notre propos sur les universaux, mentionnons-en deux :

– TOUT GROUPE NOMINAL DOIT RECEVOIR UN CAS et seul un verbe *conjugué* donne le cas sujet. Cette formulation présuppose que toutes les langues aient des verbes ainsi que des groupes nominaux, ce qui est assez vrai. Elle suppose aussi que les verbes se conjuguent, et là encore on peut le prétendre, quitte à dire que leur conjugaison n'est pas forcément visible. C'est ainsi qu'on peut parler de cas en français, celui-ci étant invisible, sauf sur les pronoms : « ils », « les » et « leur » sont bien des formes différentes. Une fois ceci admis, on peut effectivement expliquer par ce principe, la grammaticalité et l'agrammaticalité des phrases ci-dessous : lorsque « arriver » est à l'infinitif le mot « été » vient obligatoirement prendre la place de « il » (du reste vide) pour recevoir le cas sujet du verbe conjugué « semble » (on notera que la situation est similaire en anglais)

(1) Il semble que l'été arrive. It seems that summer is arriving.
(2) L'été semble arriver. Summer seems to arrive.
(3) * Il semble que l'été arriver. * It seems that summer to arrive.
(4) * Il semble l'été arriver. * It seems summer to arrive.

– LIAGE Une expression α en lie une autre β si, elle sont coréférentes, et si, dans l'arbre syntaxique, α est le descendant immédiat de l'un des ancêtres de β. Les principes A, B, et C affirment respectivement qu'un pronom réfléchi doit être lié dans sa clause (A), qu'un pronom non réfléchi ne doit jamais être lié dans sa clause (B), et qu'une expression référentielle (non pronom, identifiable) ne doit jamais être liée.

(1) Le chien de Carlos pense que il ne l'aime pas.
(2) Le chien de Carlos pense que il ne s'aime pas.

Dans le premier cas, on sait que « il$\neq$ l' » sinon le principe (B) serait enfreint, mais toutes les autres égalités et inégalités avec « chien » et « Carlos » à la place de « il » et « l » sont possibles et dans le second cas, on sait d'après le principe (B) que « il = s' » mais toutes les autres égalités et inégalités avec « chien », « Carlos », voire même avec d'autres référents du discours à la place de « il=s » sont possibles.

D'autres principes plus sophistiqués régissent les coréférences possibles et impossibles entre pronoms et groupes nominaux. Ainsi dans l'exemple 1 « il » ne peut être « Chomsky » alors que dans l'exemple 2 « il » peut être « Chomsky » bien que cela ne soit pas obligatoire.

(1) Il a aimé trois livres que Chomsky a écrit.
(2) Combien de livres que Chomsky a écrit a-t-il aimé ?

4. Extensions des grammaires formelles classiques

4.1. Grammaires d'arbres

Bien évidemment, les arbres sont d'un intérêt tout particulier pour décrire la syntaxe de la phrase, comme en témoignent les arbres que nous avons dessinés à l'école, somme toute peu différents de celui de la figure 4. Leur sommets intermédiaires correspondent aux constituants, aussi appelés syntagmes. Les règles se formulent agréablement en termes de constituants. Par exemple, dans les grammaires transformationnelles, on peut spécifier qu'un groupe nominal interrogatif (un sous arbre) doit se déplacer en tête de phrase. Une grammaire transformationnelle produit d'abord la structure profonde similaire à celle de la phrase affirmative, sorte d'étape intermédiaire incorrecte « *il a aimé combien de livres que Chomsky a écrit* », et l'interrogatif "combien" déclenche alors une transformation qui déplace le constituant « *combien de livres que Chomsky a écrit* » en tête de phrase, produisant la phrase correcte « *combien de livres que Chomsky a écrit a-t-il aimé ?* » – avec en sus une inversion du sujet *il*.

Des arbres sont naturellement produits par les dérivations dans les grammaires non contextuelles. Mais on peut aussi travailler directement sur les arbres. Ainsi Aravind Joshi a-t-il défini des grammaires dites grammaires d'arbres adjoints qui opèrent directement sur les arbres. [30, 1, 31] Une grammaire d'arbres adjoints est définie par la donnée d'arbres élémentaires et d'arbres adjoints. Chaque arbre, adjoint ou élémentaire porte en chaque nœud interne et sur les feuilles des non terminaux (des catégories grammaticales) et l'une au moins des feuilles est un terminal (un mot). Les arbres adjoints satisfont une contrainte supplémentaire : le non terminal de la racine se retrouve sur l'une des feuilles, et pour distinguer

cette feuille au cas où ce non terminal figure sur plusieurs feuilles, on inscrit une étoile sur cette feuille que l'on appelle pied de l'arbre adjoint. On trouvera des exemples d'arbres élémentaires et adjoints, une susbstitution et une adjonction en figure 6 ci-après. Les arbres se composent par substitution ou par adjonction. La substitution consiste à mettre à la place d'une feuille d'un arbre déjà construit un arbre élémentaire étiqueté par un non terminal X un arbre de racine X. L'adjonction consiste à insérer (adjoindre) sur un nœud X de l'arbre déjà construit un arbre adjoint dont la racine et le pied sont étiquetée X : le nœud de l'arbre principal, disons X, est scindé en deux nœuds X, celui du haut devient la racine de l'arbre adjoint tandis que celui du bas devient le pied de l'arbre adjoint. On peut aussi insérer des marques sur certains nœuds des arbres de la grammaire qui interdisent l'adjonction.

Les grammaires d'arbres adjoints (TAG) sont un formalisme bien adapté à la syntaxe du langage naturel, car il constituent une classe plus riche que les grammaires non contextuelles (engendrant des langages comme $a^n b^n c^n$) et pourtant analysable en temps polynomial : l'analyse d'une phrase de n mots se fait en temps $O(n^6)$. De plus, d'un point de vue pratique, Aravind Joshi et ces collègues ont constitué la *Penn Tree Bank* qui compte deux cent mille arbres pour décrire l'anglais et Anne Abeillé a développé une grammaire conséquente du français. [2]

4.2. Grammaires d'unification

Dans l'exemple de grammaire non contextuelle donné précédemment, rien ne permet d'éviter de dériver *Pierre dorment* ou *Les pommes tombe* ou *La pommes tombe*. Une solution inélégante est d'avoir un non terminal pour les verbes au singulier et un pour les verbes au pluriel ; un non terminal pour les noms masculins singuliers, un pour les noms féminins singuliers, un pour les noms masculins pluriels, un pour les noms féminins pluriel,... On imagine l'allure de la grammaire si on prend en compte toutes les sortes d'accords possibles ! Une solution apparue conjointement avec Prolog est de voir les non terminaux comme des prédicats et d'utiliser des variables qui peuvent prendre les valeurs pour factoriser les règles. Une règle comme $sn[N, G] \to det[N, G]n[N, G]$, en notant les variables par des majuscules) permet de spécifier que le déterminant et le nom d'un syntagme nominal doivent avoir le même nombre et le même genre qui sont aussi ceux du syntagme nominal tout entier. Une règle comme $sn[sg, f] \to pomme$ signifie que « pomme » est un nom féminin singulier, sg et f étant des constantes. C'est ainsi qu'on pourra dériver *la pomme* et non *les pomme* car *les* assigne la valeur pl à N, tandis que *pomme* lui assigne la valeur singulier. De telles grammaires s'appellent des grammaires de clauses définies (DCG) et ont été particulièrement étudiées par David Warren, Fernando Pereira et Stuart Shieber [42, 43] et on y retrouve les pionnières, les grammaires de métamorphose d'Alain Colmerauer [25]. Si on modifie la grammaire non contextuelle de la figure 4 afin qu'elle gère les phénomènes d'accord, on obtient la DCG donnée en figure 7.

Rapidement, ces techniques ont aussi été utilisées pour traiter simultanément, c'est-à-dire dans la grammaire même, de la structure combinatoire syntaxique mais aussi sémantique et pragmatique. Le principe utilisé, l'unification, est à l'origine même de Prolog. Ce langage de programmation permet, d'unifier non seulement des variables et des constantes comme on l'a vu ci-dessus, mais aussi d'unifier

des termes. Unifier deux termes t_1 et t_2, c'est trouver une substitution σ, c'est-à-dire une application des variables vers les termes, qui appliquée à t_1 et t_2 donne le même terme : $\sigma(t_1) = \sigma(t_2)$. L'unification de deux termes t_1 et t_2 n'est pas toujours possible, mais lorsqu'elle l'est, il existe une substitution unique υ, appelé unificateur principal, telle que tout autre unificateur τ de t_1 et t_2 s'écrit $\tau = \sigma \circ \upsilon$.

Dans les années quatre-vingts Prolog et l'unification ont donné naissance aux grammaires d'unification. Celle-ci usent du procédé ci-dessus, l'unification, non seulement pour spécifier que certaines valeurs immédiatement accessibles doivent

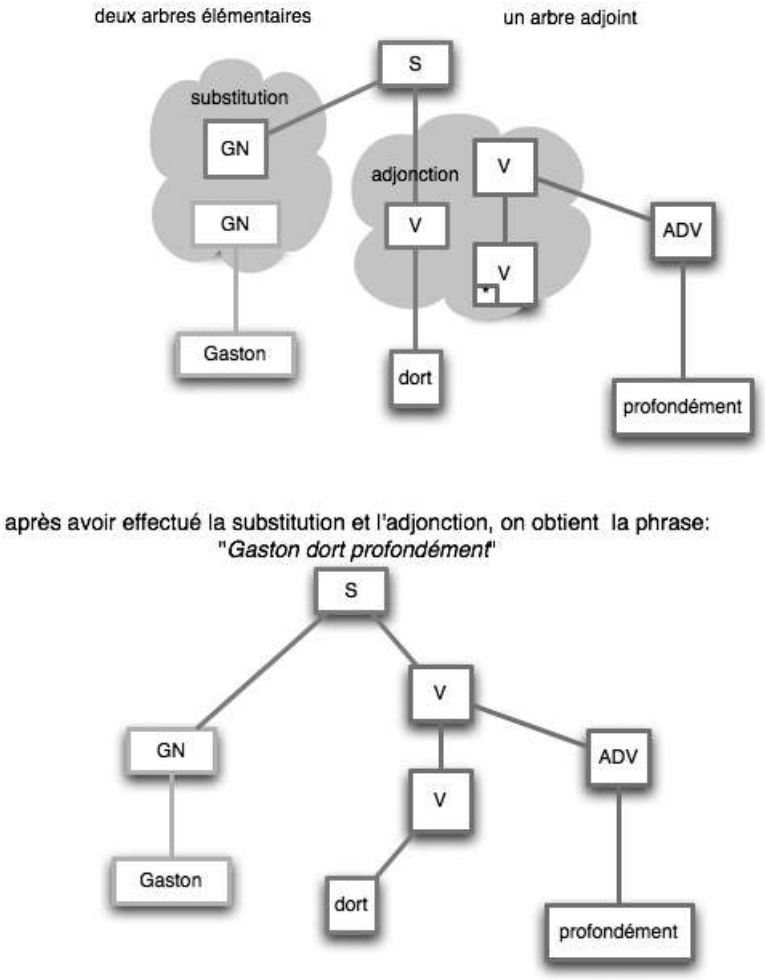


FIG. 6. Une mini grammaire TAG

être partagées, mais aussi que des valeurs enchâssées dans de telles structures doivent être égales. On peut ainsi spécifier que dans une construction comme *Je te permets de venir* le pronom clitique *te* est à la fois l'objet de *permets* et le sujet de *venir*.

4.3. Model-theoretic syntax

On pourrait dire que les grammaires d'unification abusent de ce procédé, car en toute généralité, elles décrivent toutes les langages récursivement énumérables : elle caractérisent donc mal les langues humaines parmi tous les langages formels possibles, et du point de vue pratique, la complexité algorithmique de l'analyse est sans borne : indécidable ou exponentielle pour certains fragments. Les grammaires lexicales-fonctionnelles de Joan Bresnan [13] se distinguent dans cette famille de grammaires par leur réalisme cognitif et linguistique ainsi que par l'incorporation de la structure sémantique. Elles engendrent tous les langages contextuels : la correction et l'analyse d'une phrase est donc décidable mais en en temps pas forcément polynomial en fonction du nombre de mots dans la phrase.

Dans l'industrie les DCG sont toujours populaires et les linguistes apprécient, pour décrire la langue, la plasticité des grammaires d'unification et en particulier des populaires *Head-driven Phrase Structure Grammars*, HPSG [47, 1]. Les conditions qu'on peut exprimer par l'unification sont si variées qu'on perd l'intérêt pour les règles, et qu'on débouche alors sur les grammaires de construction fortement liées

$s \to sn[N, G]\ sv[N]$
$sn[N, G] \to det[N, G]\ n[N, G] \mid np[N, G] \mid det[N, G]\ n[N, G]\ rel\_s[N] \mid det[N, G]\ n[N, G]\ rel\_o$
$\quad \mid np[N, G]\ rel\_s[N] \mid np[N, G]\ rel\_o$
$rel\_s[N] \to pro\_s\ sv[N]$
$rel\_o \to pro\_o\ sn[N, G]\ vt[N]$
$sv[N] \to vi[N] \mid vt[N]\ sn[N', G']$
$pro\_o \to que$
$pro\_s \to qui$
$vt[sg] \to regarde \mid mange$
$vt[pl] \to regardent \mid mangent$
$vi[sg] \to dort \mid tombe$
$vi[pl] \to dorment \mid tombent$
$det[sg, f] \to une \mid la$
$det[sg, m] \to un \mid le$
$det[pl, G] \mid des \mid les$
$n[pl, f] \to pommes \mid femmes$
$n[sg, f] \to pomme \mid femme$
$np[m] \to pierre$
$np[f] \to marie$
$G \to$ Pierre que Marie regarde dort.
$G \not\to$ Pierre que Marie regarde dorment.

FIG. 7. Une grammaire de clauses définies G

à la programmation logique par contraintes et à une vision qui donne son nom au paragraphe suivant.

En liaison avec le paragraphe précédent, et en avance sur le second volet de notre présentation, finissons notre tour d'horizon de la théorie des langages en linguistique computationnelle par une note de théorie des modèles. Depuis les années soixante, on sait décrire un langage régulier comme l'ensemble des modèles finis d'une théorie logique. Plus précisément, on peut dire que les chaînes ou les arbres du langage sont ceux qui satisfont certaines propriétés, généralement exprimées en terme de précédence (pour les suites et les arbres) et de dominance (pour les arbres uniquement). Relativement récemment, des résultats classiques sur la description logique de langages de chaînes ou d'arbres ont trouvé un écho linguistique, à mettre en rapport avec la programmation logique par contraintes.

L'un des pionniers à œuvrer en ce sens a été James Rogers [51]. Il a réussi à donner une description en logique monadique du second ordre — une logique où l'on peut quantifier sur les prédicats à une place, c'est-à-dire les sous ensembles — d'une partie conséquente du modèle proposé par Noam Chomsky dans les années quatre-vingts, *Government and Binding* [22, 23]. Ce travail a été poursuivi par Uwe Mönnich, Jens Michaelis et Frank Morawietz [39, 41] puis étendu et simplifié par nous mêmes dans [35]. En se plaçant dans le cadre du programme de Chomsky [24], on montre que les arbres produits par les grammaires minimalistes d'Edward Stabler [55] comme l'image d'un ensemble d'arbres définissable en logique monadique du second ordre par une relation binaire entre arbres, relation elle-aussi définissable en logique monadique du second ordre. On peut aussi décrire par des contraintes des formalismes issus de HSPG et les grammaires de construction, lignée dans laquelle s'inscrivent les grammaires de propriétés de Philippe Blache. [10]

Cette approche, où les structures ne sont plus vu comme les productions d'un mécanisme génératif mais comme les structures satisfaisant certaines propriétés, ouvre une perspective intéressante. Elle permet d'envisager une caractérisation des langues humaines par les logiques qui peuvent décrire leurs constructions plutôt que par le type de grammaire qui les produit. Comme les langues humaines sont plutôt transverses à la hiérarchie de Chomsky, c'est une piste à explorer. Cette approche permet aussi de traiter plus facilement des énoncés relativement corrects, en posant que certaines contraintes doivent absolument être satisfaites, tandis que d'autres peuvent ne pas l'être donnant alors des énoncés moins corrects, par exemple affectés d'un coefficient de correction moindre. On notera toutefois que du point de vue algorithmique, ces descriptions sont généralement inutilisables que ce soit pour l'analyse automatique ou l'acquisition de la grammaire. Certains formalismes, comme les grammaires non contextuelles ou les grammaires minimalistes admettent les deux types de présentation, dont les avantages et inconvénients comparés sont discutés dans [49].

5. Conclusions et perspectives du premier volet

La théorie des langages, issue de préoccupations linguistiques, est devenu un sujet d'intérêt par elle-même. Elle s'applique désormais à bien d'autres domaines comme la bio-informatique ou la théorie des groupes. Pour la linguistique computationnelle, on peut dire qu'on dispose à l'heure actuelle de formalismes morpho-syntaxiques efficaces qui peuvent analyser automatiquement de larges parts des

langues humaines : si on dispose de corpus correctement annotés pour construire des grammaires de taille suffisante et si on s'aide de probabilités pour ne retenir que les meilleures analyses, l'analyse morpho-syntaxique à large échelle est un traitement automatique envisageable. [11, 12, 40]

Bien des questions restent ouvertes, telle la question initiale : même si nous connaissons des éléments de réponse, savons nous réellement à quelle classe de langages formels correspondent nos langues humaines ? Du point de vue des phrases conçues comme de simples suites de mots, on sait que ce sont des langages un peu plus riches que ceux qu'engendrent les grammaires non contextuelles, analysables en temps polynomial en fonction du nombre de mots, mais quels sont exactement ces langages ? Surtout, que savons nous des arbres décrivant la structure syntaxique des phrases ? Hormis l'insuffisance des langages réguliers d'arbres, dont les langages de chaînes sont non contextuels, bien peu est connu.

Des deux principes naturels posés par Chomsky, qui contraignent la classe de langages nécessaires, l'analyse rapide (polynomiale), et l'acquisition automatique, le dernier me semble avoir complètement découragé les approches formelles, malgré quelques travaux sur les grammaires catégorielles que nous aborderons dans le second volet. C'est pourtant une question linguistiquement pertinente de modéliser l'acquisition de la syntaxe par le jeune enfant. C'est aussi une question pratique importante de pouvoir acquérir automatiquement une grammaire électronique conséquente d'après des exemples que fournissent les corpus électroniques (Internet, archives des grands quotidiens,...). En effet, les solutions standard basées sur des méthodes statistiques ne donnent pas de bons résultats. Il serait donc temps de se pencher sur les méthodes symboliques proposées par les linguistes pour modéliser l'acquisition de la syntaxe par l'enfant, notamment parce qu'elles possèdent de réels critères de convergence.

Finalement, afin d'ouvrir la voie aux méthodes logiques du second volet, c'est sans doute l'absence de lien vers la sémantique des phrases analysées qui est la plus frustrante dans la théorie des langages classique pour la syntaxe des langues. Pourtant, dans ce que Noam Chomsky appelle syntaxe, y compris dans les deux principes de la grammaire universelle que nous avons donnés, la sémantique joue déjà un rôle : qui fait quoi, quel est le rôle de tel ou tel constituant dans les prédicats de la phrase ? À qui ou à quoi renvoient les pronoms ? Pis encore, quelle est la structure d'un discours ou un dialogue, et quel relation entretient-elle avec les phrases qui le constituent ? D'un point de vue pratique, que faire des milliers d'arbres d'analyse produits par les analyseurs à large couverture ?

C'est à travers le lien entre théorie des langages et logique qu'on peut joindre syntaxe et sémantique. Ce sera le sujet du second volet de notre présentation, avant de conclure par une discussion des mérites et apports des méthodes symboliques, théorie des langages et logique, au traitement automatique des langues, à la linguistique et à l'informatique fondamentale.

6. Références

[1] A. Abeillé – *Les nouvelles syntaxes*, Armand Colin, 1993.
[2] ———, *Une grammaire électronique du français*, C.N.R.S. Editions, 2002.
[3] A. Abeillé & P. Blache – « Grammaires et analyseurs syntaxiques », in *Ingénierie des langues* (J.-M. Pierrel, éd.), Hermès Sciences, 2000.

[4] A. Arnauld & C. Lancelot – *Grammaire générale et raisonnée*, Le Petit, 1660, Appelée *Grammaire de Port-Royal*. Rééditée en 1997 par les Éditions Allia.
[5] A. Arnauld & P. Nicole – *La logique ou l'art de penser : contenant outre les règles communes, plusieurs observations nouvelles, propres à former le jugement.*, G. Desprez, 1683, Réédité par Vrin en 2002.
[6] Y. Bar-Hillel – « The present status of automatic translation of languages. », *Advances in Computers* **1** (1960), p. 91–163.
[7] M. Baratin & F. Desbordes – *L'analyse linguistique dans l'antiquité classique – I les théories*, Klincksieck, 1981.
[8] L. E. Baum, T. Petrie, G. Soules & N. Weiss – « A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. », *Ann. Math. Stat.* **41** (1970), p. 164–171.
[9] D. Bechet, R. Bonato, A. Dikovsky, A. Foret, Y. Le Nir, E. Moreau, C. Retoré & I. Tellier – « Modèles algorithmiques de l'acquisition de la syntaxe : concepts et méthodes, résultats et problèmes », *Recherches Linguistiques de Vincennes* (2007).
[10] P. Blache – *Les grammaires de propriétés : des contraintes pour le traitement automatique des langues naturelles*, Hermès, 2001.
[11] P. Boullier, L. Clément & B. Sagot – « Chaîne de traitement syntaxique », in *TALN*, 2005, p. 103–112.
[12] ———, « Un analyseur lfg efficace pour le français : Sxlfg », in *TALN*, 2005, p. 403–408.
[13] J. Bresnan (éd.) – *The mental representation of grammatical relations*, MIT Press, Cambridge, MA, 1982.
[14] J. Bresnan, R. M. Kaplan, S. Peters & A. Zaenen – « Cross-serial dependencies in Dutch », *Linguistic Inquiry* **13** (1982), no. 4, p. 613–635.
[15] L. Bréhelin & O. Gascuel – « Modèles de markov cachés et apprentissage de séquences », in *Le Temps, l'Espace, l'Evolutif, dans les sciences du traitement de l'information* (H. Prade, R. Jeansoulin & C. Garbay, éds.), Cepadues, Toulouse, 2000.
[16] J. Busquets – *Logique et langage : apports de la philosophie médiévale*, Presses Universitaires de Bordeaux, 2006.
[17] C. Casadio – « Semantic categories and the development of categorial grammars », in *Categorial Grammars and Natural Language Structures* (R. Oehrle, E. Bach & D. Wheeler, éds.), Reidel, Dordrecht, 1988, p. 95–124.
[18] N. Chomsky & M.-P. Schützenberger – « The algebraic theory of context-free languages », in *Computer Programming and Formal Systems* (P. Braffort & D. Hirschberg, éds.), North-Holland, Amsterdam, The Netherlands, 1963, p. 118–161.
[19] N. Chomsky – « The logical structure of linguistic theory », Revised 1956 version published in part by Plenum Press, 1975 ; University of Chicago Press, 1985, 1955.
[20] ———, « Three models for the description of language », *IRE Transactions on Information Theory* **IT2** (1956), p. 113–124.
[21] ———, « On certain formal properties of grammars », *Information and Control* **2** (1959), no. 2, p. 137–167.
[22] ———, *Some concepts and consequences of the theory of government and binding*, MIT Press, Cambridge, MA, 1982.
[23] ———, *La nouvelle syntaxe*, Seuil, Paris, 1987, Traduction de [22].
[24] ———, *The minimalist program*, MIT Press, Cambridge, MA, 1995.
[25] A. Colmerauer – « les grammaires de métamorphose », Tech. report, université d'Aix-Marseille, 1975.
[26] L. Gleitman & M. Liberman (éds.) – *An invitation to cognitive sciences, vol. 1 : Language*, MIT Press, 1995.
[27] S. A. Greibach – « A new normal-form theorem for context-free phrase structure grammars », *Journal of the ACM* **12** (1965), no. 1, p. 42–52.
[28] M. Grevisse – *Le bon usage*, 13^e édition par André Goosse éd., Duculot, Paris – Louvain-la-Neuve, 1993.
[29] M. A. Harrison – *Introduction to formal language theory*, Addison Wesley, 1978.
[30] A. Joshi, L. Levy & M. Takahashi – « Tree adjunct grammar », *Journal of Computer and System Sciences* **10** (1975), p. 136–163.

[31] A. Joshi & Y. Schabes – « Tree adjoining grammars », in *Handbook of Formal Languages* [52].
[32] D. Jurafsky & J. H. Martin – *Speech and langage processing : an introduction to natural language processing, computational linguistics and speech recognition*, Prentice Hall, 2000.
[33] M. Kanazawa – *Learnable classes of categorial grammars*, Studies in Logic, Language and Information, FoLLI & CSLI, 1998, distributed by Cambridge University Press.
[34] W. Kneale & M. Kneale – *The development of logic*, 3rd éd., Oxford University Press, 1986.
[35] G. Kobele, C. Retoré & S. Salvati – « An automata-theoretic approach to minimalism », in *Model Theoretic Syntax at 10 -ESSLLI 2007 Workshop* (F. of Logic Language & Information, éds.), 2007.
[36] S.-Y. Kuroda – « Classes of languages and linear-bounded automata », *Information and Control* **7** (1964), no. 2, p. 207–223.
[37] J. Léon & M. Cori – « La constitution du TAL — Étude historique des dénominations et des concepts », *Traitement Automatique des Langues* **43** (2002), no. 3, p. 21–55.
[38] C. Manning & H. Schütze – *Foundations of statistical natural language processing*, MIT Press, 1999.
[39] U. Mönnich, J. Michaelis & F. Morawietz – « On minimalist attribute grammars and macro tree transducers », in *Linguistic Form and its Computation* (C. Rohrer, A. Rossdeutscher & H. Kamp, éds.), CSLI Publications, 2004.
[40] R. Moot – « Automated extraction of type-logical supertags from the spoken dutch corpus », in *The Complexity of Lexical Descriptions and its Relevance to Natural Language Processing : A Supertagging Approach* (S. Bangalore & A. Joshi, éds.), MIT Press, 2007.
[41] F. Morawietz – *Two-step approaches ot natural language formalisms*, Studies in Generative Grammar, Mouton de Gruyter, Berlin · New York, 2003.
[42] F. C. N. Pereira & S. M. Shieber – *Prolog and natural-language analysis*, CSLI Lecture Notes, no. 10, University of Chicago Press, Chicago, IL, 1987.
[43] F. C. N. Pereira & D. H. D. Warren – « Definite clause grammars for language analysis », in *Readings in Natural Language Processing* (K. S.-J. B. J. Grosz & B. L. Webber, éds.), Morgan Kaufmann, Los Altos, 1980, p. 101–124.
[44] S. Pinker – *The language instinct*, Penguin Science, 1994.
[45] S. Pinker – « Language acquisition », in *An invitation to cognitive sciences – Volume 1 : Language* [26], p. 135–182.
[46] ———, « Why the child holded the baby rabbits », in *An invitation to cognitive sciences – Volume 1 : Language* [26], p. 107–133.
[47] C. Pollard & I. A. Sag – « Head-driven phrase structure grammar », Tech. Report LI221, Linguistic Institute, Stanford, CA, USA, 1987.
[48] J.-Y. Pollock – *Langage et cognition : le programme minimaliste de la grammaire générative*, Presses Universitaires de France, Paris, 1997.
[49] G. K. Pullum & B. C. Scholz – « On the distinction bewteen model-theoretic and generative-enumerative syntax », in *Logical Aspects of Computational Linguistics, LACL'2001* (P. de Groote, G. Morrill & C. Retoré, éds.), LNCS/LNAI, no. 2099, Springer-Verlag, 2001, p. 17–43.
[50] E. Roche & Y. Schabes – *Finite-state language processing*, MIT Press, 1997.
[51] J. Rogers – *A descriptive approach to language-theoretic complexity*, Studies in Logic, Language and Information, CSLI, 1998.
[52] G. Rozenberg & A. Salomaa (éds.) – *Handbook of formal languages*, Springer Verlag, Berlin, 1997.
[53] F. de Saussure – *Cours de linguisitique générale*, seconde éd., Editions Payot, 1981.
[54] S. M. Shieber – « Evidence against the context-freeness of natural language », *Linguistics and Philosophy* **8** (1985), p. 333–343.
[55] E. Stabler – « Derivational minimalism », in *Logical Aspects of Computational Linguistics, LACL'96* (C. Retoré, éd.), LNCS/LNAI, vol. 1328, Springer-Verlag, 1997, p. 68–95.
[56] E. P. Stabler – *The logical approach to syntax : foundations, specifications, and implementations of theories of government and binding*, MIT Press, 1992.